3GPP TS 26.253 V0.0.1 (2023-11)

Technical Specification

3rd Generation Partnership Project;
Technical Specification Group Services and System Aspects;
Codec for Immersive Voice and Audio Services;
Detailed Algorithmic Description incl. RTP payload format and SDP parameter definitions
(Release 18)

Contents

Foreword

This Technical Specification has been produced by the 3rd Generation Partnership Project (3GPP).

The contents of the present document are subject to continuing work within the TSG and may change following formal TSG approval. Should the TSG modify the contents of the present document, it will be re-released by the TSG with an identifying change of release date and an increase in version number as follows:

Version x.y.z

where:

x the first digit:

1 presented to TSG for information;

2 presented to TSG for approval;

3 or greater indicates TSG approved document under change control.

y the second digit is incremented for all changes of substance, i.e. technical enhancements, corrections, updates, etc.

z the third digit is incremented when editorial only changes have been incorporated in the document.

In the present document, modal verbs have the following meanings:

shall	indicates a mandatory requirement to do something
shall not	indicates an interdiction (prohibition) to do something

The constructions "shall" and "shall not" are confined to the context of normative provisions, and do not appear in Technical Reports.

The constructions "must" and "must not" are not used as substitutes for "shall" and "shall not". Their use is avoided insofar as possible, and they are not used in a normative context except in a direct citation from an external, referenced, non-3GPP document, or so as to maintain continuity of style when extending or modifying the provisions of such a referenced document.

should	indicates a recommendation to do something
should not	indicates a recommendation not to do something
may	indicates permission to do something
need not	indicates permission not to do something

The construction "may not" is ambiguous and is not used in normative elements. The unambiguous constructions "might not" or "shall not" are used instead, depending upon the meaning intended.

can	indicates that something is possible
cannot	indicates that something is impossible

The constructions "can" and "cannot" are not substitutes for "may" and "need not".

will	indicates that something is certain or expected to happen as a result of action taken by an agency the behaviour of which is outside the scope of the present document
will not	indicates that something is certain or expected not to happen as a result of action taken by an agency the behaviour of which is outside the scope of the present document
might	indicates a likelihood that something will happen as a result of action taken by some agency the behaviour of which is outside the scope of the present document

might not	indicates a likelihood that something will not happen as a result of action taken by some agency the behaviour of which is outside the scope of the present document

In addition:

is	(or any other verb in the indicative mood) indicates a statement of fact
is not	(or any other negative verb in the indicative mood) indicates a statement of fact

The constructions "is" and "is not" do not indicate requirements.

1 Scope

The present document is a detailed description of the signal processing algorithms of the Immersive Voice and Audio Services (IVAS) coder including the IVAS renderer.

2 References

The following documents contain provisions which, through reference in this text, constitute provisions of the present document.

- References are either specific (identified by date of publication, edition number, version number, etc.) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies. In the case of a reference to a 3GPP document (including a GSM document), a non-specific reference implicitly refers to the latest version of that document *in the same Release as the present document*.

[1] 3GPP TR 21.905: "Vocabulary for 3GPP Specifications".

[2] 3GPP TS 26.441: "Codec for Enhanced Voice Services (EVS); General Overview".

[3] 3GPP TS 26.445: "Codec for Enhanced Voice Services (EVS); Detailed Algorithmic Description".

[4] 3GPP TS 26.447: "Codec for Enhanced Voice Services (EVS); Error concealment of lost packets".

[5] 3GPP TS 26.250: "Codec for Immersive Voice and Audio Services (IVAS); General overview".

[6] 3GPP TS 26.251: "Codec for Immersive Voice and Audio Services (IVAS); C code (fixed-point)".

[7] 3GPP TS 26.252: "Codec for Immersive Voice and Audio Services (IVAS); Test Sequences".

[8] 3GPP TS 26.254: "Codec for Immersive Voice and Audio Services (IVAS); Rendering".

[9] 3GPP TS 26.255: "Codec for Immersive Voice and Audio Services (IVAS); Error concealment of lost packets".

[10] 3GPP TS 26.256: "Codec for Immersive Voice and Audio Services (IVAS); Jitter Buffer Management".

[11] 3GPP TS 26.258: "Codec for Immersive Voice and Audio Services (IVAS); C code (floating point)".

[12] De Boor, C., B(asic)-Spline Basics; https://ftp.cs.wisc.edu/Approx/bsplbasic.pdf

3 Definitions of terms, symbols and abbreviations

3.1 Terms

For the purposes of the present document, the terms given in TR 21.905 [1] and the following apply. A term defined in the present document takes precedence over the definition of the same term, if any, in TR 21.905 [1].

frame: an array of audio samples or metadata spanning a 20-ms time duration.

3.2 Symbols

For the purposes of the present document, the following symbols and conventions to mathematical expressions apply:

E	Energy

F	Sample rate
g	Gain
I	Metadata
L	Length of an audio buffer in samples (e.g., L_{frame} is the length of a frame in samples)
M	Mode (e.g., $M_{element}$ is the element mode or M_{core} is the core coder mode)
N	Number of audio channels
R	Bitrate
s	Audio signal in time domain
S	Audio signal in frequency domain (spectrum)

$s(n)$	(n) indicates the nth sample of the audio signal s
$E(b)$	(b) indicates the bth band in the energy vector E
$S(k)$	(k) indicates the kth frequency bin
$S(k,n)$	(k,n) indicates the nth time slot of the kth frequency bin of the discrete time-frequency spectrum
$s(m;n)$	$(m;n)$ indicates the nth sample within mth subframe
$s(t)$	(t) indicates the time instant t in the continuous time domain
S_i, s_i	Lower index i indicates the ith channel (input or transport) of a multi-channel signal. The indexing starts from 1
$s_{HP20}(n)$	HP20 filtered time domain signal
$s_{inp}(n)$	Input signal to IVAS encoder
S^{MDFT}	Superscript $MDFT$ indicates the type of frequency-domain transform; also FFT and $CLDFB$
$\hat{S}$	Quantized (coded) version (of frequency-domain audio signal)
$\bar{E}$	Mean value (of energy)
$s^{[-1]}(n)$	Upper index indicates a particular frame, e.g., $[-1]$ refers to the previous frame. When omitted, current frame is assumed by default

3.3 Abbreviations

For the purposes of the present document, the abbreviations given in TR 21.905 [1] and the following apply. An abbreviation defined in the present document takes precedence over the definition of the same abbreviation, if any, in TR 21.905 [1].

ACELP	Algebraic Code-Excited Linear Prediction
AGC	Adaptive Gain Control
AllRAD	All-Round Ambisonics Decoding
BRIR	Binaural Room Impulse Response
DirAC	Directional Audio Coding
CBR	Constant Bit Rate
CNA	Comfort Noise Addition
CNG	Comfort Noise Generation
CPE	Channel Pair Element
DFT	Discrete Fourier Transform
DoA	Direction of Arrival
DTX	Discontinuous Transmission
EC	Entropy Coding
EFAP	Edge Fading Amplitude Panning
FEC	Frame Erasure Concealment
FOA	First-Order Ambisonics
GCC-PHAT	Generalized Cross-Correlation PHAse Transform
HOA	Higher-Order Ambisonics
HRIR	Head-Related Room Impulse Response
HRTF	Head-Related Transfer Function
IC-BWE	Inter-Channel BandWidth Extension
ILD	Inter-Channel Level Difference
ITD	Inter-Channel Time Delay
ISM	Independent Streams with Metadata
LFE	Low-Frequency Effects
MASA	Metadata-Assisted Spatial Audio
McMASA	Multi-Channel MASA

MC	Multi-Channel
MCT	Multi-channel Coding Tool
MDCT	Modified Discrete Cosine Transform
MDFT	Modified Discrete Fourier Transform
MDST	Modified Discrete Sine Transform
OMASA	Objects with MASA
OSBA	Objects with SBA
PCA	Principal Component Analysis
PLC	Packet Loss Concealment
SAD	Sound Activity Detection
SBA	Scene-Based Audio
SCE	Single Channel Element
SNS	Spectral Noise Shaping
SPAR	Spatial Reconstruction
TCX	Transform-Coded eXcitation
TD	Time-Domain
TNS	Temporal Noise Shaping
VAD	Voice Activity Detection
VBAP	Vector Base Amplitude Panning
VBR	Variable Bit Rate

4 General description of the coder

4.1 Introduction

The present document is a detailed algorithmic description of the Immersive Voice and Audio Services (IVAS) coder. The IVAS coder is a framework for low-delay speech- and audio coding and rendering targeting stereo or immersive audio communication. It comprises:

- encoder,
- decoder, and
- renderer.

The procedure of this document is mandatory for implementation in all network entities and User Equipment (UE)s supporting the IVAS coder.

The present document does not describe the C code of the IVAS coder. In the case of discrepancy between the algorithmic description in the present document and its C code specifications contained in [6], [11] the C code specifications prevail.

4.2 IVAS codec overview

4.2.1 General

The IVAS codec is an extension of the 3GPP Enhanced Voice Services (EVS) codec [2]. It provides full and bit exact EVS codec functionality for mono speech/audio signal input. It further provides:

- Encoding and decoding of stereo and immersive audio formats such as multi-channel audio, scene-based audio (Ambisonics), metadata-assisted spatial audio (MASA), object-based audio (ISM), and their combination.
- VAD/DTX/CNG for rate efficient stereo and immersive conversational voice transmissions
- Error concealment mechanisms to combat the effects of transmission errors and lost packets. Jitter buffer management is also provided.
- The IVAS codec operates on 20-ms audio frames. In addition, rendering is possible with 5-ms granularity.
- Support for bit rate switching upon command.
- Stereo and immersive audio coding at the following discrete bit rates [kbps]: 13.2, 16.4, 24.4, 32, 48, 64, 80, 128, 160, 192, 256, 384, and 512, with supported bit rate ranges listed in Table 4.2-1.

Table 4.2-1: Ranges of supported bitrates for stereo and immersive coding of the IVAS codec

Input audio format	Range of supported bitrates [kbps]
Stereo	13.2 – 256
Scene-based audio (SBA)	13.2 – 512
Metadata assisted spatial audio (MASA)	13.2 – 512
Object-based audio (ISM)(1)	13.2 – 512
Multi-channel audio (MC)	13.2 – 512
Combined ISM and MASA (OMASA)	13.2 – 512
Combined ISM and SBA (OSBA)	13.2 – 512

(1) 13.2 kbps – 128 kbps for 1 ISM, 16.4 kbps – 256 kbps for 2 ISMs, 24.4 kbps – 384 kbps for 3 ISMs, 24.4 kbps – 512 kbps for 4 ISMs

4.2.2 Mono (EVS-compatible) Operation

The IVAS codec supports mono operation with EVS compatibility by implementing all EVS functionality in a bit-exact manner. This mode is suitable for applications that require low bitrate and high-quality speech and audio. The codec provides the full range of EVS primary bitrates, from 7.2 kbps CBR / 5.9 kbps VBR to 128 kbps and can operate at narrowband, wideband, super-wideband and full-band sampling rates. EVS compatibility includes the AMR-WB IO modes, with bitrates from 6.6 kbps to 23.85 kbps. All functionalities already present in the EVS codec are retained, as also outlined in the subsequent Clauses. While staying bit-exact, some EVS code portions were reorganized to allow the extension towards immersive speech and audio services.

The EVS-compatible mono operation is interoperable to other implementations of the EVS codec, which enables seamless integration with existing systems and devices that use the EVS codec.

4.2.3 Stereo Operation

While the EVS codec only supports mono operation that could be combined for dual-mono operation, the IVAS codec supports native stereo coding at bitrates starting from 13.2 kbps to 256 kbps, offering high audio quality operation for stereo signals with all kinds of different stereo cues.

The stereo coding consists of coding modules operating in the time domain and frequency domain.

Low-rate stereo coding (13.2 to 32 kbps) mainly operates based on a combination of DFT-based processing for correlated signals and a time domain approach for uncorrelated signals. Which method is used to code the current frame is decided by a prior stereo classification stage.

For the DFT-based stereo approach, (band-wise) side parameters - including time, phase and loudness differences between the channels, as well as prediction parameters for the side residual - are extracted at the encoder followed by a downmix stage to obtain a mid-signal which is then given to the core coder. At the decoder, the stereo signal is reconstructed from the downmix signal and the transmitted stereo parameters. For the mid-bitrate of 32 kbps part of the residual side signal is additionally coded and directly transmitted in the bitstream to further improve quality.

For the time-domain stereo approach, time-domain parameters are extracted and a weighted downmix is created resulting in a primary and a secondary channel which are both given to the core coder but with the primary channel receiving a higher number of bits.

High-rate stereo coding (48-256 kbps) operates based on MDCT-based processing for band-wise stereo encoding. This can be seen as a sort of joint core coding as both stereo and discrete channel operations are done in the same domain without intermediate transforms. For stereo, additional loudness differences are transmitted and each stereo band can be transformed adaptively to a mid/side representation, depending on what is most efficient to code.

In addition, stereo downmix is supported to generate a mono signal for EVS interoperable stream with no extra delay.

4.2.4 Objects (Independent Streams with Metadata) Operation

The IVAS codec supports coding of 1 to 4 independent audio objects with associated metadata. The coding is based on input Independent Streams (ISMs) analysis, metadata coding, and inter-object core-coder SCEs bitrate adaptation. The

ISM coding employs two coding schemes, namely the “discrete ISM” mode in which each object is coded by one SCE, and parametric ISM (“ParamISM”) mode which downmixes 3 or 4 objects to two transport channels coded by two SCEs. The coded and transmitted metadata consists of azimuth and elevation (up to 48 kbps) or of azimuth, elevation, radius, pitch, and yaw (64 kbps and up). Alternatively, the metadata can consist of a non-diegetic panning gain. On the decoder side, the objects can then be rendered to the requested output configuration.

4.2.5 Multi-Channel (MC) Operation

Coding of multi-channel inputs is available for the channel layouts 5.1, 7.1, 5.1+2, 5.1+4, and 7.1+4. The coding technique is selected from a set of coding modes based on the available bitrate and specified channel layout. The general principle in technique selection is to aim for best possible quality given the allowed bitrate. For all techniques, LFE channel coding is also offered either separately or within the technique. The multi-channel operation supports output to mono, stereo, multi-channel (at the same or any other layout with up to 16 speakers), Ambisonics (at up to order 3), and binaural.

4.2.6 Scene-based Audio (Ambisonics) Operation

Coding of ambisonics signals is supported for 1st- to 3rd-order inputs throughout the full bitrate range. The decoder's output can be SBA (of order 1, 2, or 3), mono, stereo, binaural or multi-channel. This flexibility of input order, bitrate and output format combinations is in part achieved by the combination of covariance-based and directional analysis at different frequencies.

For the lowest 8 bands, covariance analysis with a 20 ms stride is performed at the encoder and corresponding reconstruction is performed at the decoder. For the highest 4 bands, an estimation of the parameters of a psychoacoustic model is implemented with a time resolution of 5 ms.

At the encoder, the covariance-analysis metadata for the higher bands are estimated from these model parameters and combined with the directly calculated metadata for the lower bands. Based on these metadata, a downmix to 1 to 4 channels (dependent on bitrate) is obtained. The downmix channels are then coded with the appropriate core coder.

At the decoder, the downmix channels plus the metadata are received. The latter comprise the transmitted covariance-analysis metadata and model parameters for the lower and higher bands, respectively. These metadata are used to reconstruct the HOA signal and render to the requested output format. In this, the psychoacoustic model parameters for the 8 lower frequency bands are estimated from the reconstructed audio channels. The model-based reconstruction allows for the output SBA order on the decoder side to be higher than the input order on the encoder side.

Low-latency operation (less than or equal to 38 ms end-to-end) is achieved by using a very-low-latency 1-ms MDFT-based filterbank at the encoder and a 5-ms CLDFB-based filterbank at the decoder, which additionally enables the type of signal modifications that are necessary for rendering to a broad variety of output configuration.

4.2.7 Metadata-assisted Spatial Audio (MASA) Operation

The IVAS codec supports coding of parametric spatial audio format called metadata-assisted spatial audio (MASA). This format is specifically optimized for the direct immersive audio capture from smartphones and other form factors that can be unsuitable for dedicated spherical microphone arrays.

The MASA format is based on 1-2 audio channels and associated metadata that is provided for each audio frame. The MASA spatial metadata describes the spatial audio characteristics of the captured immersive audio using several spatial parameters including spatial direction information, directional and non-directional energy ratios, and two types of coherence information. The spatial metadata is provided in each frame according to a time-frequency resolution of 4 subframes and 24 frequency bands. The MASA descriptive metadata provides additional information relating to the creation and understanding of the MASA audio signal.

The coding in this operation is based on compression of the metadata exploiting detected redundancies and prioritization of selected parameters at each bitrate. The 1 or 2 audio transport channels are coded using the SCE and CPE coding capabilities. A joint bitrate allocation between these two coding blocks is based on a metadata analysis and simplification processing.

The MASA format can be flexibly rendered for binaural or loudspeaker reproduction, including mono and stereo playback. Rendering to Ambisonics is also supported. Furthermore, decoded MASA format bitstream can be directly output from the decoder without rendering as a fully compliant MASA format output for further processing.

4.2.8 Combined Objects and MASA (OMASA) Operation

The IVAS codec supports combined input format of ISMs and MASA called OMASA (Objects with Metadata-Assisted Spatial Audio). The input consists of 1 – 4 independent objects with associated metadata and the 1 or 2 transport channels of MASA and the associated metadata. The encoding configuration for OMASA is decided based on the codec bitrate and the number of objects. There are 4 encoding configurations designed such that the encoded output quality is optimal. Based on the decided configuration a combination of one audio channel pair with MASA metadata is encoded as MASA format data and none, one or all objects are encoded within ISM format encoding framework. When one or more objects are encoded, the adaptive inter-format bitrate allocation between the object(s) and the MASA format data is used at each frame based on the audio content. On the decoder side, the audio can then be rendered to the requested output configuration.

4.2.9 Combined Objects and SBA (OSBA) Operation

The IVAS codec supports combined input format of ISMs and SBA called OSBA (Objects with Scene Based Audio). There are two different operation modes for this type of input: a low- (below 256 kbps) and a high- (256 kbps and up) bitrate one. In the low-bitrate mode the objects are pre-rendered into the SBA input, which is then processed in exactly the same way as in the native SBA format. The high-bitrate mode features separate coding of 1 – 4 independent objects with associated metadata. Specifically, the object metadata are encoded in the same way as in the native ISM format and written into the bitstream alongside the SBA metadata. The object audio channels are input to the MCT coding tool together with the downmix channels of the SBA coder. Therefore, efficient bit allocation techniques of MCT are applied to all coded audio channels together. On the decoder side, the audio can then be rendered to the requested output configuration.

4.2.10 Discontinuous Transmission (DTX) Operation

DTX is a functionality of operation where the encoder encodes speech frames containing only background noise with a lower bit rate and lower packet frequency than normally used for encoding speech. A terminal and the network may adapt their transmission scheme to take advantage of the smaller frames and higher frame interval to reduce power consumption, average bit rate and network activity. The discontinuous transmission (DTX) functionality of the IVAS codec includes voice activity detection (VAD) and comfort noise generation (CNG). DTX functionality is supported for IVAS operation points, i.e., audio formats and bitrates, that are especially optimized for efficient stereo and immersive conversational voice transmissions.

The size of the SID frames for EVS interoperable modes is unchanged relative to EVS, 48 bits for EVS primary modes. For IVAS modes, the SID frame size is 104 bits. The default SID frame interval is once per 8 frames, but other update intervals are also supported.

4.3 Input/output audio configurations

4.3.1 Input/output audio sampling rate

The IVAS coder is capable of processing input audio signals sampled at 16, 32 and 48 kHz. The sampling frequency is denoted as F_s. The same set of sample rates is also supported by the IVAS decoder. In case of EVS-compatible mono operation the IVAS coder supports also 8 kHz sample rate both for input and output signals. The input audio is processed in frames of 20 ms.

4.3.2 Input/output audio formats

The IVAS coder accepts the following input audio formats:

- single-channel mono audio format denoted as $s(n)$
- two-channel stereo and binaural input audio format, where the left channel is denoted as $s_1(n)$ and the right channel is denoted as $s_2(n)$
- scene-based (ambisonic) input audio format, where the ambisonic order is denoted as l and the number of individual input channels is $(l+1)^2$. The individual input channels are stored in the ACN component ordering, denoted as $s_{ml}(n)$, where $m \in \langle -l, \dots, +l \rangle$ is the ambisonic degree. In case of the first-order ambisonic format (FOA), the individual input channels may also be denoted as:

$$s_{00}(n) = W(n)$$
$$s_{-11}(n) = X(n)$$
$$s_{01}(n) = Y(n)$$
$$s_{11}(n) = Z(n)$$

For both, first-order ambisonic signals and higher-order ambisonic signals (HOA), the notation $s_{ml}(n)$ may be simplified to:

$$s_k(n) = s_{ml}(n), \qquad k = l^2 + l + m$$

- object-based input audio format (ISM), where the number of objects is denoted as N_{obj} and the individual "streams" related to the objects are denoted as $s_k(n)$ where $k \in \langle 1, \dots, N_{obj} \rangle$. Each individual stream is associated with its input metadata signal, denoted as $I_k(m)$ where m is the frame index.

- multi-channel input audio format (MC), where the individual channels are denoted as $s_k(n)$ where $k \in \langle 1, \dots, N_{chan} \rangle$. The input channels correspond to one of the following loudspeaker layouts as per the CICP notation [XX]. The ordering of input channels is defined in the ITU specification [XX]:

loudspeaker layout	CICP layout
2.0	CICP2
5.1	CICP6
5.1+2	CICP14, 35°elevation
5.1+4	CICP16, 35°elevation
7.1	CICP12
7.1+4	CICP19, 35°elevation

- metadata-assisted spatial audio (MASA) format, where, for mono-MASA (MASA1), the mono channel is denoted as $s(n)$ and, for stereo-MASA (MASA2), the left channel is denoted as $s_1(n)$ and the right channel is denoted as $s_2(n)$. The MASA audio channel(s) is/are associated with the MASA metadata that is denoted as $I(m)$ where m is the frame index
- objects with metadata-assisted spatial audio (OMASA), which is a combination of MASA and object-based input audio format (ISM)
- objects with scene-based (ambisonic) input audio format (OSBA), which is a combination of SBA and object-based input audio format (ISM)

4.4 Algorithmic delay

The input signals (audio, or audio and metadata) are processed using 20-ms frames. The codec algorithmic delay depends on the input/output audio formats as described in Table 4.4-1.

Table 4.4-1: IVAS algorithmic delay for different input/output format combinations (rounded to integer milliseconds; in case multiple values are provided they depend on the bitrate)

		Decoder output format						
		Mono	Stereo	Multi-Channel	Binaural audio	Scene-based audio	Object-based audio	Metadata-assisted spatial audio
Encoder input format	Mono	32						
	Stereo	32	32	32				
	Binaural Audio	32			32			
	Multi-channel	32	32	32 / 37	32 / 37 / 38	32 / 37		
	Scene-based audio	33	33	38	38	38		
	Object-based audio	32	32	32 / 37	32 / 37	32 / 37	32 / 37	
	Metadata-assisted spatial audio	32 / 37	37	37	37	37		32 (NOTE
	OSBA	33	33	38	38	38		
	OMASA	32 / 37	37	37	37	37		

NOTE: Metadata-assisted spatial audio (MASA) decoder output allows also for mono or stereo decoder output at 32ms algorithmic delay by stripping the metadata file.

The algorithmic delay related to the core-coder coding in IVAS is 32 ms similarly as in EVS though its splitting between the encoder and the decoder is slightly different. It consists of 8.75 ms for the encoder look-ahead and 3.25 ms for the decoder delay related to the time-domain BWE and resampling in the DFT domain.

Further, the IVAS delay consists of 5 ms delay related to the rendering to the related output configuration, thus making the overall delay of 32 ms in some set-ups and 37 ms in other set-ups.

Finally, in SBA format and Parametric multichannel upmix coding mode in MC format, an additional encoder delay of 1 ms is present and it is related to the filter-bank analyses prior to the encoding.

It is also noted that the delay figures exclude any HRIR/BRIR induced delay.

The codec delay for mono (EVS) operation and stereo downmix operation in EVS compatible operation is 32 ms described in Clause 4.3 in [3].

4.5 Rendering overview

4.5.1 Rendering introduction

The codec for Immersive Voice and Audio Services is part of a framework comprising of an encoder, decoder, and renderer. An overview of the audio processing functions of the receive side of the codec is shown in 4.5-1. This diagram is based on [5], with rendering features highlighted.

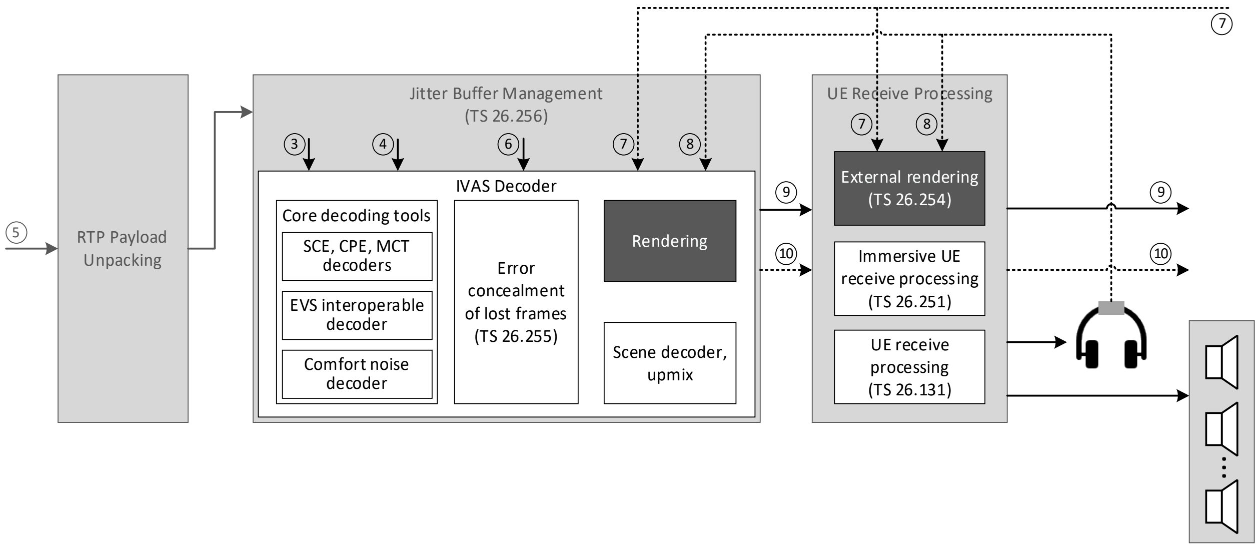


Figure 4.5-1: Overview of IVAS audio processing functions – receiver side

The interfaces are marked consistently with [5] using the following numbers:

3: Encoded audio frames (50 frames/s), number of bits depending on IVAS codec mode,
4: Encoded Silence Insertion Descriptor (SID) frames,
5: RTP Payload packets,
6: Lost Frame Indicator (BFI),
7: Renderer config data,
8: Head-tracker pose information and scene orientation control data,
9: Audio output channels (16-bit linear PCM, sampled at 8 (only EVS), 16, 32, or 48 kHz),
10: Metadata associated with output audio.

Rendering is the process of generating digital audio output from the decoded digital audio signal. Rendering is used when output format is different than input format. In case output format is the same as input format, the decoded audio channels are simply passed through to the output channels. Binaural rendering is a special case, where binaural output channels are prepared for headphone reproduction. This process includes head-tracking and scene orientation control, head-related transfer function processing, and room acoustic synthesis. IVAS rendering is integrated with IVAS decoder but can also be operated standalone as external rendering while bypassing the internal renderer. The external renderer can be applied e.g., in the case of rendering outputs originating from multiple sources, such as decoders or audio streams.

4.5.2 Internal IVAS renderer

The internal IVAS renderer is integrated into the IVAS decoder. In case of specific operating points, this integration allows for combining decoding and rendering processes, resulting in efficient processing. Therefore, rendering algorithm descriptions associated with specific audio formats are discussed in clause 6. The rendering modes and rendering control are discussed separately in clause 7.

4.5.3 External IVAS renderer

The external IVAS renderer supports all the functionality of the internal renderer. However, since the external renderer operates stand-alone, combined decoding and rendering processing is not available.

4.5.4 Interface for external rendering

IVAS renderer and its interface provide support to IVAS codec design constraints. The details of the rendering library API are provided in [6] for the fixed-point code and [11] for the floating-point code. The details of the rendering library API are provided in [8].

4.6 Organization of the rest of the Technical Standard

The detailed description is organized as follows:

- In Clause 5, the detailed functional description of the encoder is given. This begins with a description of the common processing and coding tools, followed by each of the coding operations based on the input audio formats.
- In Clause 6, the detailed functional description of the decoding is given. This description begins with common decoder processing, followed by each of the decoding operations for processing bitstream. The decoding processing also describes the rendering specific for the audio input format coding modes and the audio output format.
- In Clause 7, the detailed functional description of the rendering operations is given. This includes the rendering modes and rendering control functionalities.
- Bit allocation is summarized in Clause 8.

Annex A provides the RTP payload format and SDP parameter definitions for the IVAS codec.

5 Functional description of the encoder

5.1 Encoder overview

The IVAS codec encoder expects mono, stereo, objects, multichannel, ambisonics, MASA, combination of objects and MASA, or combination of objects and SBA as input audio channels. In case of objects or MASA, also input metadata are expected. The encoder analyzes the scene, derives the spatial audio parameters and downmixes the input channels to the so-called transport channels which are subsequently processed by the encoding tools. These tools comprise Single Channel Elements (SCE comprising one core-coder, see Clause 5.2.3.1), Channel Pair Elements (CPE comprising two core-coders, see Clause 5.2.3.2), and Multichannel Coding Tool (MCT comprising a joint coding of multiple core-coders, see Clause 5.2.3.3) while the core-coder is inherited from the EVS codec with additional flexibility and variable bitrate (Clause 5.2.2).

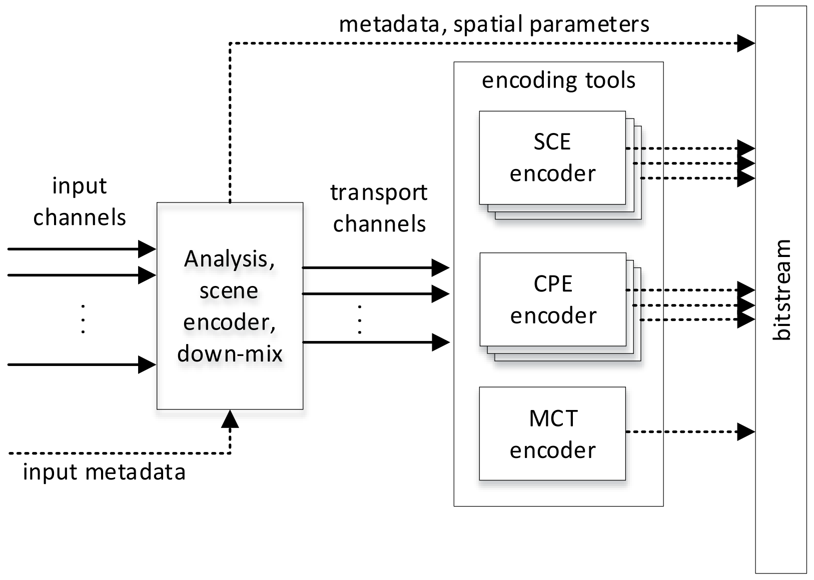


Figure 5.1-1: Encoder Data Flow from input data to IVAS bitstream.

5.2 Common processing and coding tools

5.2.1 Common processing overview

5.2.1.1 High-pass filtering

The input signal $s_{in}(n)$ sampled at the input sampling frequency F_s is high-pass filtered to supress undesired low-frequency components. The transfer function of the HP filter has a cut-off frequency of 20 Hz (–3 dB) and is given by

$$H_{20Hz} = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}}. \qquad (5.2\text{-}1)$$

where a_i and b_i are the coefficients of the HP filter. The coefficients of the HP filter are constant and defined in Table 1 of Reference [3].

In case of multi-channel input signal the HP filter is not applied on the LFE channel (index 3). In case of ambisonic input format the HP filter is only applied on analyzed channels (see clause XX).

The input signal, filtered with the HP filter, is denoted as $s_{HP}(n), n = 0, \dots, N-1$. In case of skipping the HP filtering operation $s_{HP}(n) = s_{in}(n)$ in channel n.

5.2.2 Core-coder processing

The core-coder in the IVAS codec is based on the EVS core-codec [3] with added flexibility and adaptation for multichannel processing. In the remaining part of Clause 5.2.2, only the differences of the IVAS core-coder wrt. the EVS core-coder [3] are provided.

The core-coder module bitrate (one per one core-coder) is not directly signalled in the bitstream but it is derived from the bit-budgets of all codec modules in the following steps:

First, an IVAS format signalling is determined and written to the bitstream.

Second, stereo or spatial coding tools processing (if present) is done and their respective bit-budgets computed.

Similarly, a metadata coding module (if present) is processed and its bit-budget is computed.

The bit-budgets for IVAS format signalling, the stereo / spatial coding tools and metadata coding module are then subtracted from the IVAS total bit-budget.

Finally, the remaining bit-budget is distributed among core-coders based on the type of the core. The details of the LP-based core-coding are provided in Clause 5.2.2.2.2 while the MDCT-based core coder details are provided in Clause 5.2.2.2.3.

5.2.2.1 Core-coder front pre-processing

5.2.2.1.1 Sample rate conversion

5.2.2.1.2 Preemphasis

5.2.2.1.3 Spectral analysis

5.2.2.1.4 Signal activity detection

When operating in EVS mode, the signal activity detection operates as in [EVS CNG ref]. When the IVAS core encoding is active, the CLDFB domain analysis is not present. Hence, the signal activity detection related to the CLDFB domain parameters is disabled [Ref to EVS CLDFB VAD].

5.2.2.1.5 Bandwidth detector

The Audio Bandwidth Detection (BWD) algorithm in IVAS is similar to the BWD algorithm in EVS (see Clause 5.1.6 in [3]) and it is applied in its EVS-based form in the ISM format, DFT stereo and TD stereo modes. However, in the MDCT stereo mode and MCT coding tool (including higher-bitrate SBA, higher-bitrate MASA, multi-channel format, or combined formats), updates to the BWD were introduced in order the BWD is present at all operating points in IVAS. The updates consist of a BWD that comprises an analysis of the sound signal module and a final bandwidth (BW) decision module which is located upstream of the sound analysis module while the analysis module is integrated to the core-coder stage and the final BW decision module is integrated to the front pre-processing stage.

The BWD algorithm comprises several operations: a) computation of mean and maximum energy values in a number of spectral regions of the input signal; 2) updating long-term parameters and counters; and 3) final decision about the detected and thus coded audio bandwidth while operating in a transform domain. The transform domain is the CLDFB domain in case of EVS or DCT domain in case of AMR-WB IO mode within IVAS (see Clause 5.1.6 in [3]).

In IVAS, the BWD is run on all transport channels (except of LFE channel in the MC format) and the transform domain is the CLDFB domain in case of the SCE coding tool and DFT stereo and TD stereo within the CPE coding tool similarly as in EVS. However, in case of MDCT stereo within the CPE tool or the MCT tool, it is the MDCT spectrum which forms the BWD energy vector.

In the case of the MDCT energy vector, there are nine frequency bands of interest whereby the width of each band is 1500 Hz. One to four frequency bands are assigned to each of the spectral regions as defined in Table 5.2-4 where i is the index of the frequency band and k_{start} is the energy band start index and k_{stop} is the energy band end index (see Clause 5.1.6.1 in [3]).

Table 5.2-1: MDCT bands for energy calculation in BWD

i	k_{start}	k_{stop}	band-width in kHz	spectral region
0	1	1	1.5 – 3.0	*nb*
1	3	3	4.5 – 7.5	*wb*
2	4	4		
3	6	6	9.0 – 15.0	*swb*
4	7	7		
5	8	8		
6	9	9		
7	11	11	16.5 – 19.5	*fb*
8	12	12		

The BWD analysis is then adjusted from the EVS native BWD algorithm such that the MDCT spectrum is scaled proportionally to the input sampling rate. The energy $E_{bin}(i)$ (see Clause 5.1.6.1 in [3]) of the MDCT spectrum of each transport channel signal in the MDCT stereo mode is thus computed in the nine frequency bands as follows:

$$E_{bin}(i) = \sum_{k=k_{start}(i)\cdot b}^{k=k_{stop}(i)\cdot b+b} S^2(k)\,, i = 0, \dots ,8, \qquad (5.2\text{-}2)$$

where $S(k)$ is the MDCT spectrum and the width of the energy band is $b = 60$ samples (which corresponds to 1500 Hz regardless of the sampling rate). The rest of the BWD algorithm is the same as in EVS and consists of computing long-term values of the mean energy of spectral regions, updating spectral region counters, and comparing them to given thresholds (see Clause 5.1.6.1 in [3][2] for more details).

There are however three other particularities in IVAS: 1) the BWD analysis in the MDCT domain is not done at the front pre-processing stage but only at the beginning of the TCX core coding where the MDCT spectrum is available from the TCX core processing. 2) Consequently, the final BWD decision operation is postponed to the front pre-processing of the next frame. Thus, the former EVS BWD algorithm is split into two parts; the BWD analysis operation (i.e. computing energy values per frequency band and updating long-term counters) is done at the beginning of current TCX core coding and the final BWD decision operation is done only in the next frame before the TCX core encoding starts. 3) The BWD analysis is skipped in switching frames (tcx_last_overlap_mode == TRANSITION_OVERLAP) having a longer duration.

Next, in MDCT stereo coding, the final BWD decision from the decision module about the input and thus coded audio bandwidth is done not separately for each of the two channels (or multiple channels in case of MCT) but as a joint decision for both channels. In other words, in the MDCT stereo coding, both channels are always coded using the same audio bandwidth and the information about the coded audio bandwidth is transmitted only once per one CPE. If the final BWD decision is different between the two CPE channels, both CPE channels are coded using the broader audio bandwidth BW of the two channels. E.g. in case that the detected audio bandwidth BW is the WB bandwidth for the first channel and the SWB bandwidth for the second channel, the coded audio bandwidth BW of the first channel is rewritten to SWB bandwidth and the SWB bandwidth information is transmitted in the bitstream. The only exception is a case when one of the MDCT stereo channels corresponds to the LFE channel, then the coded audio bandwidth of the other channel is set to the audio bandwidth of this channel. This is applied mostly in the MC format mode when multiple MC channels are coded using several MDCT stereo CPEs. The information about the coded audio bandwidth is then sent in the bitstream as one joint parameter for the multichannel coding.

Finally, the BWD uses a hysteresis for switching between coded bandwidths. In general, a shorter hysteresis of 10 frames is used when switching from a lower bandwidth to a higher one while a longer hysteresis of 100 frames is used when switching from a higher bandwidth to a lower one. In IVAS, the lower hysteresis is further adjusted in the following set-ups: a) MCT is present, b) the MDCT stereo with element bitrate equal or higher than 48 kbps, c) ISM format with the element bitrate equal or higher than 32 kbps. In these set-ups the shorter hysteresis is set to zero frames.

5.2.2.1.6 Time domain transient detection

5.2.2.1.6.1 Low-rate adaptation of transient detection

Low-rate adaption of transient detection is based on forward and reverse time direction analysis to detect a transient attack and transient release in the input signal. First the subframe energies, where each subframe is 2.5 ms long, are computed on a high pass filtered signal of the input as described in clause 5.1.8 (see equation 36) of Reference [3]. For each subframe a low pass filtered max energy envelope or accumulated energy is computed according to equation 37 of Reference [3].

For the forward analysis a transient attack is detected if the energy of the subframe, $E_{TD}(i)$, where $i = -2, \ldots, 6$ is above the low pass filtered max energy envelope by factor of ϑ_{fwd}. ϑ_{fwd} is set to 8.5 if condition c_1 below is satisfied otherwise it is set to 8.0.

$$m_{sum} > th_{tonal} * 0.8 \qquad (5.2\text{-}3)$$

Where m_{sum} and th_{tonal} are defined in clause 5.1.11.2.5 of Reference [3].

For transient release analysis, transient detection is performed in the reverse time direction. A low pass filtered energy envelope in the time reversed direction is computed as follows:

$$E_{accRev} := \max\left(E_{TD}(i+1), 0.8125 E_{accRev},\ i = 3, .., -4\right. \qquad (5.2\text{-}4)$$

An additional high-resolution analysis within the 5th subframe of the preceding frame, denoted as $E(-4)$, is performed to set a $ramp\_up\_flag$ indicating higher energy in the second half of the subframe compared to the first half. The $ramp\_up\_flag$ is set when the condition below is satisfied.

$$\sum_{j=\frac{k}{2}}^{j=k-1} sb_{-4}(j)^2 > \sum_{j=0}^{\frac{k}{2}-1} sb_{-4}(j)^2 \qquad (5.2\text{-}5)$$

Where the sb_{-4} is the time domain samples in subframe -4 and k is the total number samples in each subframe.

To detect a release, which can be viewed as a transient in the time reversed analysis. Transient release thresholds, ϑ_{rev_low} and ϑ_{rev_high}, are set based on condition c_2 below being satisfied.

$$m_{sum} > th_{tonal} * 0.6 \qquad (5.2\text{-}6)$$

ϑ_{rev_low} and ϑ_{rev_high} are set to 4.5 and 5.5 respectively if condition c_2 above is satisfied, otherwise they are set to 4.25 and 5.25.

A transient release in subframe -3 is detected if energy in $E_{TD}(-3)$ is above E_{accRev}(-3) by a factor of ϑ_{rev_low}.

A transient release in subframe $E_{TD}(-4)$ is detected if energy in $E_{TD}(-4)$ is above E_{accRev}(-4) by a factor of ϑ_{rev_low} and the $ramp\_up\_flag$ being set or if energy in $E_{TD}(-4)$ is above $E_{accRev}(-4)$ by a factor of ϑ_{rev_high}.

5.2.2.1.7 Linear prediction analysis

5.2.2.1.8 Open-loop pitch analysis

5.2.2.1.9 Speech/Music classifier

5.2.2.1.9.1 GMM-based speech/music classifier

5.2.2.1.9.2 SNR-based speech/music classifier

5.2.2.1.9.3 Correction for non-harmonic transient signals

For low bitrates a determination is made to identify if the input signal contains a problematic transient (attack or release) based on forward and reverse time direction signal analysis to adjust the encoding scheme selection through frame classification. The problematic transient cause annoying artifacts if encoded in FD domain and a correction of the frame classification is needed. Typically frames classified as speech are encoded with a TD domain encoding scheme and music classified frames are encoded with a FD domain encoding scheme. A transient attack or a transient release is detected in the input signal, the location in the current and previous frame is used to adaptively adjust a frame classification. Additionally, an analysis of the harmonicity of the input signal is performed to adjust the frame classification. Finally, three conditions, $(\mathrm{c}_1,\ \mathrm{c}_2,\ \mathrm{c}_3)$, are evaluated to get a decision on whether to override the frame classification as speech.

The first condition, c_1, is whether a transient is detected in the current frame, F_N, excluding the last subframe. The second condition, c_2, to be checked especially when the last frame was a TD frame is whether a transient is detected in the last half of the previous frame. At low rates, both conditions are checked by performing a refined transient analysis in both the forward and reversed time direction as described in clause 5.1.2.6.1. The third condition, c_3, is whether the signal is harmonic which is determined by a harmonicity flag set according to equation 136 in Reference [3]. The decision to change the frame classification to speech is determined by the flag below.

$$forceTD = (\mathrm{c}_1 \mid \mathrm{c}_2)\ \&\ !\ \mathrm{c}_3 \qquad (5.2\text{-}7)$$

The third condition c_3 not being fulfilled (false), indicates the signal is not harmonic. When the flag $forceTD$ is set, the frame is classified as speech, otherwise the frame classification is left unchanged to that determined by the GMM-based speech/music classifier. The first condition, c_1, addresses both forward and backward spreading (and smearing) caused by the scarcity of bits in the low-rate FD TCX20 compression scheme, where TCX20 is a regular MDCT frame type producing 20 ms of synthesized output signal. The second condition, c_2, addresses smearing caused by the suboptimal transition window (TCX25) used when switching from TD to FD coding. If there is a strong transient in the end of the TD coded frame, part of its energy might be included in the beginning of the FD coded frame, which then causes smearing. The third condition, c_3, is restricting the switch to TD for signals with high harmonicity when the low-rate TD coding mode is likely not performing as well as the FD coding.

First, condition c_3 is evaluated by checking harmonicity of the signal based on the long-term correlation map, m_{sum}, and an adaptive threshold, th_{tonal}, according to equation 136 in Reference [3]. If the harmonicity flag is set, meaning there is a high degree of harmonicity in the signal, the classification is not changed with respect to potential transients and further analysis is not carried out. However, if the harmonicity flag is not set, a transient analysis is performed to determine whether to classify the frame as speech.

The subframes analysed for transients are the ones that fall within the transform window that would be encoded if an FD encoding scheme would be used. This corresponds to subframes {-4, 6}. For subframes {-2, 6} a refined transient detector described in clause 5.1.2.1.6.1 is run in the forward direction. If a transient is detected, which corresponds to $\mathrm{c}_1$in equation (5.2-7) being fulfilled, the $forceTD$ flag is set.

For subframes {-3} and {-4} an enhanced transient detector is ran in the reverse time direction as described in clause 5.1.2.1.6.1 to detect a potentially harmful transient release whose energy might spread too much into the current frame to be encoded.

For the reverse analysis of subframe {-3} it is checked whether the transient release energy is above ϑ_{rev_low}, and if that is the case, $forceTD$ is set, adjusting the coding scheme to be TD as using an FD encoding scheme might lead to smearing. For the reverse analysis of subframe {-4} it is checked whether a transient release is detected with both thresholds ϑ_{rev_low} and ϑ_{rev_high}. If a transient release is detected with threshold ϑ_{rev_high}, the $forceTD$ flag is set. If a transient release is detected with only threshold ϑ_{rev_low}, it is additionally checked whether the energy of the second half is greater than the first half of subframe -4, and if that is the case, the $forceTD$ flag is set. The reason for the additional high resolution time domain analysis within subframe {-4} is that: only a part of the 1st half of the subframe actually falls within the TCX25 transform window and that the signal is weighted by the TCX25 window, so if most energy is located in the 2nd part of subframe {-4} then we might get smearing even though we are lower than limit ϑ_{rev_high}. In all other cases $forceTD$ flag is not set.

5.2.2.1.9.4 Onsets detection

5.2.2.1.9.5 HQ classifier

For SWB bitrates at 24.4 and 32, there are 4 modes supported, Transient, Harmonic, HVQ and Generic as described in 5.3.4.2 [3]. A frame is considered harmonic following the classification described in 5.3.4.2.3 [3]. A HQ classifier is used to switch to Generic mode for harmonic frames if a sparse harmonic structure is not detected in the spectrum, otherwise the classification follows that described in 5.3.4.2.3 [3]. A spectrum sparseness analysis is performed based on obtaining a peakyness measure and a noise band detection measure derived from the MDCT coefficients.

First, the magnitude A_i of a critical frequency region is obtained by

$$A_i = |X(k_{start} + i)|, i = 0, \dots, k_{end} \tag{5.2-8}$$

Where $X(k)$ is the MDCTs spectrum computed, k_{start} is the first bin the critical frequency region and k_{end} is the last bin in the critical frequency region. k_{start} and k_{end} are set to 320 and 639 respectively where the input sampling rate is 32 kHz. The critical frequency region is the upper half of the MDCT spectrum.

The peakyness measure is obtained by obtaining the crest in accordance with equation (5.2-9) below

$$crest = \frac{\max(A_i)}{\sqrt{\frac{1}{M}\sum_{i=0}^{M-1} A_i^2}} \tag{5.2-9}$$

Where $M = k_{end} - k_{start} + 1$ is the number of bins in the critical band. A complimentary peakyness measure pk_{low} is obtained in accordance with equation (5.2-10)

$$pk_{low} = \sum_{i=0}^{M-1} low(A_i) \tag{5.2-10}$$

Where

$$low(A_i) = \begin{cases} 1, A_i < 0.1\max(A_i) \\ 0, A_i \geq 0.1\max(A_i) \end{cases} \tag{5.2-11}$$

A noise band detection measure is obtained according to equation.

$$crest_{mod} = \frac{\max\left(movmean(A_i,W)\right)}{\sqrt{\frac{1}{M}\sum_{i=0}^{M-1} A_i^2}} \tag{5.2-12}$$

Where $movmean(A_i, W)$ is a moving mean of the absolute spectrum A_i using a window size W fixed to 21. The moving mean is computed according to equation (5.2-13) below.

$$movmean(A_i, W) = \frac{1}{b-a+1}\sum_{i=a}^{b} A_i(m)\,,\ where \begin{cases} a = \max(0, i - (W-1)/2 \\ b = \min(M-1, i + (W-1)/2 \end{cases} \tag{5.2-13}$$

$crest_{mod}$ gives a measure of local concentration of energy, indicating a noise band in the spectrum. To stabilize the decision, $crest$ and $crest_{mod}$ are low pass filtered according to equation.

$$crest_{LP} = (1-\alpha)\cdot crest + \alpha \cdot crest_{LP}^{-1} \tag{5.2-14}$$

$$crest_{mod,LP} = (1-\alpha)\cdot crest_{mod} + \alpha \cdot crest_{mod,LP}^{-1} \quad (5.2\text{-}15)$$

Where α is set to 0.97.

The spectrum sparseness analysis obtains a harmonic decision in accordance with equation (5.2-16)

$$Harmonic\_decision(m) = \begin{cases} FALSE, & crest_{LP} > crest_{thr}, crest_{mod,LP} > crest_{mod,thr}, pk_{low} > pk_{low-thr} \\ TRUE, & otherwise \end{cases} \quad (5.2\text{-}16)$$

Table 5.2-2: Thresholds for HQ classifier

threshold	value
$crest_{thr}$	7.0
$crest_{mod,thr}$	2.128
$pk_{low-thr}$	220

The Generic mode is selected if the condition below is fulfilled, otherwise the mode decision is left unchanged and follows that described in 5.3.4.2.3 [3].

$$mode == \text{Harmonic} \; \&\& \; !Harmonic\_decision \quad (5.2\text{-}17)$$

5.2.2.2 Core-coder modules

5.2.2.2.1 Core-coder pre-processing

5.2.2.2.1.1 Selection of internal sampling rate

5.2.2.2.1.2 Coder technology selection

5.2.2.2.2 LP based coding

5.2.2.2.2.1 Variable bitrate ACELP coding

The ACELP core coder in EVS [3] is mainly based on a constant bitrate principle where a bit-budget to encode a given frame is constant during the encoding. In order to obtain the best possible quality at a given constant bitrate, the bit-budget is carefully distributed among the different coding parts. Specifically, the bit-budget per coding part at a given bitrate is fixed and stored in codec ROM tables.

On the other hand, IVAS is a complex multi-module codec where the bit-budget at a constant bitrate is allocated between different modules based on, for example, a number of input audio channels, audio bandwidth, spatial audio characteristics of input signal, presence of metadata, available codec total bitrate, etc. Then in IVAS, the codec total bit-budget is distributed among the core module tools (see Clause 5.2.3) and other different modules like a bandwidth extension (BWE), stereo or spatial parameters module(s), metadata module(s), etc. which are collectively referred to in the further text as "supplementary codec modules". In order to get the most efficiency and flexibility in IVAS, the allocated bit-budget per supplementary module is variable by default making a demand for the core-codec to support a variable bitrate coding as well. Consequently, the bit-budget allocated to the ACELP core module can fluctuate between a relatively large minimum and maximum bitrate span with a granularity of 1 bit (i.e. 0.05 kbps at a frame length of 20 ms).

Dedicated ROM table entries for all possible ACELP core module bitrates as done in EVS are obviously inefficient in IVAS with variable bitrate coding. Thus, a more efficient and flexible distribution of the bit-budget among the different modules with fine bitrate granularity based on a limited number of intermediate bitrates is employed in IVAS and described in the next Clauses.

5.2.2.2.2.1.1 Bit-budget allocator

Figure 5.6-1 is a block diagram illustrating the bit-budget allocator in the IVAS codec.

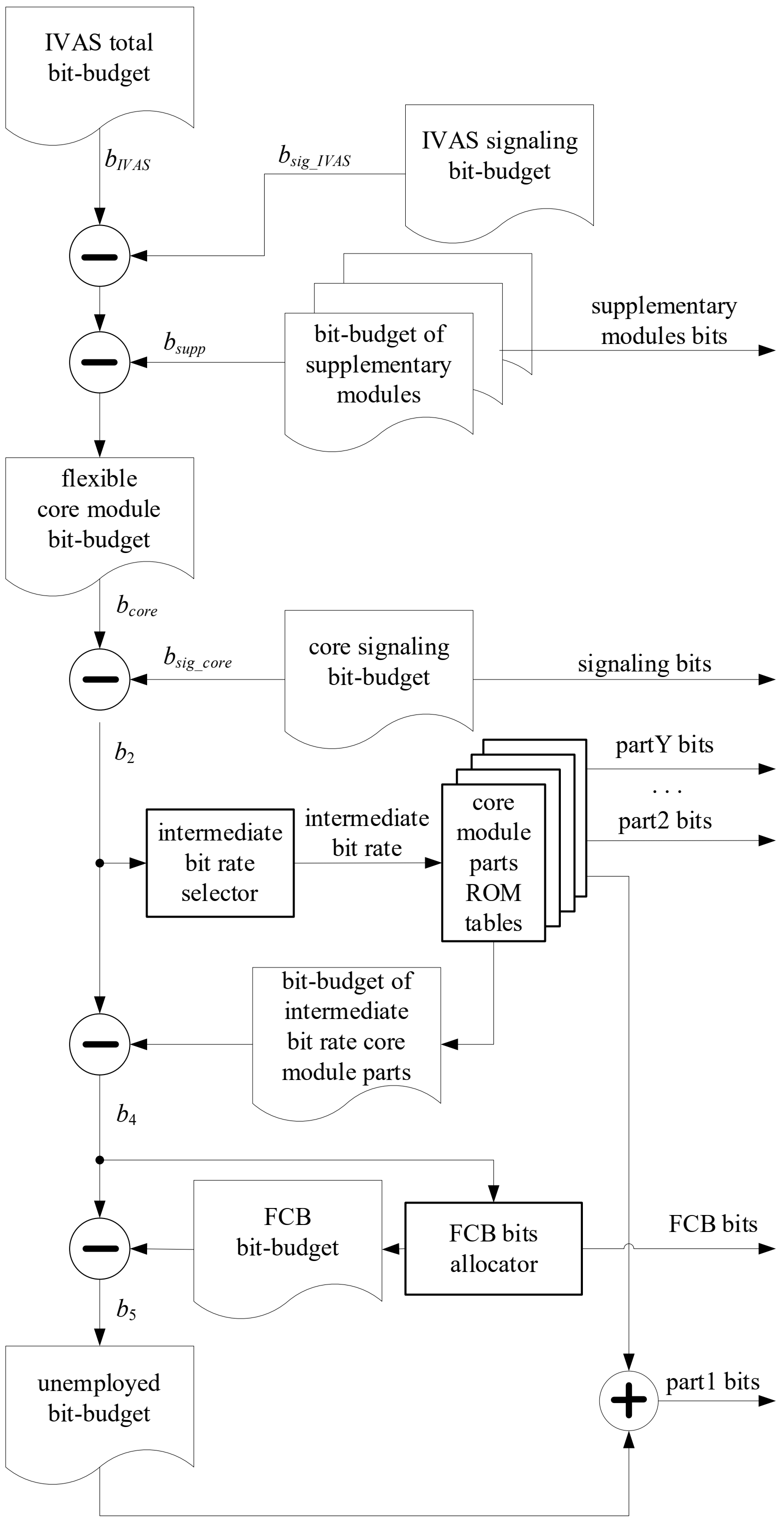

IVAS total bit-budget
b_{IVAS}
b_{sig_IVAS}
IVAS signaling bit-budget
b_{supp}
bit-budget of supplementary modules
supplementary modules bits
flexible core module bit-budget
b_{core}
b_{sig_core}
core signaling bit-budget
signaling bits
b_2
partY bits
. . .
part2 bits
intermediate bit rate selector
intermediate bit rate
core module parts ROM tables
bit-budget of intermediate bit rate core module parts
b_4
FCB bit-budget
FCB bits allocator
FCB bits
b_5
unemployed bit-budget
part1 bits

Figure 5.2-1: Block diagram of the bit-budget allocator.

The ACELP bit-budget allocator from Figure 5.6-1 operates on a frame-by-frame basis and consists of several operations described below. Its fluctuating bit-budget is a result of a fluctuating number of bits used for encoding supplementary codec modules while these modules are processed and their bit-budget set before the core-coder bit-budget allocator is run. It is noted that the distribution of bit-budget among the different core module parts is symmetrically done at the encoder and the decoder using the same allocation algorithm.

Referring to Figure 5.6-1, an IVAS total bit-budget b_{IVAS} is allocated to the codec for each frame.

First, there are determined the number of bits (bit-budget) of all supplementary modules, b_{supp}, used for encoding the supplementary codec modules like stereo or spatial information, metadata, or BWE, and the number of bits b_{sig_IVAS}

for transmitting IVAS format signalling to the decoder. These two bit-budgets are then subtracted from the IVAS total bit-budget to obtain a bit-budget of the ACELP core module, b_{core}, using the following relation:

$$b_{core} = b_{IVAS} - b_{sig\_IVAS} - b_{supp}. \tag{5.2-18}$$

Next operation subtracts the bit-budget b_{sig_core} related to the transmission of ACELP core module signalling parameters like core-coder type or audio bandwidth, similarly as done in EVS:

$$b_2 = b_{core} - b_{sig\_core}. \tag{5.2-19}$$

Then, the bit-budget b_2 is converted in an intermediate bitrate selector from Figure 5.6-1 to bitrate $brate_2$ and its related so-called candidate bitrate is found. Specifically, in IVAS, a predetermined number of twenty-two (22) candidate bitrates is used. These bitrates are as follows (note an overlap with defined EVS core-coder bitrates): 5.00 kbps, 6.15 kbps, 7.20 kbps, 8.00 kbps, 9.60 kbps, 11.60 kbps, 12.15 kbps, 12.85 kbps, 13.20 kbps, 14.80 kbps, 16.40 kbps, 22.60 kbps, 24.40 kbps, 29.00 kbps, 29.20 kbps, 30.20 kbps, 30.40 kbps, 32.00 kbps, 48.00 kbps, 64 kbps, 96 kbps, and 128 kbps. The found intermediate bitrate for core-coder parts is then the nearest higher candidate intermediate bitrate to the ACELP core module bitrate. For example, for $brate_2 = 9.00$ kbps bitrate the found intermediate bitrate would be 9.60 kbps using the candidate intermediate bitrates from the list of 22 IVAS candidate bitrates.

Then, for each candidate intermediate bitrate, pre-determined bit-budgets for encoding first parts of the ACELP core module are stored in ROM tables. These ACELP core module first parts comprise the LP filter coefficients, the adaptive codebook, the adaptive codebook gain, and the innovation codebook gain. Note that the ROM tables, one ROM table per one first part parameter, used to allocate the ACELP first parts bit-budgets are ROM tables as used in EVS complemented by five (5) other candidate bitrates. Also note that no bit-budget for encoding the innovation codebook is stored in the ROM tables but its bit-budget is computed as described in the next Clause 5.2.2.2.2.1.2.

The bit-budget selector thus allocates for encoding the ACELP core module first parts the bit-budgets stored in the ROM tables and associated to the intermediate bitrate selected by the bit-budget selector.

In the next step, the bit-budget for intermediate bitrate core module first parts is subtracted from the bit-budget b_2. More specifically, there are subtracted from bit-budget b_2 the following bit-budgets: (a) bit-budget b_{LPC} for encoding the LP filter coefficients associated to the candidate intermediate bitrate selected by the bit-budget selector, (b) the sum of the bit-budgets $b_{ACB}[m], m = 0,..,M-1,$ of the M subframes associated to the selected candidate intermediate bitrate, (c) the sum of the bit-budgets $b_{gain}[m]$ for quantizing the adaptive and innovation codebook gains of the M subframes associated to the selected candidate intermediate bitrate, (d) the bit-budget, associated to the selected intermediate bitrate, for encoding other ACELP core module first parts that are present like FEC bits or the low-pass adaptive excitation filtering flag. Thus, a remaining bit-budget b_4 still available for encoding the innovation codebook (which relates to the second ACELP core module part) is found as:

$$b_4 = b_2 - b_{LPC} - \sum_{m=0}^{M-1} b_{ACB}[m] - \sum_{m=0}^{M-1} b_{gain}[m] - \cdots. \tag{5.2-20}$$

5.2.2.2.2.1.2 Innovation codebook bit-budget distribution

The distribution of bit-budget for innovation codebook coding is done in Fixed Codebook (FCB) allocator as part of the bit-budget allocator from Figure 5.6-1. The FCB allocator distributes the remaining bit-budget b_4 from (5.2-20) for encoding the innovation codebook (aka second ACELP core module part) between the M subframes of the current frame. Specifically, the bit-budget b_4 is divided into bit-budgets $b_{FCB}[m]$ allocated to the various subframes m. At the same time, there must be kept in mind that only some bit-budget values are supported by the innovative codebook. These supported bit-budgets per subframe are 7, 10, 12, 15, 17, 20, 24, 26, 28, 30, 32, 34, 36, 40, 43, 46, 47, 49, 50, 53, 55, 56, 58, 59, 61, 62, 65, 68, 70, 73, 75, 78, 80, 83, 85, 87, 89, 92, 94, 96, and 98 bits.

The FCB allocator then distributes the bit-budget between subframes in an iterative procedure which divides the bit-budget b_4 between the M subframes as equally as possible while assuming the following principles:

(1) In case the bit-budget b_4 cannot be distributed equally between all the subframes, a highest possible (i.e. a larger) bit-budget is allocated to the first subframe. As an example, if $b_4 = 106$ bits, the FCB bit-budget per 4 subframes is allocated as 28-26-26-26 bits.

(2) If there are more bits available to potentially increase other subframe FCB codebooks, the FCB bit-budget allocated to at least one next subframes after the first subframe is increased. As an example, if $b_4 = 108$ bits, the FCB bit-budget per 4 subframes is allocated as 28-28-26-26 bits. In another example, if $b_4 = 110$ bits, the FCB bit-budget per 4 sub-frames is allocated as 28-28-28-26 bits, etc.

(3) The bit-budget b_4 is not necessarily distributed as equally as possible between all the subframes but rather to use as much as possible the bit-budget b_4. As an example, if $b_4 = 87$ bits, the FCB bit-budget per 4 subframes is allocated as 26-20-20-20 bits rather than e.g. 24-20-20-20 bits when the principle (3) would not be considered. In another example, if $b_4 = 91$ bits, the FCB bit-budget per 4 subframes is allocated as 26-24-20-20 bits in contradiction to e.g. 24-24-20-20 bits would be allocated if the principle (3) is not considered. Consequently, only 1 bit remains unused when the principle (3) is considered while 3 bits would remain unused otherwise.

(4) In case the bit-budget cannot be equally distributed between all the subframes when encoding using a Transition Coding (TC) mode (see Clause 5.2.3.2 in [3][2]), the largest possible (i.e. larger) bit-budget is allocated to the subframe using a glottal-impulse-shape codebook. As an example, if $b_4 = 122$ bits and the glottal-impulse-shape codebook is used in the third subframe, the FCB bit-budget per 4 subframes is allocated as 30-30-32-30 bits.

(5) If, after applying the principle (4), there are more bits available to potentially increase another FCB codebook in a TC mode frame, the FCB bit-budget allocated to the last subframe is increased. As an example, if $b_4 = 116$ bits and the glottal-impulse-shape codebook is used in the second subframe, the FCB bit-budget per 4 subframes is allocated as 28-30-28-30 bits.

The sum of the bit-budgets (number of bits) $b_{FCB}[m], m = 0,..,M-1,$ allocated to the M various subframes for encoding the innovation codebook is thus

$$\sum_{m=0}^{M-1} b_{FCB}[m] \qquad (5.2\text{-}21)$$

Then, a subtractor at the very bottom of Figure 5.6-1 determines the number of bits b_5 remaining after encoding of the innovation codebook, using the following relation:

$$b_5 = b_4 - \sum_{m=0}^{M-1} b_{FCB}[m]. \qquad (5.2\text{-}22)$$

Ideally, after encoding of the innovation codebook, the number of remaining bits b_5 is equal to zero. However, it may not be possible to achieve this result because the granularity of the innovation codebook index is greater than 1 (usually 2-3 bits). Consequently, a small number of bits often remain unemployed after encoding of the innovation codebook.

Finally, a bit-budget allocator from Figure 5.6-1 thus assigns the unemployed bit-budget b_5 to increase the bit-budget of one of the ACELP core first module parts. Specifically, the unemployed bit-budget b_5 is used to increase the bit-budget b_{LPC} obtained from the ROM tables, using the following relation:

$$b_{LPC}' = b_{LPC} + b_5. \qquad (5.2\text{-}23)$$

5.2.2.2.2.1.3 Bit-budget distribution in high bitrate ACELP

At high bitrate ACELP core, there is used a combined algebraic codebook as described in Clause 5.2.3.1.6 of [3] which contains a transform-domain AVQ-based codebook. In high bitrate ACELP, bit-budget is allocated to the ACELP core module parts using an extended version of the procedure as described in previous Clause 5.2.2.2.2.1.1. Following this procedure, the sum of the bit-budgets $b_{FCB}[m], m = 0, \ldots, M-1,$ from (5.2-21) for encoding the FCB codebook in the M subframes should be equal or approach bit-budget b_4. In the high bitrate ACELP, the bit-budgets $b_{FCB}[m]$ are however modest, and the number of unemployed bits b_5 from (5.2-22) is relatively high and is used to encode the transform-domain codebook parameters as follows.

First, the sum of the bit-budget $b_{TDgain}[m]$ for encoding the transform-domain pre-quantizer gain (see Clause 5.2.3.1.6.2 in [3]) in the M subframes are subtracted from the unemployed bit-budget b_5, using the following relation:

$$b_7 = b_5 - \sum_{m=0}^{M-1} b_{TDgain}[m]. \qquad (5.2\text{-}24)$$

Then, the remaining bit-budget b_7 is allocated to the AVQ vector quantizer within the transform-domain codebook (see Clause 5.2.3.1.6.9 in [3]) and distributed among all M subframes. The bit-budget by subframe of the vector quantizer is denoted as $b_{VQ}[m]$. Giving the AVQ quantization steps, the vector quantizer does not consume all of the allocated bit-budget $b_{VQ}[m]$ leaving a small variable number of bits available in each subframe. These bits are floating bits employed in the following subframe within the same frame. For a better effectiveness of the transform-domain codebook, a slightly higher bit-budget is allocated to the vector quantizer in the first subframe. The implementation of this logic is given in the following pseudocode:

```
b_tmp = floor(b_7 / M);
for( m = 0; m < M; m++ )
{
   b_VQ[m] = btmp;
}
b_VQ[0] = b_tmp + (b_7 - M * b_tmp)
```

where $floor(x)$ denotes the largest integer less than or equal to x and M is the number of subframes in one frame. Bit-budget b_7 is thus distributed equally between all the subframes while the bit-budget for the first subframe is eventually slightly increased by up to $M-1$ bits. Consequently, in high bitrate ACELP, there are no remaining bits after this operation.

5.2.2.2.2.2 Fast algebraic codebook search

The existence of a variable bitrate ACELP coding and a very low bitrate available for core coding in some configurations make a demand for algebraic codebooks that support algebraic codebooks sizes which are not part of the EVS codec. Thus, in addition to the algebraic codebooks as described in Clause 5.2.3.1.5 in [3], an additional so-called fast algebraic codebook search is present in IVAS.

The fast algebraic codebook search is employed for subframe lengths $L_{subfr}=64 \text{ samples}$ and $L_{subfr}=128 \text{ samples}$ at internal sampling length of 12.8 kHz and its supported configurations are summarized in Table 5.2-3.

Table 5.2-3: Fast algebraic codebook configurations

subframe length	bits/subframe
L_subfr = 64	10, 12, 15, 17
L_subfr = 128	12, 14, 18, 20, 24

The structure of fast algebraic codebook is based on interleaved single-pulse permutation (ISSP) design where the pulse positions are divided into several tracks of interleaved positions. The details about the structure including number of pulses and number of tracks is summarized in Table 5.2-4 for codebooks with $L_{subfr}=64 \text{ samples}$ resp. in Table 5.2-5 for codebooks with $L_{subfr}=128 \text{ samples}$.

Table 5.2-4: Structure of fast algebraic codebook search for $L_{subfr}=\mathbf{64\ samples}$

bits/subframe	number of pulses	number of tracks	note
10	2	4	even tracks used
12	2	2	all tracks used
15	3	4	first three tracks used
17	3	4	all tracks used

Table 5.2-5: Structure of fast algebraic codebook search for $L_{subfr}=\mathbf{128\ samples}$

bits/subframe	number of pulses	number of tracks	note
12	2	4	even tracks used
14	2	2	all tracks used
18	3	4	first three tracks used
20	3	4	all tracks used
24	4	4	all tracks used

The fundamental principle of the fast algebraic codebook search uses a sequential search of the pulses by maximizing a criterion based on a certain reference signal. The optimum fixed codebook gain, filtered target vector and reference signal are then recomputed after each new pulse is determined.

Similar to Clause 5.2.3.1.5.9 in [3], let us define a reference signal $b(n)$ as a weighted sum of the signal $d(n)$ and an ideal excitation signal $r(n)$ as

$$b(n)=\sqrt{\frac{E_d}{E_r}}r(n)+\beta d(n) \qquad (5.2\text{-}25)$$

where $E_d=\mathbf{d}^T\mathbf{d}$ is the energy of the signal $d(n)$ and $E_r=\mathbf{r}^T\mathbf{r}$ is the energy of the ideal excitation signal $r(n)$. The value of the scaling factor $\beta=2.0$.

The signal $d(n)$ in (5.2-25) is the backward filtered target vector defined in Equation (524) of [3], i.e.:

$$d(n) = \sum_{i=n}^{L-1} x_{11}(i)h(i-n)\,, n = 0, \dots, L-1 \qquad (5.2\text{-}26)$$

where $x_{11}(n)$ is an updated target signal computed in Equation (523) of [3] and $h(n)$ is the impulse response of the weighted synthesis filter. Then $n, n = 0, \dots, L-1,$ is a time sample index and L is the subframe length L_{subfr} (the subscript is avoided in this Clause for simplicity).

Further, the pulse signs are pre-determined using an approach where the sign of a pulse at specific position is set a priori equal to the sign of the reference signal $b(n)$ at that position. That is, there is defined the sign vector $z_b(n)$ containing the signs of the reference signal $b(n)$.

Further, there is employed an autocorrelation method for error minimization in the fixed codebook search procedure in which the matrix of correlations with elements

$$\varphi(i,j) = \sum_{n=j}^{L-1} h(n-i)h(n-j)\,, i = 0, \dots, L-1, j = i, \dots, L-1 \qquad (5.2\text{-}27)$$

is reduced to a Toeplitz form by modifying the summation limits in Equation (5.2-27) so that $\varphi(i,j) = \alpha(|i-j|)$, where

$$\alpha(n) = \sum_{i=n}^{L-1} h(i)h(i-n) \qquad (5.2\text{-}28)$$

is the correlation vector.

Now, the fast algebraic codebook search of M pulses can be summarized in the following steps:

Step I: The backward filtered target vector $d(n)$, the correlation vector $\alpha(n)$, the reference signal $b(n)$, and the sign vector $z_b(n)$ are computed.

Step II: Find the first pulse: the first pulse is found as the index of maximum absolute value of the reference signal $b(n)$. Using the sign vector $z_b(n)$, this can be expressed as

$$m_0 = \text{index}[\max(z_b(n)b(n))] \qquad (5.2\text{-}29)$$

$$s_0 = z_b(m_0) \qquad (5.2\text{-}30)$$

where m_0 is the index of the first pulse position and s_0 its sign.

Step III: Find following pulses: The other pulses are searched sequentially for $j = 1, \dots, M-1$. The search of each new pulse starts with computing the fixed codebook gain g_c and the update of the reference signal $b(n)$. The fixed codebook gain for the previously found pulses (pulses $m_0, \dots, m_{j-1}$) is given by

$$g_c^{(j-1)} = \frac{g_N^{(j-1)}}{g_D^{(j-1)}} \qquad (5.2\text{-}31)$$

where numerator and denominator are expressed as

$$g_N^{(j-1)} = g_N^{(j-2)} + s_{j-1}d(m_{j-1}) \qquad (5.2\text{-}32)$$

and

$$g_D^{(j-1)} = g_D^{(j-2)} + \alpha(0) + 2\sum_{i=0}^{j-2} s_i s_{j-1}\alpha(|m_i - m_{j-1}|) \qquad (5.2\text{-}33)$$

with the initialization $g_N^{-1} = 0$ and $g_D^{-1} = 0$.

The target signal is then updated by subtracting the contributions of the found pulses from the original target signal $x_{11}(n)$. This can be written as

$$x'_{11}(n) = x_{11}(n) - g_c^{(j-1)} \sum_{i=0}^{j-1} s_i h(n - m_i) \qquad (5.2\text{-}34)$$

and substituting $x'_{11}(n)$ from (5.2-34) in (5.2-25) and using (5.2-28) we get an update of the signal $d(n)$, i.e.

$$\acute{d}(n) = d(n) - g_c^{(j-1)} \sum_{i=0}^{j-1} s_i \alpha(|n - m_i|) \qquad (5.2\text{-}35)$$

Next, the reference signal $b(n)$ is updated as

$$\acute{b}(n) = \sqrt{\frac{E_d}{E_r}}r(n) + \beta\acute{d}(n) \tag{5.2-36}$$

and the new pulse m_j is found similarly to (5.2-29) and (5.2-30) as

$$m_j = \text{index}\big[\max\big(z_b(n)\acute{b}(n)\big)\big] \tag{5.2-37}$$

$$s_j = z_b(m_j) \tag{5.2-38}$$

Step IV: Compute the fixed codevector and the filtered fixed codevector.

Giving the use of mentioned ISSP design, the steps II to IV are repeated. Let's first suppose that the number of tracks is equal to the number of searched pulses M (one pulse per track) while it is then extended for other configurations. The first iteration is initialized in step II by assigning m_0 to T_0, m_1 to T_0, …, m_{M-1} to T_{M-1}, where T_z is the track number. The procedure is then repeated from step II to step IV by assigning the pulses in the initialization to different tracks and computing equations (5.2-34) to (5.2-38) for $n \in T_z$ only. The number of iterations is equal to M. Finally, the set of pulses selected in the iteration that maximizes the criterion from (526) in [3] is chosen.

5.2.2.2.2.3 Comfort Noise Addition (CNA)

5.2.2.2.2.4 AVQ Bit Savings Encoder

5.2.2.2.2.4.1 Overview

Like EVS [3], the time domain or transform-domain coefficients are split into sub-frames, prior to the AVQ quantization. In every sub-frame, a bit budget allocated to the AVQ subframe is composed of a first number of bits which is a sum of a fixed bit-budget and a floating number of bits. AVQ subframes usually does not consume all the allocated bits, leaving a variable number of bits available in each subframe. These bits are floating bits employed in the following AVQ sub-frame. The floating number of bits is equal to 0 in the first sub-frame and the floating bits resulting from AVQ in the last sub-frame in each frame remain unused when coding WB signals or are re-used in coding of upper band. Allocated bits for each AVQ subframe is used for quantizing subframe coefficients, which are split into 8 consecutive sub vectors of 8 coefficients each. The sub-vectors are quantized with an 8-dimensional multi-rate algebraic vector quantizer (AVQ). The parameter of each sub vector sv consist of the codebook number n_{sv} and a corresponding code vector index v_{sv}. The codebook number is in the set of integers {0, 2, 3, 4, 5, 6, 7, 8, ...} and the size of its unary code representation is n_{sv} bits and the size of each index v_{sv} is given by $4n_{sv}$ bits, equalling to $5n_{sv}$ bits per sub-vector with exception of the case of codebook number with the integer 0 that requires 1 bit. Based on the number of bits-available for encoding a sub-vector in the AVQ, one of two encoding methods is selected for encoding the sub-vector. The one is encoding the codebook number of the sub-vector (i.e., the original AVQ encoding [3]), and the other is encoding "unused" bits which is obtained by computing the difference between the number of bits allocated to the AVQ subframe and the number of bits consumed by the AVQ parameters. If the number of bits for encoding the "unused" bits is smaller than the number of bits for encoding the codebook number, the total number of AVQ bits can be reduced and saved. Such condition can be determined using the number of bits available for encoding the sub-vector in the AVQ. Detailed algorithm will be described in the following sub clauses. Figure 5.2-2 shows an example of an AVQ encoder for encoding DCT coefficients. The AVQ bit-saving algorithm is integrated in the code converter block in the figure.

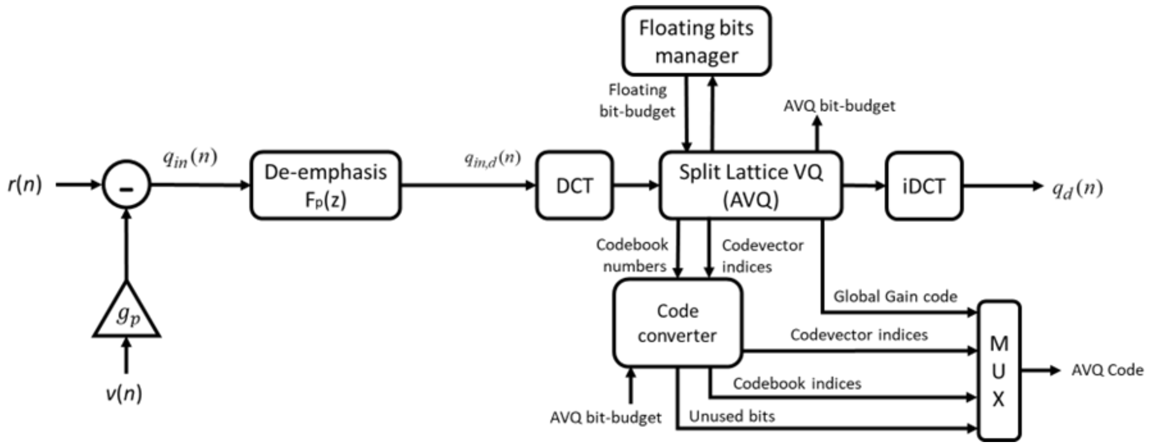


Figure 5.2-3: Overview of AVQ Bit Savings Encoder

5.2.2.2.2.4.2 Code Converter

5.2.2.2.2.4.3 Unused Bit Encoding

5.2.2.2.2.5 Bandwidth extension in time domain

5.2.2.2.2.6 Multimode FD bandwidth extension coding

5.2.2.2.3 MDCT based coding

5.2.2.2.3.1 Variable bitrate MDCT coding

5.2.2.2.3.2 TCX Entropy Coding

5.2.2.2.3.3 Intelligent Gain Filling (IGF)

5.2.2.2.3.3.1 General

The general functionality of IGF is described in clause 5.3.3.2.11 of [3]. In the following all changes and additions that were made for IVAS are described.

For IVAS, activation of IGF and selection of individual configuration is not dependent on fixed bitrates, but instead bitrate intervals are used depending on the selected bandwidth for processing, the IVAS format and the type and number of IVAS elements (SCEs/CPEs). For modes where multiple elements are coded, the IGF configuration is called separately for each element. Accordingly, the bitrates given to the IGF configuration do not correspond to the overall bitrate but are bitrates specified for each element.

The maximum configuration bitrates for which IGF is active for SCEs and CPEs are shown in tables 5.2-6 and 5.2-7:

Table 5.2-6: IGF application modes – IVAS SCE

Mode	Bitrate
WB	≤ 9.6 kbps
SWB	≤ 64 kbps
FB	≤ 128 kbps

Table 5.2-7: IGF application modes – IVAS CPE

Mode	Bitrate
WB	≤ 13.2 kbps
SWB	≤ 96 kbps
FB	≤ 128 kbps

NOTE: Some low bitrate modes do not allow FB operation. Accordingly, IGF is only applied up to SWB for such cases. [reference clause where maximum allowed bandwidth is described].

5.2.2.2.3.3.2 IGF Damping

5.2.2.2.3.3.2.1 Introduction

Typically, signal spectra become more noise-like toward higher frequencies. Therefore, the IGF source regions, which are located in a lower part of the spectrum than the higher-frequency target regions they are used to fill, are often too tonal compared to the original high-frequency spectrum. This is remedied by whitening of the tiles (as described in clause 5.3.3.2.11.6 of [3]). In much rarer cases, however, it can also happen that a very noisy source spectrum is copied to a much more tonal target spectrum. In the most extreme cases, pure background noise might be copied to a place where the original signal has a tonal component, like an overtone. This background noise will be scaled to the level of the tonal component by the transmitted scale-factor (which adjusts the spectral envelope of the source tile to the level of the target tile) inside the according scale-factor band (SFB). This will result in a noise band in the high-frequency spectrum which can be very audible and detrimental to the overall quality. An illustration of this effect is shown in Figure 5.2-4.of noise bands due to tonality mismatch

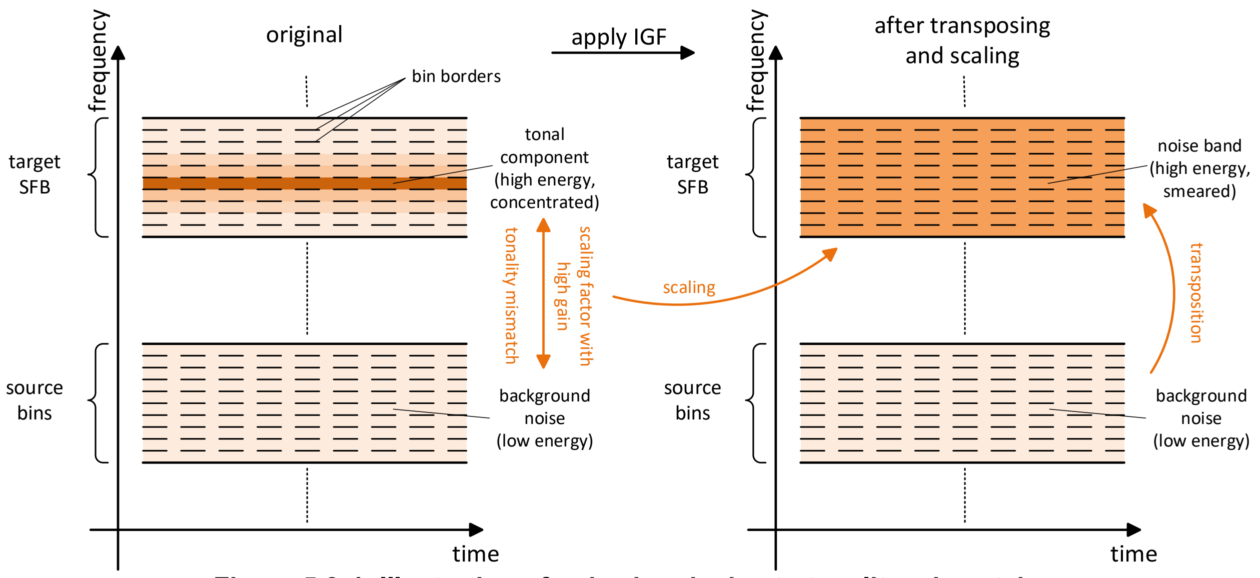

original
apply IGF
after transposing and scaling
frequency
bin borders
target SFB
tonal component (high energy, concentrated)
tonality mismatch
scaling factor with high gain
scaling
noise band (high energy, smeared)
transposition
source bins
background noise (low energy)
time

Figure 5.2-4: Illustration of noise bands due to tonality mismatch

To improve quality in such cases, a scale-factor damping is applied at the IGF encoder. The idea is to detect cases where such a mismatch occurs by analyzing tonality in both source and target SFBs and use the results to calculate a compensation value depending on the severity of the mismatch. With this compensation value and the original scale factor, a reduced scale-factor is calculated for the target SFB which will be transmitted in the bitstream and which will lead to a significantly reduced noise band when applied during the decoder IGF processing. There are no additional processing steps at the decoder, the technique is therefore encoder-only. A block diagram of the IGF encoder with the necessary adaptations is shown in Figure 5.2-5 and the individual processing steps are described in more detail in the following.

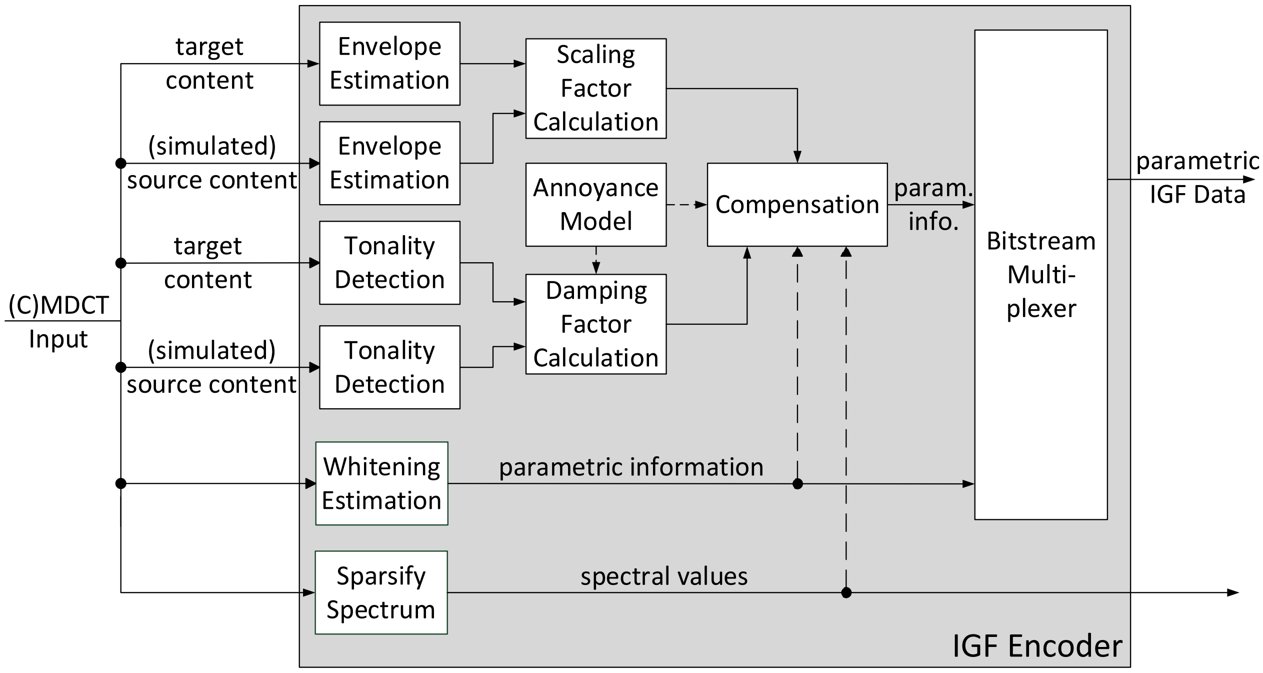


Figure 5.2-5: IGF encoder with added damping

5.2.2.2.3.3.2.2 Tonality mismatch detection

In a first step, those SFBs, where a tonality mismatch might cause noise band artefacts, have to be identified. In order to do so, the tonality in each SFB of the IGF range and the corresponding source bands has to be determined. The tonality is calculated in the same manner as it is already done for IGF Whitening using both the spectral flatness measure (SFM) and the crest factor. Spectral flatness and crest factor calculations are described in clauses 5.3.3.2.11.1.3 and 5.3.3.2.11.1.4 of [3], respectively. Smoothed spectral flatness values SFM_{src} and SFM_{dest} for source and target SFBs are obtained by following the calculation of *s(k)* in step 1) of clause 5.3.3.2.11.6.3 of [3]. However, in this case the values are calculated per SFB *b* and not per tile *k*.

Now the difference in tonality between source and destination is calculated:

$$SFM_{diff} = SFM_{src} - SFM_{dest} \qquad (5.2\text{-}39)$$

Positive values of this difference indicate that the copied-up source SFB is noisier than the target SFB. Such an SFB becomes a likely candidate for damping.

In order to avoid damping in some unwanted cases, e.g. band-limitation inside an SFB, possibly affected SFBs the spectral tilt of the energy in all target bands with positive SFM differences is calculated as:

$$slope = \left(\sum_i x_i P_i - \frac{1}{e-b}\sum_i x_i \sum_i P_i\right)\left(\sum_i {x_i}^2 - \frac{1}{e-b}\left(\sum_i x_i\right)^2\right)^{-1}, \quad i = b, \ldots., e-1 \qquad (5.2\text{-}40)$$

with x as the bin number, P the TCX power spectrum,   the start line and e the stop line of the current SFB.

However, a tonal component close to a border of an SFB might also cause a steep tilt, but should still be subjected to damping. To separate these two cases, another shifted SFM calculation is performed for bands with steep tilt.

The threshold for the slope value is defined as

$$thresh_{tilt} = \frac{60}{e-b} \qquad (5.2\text{-}41)$$

with division by the SFB width *e-b* as normalization.

If there is a strong decline slope < -$thresh_{tilt}$, the frequency region for which SFM is calculated will be shifted down by half the width of the SFB; for a strong incline slope > -$thresh_{tilt}$ it is shifted up. In this way, tonal components that are supposed to be damped can still be correctly detected due to low SFM while for higher SFM values damping will not be applied. The threshold here is defined as the value 0.04, where damping is only applied if the shifted SFM falls below the threshold.

5.2.2.2.3.3.2.3 Perceptual annoyance model

To ensure application within reasonable bounds, damping should only be used if the target SFB is indeed very tonal. So only whenever both

$$SFM_{diff} > 0 \qquad (5.2\text{-}42)$$

and

$$SFM_{dest} < 0.1 \qquad (5.2\text{-}43)$$

hold, damping should be applied.

Additionally, the amount of damping is dependent on the tonal-to-noise ratio in the SFB. For SFBs with a higher ratio, i.e. lower background noise, more damping is applied, and vice versa.

For the calculation of this tonal-to-noise ratio the squared TCX power spectral values P of all bins i in an SFB are summed up and divided by the width of the SFB (given by start line b and stop line e) to get the average energy of the band. This average is subsequently used to normalize all the energies in the band.

$$P_{norm,k} = \sqrt{P_k}\left(\frac{1}{e-b} * \sum_{sb=b}^{e-1} \sqrt{P_{sb}}\right)^{-1}, \qquad k = b, \dots, e-1 \qquad (5.2\text{-}44)$$

All bins with a normalized energy $P_{norm,k}$ below 1 are then summed up and counted as the noise part P_{noise} while everything above a threshold of 1 + adap with

$$adap = \frac{e-b}{40} \qquad (5.2\text{-}45)$$

is counted as the tonal part P_{tonal}. From the tonal and the noise part the final tonal-to-noise log-ratio is computed:

$$tonalToNoise = 20 * \log_{10}\left(\frac{P_{tonal}}{P_{noise}}\right) \qquad (5.2\text{-}46)$$

5.2.2.2.3.3.2.4 Calculation of damping factor and compensation of tonality mismatch

Now the damping factor d_{curr} is calculated as

$$d_{curr} = \left(\frac{SFM_{dest}}{SFM_{src}}\right)^{\alpha} + \beta, \qquad (5.2\text{-}47)$$

where α and β are damping adjustment parameters that are calculated as

$$\alpha = min\left(\frac{320}{e-1}, 1.25\right) \qquad (5.2\text{-}48)$$

with e is the stop line of the current SFB and

$$\beta = \begin{cases} 0.1 * \left((10 + adap) - tonalToNoise\right), & \text{if } \left((10 + adap) - tonalToNoise\right) > 0 \\ 0, & else \end{cases} \qquad (5.2\text{-}49)$$

where adap is again calculated as

$$adap = \frac{e-b}{40} \qquad (5.2\text{-}50)$$

The parameter α decreases with frequency in order to apply less damping for higher frequencies while β is used to further reduce the strength of the damping if the tonal-to-noise ratio of the SFB falls below a threshold. The more significantly it falls below this threshold, the more the damping is reduced.

As damping is only activated within certain constraints, it is necessary to apply smoothing in order to prevent abrupt on/off transitions. To realize this, several smoothing mechanisms are active.

Directly after a transient, a core switch to TCX or an undamped previous frame damping is only gradually applied to avoid extreme energy drops after high-energy transients. Furthermore, a forgetting factor in the form of an IIR filter is utilized to also take the results of previous frames into account.

All smoothing techniques are comprised in the following formula:

$$d = min\left(\frac{d_{curr}+d_{prev}}{2} + 0.1 * smooth, 1\right), \tag{5.2-51}$$

where d_{prev} is the damping factor of the previous frame. If damping was not active in the previous frame d_{prev} is overwritten with d_{curr} but limited to a minimum of 0.1. The variable smooth is an additional smoothing factor that will be set to 2 during transient frames (flag isTransient active) or after core switches (flag isCelpToTCX active), to 1 if in the previous frame damping was inactive. In each frame with damping the variable will be decreased by 1, but may not fall below 0. If a power spectrum is not available or if the frame is transient, no damping will be applied.

In the final step, the damping factor *d* is multiplied with the scaling gain:

$$g_{damped} = g * d \tag{5.2-52}$$

The resulting compensated gain factor is then transmitted in the bitstream as part of the core-encoded IGF side data.

5.2.2.2.3.4 Stabilization of IGF Whitening levels

In some cases, the calculated IGF whitening levels *currWLevel(k)* for each tile *k* (as described in clause 5.3.3.2.11.6.3 of [3]) can fluctuate quite strongly by changing their value every few frames. This leads to an unstable listening impression detrimental to the overall quality. Therefore, additional stability mechanisms are introduced for IVAS as described below.

5.2.2.2.3.4.1 Margin-based Whitening thresholds and minimum deviation from past SFM mean

In cases where $currWLevel(k) \neq prevWLevel(k)$, switching to the new whitening level $currWLevel(k)$ is allowed if the spectral flatness *SFM* of tile *k* (see calculation of $s(k)$ in clause 5.3.3.2.11.6.3 of [3]) lies outside a margin of ± 0.15 around the whitening thresholds ThM_k and ThS_k (see mapping in clause 5.3.3.2.11.1.5 of [3]).

If it falls inside either margin, as in

$$ThM_k - 0.15 < SFM_k < ThM_k + 0.15 \tag{5.2-53}$$

or

$$ThM_s - 0.15 < SFM_k < ThM_s + 0.15 \tag{5.2-54}$$

switching is prohibited if additionally the deviation of SFM_k from the mean $SFM_{mean,k}$ of the *SFM* values of the n_{prev} previous frames is sufficiently small:

$$currWLevel(k) = \begin{cases} prevWLevel(k), & SFM_{mean,k} < 0.2 \\ currWLevel(k), & else \end{cases} \tag{5.2-55}$$

$SFM_{mean,k}$ is calculated for each tile *k*:

$$SFM_{mean,k} = \frac{\sum_{i=0}^{n_{prev}-1} SFM_{prev,k}(i)}{n_{prev}} \tag{5.2-56}$$

where n_{prev} is the number of available past values $SFM_{prev,k}$ in the previous 5 frames. If past SFM values are not available for one the past 5 frames - due to the frame being ACELP or transient – these frames will be excluded and n_{prev} reduced accordingly.

5.2.2.2.3.4.2 Oscillating pitch handling

If the pitch of a signal is not stable but oscillates rapidly (e.g. a voice or instrument employing vibrato) overtones of the signal might constantly cross tile borders. This can strongly affect the SFM values in the two tiles between which the overtone oscillates and, in the worst case, can lead to constantly changing whitening levels. If such a case is detected, whitening is turned off for both tiles.

The detection is done for pairs of tiles *k* and *k+1*. First, the change in *SFM* for both tiles is calculated over the 3 previous frames (provided and SFM value is available for all 3 frames) by summing up the SFM differences:

$$SFM_{diff,k} = \sum_{i=0}^{2}\left(SFM_k(i) - SFM_k(i+1)\right) \qquad (5.2\text{-}57)$$

and

$$SFM_{diff,k+1} = \sum_{i=0}^{2}\left(SFM_{k+1}(i) - SFM_{k+1}(i+1)\right) \qquad (5.2\text{-}58)$$

If either

$$SFM_{diff,k} < 0 \quad \&\& \quad SFM_{diff,k+1} > 0 \qquad (5.2\text{-}59)$$

or

$$SFM_{diff,k} > 0 \quad \&\& \quad SFM_{diff,k+1} < 0 \qquad (5.2\text{-}60)$$

holds – indicating that the tonality changes in opposite directions between the tiles – and the difference between $SFM_{diff,k}$ and $SFM_{diff,k+1}$ is sufficiently large as in

$$\left|SFM_{diff,k} - SFM_{diff,k+1}\right| > 0.5 \qquad (5.2\text{-}61)$$

and if $currWLevel(k) = 0$ and $currWLevel(k+1) \neq 0$, or vice versa, is true – meaning whitening is off only in one of the tiles – the whitening will be turned off in both tiles:

$$currWLevel(k) = currWLevel(k+1) = 0 \qquad (5.2\text{-}62)$$

Editors note: Whitening hangover

Additionally to the stabilization methods described above, a hangover of 2 frames is employed. This means that any conditions for a switch of the whitening level have to be fulfilled for 3 frames in a row before the switch is allowed in the third frame.

5.2.2.2.3.5 Whitening processing for CPE bitrate range of]32.0-48.0] kbps

In cases where the IVAS element is a CPE and the configuration bitrate falls in the]32.0-48.0] kbps range, a different processing is done to determine the whitening levels.

Instead of calculating a separate whitening level for each tile, this is done only for the first tile. The whitening levels of the other tiles are set to the same value. Additionally, only whitening levels 0 and 1 – indicating no whitening and mid-whitening, respectively – are allowed.

The calculation of this one whitening level based on the first tile is also done differently from other modes. Instead of setting the level by comparing the SFM value of the target tile to fixed thresholds (see clause 5.3.3.2.11.1.5 of [3] and clause 5.2.2.2.3.6 in the present document), the level is determined adaptively by comparing the SFM value of the target tile against the SFM value in the source tile that is used for copy-up. This comparison further depends on the decision of the speech-music classifier [reference to this classifier, IVAS chapter classifier].

If the signal is classified as **music**, the whitening level will be set to 0 – indicating no whitening – for

$$SFM_{tar} \leq SFM_{src} + 0.5 \qquad (5.2\text{-}63)$$

and to 1 – indicating mid-whitening – for

$$SFM_{tar} > SFM_{src} + 0.5 \qquad (5.2\text{-}64)$$

If the signal is classified as **speech**, the whitening level will be set to 0 – indicating no whitening – for

$$SFM_{tar} \leq SFM_{src} + 0.1 \qquad (5.2\text{-}65)$$

and to 1 – indicating mid-whitening – for

$$SFM_{tar} > SFM_{src} + 0.1 \tag{5.2-66}$$

To avoid redundancies, only this one level is transmitted in the bitstream.

5.2.2.2.3.6 IVAS IGF whitening thresholds

The EVS IGF whitening mapping and thresholds are described in clause 5.3.3.2.11.1.5 of [3]. For IVAS, different tables are used depending on whether an SCE or a CPE element is processed, see tables 5.2-8 and 5.2-9.

NOTE: The bitrate given to the IGF configuration depends on the IVAS format and is not always the same as the overall bitrate.

Table 5.2-8: Thresholds for whitening – IVAS SCE

Bitrate	Mode	nT	ThM	ThS
≤ 9.6 kbps	WB	2	0.36, 0.36	1.41, 1.41
≤ 9.6 kbps	SWB	4	0.89, 0.89, 0.80 ,0.80	1.25, 1.25, 1.19, 1.19
]9.6-13.2] kbps	SWB	6	1.05, 1.05, 1.10, 1.10, 1.05, 1.05	1.70, 1.70, 1.65, 1.65, 1.60, 1.50
]13.2-16.4] kbps	SWB	7	1.20, 1.20, 1.15, 1.10, 1.05, 1.00, 0.90	1.70, 1.70, 1.70, 1.70, 1.55, 1.45, 1.20
]16.4-24.4] kbps	SWB	8	1.20, 1.20, 1.15, 1.15, 1.10, 1.05, 1.00, 0.90	1.80, 1.80, 1.80, 1.80, 1.65, 1.55, 1.45, 1.20
]24.4-32.0] kbps	SWB	3	0.91, 0.85, 0.85	1.34, 1.35, 1.35
]32.0-48.0] kbps	SWB	1	1.15	1.19
]48.0-64.0] kbps	SWB	1	1.15	1.19
≤ 16.4 kbps	FB	9	1.20, 1.20, 1.15, 1.10, 1.05, 1.00, 0.90, 0.34, 0.34	1.70, 1.70, 1.70, 1.70, 1.55, 1.45, 1.20, 0.65, 0.65
]16.4-24.4] kbps	FB	10	1.20, 1.20, 1.15, 1.15, 1.10, 1.05, 1.00, 0.90, 0.34, 0.34	1.80, 1.80, 1.80, 1.80, 1.65, 1.55, 1.45, 1.20, 0.65, 0.65
]24.4-32.0] kbps	FB	4	0.78, 0.31, 0.34, 0.34	1.49, 1.38, 0.65, 0.65
]32.0-48.0] kbps	FB	1	0.80	1.0
]48.0-64.0] kbps	FB	1	0.80	1.0
]64.0-96.0] kbps	FB	1	0	2.82
]96.0-128.0] kbps	FB	1	0	2.82

Table 5.2-9: Thresholds for whitening – IVAS CPE

Bitrate	Mode	nT	ThM	ThS
≤ 9.6 kbps	WB	2	0.80, 0.75	1.50, 1.45
]9.6-13.2] kbps	WB	2	0.90, 0.85	1.60, 1.50
≤ 9.6 kbps	SWB	4	0.89, 0.89, 0.80 ,0.80	1.25, 1.25, 1.19, 1.19
]9.6-13.2] kbps	SWB	6	1.05, 1.05, 1.10, 1.10, 1.05, 1.05	1.70, 1.70, 1.65, 1.65, 1.60, 1.50
]13.2-16.4] kbps	SWB	7	1.20, 1.20, 1.15, 1.10, 1.05, 1.00, 0.90	1.70, 1.70, 1.70, 1.70, 1.55, 1.45, 1.20
]16.4-24.4] kbps	SWB	8	1.20, 1.20, 1.15, 1.15, 1.10, 1.05, 1.00, 0.90	1.80, 1.80, 1.80, 1.80, 1.65, 1.55, 1.45, 1.20
]24.4-32.0] kbps	SWB	3	0.91, 0.85, 0.85	1.34, 1.35, 1.35
]32.0-48.0] kbps	SWB	6	adaptive	adaptive
]48.0-64.0] kbps	SWB	4	1.00, 1.00, 1.20, 1.25	1.50, 1.50, 1.60, 1.60
]64.0-80.0] kbps	SWB	2	1.20, 1.25	1.60, 1.60
]80.0-96.0] kbps	SWB	1	1.15	1.19
≤ 16.4 kbps	FB	9	1.20, 1.20, 1.15, 1.10, 1.05, 1.00, 0.90, 0.34, 0.34	1.70, 1.70, 1.70, 1.70, 1.55, 1.45, 1.20, 0.65, 0.65
]16.4-24.4] kbps	FB	10	1.20, 1.20, 1.15, 1.15, 1.10, 1.05, 1.00, 0.90, 0.34, 0.34	1.80, 1.80, 1.80, 1.80, 1.65, 1.55, 1.45, 1.20, 0.65, 0.65
]24.4-32.0] kbps	FB	4	0.78, 0.31, 0.34, 0.34	1.49, 1.38, 0.65, 0.65
]32.0-48.0] kbps	FB	7	adaptive	adaptive
]48.0-64.0] kbps	FB	5	1.00, 1.00, 1.20, 1.25, 0.75	1.50, 1.50, 1.60, 1.60, 0.75
]64.0-80.0] kbps	FB	3	1.20, 1.25, 0.75	1.60, 1.60, 1.00
]80.0-96.0] kbps	FB	2	0.91, 0.85	1.34, 1.35
]96.0-128.0] kbps	FB	1	0	2.82

NOTE: For CPEs within bitrate interval]32.0-48] kbps, the whitening is not determined via fixed thresholds but calculated adaptively as described in clause 5.2.2.2.3.5.

5.2.2.2.3.7 IVAS IGF scale factor tables

The EVS IGF scale factor tables are described in 5.3.3.2.11.1.7, table 94 of [3]. For IVAS, different tables are used depending on whether an SCE or a CPE element is processed, see tables 5.2-10 and 5.2-11.

NOTE: The bitrate given to the IGF configuration depends on the IVAS format and is not always the same as the overall bitrate.

Table 5.2-10: Scale factor band offset table – IVAS SCE

Bitrate	Mode	Number of bands (nB)	Scale factor band offsets (t[0],t[1],…,t[nB])
≤ 9.6 kbps	WB	3	164, 186, 242, 320
≤ 9.6 kbps	SWB	4	200, 264, 344, 440, 566
]9.6-13.2] kbps	SWB	6	228, 264, 308, 360, 420, 488, 566
]13.2-16.4] kbps	SWB	7	256, 288, 328, 376, 432, 496, 576, 640
]16.4-24.4] kbps	SWB	8	256, 284, 320, 360, 404, 452, 508, 576, 640
]24.4-32.0] kbps	SWB	8	256, 284, 318, 358, 402, 450, 508, 576, 640
]32.0-48.0] kbps	SWB	3	512, 534, 576, 640
]48.0-64.0] kbps	SWB	3	512, 534, 576, 640
≤ 16.4 kbps	FB	9	256, 288, 328, 376, 432, 496, 576, 640, 720, 800
]16.4-24.4] kbps	FB	10	256, 284, 320, 360, 404, 452, 508, 576, 640, 720, 800
]24.4-32.0] kbps	FB	10	256, 284, 318, 358, 402, 450, 508, 576, 640, 720, 800
]32.0-48.0] kbps	FB	4	512, 584, 656, 728, 800
]48.0-64.0] kbps	FB	4	512, 584, 656, 728, 800
]64.0-96.0] kbps	FB	2	640, 720, 800
]96.0-128.0] kbps	FB	2	640, 720, 800

Table 5.2-11: Scale factor band offset table – IVAS CPE

Bitrate	Mode	Number of bands (nB)	Scale factor band offsets (t[0],t[1],…,t[nB])
≤ 9.6 kbps	WB	2	196, 248, 320
]9.6-13.2] kbps	WB	2	228, 268, 320
≤ 9.6 kbps	SWB	4	200, 264, 344, 440, 566
]9.6-13.2] kbps	SWB	6	228, 264, 308, 360, 420, 488, 566
]13.2-16.4] kbps	SWB	7	256, 288, 328, 376, 432, 496, 576, 640
]16.4-24.4] kbps	SWB	8	256, 284, 320, 360, 404, 452, 508, 576, 640
]24.4-32.0] kbps	SWB	8	256, 284, 318, 358, 402, 450, 508, 576, 640
]32.0-48.0] kbps	SWB	6	360, 392, 424, 464, 508, 560, 640
]48.0-64.0] kbps	SWB	7	400, 424, 448, 476, 508, 540, 576, 640
]64.0-80.0] kbps	SWB	4	464, 496, 532, 576, 640
]80.0-96.0] kbps	SWB	3	512, 536, 576, 640
≤ 16.4 kbps	FB	9	256, 288, 328, 376, 432, 496, 576, 640, 720, 800
]16.4-24.4] kbps	FB	10	256, 284, 320, 360, 404, 452, 508, 576, 640, 720, 800
]24.4-32.0] kbps	FB	10	256, 284, 318, 358, 402, 450, 508, 576, 640, 720, 800
]32.0-48.0] kbps	FB	8	360, 392, 424, 464, 508, 560, 640, 720, 800
]48.0-64.0] kbps	FB	9	400, 424, 448, 476, 508, 540, 576, 640, 720, 800
]64.0-80.0] kbps	FB	6	464, 496, 532, 576, 640, 720, 800
]80.0-96.0] kbps	FB	5	512, 536, 576, 640, 720, 800
]96.0-128.0] kbps	FB	2	640, 720, 800

5.2.2.2.3.8 IVAS IGF source band mapping

The EVS IGF source band mapping is described in clause 5.3.3.2.11.1.8 of [3]. For IVAS, different mappings are used depending on whether an SCE or a CPE element is processed, see new tables for minimal source bands (tables 5.2-12 and 5.2-13) and mapping functions (tables 5.2-14 and 5.2-15).

NOTE: The bitrate given to the IGF configuration depends on the IVAS format and is not always the same as the overall bitrate.

Table 5.2-12: IGF minimal source subband – IVAS SCE

Bitrate	mode	*minSb*
≤ 9.6 kbps	WB	30
≤ 9.6 kbps	SWB	32
]9.6-13.2] kbps	SWB	32
]13.2-16.4] kbps	SWB	32
]16.4-24.4] kbps	SWB	32
]24.4-32.0] kbps	SWB	32
]32.0-48.0] kbps	SWB	64
]48.0-64.0] kbps	SWB	64
≤ 16.4 kbps	FB	32
]16.4-24.4] kbps	FB	32
]24.4-32.0] kbps	FB	32
]32.0-48.0] kbps	FB	64
]48.0-64.0] kbps	FB	64
]64.0-96.0] kbps	FB	64
]96.0-128.0] kbps	FB	64

Table 5.2-13: IGF minimal source subband – IVAS CPE

Bitrate	mode	*minSb*
≤ 9.6 kbps	WB	32
]9.6-13.2] kbps	WB	32
≤ 9.6 kbps	SWB	32
]9.6-13.2] kbps	SWB	32
]13.2-16.4] kbps	SWB	32
]16.4-24.4] kbps	SWB	32
]24.4-32.0] kbps	SWB	32
]32.0-48.0] kbps	SWB	48
]48.0-64.0] kbps	SWB	48
]64.0-80.0] kbps	SWB	64
]80.0-96.0] kbps	SWB	64
≤ 16.4 kbps	FB	32
]16.4-24.4] kbps	FB	32
]24.4-32.0] kbps	FB	32
]32.0-48.0] kbps	FB	48
]48.0-64.0] kbps	FB	48
]64.0-80.0] kbps	FB	64
]80.0-96.0] kbps	FB	64
]96.0-128.0] kbps	FB	64

Table 5.2-14: Mapping functions for IVAS SCE

Bitrate	Mode	nT	mapping Function
≤ 9.6 kbps	WB	2	*m2a*
≤ 9.6 kbps	SWB	4	*m4b*
]9.6-13.2] kbps	SWB	6	*m2a*
]13.2-16.4] kbps	SWB	7	*m3b*
]16.4-24.4] kbps	SWB	8	*m3c*
]24.4-32.0] kbps	SWB	3	*m3c*
]32.0-48.0] kbps	SWB	1	*m1*
]48.0-64.0] kbps	SWB	1	*m1*
≤ 16.4 kbps	FB	9	*m4c*
]16.4-24.4] kbps	FB	10	*m4d*
]24.4-32.0] kbps	FB	4	*m4*
]32.0-48.0] kbps	FB	1	*m1*
]48.0-64.0] kbps	FB	1	*m1*
]64.0-96.0] kbps	FB	1	*m1*
]96.0-128.0] kbps	FB	1	*m1*

Table 5.2-15: Mapping functions for IVAS CPE

Bitrate	Mode	nT	mapping Function
≤ 9.6 kbps	WB	2	*m1*
]9.6-13.2] kbps	WB	2	*m1b*
≤ 9.6 kbps	SWB	4	*m4b*
]9.6-13.2] kbps	SWB	6	*m2a*
]13.2-16.4] kbps	SWB	7	*m3b*
]16.4-24.4] kbps	SWB	8	*m3c*
]24.4-32.0] kbps	SWB	3	*m3c*
]32.0-48.0] kbps	SWB	6	*m2c*
]48.0-64.0] kbps	SWB	4	*m2d*
]64.0-80.0] kbps	SWB	2	*m1c*
]80.0-96.0] kbps	SWB	1	*m1d*
≤ 16.4 kbps	FB	9	*m4c*
]16.4-24.4] kbps	FB	10	*m4d*
]24.4-32.0] kbps	FB	4	*m4*
]32.0-48.0] kbps	FB	7	*m4e*
]48.0-64.0] kbps	FB	5	*m3f*
]64.0-80.0] kbps	FB	3	*m2e*
]80.0-96.0] kbps	FB	2	*m2f*
]96.0-128.0] kbps	FB	1	*m1*

In the following the newly added mapping functions are given. For mappings reused from EVS, please refer to clause 5.3.3.2.11.1.8 of [3]. New tables for tile number and tile width are given in 5.2-16and 5.2-17.

NOTE: For some IVAS modes (e.g., SWB for]16.4-24.4] kbps range) the number of tiles was increased with tiles consisting of only SFB. In these cases, some tiles are often contiguous and not overlapping, so they can be grouped together in one case of the according mapping function. Accordingly, there the number of cases per mapping function does not correspond to the number of tiles.

The mapping function $m1b$ is defined with:

$$m1b \coloneqq minSB + tF(80, f) + \left(x - t(0)\right) \quad for \quad t(0) \le x < t(2) \qquad (5.2\text{-}67)$$

The mapping function $m1c$ is defined with:

$$m1c \coloneqq minSB + tF(212, f) + \left(x - t(0)\right) \quad for \quad t(0) \le x < t(4) \qquad (5.2\text{-}68)$$

The mapping function $m1d$ is defined with:

$$m1d \coloneqq minSB + tF(200, f) + \left(x - t(0)\right) \quad for \quad t(0) \le x < t(3) \qquad (5.2\text{-}69)$$

The mapping function $m2c$ is defined with:

$$m2c \coloneqq \begin{cases} minSb + tF(120, f) + \left(x - t(0)\right) & for \quad t(0) \le x < t(4) \\ minSb + tF(140, f) + \left(x - t(4)\right) & for \quad t(0) \le x < t(nB) \end{cases} \qquad (5.2\text{-}70)$$

The mapping function $m2d$ is defined with:

$$m2d \coloneqq \begin{cases} minSb + tF(80, f) + \left(x - t(0)\right) & for \quad t(0) \le x < t(4) \\ minSb + tF(144, f) + \left(x - t(4)\right) & for \quad t(0) \le x < t(nB) \end{cases} \qquad (5.2\text{-}71)$$

The mapping function $m2c$ is defined with:

$$m2e \coloneqq \begin{cases} minSb + tF(212, f) + \left(x - t(0)\right) & for \quad t(0) \le x < t(4) \\ minSb + tF(200, f) + \left(x - t(4)\right) & for \quad t(0) \le x < t(nB) \end{cases} \qquad (5.2\text{-}72)$$

The mapping function $m2f$ is defined with:

$$m2f := \begin{cases} minSb + tF(200,f) + (x - t(0)) & for \quad t(0) \le x < t(3) \\ minSb + tF(240,f) + (x - t(4)) & for \quad t(0) \le x < t(nB) \end{cases}$$

The mapping function $m3e$ is defined with:

$$m3e := \begin{cases} minSb + tF(120,f) + (x - t(0)) & for \quad t(0) \le x < t(4) \\ minSb + tF(140,f) + (x - t(4)) & for \quad t(0) \le x < t(6) \\ minSb + tF(140,f) + (x - t(6)) & for \quad t(0) \le x < t(nB) \end{cases} \quad (5.2\text{-}74)$$

The mapping function $m3f$ is defined with:

$$m3f := \begin{cases} minSb + tF(80,f) + (x - t(0)) & for \quad t(0) \le x < t(4) \\ minSb + tF(144,f) + (x - t(4)) & for \quad t(0) \le x < t(6) \\ minSb + tF(160,f) + (x - t(6)) & for \quad t(0) \le x < t(nB) \end{cases} \quad (5.2\text{-}75)$$

The mapping function $m4b$ is defined with:

$$m4b(x) := \begin{cases} minSb + (x - t(0)) & for \quad t(0) \le x < t(1) \\ minSb + tF(32,f) + (x - t(1)) & for \quad t(0) \le x < t(2) \\ minSb + tF(46,f) + (x - t(2)) & for \quad t(0) \le x < t(3) \\ minSb + tF(40,f) + (x - t(3)) & for \quad t(0) \le x < t(nB) \end{cases}$$

The mapping function $m4c$ is defined with:

$$m4c(x) := \begin{cases} minSb + (x - t(0)) & for \quad t(0) \le x < t(4) \\ minSb + tF(48,f) + (x - t(4)) & for \quad t(4) \le x < t(6) \\ minSb + tF(64,f) + (x - t(6)) & for \quad t(6) \le x < t(7) \\ minSb + (x - t(7)) & for \quad t(7) \le x < t(nB) \end{cases} \quad (5.2\text{-}77)$$

The mapping function $m4d$ is defined with:

$$m4d(x) := \begin{cases} minSb + (x - t(0)) & for \quad t(0) \le x < t(4) \\ minSb + tF(32,f) + (x - t(4)) & for \quad t(4) \le x < t(7) \\ minSb + tF(64,f) + (x - t(7)) & for \quad t(6) \le x < t(8) \\ minSb + (x - t(8)) & for \quad t(8) \le x < t(nB) \end{cases} \quad (5.2\text{-}78)$$

Table 5.2-16: Number of tiles *nT* and tile width *wT* – IVAS SCE

Bitrate	Mode	nT	wT
≤ 9.6 kbps	WB	2	*t(2)-t(0), t(nB)-t(2)*
≤ 9.6 kbps	SWB	4	*t(1)-t(0), t(2)-t(1), t(3)-t(2), t(nB)-t(3)*
]9.6-13.2] kbps	SWB	6	*t(1)-t(0), t(2)-t(1), t(3)-t(2), t(4)-t(3), t(5)-t(4), t(nB)-t(5)*
]13.2-16.4] kbps	SWB	7	*t(1)-t(0), t(2)-t(1), t(3)-t(2), t(4)-t(3), t(5)-t(4), t(6)-t(5), t(nB)-t(6)*
]16.4-24.4] kbps	SWB	8	*t(1)-t(0), t(2)-t(1), t(3)-t(2), t(4)-t(3), t(5)-t(4), t(6)-t(5), t(7)-t(6), t(nB)-t(7)*
]24.4-32.0] kbps	SWB	3	*t(4)-t(0), t(7)-t(4), t(nB)-t(7)*
]32.0-48.0] kbps	SWB	1	*t(nB)-t(0)*
]48.0-64.0] kbps	SWB	1	*t(nB)-t(0)*
≤ 16.4 kbps	FB	9	*t(1)-t(0), t(2)-t(1), t(3)-t(2), t(4)-t(3), t(5)-t(4), t(6)-t(5), t(7)-t(6), t(8)-t(7), t(nB)-t(9)*
]16.4-24.4] kbps	FB	10	*t(1)-t(0), t(2)-t(1), t(3)-t(2), t(4)-t(3), t(5)-t(4), t(6)-t(5), t(7)-t(6), t(8)-t(7), t(9)-t(8), t(nB)-t(9)*
]24.4-32.0] kbps	FB	4	*t(4)-t(0), t(6)-t(4), t(9)-t(6), t(nB)-t(9)*
]32.0-48.0] kbps	FB	1	*t(nB)-t(0)*
]48.0-64.0] kbps	FB	1	*t(nB)-t(0)*
]64.0-96.0] kbps	FB	1	*t(nB)-t(0)*
]96.0-128.0] kbps	FB	1	*t(nB)-t(0)*

Table 5.2-17: Number of tiles *nT* and tile width *wT* – IVAS CPE

Bitrate	Mode	nT	wT
≤ 9.6 kbps	WB	2	*t(1)-t(0), t(nB)-t(1)*
]9.6-13.2] kbps	WB	2	*t(1)-t(0), t(nB)-t(1)*
≤ 9.6 kbps	SWB	4	*t(1)-t(0), t(2)-t(1), t(3)-t(2), t(nB)-t(3)*
]9.6-13.2] kbps	SWB	6	*t(1)-t(0), t(2)-t(1), t(3)-t(2), t(4)-t(3), t(5)-t(4), t(nB)-t(5)*
]13.2-16.4] kbps	SWB	7	*t(1)-t(0), t(2)-t(1), t(3)-t(2), t(4)-t(3), t(5)-t(4), t(6)-t(5), t(nB)-t(6)*
]16.4-24.4] kbps	SWB	8	*t(1)-t(0), t(2)-t(1), t(3)-t(2), t(4)-t(3), t(5)-t(4), t(6)-t(5), t(7)-t(6), t(nB)-t(7)*
]24.4-32.0] kbps	SWB	3	*t(4)-t(0), t(7)-t(4), t(nB)-t(7)*
]32.0-48.0] kbps	SWB	6	*t(1)-t(0), t(2)-t(1), t(3)-t(2), t(4)-t(3), t(5)-t(4), t(nB)-t(5)*
]48.0-64.0] kbps	SWB	4	*t(2)-t(0), t(4)-t(2), t(6)-t(4), t(nB)-t(6)*
]64.0-80.0] kbps	SWB	2	*t(2)-t(0), t(nB)-t(2)*
]80.0-96.0] kbps	SWB	1	*t(nB)-t(0)*
≤ 16.4 kbps	FB	9	*t(1)-t(0), t(2)-t(1), t(3)-t(2), t(4)-t(3), t(5)-t(4), t(6)-t(5), t(7)-t(6), t(8)-t(7), t(nB)-t(9)*
]16.4-24.4] kbps	FB	10	*t(1)-t(0), t(2)-t(1), t(3)-t(2), t(4)-t(3), t(5)-t(4), t(6)-t(5), t(7)-t(6), t(8)-t(7), t(9)-t(8), t(nB)-t(9)*
]24.4-32.0] kbps	FB	4	*t(4)-t(0), t(6)-t(4), t(9)-t(6), t(nB)-t(9)*
]32.0-48.0] kbps	FB	7	*t(1)-t(0), t(2)-t(1), t(3)-t(2), t(4)-t(3), t(5)-t(4), t(6)-t(5), t(nB)-t(6)*
]48.0-64.0] kbps	FB	5	*t(2)-t(0), t(4)-t(2), t(6)-t(4), t(7)-t(6), t(nB)-t(7)*
]64.0-80.0] kbps	FB	3	*t(2)-t(0), t(4)-t(2), t(nB)-t(4)*
]80.0-96.0] kbps	FB	2	*t(3)-t(0), t(nB)-t(3)*
]96.0-128.0] kbps	FB	1	*t(nB)-t(0)*

5.2.2.2.4 Switching coding modes

5.2.2.2.4.1 Transitions between ACELP and HQ MDCT

The transitions between ACELP and HQ Core in IVAS operations has been harmonized to be more similar to the transitions between ACELP and TCX Core. When switching from HQ MDCT to ACELP, the transition is based on MC2 as described in 5.4.2.2. Similar to MC2, a synthesis at the ACELP sampling rate sr_{celp} is generated by performing an additional MDCT synthesis and overlap-add operation corresponding to [ref?] using the portion of the spectrum corresponding to sr_{celp}. This synthesis is stored in the ACELP excitation memory. Upon a switch from HQ MDCT to ACELP, the content of the excitation memory is filtered through the pre-emphasis filter and the decoded filter coefficients of the current ACELP frame to form the excitation memory for the current frame.

Editor's note: Enforcing of TC frame

Editor's note: Longer frame for HQ->ACELP transition

5.2.2.2.4.2 Switching from TCX/HQ to ACELP in DFT-based stereo

The time-domain bandwith extension (BWE) process is defined for ACELP mode in [3] Section 5.2.6.2. When IVAS is operating in ACELP coding mode, BWE buffer is updated with the high band down-mix signal, $x_{HI}(n)$ generated by a DFT synthesis. Meanwhile TCX/HQ coding mode operates on the full band down-mix signal $x(n)$ also generated by a DFT synthesis. For IVAS operations, a BWE process is defined to extract the target bands to update the BWE buffer.

To adapt the BWE for different target bands, an intermediate length of interpolation, N_{inter}, is determined by rescaling the length of the input frame N in the input sampling frequency, f_{in}. The upper limit of the target band is set to be the Nyquist frequency in the interpolated intermediate frame. For the target band with frequency limits of $[f_{lo}, f_{hi}]$, the intermediate length N_{inter} is:

$$N_{inter} = N\frac{f_{inter}}{f_{in}} \tag{5.2-79}$$

The intermediate sampling frequency f_{inter} is twice the upper frequency limit of the target band, f_{hi}:

$$f_{inter} = 2f_{hi} \tag{5.2-80}$$

The full band input $x(n)$ with length N is then linearly interpolated to the intermediate frame $x_{int}(n)$ with the length N_{int} following the equations:

$$x_{int}(n) = \begin{cases} x(0) + n_{frac}\big(x(1) - x(0)\big), & n_{frac} < 0 \\ x(N-1) + \big(n_{frac} - n_{floor}\big)\big(x(N-1) - x(N-2)\big), & n_{frac} > N-1 \\ x\big(n_{floor}\big) + \big(n_{frac} - n_{floor}\big)\Big(x\big(n_{floor} + 1\big) - x\big(n_{floor}\big)\Big), & otherwise \end{cases} \tag{5.2-81}$$

The interpolated sampling value at point n is $x_{int}(n)$, and n_{frac} is a fractional point where the source vector x is to be estimated. The first two cases of $x_{int}(n)$ handles the samples on the edges of the frame where the new sampling point is extrapolated from either the last or the first two samples of the frame. An index offset n_{offset} handles the case when $N_{int} > N$ and the first sample point in the stretched frame would be below $n = 0$ and the last sample point exceeding $n = N - 1$. The fractional point, n_{frac} is calculated using the displacement ϵ and index offset n_{offset}:

$$n_{floor} = \lfloor n_{frac} \rfloor \tag{5.2-82}$$

$$n_{frac} = n\frac{N}{N_{int}} + n_{offset} \tag{5.2-83}$$

$$n_{offset} = 0.5\frac{N}{N_{int}} - 0.5 + \epsilon \tag{5.2-84}$$

A displacement denoted as ϵ for n_{offset} is defined according to:

$$\epsilon = \begin{cases} -0.13, & N/N_{int} > 0.3 \\ 0, & otherwise \end{cases} \tag{5.2-85}$$

The spectrum of the resulting intermediate frame is reversed for $n = 0, 1, \ldots, N_{inter} - 1$ by changing the sign of every second sample:

$$x_{inter,rev}(n) = \begin{cases} -x_{inter}(n), & n \text{ is even} \\ x_{inter}(n), & n \text{ is odd} \end{cases} \tag{5.2-86}$$

A second linear interpolation is then performed on $x_{inter,rev}(n)$ to adjust the frame length to match the sampling frequency of the following low pass filtering and decimation operation. The cut-off frequency of the low-pass filter is selected to be in the middle of the spectrum. The decimator performs decimation by 2 which gives an output sampling frequency of half the input sampling frequency. When the frequency of decimation is f_{dec}, the decimation frame length N_{dec} is:

$$N_{dec} = N_{inter}\frac{f_{dec}}{f_{inter}} \tag{5.2-87}$$

The decimation process is explained in detail in Section 5.2.6.1.9 in [3]. In the frequency spectrum of the second interpolation output, $x_{dec}(n)$, the target band is aligned in the lower half of the spectrum. The low-pass filter and decimation extracts the target frequency band for the BWE target signal.

5.2.2.2.4.3 Bandwidth switching

A change of the audio bandwidth (BW) may happen as a consequence of a bitrate change or a coded audio bandwidth change. In EVS, when a change from WB to SWB occurs, or from SWB to WB, an audio bandwidth switching post-processing at the decoder is performed in order to improve the perceptual quality. A smoothing is applied for switching from WB to SWB, and a blind audio BWE is employed for switching from SWB to WB. More information can be found in Section 6.3.7 of Reference [3].

In IVAS, a different bandwidth switching (BWS) technique is employed. First, the new BWS algorithm is implemented at the encoder part of the IVAS codec in the core-coder (i.e. there is one BWS module per one transport channel). Then, the BWS algorithm in IVAS is implemented only for switching from a lower BW to a higher BW (for example from WB to SWB). In this direction, the switching is relatively fast (see Clause 5.2.2.1.5) and an abrupt high-frequency (HF) content change can be annoying. The IVAS BWS algorithm is thus designed to smooth such switching. On the other hand, no special treatment is implemented for switching from a higher BW to a lower BW because in this direction there is practically no important HF content in the spectrum, so the change of the spectrum content is not unnaturally abrupt and annoying.

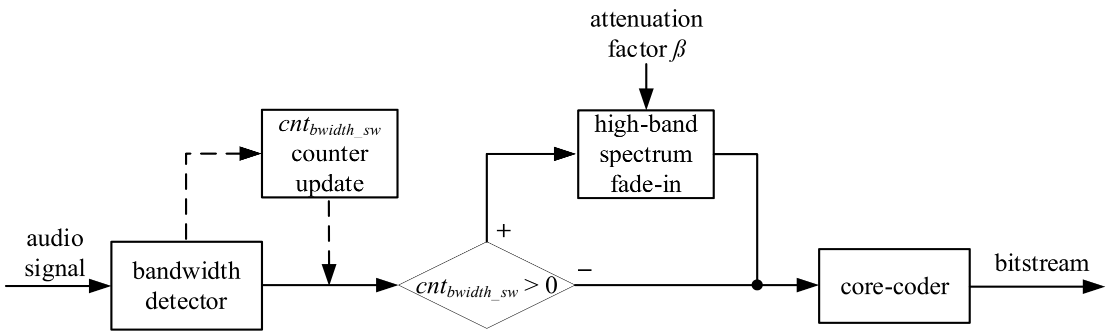

attenuation factor β
cnt_{bwidth_sw} counter update
high-band spectrum fade-in
audio signal
bandwidth detector
+
−
$cnt_{bwidth\_sw} > 0$
core-coder
bitstream

Figure 5.2-6: Schematic block diagram of the bandwidth switching algorithm.

A schematic block diagram of the encoder-side BWS algorithm is shown in Figure 5.2-6 and it comprises the final audio bandwidth decision operation as part of the BWD from Clause 5.2.2.1.5, a cnt_{bwidth_sw} counter updating operation, a comparison operation, and a high-band spectrum fade-in operation. The idea of the BWS algorithm is to smooth the perceptual impact of an audio BW switching already at the encoder part of the IVAS codec while removing the artifacts in the synthesis. The high-band (HB > 8 kHz) part of the spectrum is attenuated in several consecutive frames after a BWS instance as indicated by the final audio BW decision module from Clause 5.2.2.1.5. More specifically, a gain of the HB spectrum is faded-in in an attenuation operation from Figure 5.2-6 and thus smartly controlled in case of a BWS in order to avoid unpleasant artifacts. The attenuation is applied before the HB spectrum is quantized and encoded in the core-coder, so the smoothed BW transitions are already present in the transmitted bitstream and no further treatment is needed at the decoder. For example, in case of audio bandwidth switching from WB to SWB, the HB spectrum corresponding to frequencies above 8 kHz is smoothed before further processing.

The BWS algorithm then works as follows.

Referring to Figure 5.2-6, the calculator first updates a counter of frames cnt_{bwidth_sw} where audio bandwidth switching occurs and attenuation is applied at the end of the pre-processing for each IVAS transport channel based on the final BWD decision as follows. The calculator initially set the value of the counter of frames cnt_{bwidth_sw} to an initialization value of "0". When there is detected – as a response to a final BWD decision from the audio bandwidth decision module (Clause 5.2.2.1.5) – a BW change from a lower BW to a higher BW the value of the counter of frames is increased by 1. In the following frames, the counter is increased by 1 in every frame until it reaches its maximum value $B_{tran} = 5$ frames. When the counter reaches its maximum value B_{tran}, the counter cnt_{bwidth_sw} is then reset to 0 and a new detection of a BW switching can occur.

Next, when $cnt_{bwidth\_sw} > 0$, an attenuation is applied to the transport channel signal in frame n with an attenuation factor β_n defined as follows:

$$\beta_n = \frac{cnt_{bwidth\_sw}}{B_{tran}}, \;\; n = 0, \dots, B_{tran} - 1. \qquad (5.2\text{-}88)$$

Finally, the attenuation factor β is applied in the BWS transition period defined by B_{tran} frames depending on the coding mode as follows.

In TCX and HQ core-coder frames the high-band gain of the spectrum $X_M(k)$ of length L as defined in Section 5.3.2 of Reference [3] is controlled and the HB part of the spectrum $X_M(k)$, right after the time-to-frequency domain transformation, is updated (faded-in) in frame n using the following relation:

$$X'_M(k + L_{WB}) = \beta_n \cdot X_M(k + L_{WB}), \;\; k = 0, \dots, K - L_{WB} - 1, \qquad (5.2\text{-}89)$$

where L_{WB} is the length of spectrum corresponding to the WB audio bandwidth, i.e. $L_{WB} = 320$ samples in the frame length of 20 ms (normal HQ, or TCX20 frame), $L_{WB} = 80$ samples in transient frames, $L_{WB} = 160$ samples in TCX10 frames, and k is the transform domain sample index in the range $[0, K - L_{WB} - 1]$ where K is the length of the whole spectrum in particular transform sub-mode (normal, transient, TCX20, TCX10).

In ACELP core with time-domain BWE (TBE) frames the attenuation is applied with the attenuation factor β_n to the SWB gain shapes parameters of the HB part of the spectrum before these parameters are additionally processed. The temporal gain shapes parameters $g_s(j)$ are defined in Section 5.2.6.1.14.2 of Reference [3] and consist of four values. Thus, in IVAS, there applies in frame n the following attenuation:

$$g_s(j) = \beta_n \cdot g_s(j) \qquad (5.2\text{-}90)$$

where $j = 0, \ldots, 3$ is the gain shape number.

In ACELP core with frequency-domain BWE (FD-BWE) frames, the high-band gain of the transformed original input signal $X_M(k)$ of length L as defined in Section 5.2.6.2.1 of Reference [3] is controlled and the HB part of the MDCT spectrum is attenuated in frame n using the following relation:

$$\acute{X}_M(k + L_{WB}) = \beta_n \cdot X_M(k + L_{WB}), \ \ k = 0, \ldots, K - L_{WB} - 1. \tag{5.2-91}$$

Note that NB coding is not considered in IVAS and SWB to FB switching is not treated as its subjective impact is negligible. However, the principles above are used to cover the other BWS scenarios.

The attenuated sound signal is finally encoded in the core-coder (SCE/CPE/MCT). If the counter cnt_{bwidth_sw} is not larger than 0, then the sound signal is encoded in the core-coder without attenuation.

5.2.2.2.5 DTX/CNG operation

Editor's note: [CNG updates]

5.2.2.2.5.1 Efficient first stage for multistage VQ for FD-CNG

This clause describes the enhanced first stage operation of the FD-CNG MSVQ which is employed in Object-based audio (ISM), Stereo audio and Scene-based Audio (SBA) for quantization of spectral shape for each given FD-CNG base SID-frame type. The subsequent stages of the FD-CNG MSVQ are following the steps as described in EVS [3] clause 5.6.3.5.

The first stage of the SID MSVQ encoder first obtains an FD-CNG target vector $N_{FD-CNG,DCT-II}^{dB}$, in the DCT-II (M=24) domain by applying a normalized DCT- type II transformation to the MSVQ target vector N_{FD-CNG}^{dB} (corresponding to $N_{L,FD-CNG}^{dB}$, $N_{R,FD-CNG}^{dB}$, $N_{M,FD-CNG}^{dB}$, $N_{S,FD-CNG}^{dB}$) in clause 5.3.3.5.2.4)*, (References add: to $N_{SBA,FD-CNG}^{dB}$ in clause SBA*

*clause SBA*_DTX-CNG_*, to N_{FD-CNG}^{dB} in clause ISM*_DTX-CNG_ *)* and to $\bar{N}_{FD-CNG}^{dB}$ in clause of the generic MSVQ description EVS [3]), of length $\boldsymbol{M} = 24$.

The optimal output of the first MSVQ stages is the list of $\boldsymbol{N_{best}}$ best candidate indexes and their mean square errors in relation to the FD-CNG domain input signal. The total number of entries in the stage one codebook size is denoted $\boldsymbol{N_{st1}}$ and the $\boldsymbol{N_{best}}$ for FDCNG is 8 for FD-CNG SID-frame MSVQ quantization. Each candidate points to an $idx \in [0 \ldots N_{st1} - 1]$ in the first stage global codebook.

To achieve an improved VQ efficiency in terms of both cycles and ROM-storage the FD-CNG VQ search is performed in a slightly sub-optimal fashion followed by a circular neighbour analysis post-optimization step. The 1st stage codebook is compactly stored in $\boldsymbol{N_{seg}}$ segments with different truncation lengths in the DCT-II (M=24) domain. The employed truncation lengths allow for different degrees of high frequency content between segments. Here $\boldsymbol{N_{seg}}$ is four and the employed truncation lengths are $\boldsymbol{N_{trunc,segm} = \{8,12,16,22\}}$ for segment $segm \in [0 \, .. \, N_{seg} - 1]$. A set of $\boldsymbol{N_{best}}$ initial stage one candidates $\boldsymbol{I_{cand}}$ are established through employing $\boldsymbol{N_{seg}}$ pair-wise inner search loops with $N_{FD-CNG,DCT-II}^{dB}$ as target, where each segment $\boldsymbol{segm}$ provides two initial candidates. Further, during the pairwise inner loop search, the mean square error of every possible codebook entry $\boldsymbol{i} \in [\boldsymbol{0} \ldots \boldsymbol{N_{st1}} - \boldsymbol{1}]$, is stored in an auxiliary vector $\boldsymbol{st1MSE_i}$.

To allow for low ROM storage and low inner loops cycle of the VQ, the first stage FD-CNG vectors are stored in the format $\boldsymbol{m_{i,segm,col}} \cdot 2^{\boldsymbol{exp_{segm,col}}}$, where the mantissa $\boldsymbol{m_{i,segm,col}}$ consists of 8 bits and the exponent (an shift factor) $\boldsymbol{exp_{segm,col}}$ is an integer. Two mantissas for a column $col \in [0 \ldots N_{trunc,segm}]$ may then be efficiently fetched from ROM as a single 16-bit word and shifted with a common integer shift factor of $\boldsymbol{exp_{segm,col}}$.

In a post-optimization step, the eight initial candidates $\boldsymbol{I_{cand}}$ (with assistance of the auxiliary vector $\boldsymbol{st1MSE}$ with MSE values) are analysed further by evaluating a set of global neighbours to the initial candidates. The post-optimization first copies the indexes of the eight initial candidates in $\boldsymbol{I_{cand}}$ as starting points in the final candidate list $\boldsymbol{I_{final}}$. An off-line neighbour list of closely located neighbours in the MSE sense with respect to each index $\boldsymbol{i}$ are stored as a circular list $\boldsymbol{L_{neighbour,i}}$ of size $\boldsymbol{N_{st1}}$, containing close neighbour indexes to each index $\boldsymbol{i}$.

If any neighbour to an index in $\boldsymbol{I_{cand}}$ within a neighbour cardinal distance of two (in both forward and reverse directions) in $\boldsymbol{L_{neighbour,i}}$ is found to be better in an MSE sense than the current worst candidate in $\boldsymbol{I_{final}}$, then the worst position in $\boldsymbol{I_{final}}$ list is replaced with the analysed neighbour.

The remaining stages in the FD-CNG-MSVQ are operating in the non-DCT-transformed FD-CNG frequency band signal domain, thus the $\boldsymbol{N_{best}}$ codebook entries represented by the indexes the list $\boldsymbol{I_{final}}$ are transformed back to the input domain using the inverse DCT-II(M=24) and then provided to the subsequent MSVQ stage (see clause EVS [3]). The list of 2nd stage input candidate vectors in the frequency band domain is provided to the 2nd stage is denoted as: $V_1(idx, i), idx \in I_{final}$ and $i \in [0 \ldots N_{best}]$.

The required $\boldsymbol{N_{best}}$ MSE's for each of the candidates in $\boldsymbol{I_{\text{final}}}$ do not need to be recomputed prior to the next stage as the employed DCT-II is an orthogonal transform.

5.2.2.2.5.2 Wideband pre-processing in the first stage FD-CNG MSVQ

For wideband input signals, the MSVQ FD-CNG target vector has a reduced dimension of M=21. To be able to maintain the efficient and compact storage and search, in case of WB input the M=21 length signal is extended by extrapolation in the DCT-II(M-24) domain, to a length of 24. The first stage search is then performed as described in clause 5.2.2.2.5.1 with the exception that the output vectors are truncated to M=21 and the MSEs are updated to only include error contribution from the first 21 FD-CNG coefficients.

5.2.3 Common audio coding tools

This Clause describes common audio coding tools used for encoding the so-called transport channels. The individual tools can be used simultaneously multiple times similarly as a combination of different tools can be used simultaneously to encode different numbers of transport channels. The exact set-up depends on the actual IVAS format, IVAS total bitrate, and its actual mode and it is described in particular IVAS format related Clauses.

5.2.3.1 Single Channel Element (SCE)

The Single Channel Element (SCE) is a coding tool that encodes one transport channel. It is based on one core-coder module. Its block diagram is shown in Figure 5.2-7.

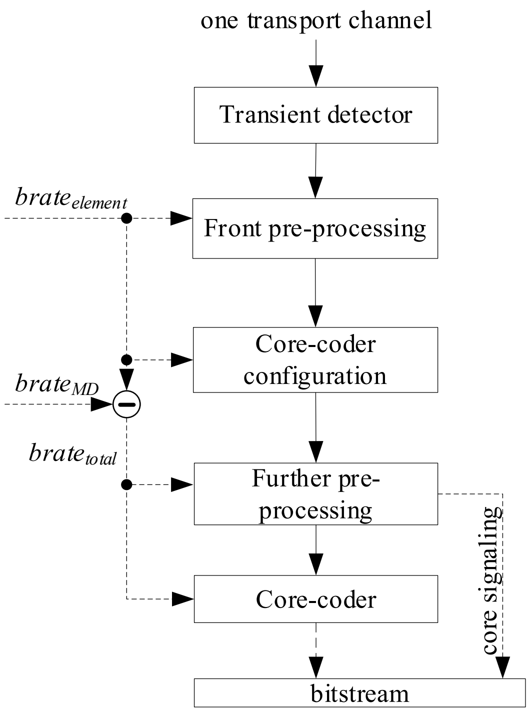


Figure 5.2-7: Block diagram of the Single Channel Element (SCE) encoder.

As seen from Figure 5.2-7, the SCE encoder consists of the following modules:

(a) Temporal transient detector (Clause 5.2.2.1.6) enables detection and therefore proper processing and encoding of transients in the transform-domain core-coder modules (TCX core, HQ core, FD-BWE).

(b) Front pre-processing (Clause 5.2.2.1). Note that it depends on the element bitrate, $brate_{element}$, which is constant at one IVAS total bitrate, $brate_{IVAS}$.

(c) Core-coder configuration module: high-level parameters like core-coder internal sampling-rate, core-coder nominal bitrate, IGF presence etc. are set. Note that it depends on the element bitrate, $brate_{element}$, which is constant at one IVAS total bitrate, $brate_{IVAS}$, and ensures that there is no frequent switching between different core-coder set-ups.

(d) Further pre-processing (Clause 5.2.2.2.1).

(e) One variable bitrate core-coder (Clause 5.2.2.2).

Note that the further pre-processing module (d) and the core-coder module (e) depend on the core-coder total bitrate $brate_{total}$ which is a difference between the element bitrate $brate_{element}$ and bitrate related to encode the metadata $brate_{MD}$ (if present), i.e.

$$brate_{total} = brate_{element} - brate_{MD}. \tag{5.2-92}$$

5.2.3.2 Channel Pair Element (CPE)

The Channel Pair Element (CPE) is a coding tool that encodes two transport channels. It is based on one or two core-coder module(s) supplemented by stereo coding tools. Its block diagram is shown in Figure 5.2-8.

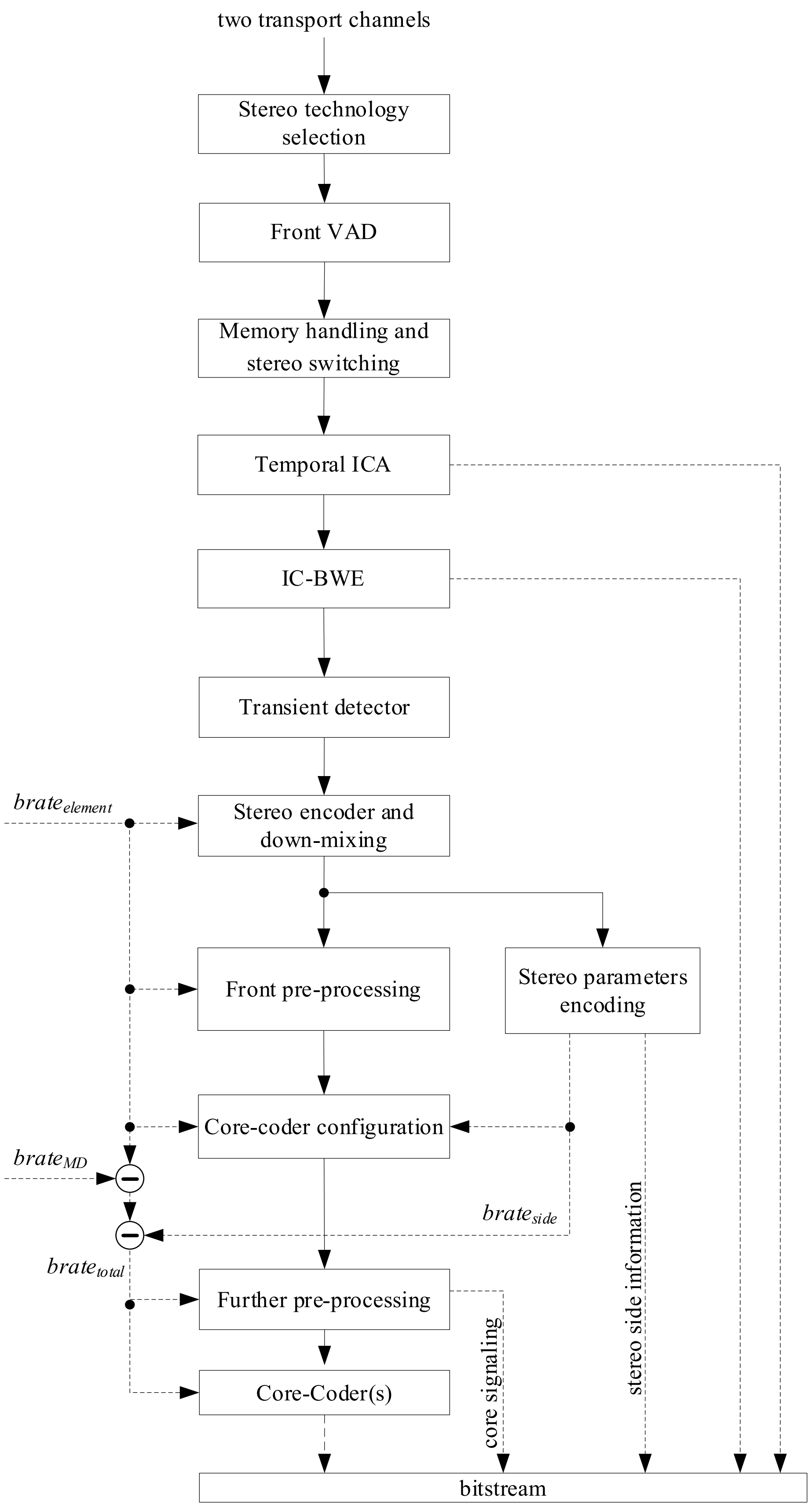


Figure 5.2-8: Block diagram of the Channel Pair Element (CPE) encoder.

As seen from Figure 5.2-8, the CPE encoder consists of the following modules:

(a) Stereo technology selection module controls the selection between DFT stereo, TD stereo and MDCT stereo coding modes and it is based on the element bitrate, IVAS format, and information from the stereo classifier (Clause 5.3.3.2.4).

(b) Front VAD module (Clause 5.3.3.2.3.3) runs on both transport channels and its output is used in the DTX operation.

(c) Memory handling and stereo switching (Clause 5.3.3.4) controls the switching between DFT stereo, TD stereo and MDCT stereo coding modes and comprises dynamic allocation/deallocation of static memory of stereo modes data structures depending on the current stereo mode. Also, the TD stereo mode is selected in this module.

(d) Temporal Inter-Channel Alignment (ICA) module to time-align the two transport channels (Clause 5.3.3.2.3.1).

(e) Inter-Channel BWE (IC-BWE) module (Clause 5.3.3.2.3.2) encode high-frequencies in DFT stereo and TD stereo modes.

(f) Temporal transient detector (Clause 5.2.2.1.6), one per transport channel, enables detection and therefore proper processing and encoding of transients in the transform-domain core-coder modules (TCX core, HQ core, FD-BWE).

(g) Stereo processing and downmixing module in case of DFT stereo (Clause 5.3.3.2.2) or TD stereo (Clause 5.3.3.2.1) while the stereo parameters are encoded in the stereo parameters encoding module.

(h) Front pre-processing (Clause 5.2.2.1) performed on one or two downmixed channels. Note that it depends on the element bitrate, $brate_{element}$, which is constant at one IVAS total bitrate, $brate_{IVAS}$.

(i) Core-coder configuration, one per downmixed channel: high-level parameters like core-coder internal sampling-rate, core-coder nominal bitrate, IGF presence etc. are set. Note that it depends on the element bitrate, $brate_{element}$, which is constant at one IVAS total bitrate, $brate_{IVAS}$, and ensures that there is no frequent switching between different core-coder set-ups.

(j) Further pre-processing (Clause 5.2.2.2.1) performed on one or two downmixed channels.

(k) One or two variable bitrate core-coder(s) (Clause 5.2.2.2) performed sequentially on one or two downmixed channels.

(l) Encoding of stereo parameters module depending on the current stereo mode being DFT stereo (Clause 5.3.3.2.2) or TD stereo (Clause 5.3.3.2.1).

Note that the further pre-processing module (j) and the core-coder module (k) depend on the core-coder total bitrate $brate_{total}[n], n = 1, 2$. The sum of total bitrates of both downmixed channels is then a difference between the element bitrate $brate_{element}$ and bitrates related to encode the metadata $brate_{MD}$ (if present) and stereo side information $brate_{side}$ (if present), i.e.

$$brate_{total}[0] + brate_{total}[1] = brate_{element} - brate_{MD} - brate_{side}. \qquad (5.2\text{-}93)$$

5.2.3.3 Multichannel Coding Tool (MCT)

5.2.3.3.1 Summary of the system

In this implementation, the codec uses the flexibility of signal adaptive joint coding of arbitrary channels and reuses the concepts of MDCT-based stereo joint coding described in [3]. These are:

- Use of perceptually whitened signals for further coding.
- Use of ILD parameters of arbitrary channels to efficiently code panned sources
- Flexible bit distribution among the processed channels based on the energy.

The codec uses Spectral Noise Shaping (SNS) to perceptually whiten the signal. The algorithm further normalizes the SNS-whitened spectrum towards the mean energy level using ILD parameters. Channel pairs for joint coding are selected in an adaptive manner, where the stereo coding consist of a band-wise M/S vs L/R decision. The band-wise M/S decision is based on the estimated bitrate in each band when coded in the L/R and in the M/S mode as described in 5.3.3.3.4.5. Bitrate distribution among the band-wise M/S processed channels is based on the energy.

5.2.3.3.2 Encoder single channel processing up to whitened spectrum

Each single channel $s_i(t)$ is analyzed and transformed to a whitened MDCT-domain spectrum $S_i(k)$ following the processing steps as shown in the block diagram of Figure 5.2-9, as is done for the MDCT-based stereo input format described in 5.3.3.3.3.

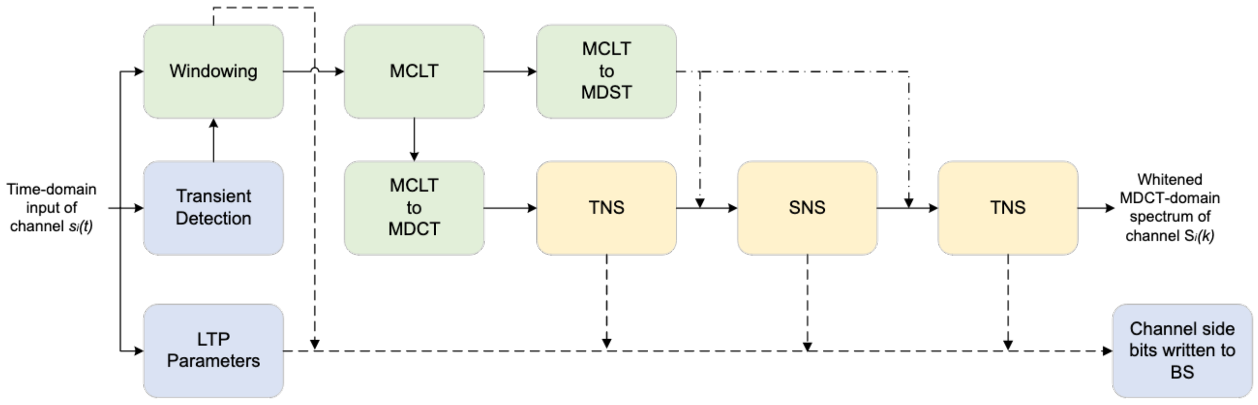

Windowing
MCLT
MCLT to MDST
Time-domain input of channel $s_i(t)$
Transient Detection
MCLT to MDCT
TNS
SNS
TNS
Whitened MDCT-domain spectrum of channel $S_i(k)$
LTP Parameters
Channel side bits written to BS

Figure 5.2-9: Block diagram of whitening processing for each channel

The processing blocks of the time-domain transient detector, windowing, MDCT, MDST and OLA are described in [3] clause 5.3.2. MDCT and MDST form modulated complex lapped transform (MCLT); "MCLT to MDCT" represents taking just the MDCT part of the MCLT and discarding MDST and vice versa.

Temporal Noise Shaping (TNS) is done similar as described in [3] clause 5.3.3.2.2 with the addition that the order of the TNS and the spectral noise shaping (SNS) is adaptive. The existence of the 2 TNS boxes in the figures is to be understood as the possibility to change the order of the SNS and the TNS. The decision of the order of the TNS and the SNS is described in the MDCT-based stereo clause 5.3.3.3.3.5.2

Frequency domain noise shaping (SNS) and the calculation of SNS parameters are described in clause 5.3.3.3.3.6..

5.2.3.3.3 Joint channel Encoding System Description

In the described system, after each channel is transformed to the whitened MDCT domain, signal-adaptive exploitation of varying similarities between arbitrary channels for joint coding is applied. From this procedure, the respective *channel-pair blocks* are detected and chosen to be jointly coded using a band-wise M/S transform as is done for stereo signals with the MDCT-based stereo approach as described in detail in clauses 5.3.3.3.4.5-5.3.3.3.4.7.

An overview of the encoding system is given in Figure 5.2-11. For simplicity block arrows represent single channel processing (i.e. the processing block is applied to each channel). Block "MDCT-domain analysis" is represented in detail in Figure 5.2-9.

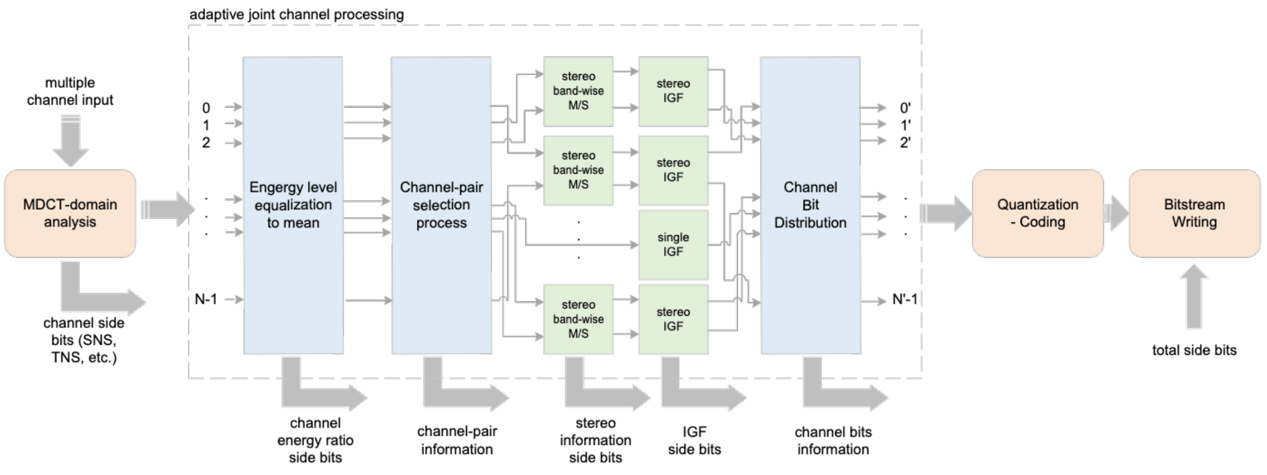


Figure 5.2-10: Block diagram of MCT and the adaptive joint channel processing

In the following paragraphs, the individual steps of the algorithm applied per frame are described in detail. A data flow graph of the algorithm described is given in Figure 5.2-11. For simplification in the graph, the cross-correlation vector is called “CC vector” and “CP” represents channel-pairs.

It should be noted, in the initial configuration of the system, there is a channel mask indicating for which channels the multi-channel joint coding tool is active. Therefore, for input where LFE channels are present, they are not considered in the processing steps of the tool.

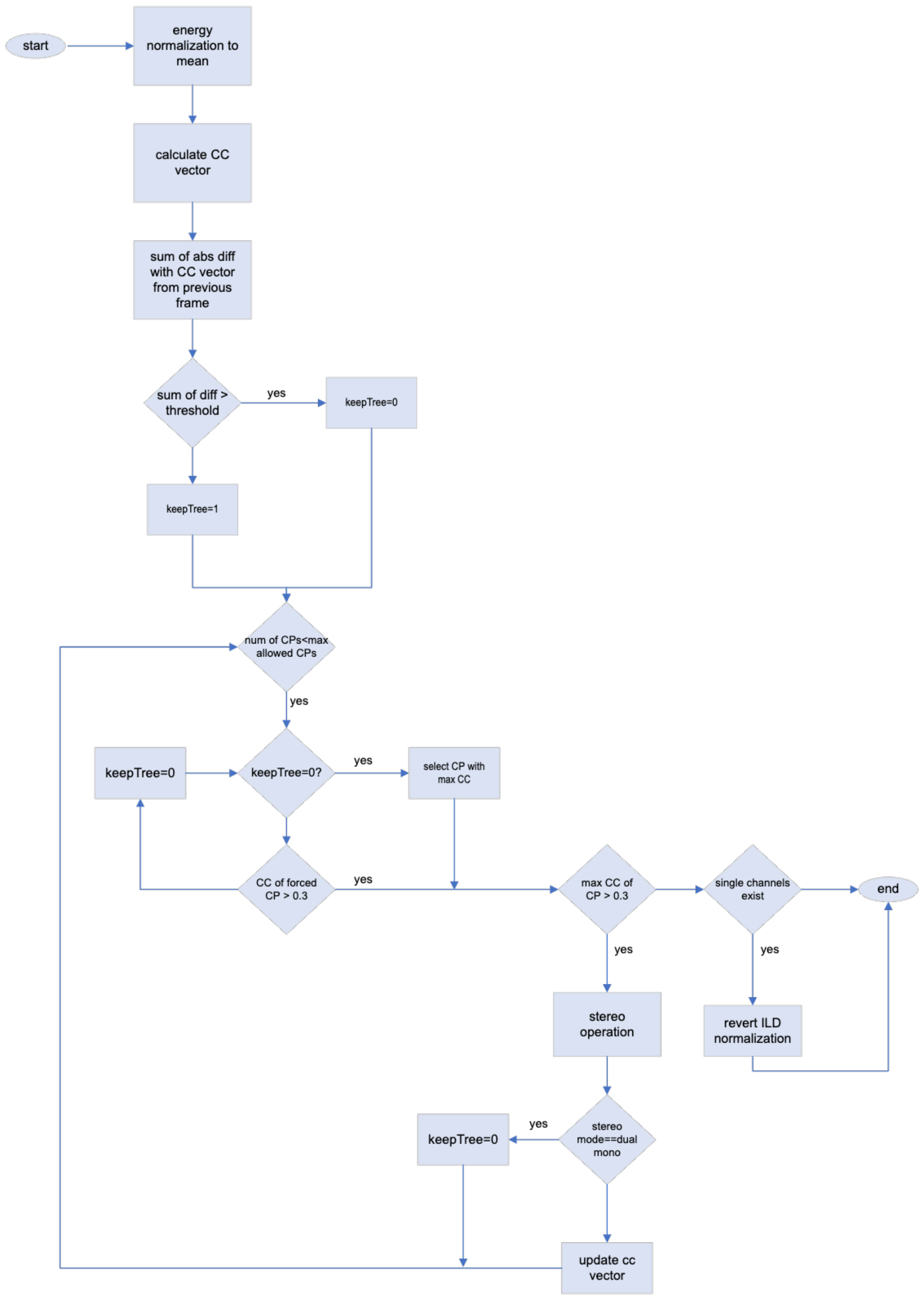


Figure 5.2-11: Data flow graph of MCT algorithm

5.2.3.3.3.1 Energy normalization of all channels towards mean energy.

An M/S transform is not efficient if ILD exists, that is if channels are panned. We avoid this problem by normalizing the amplitude of the perceptually whitened spectra of all channels to a mean energy level $\bar{E}$.

- Calculate energy E_i for each channel $i = 0, \dots, N$, where N is the total number of channels.

$$E_i = \sqrt{\sum_{k=0}^{L_{frame}} S^{MDCT}(k)^2} \quad (5.2\text{-}94)$$

where L_{frame} is the total number of spectral coefficients of the current frame;

- calculate mean energy:

$$\bar{E} = \frac{\sum_{i=0}^{N} E_i}{N} \quad (5.2\text{-}95)$$

- normalize the spectrum of each channel towards mean energy $\bar{E}$.

  If $E_i > \bar{E}$ (downscaling)

$$\alpha_i = \frac{\bar{E}}{E_i} \quad (5.2\text{-}96)$$

where a_i is the scaling ratio.

The scaling ratio is uniformly quantized and sent to the decoder as side information bits, in the same way as is done for the ILD normalization in MDCT-based stereo and Equation XX.

$$\widehat{ILD}(i) = \max(1, \min(ILD_{RANGE} - 1, \lfloor (ILD_{RANGE} * a_i + 0.5) \rfloor) \quad (5.2\text{-}97)$$

where $ILD_{RANGE} = 2^{ILD_{bits}}$ and $ILD_{bits} = 4$;

then the quantized scaling ratio with which the spectrum is finally scaled is given by

$$\hat{\alpha}_i = \frac{\widehat{ILD}(i)}{ILD_{RANGE}} \quad (5.2\text{-}98)$$

if $E_i < \bar{E}$ (upscaling)

$$\alpha_i = \frac{E_i}{\bar{E}} \quad (5.2\text{-}99)$$

and

$$\hat{\alpha}_i = \frac{ILD_{RANGE}}{\widehat{ILD}(i)} \quad (5.2\text{-}100)$$

where $\widehat{ILD}(i)$ is calculated as in Equation (5.2-97).

To distinguish whether we have downscaling/upscaling at decoder and in order to revert the normalization, besides the $\widehat{ILD}$ value for each channel, a 1-bit flag (0=downscaling/1=upscaling) is written in the bit stream as side information.

5.2.3.3.3.2 Calculation of normalized inter-channel cross-correlation values for all possible channel-pairs

In this step, in order to decide and select which channel pair has the highest degree of similarities and therefore is suitable to be selected as a pair for stereo joint coding, the inter-channel normalized cross-correlation value for each possible combination of channel pairs is calculated. The normalized cross-correlation value for each channel pair is given by the cross-spectrum as follows:

$$\tilde{r}(i_1, i_2) = \frac{r_{X_{i_1}X_{i_2}}}{\sqrt{r_{X_{i_1}X_{i_1}} r_{X_{i_2}X_{i_2}}}} \quad (5.2\text{-}101)$$

where

$$r_{X_{i_1}X_{i_2}} = \sum_{k=0}^{L_{frame}} \bar{S}_{i_1}^{MDCT}(\mathrm{k}) * \bar{\mathrm{S}}_{\mathrm{i}_2}^{\mathrm{MDCT}}(\mathrm{k}) \qquad (5.2\text{-}102)$$

and $i_1 = 0, \dots, N-1$ and $i_2 = i_1, \dots, N-1$;

$L_{f\mathrm{rame}}$ is the total number of spectral coefficients per frame;

$\bar{S}_{i_1}^{MDCT}$ and $\bar{S}_{i_2}^{MDCT}$ are the respective normalized MDCT spectra of the channel-pair i_1, i_2under consideration.

The normalized cross-correlation values for each channel pair are stored in the cross-correlation vector

$$CC = [\tilde{r}_0, \tilde{r}_1, \dots, \tilde{r}_{P-1}] \qquad (5.2\text{-}103)$$

where $P$is the maximum number of possible pairs and can be calculated by:

$$\mathrm{P} = \big(N * (N-1)\big)/2 \qquad (5.2\text{-}104)$$

As seen in Figure 5.2-9, depending on the transient detector we can have different block sizes (TCX10 or TCX20 window block sizes). Therefore, the inter-channel cross-correlation is calculated given that the spectral resolution for both channels is the same. If otherwise, then the value is set to 0, thus ensuring that no such channel pair is selected for joint coding.

An indexing scheme to uniquely represent each channel pair is used. An example of such a scheme for indexing six input channels is shown in Table 5.2-18.

Table 5.2-18: Indexing scheme of channel pairs

Channel Index	**0**	**1**	**2**	**3**	**4**	**5**
0	▨	0	1	2	3	4
1	▨	▨	5	6	7	8
2	▨	▨	▨	9	10	11
3	▨	▨	▨	▨	12	13
4	▨	▨	▨	▨	▨	14
5	▨	▨	▨	▨	▨	▨

To create the indexing scheme the logic in the following pseudo-code may be used:

```
For a given selected channel pair with channel indices chIdx1, chIdx2 respectively:

pairIdx = 0;

for ( ch2 = 1; ch2 < nChannels; ch2++ )
{
   for ( ch1 = 0; ch1 < ch2; ch1++ )
   {
      if ( ch1 == chIdx1 && ch2 == chIdx2 )
      {
         return pairIdx;
      }
      else
      {
         pairIdx++;
      }
   }
}
```

The same indexing scheme is held throughout the algorithm as is used also to signal channel pairs to the decoder. The number of bits needed for signalling one channel-pair amount to

$$bits_{idx} = \lfloor \log_2(P-1) \rfloor + 1 \qquad (5.2\text{-}105)$$

5.2.3.3.3.3 Channel-pair selection and jointly coded stereo processing

After calculating the cross-correlation vector, the first channel-pair to be considered for joint-coding is with the highest normalized cross-correlation value and higher than a minimum value threshold of 0.3.

$$\mathrm{idx}_{\mathrm{max}} = arg \max_{i|CC[i]>0.3} CC[i], \quad i = 0,1, \dots, P-1 \qquad (5.2\text{-}106)$$

The selected pair of channels serve as input to the MDCT stereo encoding procedure, namely a band-wise M/S transform. For each spectral band, the decision whether the channels will be coded using M/S or discrete L/R coding depends on the estimated bitrate for each case. The coding method that is less demanding in terms of bits is selected. This procedure is described in detail in subclause 5.3.3.3.4.5.

The output of this process results to an updated spectrum for each of the channels of the selected channel-pair as described in clause 5.3.3.3.4.7. Also, information that need to be shared with the decoder (side information) regarding this channel-pair are created, i.e., which stereo mode is selected (Full M/S, dual-mono, or band-wise M/S) and if band-wise M/S is the mode selected the respective mask of indicating whether M/S coding is chosen (1) or L/R (0).

After the stereo processing of a selected channel-pair block the cross-correlation vector CC is updated accordingly. For the selection of channel pairs, there is no "cascading" allowed. That means, the output signals of the stereo processing of a channel-pair block cannot be reused as an input of another channel pair block combination. That is ensured, by while updating the cross-correlation vector, the values of the normalized cross-correlation for all channel combinations with the channels of a previous channel-pair block, are set to 0. Therefore, it is not possible to select a new channel-pair for which one of the channels was already part of an existing channel pair.

The process continues iteratively until either there are no more possible channel combinations for which the condition of Equation (5.2-106) applies or the maximum number of possible channel pairs $CP_{max} = \lfloor N/2 \rfloor$ is met.

It should be noted that there may be cases where the stereo operation of a selected channel-pair does not alter the spectra of the channels. That happens when the M/S decision algorithm decides the stereo encoding mode is dual-mono. This can happen especially in cases where the normalized cross-correlation value $\tilde{r}(i_1, i_2)$ is close to the minimum threshold. In this case, the arbitrary channels involved are not considered a channel-pair block anymore as they will be coded separately. Also, updating the cross-correlation vector will have no effect.

To continue with the process, the channel-pair with the next highest value is considered. The steps in this case continue as described above. The whole process is described in the data flow graph of Figure 5.2-11.

5.2.3.3.3.4 Retain channel pair selection (stereo tree) of previous frame

In many cases the normalized cross-correlation values of arbitrary channel-pairs can be close and therefore the selection can switch often between this close values. That may cause frequent channel-pair tree switching, which may result to audible instabilities to the output system. Therefore, it is opted to use a stabilization mechanism, where a new set of channel pairs is selected only when there is a significant change to the signal and the similarities between arbitrary channels change. To detect this, the cross-correlation vector of the current frame is compared with the vector of the previous frame and when the difference is larger than a certain threshold then the selection of new channel pairs is allowed.

The variation in time of the cross-correlation vector is calculated as follows:

$$\mathrm{C}C_{diff} = \sum_{i=0}^{P} \left| CC[i] - CC^{[-1]}[i] \right| \qquad (5.2\text{-}107)$$

where P is the maximum number of possible combinations defined in Equation (5.2-104)

If

$$CC_{diff} > 0.15 \cdot P \qquad (5.2\text{-}108)$$

then the selection of new channel-pairs to be jointly coded, as described in the previous subclause 5.2.3.3.3.3, is allowed. Otherwise, the differences are small, then the same channel-pair tree as in the previous frame is used. For each given channel-pair, the band-wise M/S operation is applied as previously described. If, however, the normalized cross-correlation value of the given channel-pair does not exceed the threshold of 0.3 then the selection of new channel pairs creating a new tree is initiated.

5.2.3.3.3.5 Revert energy of single channels

After the termination of the iteration process for the channel pair selection there may be channels that are not part of any channel-pair block and therefore, are coded separately. For those channels the initial normalization of the energy towards the mean energy is reverted to their original energy. Depending on the flag signalling upscaling or downscaling the energy of these channels are reverted using the inverse of the quantized scaling ratio $\frac{1}{\hat{a}_{\mathrm{i}}}$ defined in clause

5.2.3.3.3.1. Additionally, $\widehat{ILD}(i) = 0$, signalling implicitly to the decoder that the i -th channel was coded separately and not as part of a channel-pair block.

5.2.3.3.4 IGF for multi-channel processing

Regarding IGF analysis, in case of stereo channel pairs an additional joint stereo processing is applied, as is thoroughly described in 5.3.3.3.6.2. This is necessary, because for a certain destination range in the IGF spectrum the signal can be a highly correlated panned sound source. In case the source regions chosen for this particular region is not well correlated, although the energies are matched for the destination regions, the spatial image can suffer due to the uncorrelated source regions.

Therefore, for each channel pair stereo IGF is applied if the stereo mode of the core region is different to the stereo mode of the IGF region or if the stereo mode of the core is flagged as band-wise M/S. If these conditions do not apply, then single channel IGF analysis is performed. If there are single channels, not coded jointly in a channel-pair, then they also undergo a single channel IGF analysis.

5.2.3.3.5 Bitrate distribution

After the process of joint channel-pair stereo processing, each channel is quantized and coded separately by an entropy coder. Therefore, for each channel the available number of bits should be given. In this step, the total available bits are distributed to each channel using the energies of the processed channels.

The energy of each channel, the calculation of which is described above in Equation (5.2-94) for the normalization step, is recalculated as the spectrum for each channel may have changed due to the joint stereo processing. The new energies are denoted $\tilde{E}_i, i = 0,1, \dots, N$. As a first step the energy-based ratio with which the bits will be distributed is calculated:

$$r_{\mathrm{i}} = \frac{\tilde{E}_{\mathrm{i}}}{\sum_{k=0}^{\mathrm{N}} \tilde{E}_{\mathrm{i}}}, \quad i = 0,1, \dots, N-1 \qquad (5.2\text{-}109)$$

Here it should be noted, that in the case where the input consists also from an LFE channel, it is not taken into account for the ratio calculations. The LFE channel is coded separately in XXX before of the call to MCT to code the remaining channels and therefore the bits for coding the LFE are already substracted from the bits available for coding the remaining channels The ratio is uniformly quantized:

$$\hat{r}_i = \max(1,\ \min(r_{RANGE} - 1, \lfloor r_{RANGE} \cdot r_i + 0.5 \rfloor)) \qquad (5.2\text{-}110)$$

$$r_{RANGE} = 2^{r_{bits}} \qquad (5.2\text{-}111)$$

The quantized ratios $\hat{r}_i$ are stored in the bitstream to be used from the decoder to assign the same amount of bits to each channel to read the transmitted channel spectra coefficients .

The bit distribution scheme is described below:

- For each channel assign the minimum amount of bits required by the entropy coder $bits_{min}$.
- The remaining bits, i.e., $bits_{remaining} = bits_{total} - \sum_{k=0}^{N} bits_{min,i}$ are divided using the quantized ratio $\hat{r}_i$:

$$\mathrm{bits}_i = \frac{\hat{r}_i}{r_{RANGE}} \cdot \mathrm{bits}_{remaining} \qquad (5.2\text{-}112)$$

- Because of the quantized ratio the bits are approximately distributed and therefore it may be

$$\mathrm{bits}_{split} = \sum_{k=0}^{\mathrm{N}} bits_i \neq bits_{total}. \qquad (5.2\text{-}113)$$

  So in a second refining step the difference $bits_{diff} = bits_{split} - bits_{total}$ are proportionally subtracted from the channel bits $bits_i$:

$$\mathrm{bits}_i = \mathrm{bits}_i - \frac{\hat{r}_i}{r_{RANGE}} \cdot \mathrm{bits}_{diff} \qquad (5.2\text{-}114)$$

- After the refinement step if there is still a discrepancy of $bits_{split}$ in comparison with $bits_{total}$ the difference (usually very few bits) is “donated” to the channel with the maximum energy.
- Finally, in the case where all channels are silent and [are flagged to be ignored] all the bits are assigned to the first input channel.

Editor's note: clarify above "flagged or ignored"

The exact same procedure is followed from the decoder in order to determine the amount of bits to be read to decode the spectrum coefficients of each channel.

5.2.3.3.6 Quantization and coding of each channel

Quantization, noise filling and the entropy encoding, including the rate-loop, are as described in 5.3.3.2 of [3] and 5.3.3.3.8 of this specification. The power spectrum (magnitude of the MCLT) is used for the tonality/noise measures in the quantization and IGF as described 5.3.3.2 of [3] and 5.3.3.3.8 of this specification. Since whitened and band-wise M/S processed MDCT spectrum is used for the power spectrum, the same SNS and M/S processing must be done on the MDST spectrum. Likewise, the same normalization scaling towards the mean energy must be done for the MDST spectrum as it was done for the MDCT. For the frames where TNS is active, MDST spectrum used for the power spectrum calculation is estimated from the whitened and M/S processed MDCT spectrum.

5.2.4 Common spatial metadata coding tools

5.2.4.1 General

Common spatial metadata encoding tools are methods shared by the SBA, MASA and McMASA coding modes for encoding their spatial metadata. The methods are steered by the configuration parameters received as in input and shown in table 5.3-1.

Table 5.2-19: Main configuration parameters for spatial metadata coding

Configuration parameter name	Description	Relevant for
nbands	Number of parameter subbands	SBA, MASA, McMASA
nblocks	Number of subframes	SBA, MASA, McMASA
start_band	Index of the first parameter band	SBA, MASA, McMASA
Ho_dirac	Flag indicating that Higher-order Dirac is used in SBA mode	SBA

The nbands and nblock variables represent the number of frequency subbands that are coded and the number of subframes that are coded, respectively. They may differ from the total number of subbands, *B,* or total number of subframes *M* used in analysis or coming from the input.

5.2.4.2 Spatial metadata composition

The codec operates with spatial metadata as part of the input alongside audio transport channels or as being derived from the input audio data. This section covers the encoding of the spatial metadata parameters related to SBA or MASA input format, as well as some MASA based codec internal models. More precisely we describe in the following coding procedures for audio direction (azimuth, elevation), energy ratio, diffuseness, spread coherence, surround coherence. The number of directions for each method is specified for the cases where these methods are used.

Table 5.2-20: Main metadata configuration parameters for different formats

Mode	Direction	Number of directions	Energy ratio/ Diffuseness	Spread coherence	Surround coherence
MASA	Yes	1 or 2	Yes	Yes	Yes
MC MASA	Yes	1	Yes	Yes	Yes
SBA	Yes	1 or 2	Yes	No	No

In all the methods presented below we will consider the subbands indexed with *b=1:nbands* and the subframes indexed by *m=1:nblocks*. One time-frequency element is denoted as TF tile. The number of subbands and subframes are defined for each mode and operation point.

5.2.4.3 Direction metadata coding tools

5.2.4.3.1 Overview

This subclause defines the coding tools used in the spatial direction encoding methods described in later subclauses. These are low-level functions that will be referred to later in the description. They include the direction parameters (azimuth and elevation) quantization, the raw or fixed rate encoding, as well as methods for variable rate encoding of the direction parameters.

5.2.4.3.2 Direction metadata quantization

5.2.4.3.2.1 Joint azimuth elevation quantization

5.2.4.3.2.2 Azimuth quantization

The azimuth codebook is uniform and the number of points is given by the $n_\phi$corresponding to the quantized elevation value. The azimuth is differently quantized if it corresponds to a direction from MASA input format or to a direction from multichannel input format in McMASA representation. Both variants are presented in this section.

If the input format is MASA, the azimuth points for two consecutive circles in the spherical grid are shifted by the value $shift$ such that for zero valued elevation the first azimuth codeword has value zero. For the following elevation value the first azimuth codeword has value equal to half of the corresponding quantization step: $\frac{\Delta\phi_{id_\theta}}{2} = \frac{360}{2\, n_\phi[id_\theta]}$. If the elevation values are numbered from 0, the first azimuth point on circles corresponding to even numbered elevation values are zero, while those corresponding to odd elevation values are $\frac{\Delta\phi_{id_\theta}}{2}$. The shifting is considered only when the number of azimuth points on the horizontal circles on the sphere is larger than 2. The value $shift$ is thus defined by:

$$shift = \begin{cases} 0, if\ id_\theta\ is\ odd \\ \frac{\Delta\phi_{id_\theta}}{2}, if\ id_\theta\ is\ even \end{cases} \tag{5.2-115}$$

The azimuth quantization for MASA input format is performed as shown in the following equation:

$$id_\phi = \left[\frac{\phi - 180 - shift}{\Delta\phi_{id_\theta}}\right], 0 \le \phi < 360$$

For obtaining the quantized value $\hat{\phi}$ and the final quantized azimuth index, the index id_ϕ goes through the following verifications that take into account that the azimuth values are cyclic:

```
if ( $id_\phi$ + ($n_\phi[id_\theta]$ >> 1) < 0 ) $id_\phi$ = $id_\phi$ + 1;
if ( $id_\phi$ - ($n_\phi[id_\theta]$ >> 1) >= 0 ) $id_\phi$ = $id_\phi$ -( $n_\phi[id_\theta]$ >> 1 );
```

In addition, the quantized azimuth index is transformed with the following pseudo code, such that it has positive values for directions pointing to the left side and negative values for directions pointing to the right side. The frontal direction has value zero.

```
if ( $id_\phi$ == -(($n_\phi[id_\theta]$ >> 1) + ( $n_\phi[id_\theta]$ % 2 )))
{
   $id_\phi$ = $id_\phi$ + ($n_\phi[id_\theta]$ % 2);
}
else
{
   if ( $id_\phi$ == (($n_\phi[id_\theta]$ >> 1) + ($n_\phi[id_\theta]$ % 2)))
   {
      if ($n_\phi[id_\theta]$ % 2)
      {
         $id_\phi$ = $id_\phi$ - 1;
      }
      else
      {
         $id_\phi$ = $id_\phi$ - $id_\phi$;
      }
   }
}
```

The quantized azimuth value is $\hat{\phi} = id_\phi * \Delta_{\phi_{id_\theta}}$. The domain of the quantized azimuth is consequently $, -180 \leq \phi < 180$. After the value of the quantized azimuth is calculated, an additional transformation is performed on the quantized azimuth index ordering the quantized values such that the first index is assigned to the zero quantized azimuth value followed by alternating positive and negative azimuth values.

If the input format is multichannel and the encoding is done through the McMASA representation (described in 5.7.3), the azimuth quantization is preceded by a companding function. The compading function definition depends on the multichannel input format type and on absolute elevation value being larger than a threshold or not. The companding function is defined for the input domain [-180, 180) and is specified below:

```
if multichannel format is 5.1 or 5.1+2 or 5.1+4
{
   pointsA = { 0.0, 50.0, 90.0, 150.0, 180.0};
   pointsB = { 0.0, 90.0, 110.0, 170.0, 180.0};
}
else
{
   pointsA = { 0.0, 60.0, 110.0, 150.0, 180.0};
   pointsB = { 0.0, 90.0, 110.0, 170.0, 180.0 };


if ( absolute elevation > 40)  ∧ (multichannel format is 5.1+2 or 5.1+4 or 7.1+4)
{
   pointsA = { 0.0, 30.0, 80.0, 150.0, 180.0 };
   pointsB = { 0.0, 10.0, 100.0, 170.0, 180.0};
}
find k = 0:4 such that $abs(\phi) \leq pA[k+1]$
calculate $comp(\phi) = sign(\phi)\left(pB[k] + \frac{pB[k+1]-pB[k]}{pA[k+1]-pA[k]}(abs(\phi) - pA[k])\right)$
```

The companded signed value is then scalar quantized and the signed index id_ϕ is transformed such that there are exactly the number of values:

```
if ( id_phi + ( $n_\phi[id_\theta]$ >> 1 ) < 0 )
{
   id_phi += 1;
}
if ( id_phi - ( $n_\phi[id_\theta]$ >> 1 ) >= 0 )
{
   id_phi = -( $n_\phi[id_\theta]$ >> 1 );
}
if ( id_phi == -( ( $n_\phi[id_\theta]$ >> 1 ) + ( $n_\phi[id_\theta]$ % 2 ) ) )
{
   id_phi += ( $n_\phi[id_\theta]$ % 2 );
}
else
{
   if ( id_phi == ( ( $n_\phi[id_\theta]$ >> 1 ) + ( $n_\phi[id_\theta]$ % 2 ) ) )
   {
      if ( $n_\phi[id_\theta]$ % 2 )
      {
         id_phi -= 1;
      }
      else
      {
         id_phi = -id_phi;
      }
   }
}
```

The resulting index is used for obtaining the quantized azimuth value in the companded domain $\hat{\phi}_c = id_\phi * \Delta_{\phi_{id_\theta}}$ and then inverse companded to obtain the real quantized azimuth value.

The azimuth quantization and indexing are wrapped together under *quantize_phi*() function that will be referred in later clauses.

5.2.4.3.3 Direction metadata raw encoding

5.2.4.3.4 Direction metadata entropy encoding tools

5.2.4.3.4.1 Quasi-uniform encoding

The *encode_quasi_uniform(value, alphabet_size)* function is used to encode $value$ with quasi-uniform probability using a punctured code. For $value \in \{0, \ldots, alphabet\_size - 1\}$, a number of the smallest ones are encoded using $\lfloor log_2(alpbabet\_size) \rfloor$ bits, and the rest using $\lfloor log_2(alpbabet\_size) \rfloor + 1$ bits. If $alphabet\_size$ is a power of two, binary coding results. The function is defined in pseudo-code as:

```
nbits = floor(log2(alphabet_size))
thresh = 2 ^ (nbits + 1) - alphabet_size
if (value < thresh)
   write_bits(value, nbits)
else
   write_bits(value + thresh, nbits + 1)
```

If *alphabet_size* is a power of 2, then *alphabet_size = 2 ^ bits*, and *thresh = 2 ^ bits*, therefore the else branch is never used, and binary coding results. Otherwise, the first *thresh* smallest values are encoded using a binary code having *nbits* bits, and the rest, starting with *value = thresh*, are encoded using a binary code having *nbits + 1* bits. The first binary code encoded using *nbits + 1* bits has the value *value + thresh = 2.thresh*, therefore the decoder can figure out, by reading only the first *nbits* bits and comparing its value with *thresh*, if it needs to read one more additional bit.

5.2.4.3.4.2 Extended Golomb-Rice coding tool

The extended Golomb-Rice entropy coding tool is based on Golomb-rice coding with an integer parameter $p \geq 0$, to code an unsigned integer u. In traditional Golomb-rice coding, first, u is split into the least significant part with p bits, $u\_lsp = u \; mod \; 2^p$, and the most significant part $u\_msp = \lfloor u \div 2^p \rfloor$. The most significant part is coded in unary, using u_msp 1-bits and a terminating zero bit. The least significant part is coded in binary.

Because arbitrarily large integers can be coded, some coding efficiency may be lost when the actual values to be coded have a known and relatively small alphabet size. The Extended Golomb-Rice method combines three improvements over the traditional Golomb-Rice coding, for coding a vector of values, each with a known and potentially different alphabet size u_alph. First, the alphabet size of the most significant part can be computed as $u\_msp\_alph = \lceil u\_alph \div 2^p \rceil$. If the maximum possible value of the most significant part is coded, $u\_msp\_alph - 1$, the terminating zero bit can be eliminated, because this condition can be implicitly detected at the decoder side, knowing the alphabet size as well. This is called the limited Golomb-rice coding. Additionally, for the same case when $u\_msp = u\_msp\_alph - 1$, the alphabet size of the least significant part u_lsp, which can be computed as $u\_alph - (u\_msb\_alph - 1) \cdot 2^p$, may be smaller than 2^p, allowing to use advantageously the *encode_quasi_uniform()* function instead of binary coding with p bits. The function *encode_quasi_uniform()* is also advantageously used when a particular value u has an alphabet u_alph smaller than 2^p. Finally, when $u\_msp\_alph \leq 3$ the Limited Golomb-Rice method produces codes having only two possible lengths, $1 + p$ and $2 + p$ bits. Once again, the function *encode_quasi_uniform* function is optimal for up to two lengths, and therefore it is used instead in this case.

The final pseudo-code of the function *encode_extended_gr()*, looks like:

```
Encode_extended_gr():
    u_msb_alph = ( alphabet_size + ( 1 << p ) - 1 ) >> p
   if ( u_msb_alph <= 3 )
   {
    encode_quasi_uniform( u, alphabet_size );
   }
   else
   {
      u_msb = u >> p;
      u_lsb = u & ( ( 1 << p ) - 1 )
      write_bit(1) x u_msb  //Leading MSB 1-bits
      if ( u_msb < u_msb_alph_size - 1 )
      {
          write_bit(0) x 1 // Terminating 0-bit for MSB
         write_binary(u_lsb) // LSB binary code
```

```
        }
        else
        {
           encode_quasi_uniform( u_lsb, alphabet_size - ( ( u_msb_alph - 1 ) << p ) );
        }
    }
```

5.2.4.4 Diffuseness and energy ratio coding methods

5.2.4.4.1 Diffuseness and energy ratio definitions

Directional component energy ratios and diffuseness parameter are quantized and coded using the same set of coding tools. Indeed, the diffuseness can be interpreted as the complement of the ratio of the sum of directional energies to the total energy, and is used to deduce the energy ratios of the directional components. If several directional components are transmitted, the ratios of their respective energies to the total energy are additionally transmitted, but only for the *N-1* directional components, where *N* is the total number of directional components. The remaining directional energy ratio can be deduced from the diffuseness and the transmitted energy ratios.

It is important to note that the diffuseness and energy ratio parameters are quantized and coded with different time resolution than the direction parameters. The diffuseness and energy ratio parameters are transmitted once per frame with a time of resolution of 20ms, while the direction parameters are transmitted *nblocks* per frame.

5.2.4.4.2 Diffuseness parameter quantization

Each diffuseness parameter bounded between 0 and 1 is quantized to one of the discrete levels, using a non-uniform quantizer producing the diffuseness index. The quantizer thresholds and levels were derived from the well-known properties of the quantization sensitivity of the Inter-Channel Coherence (ICC), which is more sensible to quantization error towards a coherence of 1 than a coherence of 0. The diffuseness, which can be seen as of a complementary nature of the coherence, shows then an inverse property: the diffuseness is more sensitive to quantization error toward 0. Moreover, a diffuseness value of 1 is avoided to be coded and transmitted because of sound rendering consideration.

Two resolutions are possible for quantizing the diffuseness parameters, using 8 and 16 levels. The two quantizers are defined by the number of their thresholds and reconstruction levels given in table 5.2-21:

Table 5.2-21: Diffuseness quantization thresholds and levels

Number of levels (*diffuseness_levels*)	Number of bits in raw coding	Thresholds (*diffuseness_thresholds*)	Reconstruction levels
8	3	0.0	0.0,
		0.01904296875	0.03955078125,
		0.06298828125	0.089599609375,
		0.119384765625	0.158935546875,
		0.22119140625	0.308349609375,
		0.399169921875	0.473876953125,
		0.547607421875	0.63232421875,
		0.734619140625	0.85009765625
		2.0	
16	4	0.0, 0.009521484375	0.00
		0.01904296875	0.0142822265625
		0.0410156250	0.030029296875
		0.06298828125	0.052001953125
		0.0911865234375	0.07708740234375
		0.119384765625	0.10528564453125
		0.1702880859375	0.14483642578125
		0.22119140625	0.19573974609375
		0.3101806640625	0.26568603515625
		0.399169921875	0.35467529296875
		0.473388671875	0.436279296875
		0.547607421875	0.510498046875
		0.641113281250	0.5943603515625
		0.734619140625	0.6878662109375
		0.8673095703125	0.80096435546875
		2.0	0.93365478515625

A placeholder out-of-range large threshold value (2.0) is added at the end of thresholds to make searching it simpler. The quantization of the diffuseness values is then achieved using the following simple procedure:

```
for ( diffuseness_index = 0; diffuseness_index < diffuseness_levels; diffuseness_index ++ )
     if ( diffuseness_value < diffuseness_thresholds [diffuseness_index + 1] )
          return diffuseness_index;
```

5.2.4.4.3 Diffuseness and energy ratio indices coding

The quantized diffuseness indices are coded using one of the three available methods: raw coding, one value only, and two consecutive values only.

By default, raw coding is used and the method is signaled by setting to 1 the *diff_use_raw_coding* bit flag, which indicates whether the raw coding method is used. For raw coding, each diffuseness index value is encoded using the *encode_quasi_uniform()* function.

If all index values are equal, the one value only method is used, and *diff_use_raw_coding* is set to 0. A second bit (*diff_have_unique_value*) is used to indicate this method, then the unique value is encoded using the *encode_quasi_uniform()* function.

Else if all index values consist only of two consecutive values, the two consecutive values only method is used, indicated by setting *diff_use_raw_coding* and *diff_have_unique_value* to zero. The smaller of the two consecutive values is encoded using the *encode_quasi_uniform* function, taking into account that its alphabet size is reduced to $diff\_alph - 1$. Then, for each value, the difference between it and the minimum value is encoded on one bit.

Table 5.2-22 summarizes the different coding schemes, as well the resulting used coding tools, where *N* represents the *nbands-start_band* diffuseness parameters to code:

Table 5.2-22: Diffuseness coding methods, associated with their signaling and the subsequent coding. N represents the number of diffuseness indices to code.

Method	*diff_use_raw_coding* bit	*diff_have_unique_value* bit	Coding of diffuseness indixes
Raw coding	1	--	*N* times *encode_quasi_uniform()*
One value only	0	1	*encode_quasi_uniform(value, alphabet_size)*
Two consecutive values	0	0	*encode_quasi_uniform(value_min, alphabet_size-1)* Nx (value – value_min) on 1 bit

The diffuseness indices transmitted are important for the subsequent quantization and coding of the direction metadata. It will dictate the accuracy of direction quantification, following the principle that more the diffuseness is high, less important is the direction accuracy.

5.2.4.4.4 Diffuseness and energy ratio coding with two concurrent directions

5.2.4.4.4.1 Coding method overview

In the case of two sets of directional parameters, the diffuseness (or the equivalent diffuse-to-total energy ratio) and the direct-to-total energy ratio parameters are quantized and transmitted together in a combined scheme to save bits and improved quantization accuracy.

The general process is that, first, a diffuseness ratio is quantized into an index value. Then, the index value of the quantized diffuseness ratio is used to select the number of bits for quantizing a second ratio parameter. This second ratio parameter and its quantization depends on whether the metadata codec is used in Ho-DirAC mode or in MASA format metadata mode.

In case of Higher-order Dirac, a diffuseness and the direct-to-total ratio is given. The direct-to-total energy is quantized on a number of bits which depends on the quantized diffuseness index as shown in Table 5.2-23.

Table 5.2-23: Number of bits to code direction energy ratio in Ho-DirAC

diff_index	>=6	>=4 && < 6	< 4
dfRatio_bits	1	2	3

The direct-to-total ratio is then quantized uniformly between 0 and 1, with step of *1/(2^dfRatio_bits-1)*.

In the case of MASA format metadata mode, a following specific method is applied for each time-frequency tile containing two directional parameter sets. As energy ratio parameters are averaged into a same value for each subframe within a frequency band in preliminary processing, further processing is done only on frequency bands when it is possible to save complexity.

As a preliminary step, the metadata of the two directional parameter are organized such that the direct-to-total energy ratio of the first direction r_{dir1} is always equal or larger than the direct-to-total energy ratio of the second direction r_{dir2}, that is, $r_{dir1} \geq r_{dir2}$. The process of reordering is described in (TODO: ref).

First, diffuseness ratio r_{diff} is obtained with the relation:

$$r_{diff} = 1 - (r_{dir1} + r_{dir2})$$

Then a distribution factor of the two direct-to-total energy ratios (dfRatio in tables) is calculated as

$$r_{df} = \frac{r_{dir1}}{r_{dir1} + r_{dir2}}$$

In the special case of $r_{dir1} + r_{dir2} \leq \epsilon$, then the distribution factor $r_{df} = 0.5$. In this special case it is assumed that the direct-to-total energy ratios are equal.

The r_{diff} is then quantized using the method described in (TODO: add ref). The quantized diffuseness is represented by the diff_index parameter. Using the diff_index, the number of bits for quantizing r_{df} is selected with a function using thresholds that fulfill the table 5.2-24. In practice, higher r_{diff} results in less bits used to quantize r_{df}. The selected dfRatio_bits value is then used to select a corresponding uniform scalar quantizer for quantizing the r_{df}. The r_{df} is then quantized using the selected uniform scalar quantizer to obtain dfRatio_index parameter which represents the quantized r_{df}. The diff_index and dfRatio_index values are then entropy encoded to the bitstream as described in (TODO: ref and ref).

Table 5.2-25: Number of bits to code direction energy ratio in MASA

diff_index	>=6	>=3 && < 6	< 3
dfRatio_bits	1	2	3

5.2.4.4.4.2 Distribution factor ratio coding for two concurrent directions

Encoding the dfRatio_index values are done similarly as encoding diff_index values but with adaptation to what methods are available and what alphabet size is used depending on the realized number of bits used to quantize each r_{df}. As the number of raw bits known, it is possible to allow only those methods to be used that can offer benefit in compression.

First, total number of bits for raw coding raw $bits_{df-raw}$ is calculated as a sum of dfRatio_bits over all coding bands. Next, maximum value is selected from dfRatio_bits values in the coding and assigned to $bits_{df-max}$ to represent the largest used number of bits to code any r_{df} for this metadata frame.

The dfRatio_index coding algorithm is then configured to use the methods in table (5.2.-24, TODO: add ref) as follows:

- If $bits_{df-raw} \geq bits_{df-max} + 2 + B_{coding}$, then all methods are used.
- Else if $bits_{df-raw} \geq bits_{df-max} + 1$, then "Two consecutive values"-method is not used
- Else, only raw coding mode is used

Here, B_{coding} represents the total number of used coding frequency bands that is configured for the metadata codec.

The alphabet_size for the coding methods is set to $2^{bits_{df-max}}$ Signalling different coding methods is also done when necessary, that is, the coding method is in use. Otherwise, the dfRatio_index is encoded similarly as the diff_index.

5.2.4.4.4.3 Direction quantization bits adjustment with two concurrent directions

In the case of two sets of directional parameters and MASA format metadata for each time-frequency tile, the quantization resolution for quantizing the direction parameters need to be determined. This is due to the combined effect of the energy ratio model of MASA format metadata in encoding and the initial quantization resolution assignment based on the quantized direct-to-total energy ratio index. The energy ratio model of MASA format metadata states that

$$r_{dir1} + r_{dir2} + r_{diff} = 1$$

and the initial quantization resolution assignment based on the quantized direct-to-total energy ratio index assigns bits for direction parameter quantization based on the value given by the index of the quantized direct-to-total energy ratio for the direction. For two directions which are essentially equal, that is, $r_{dir1} \approx r_{dir2}$, neither of the directions can individually have a high direct-to-total energy ratio, and due to the non-linear quantization of the direct-to-total energy ratios, the quantization accuracy of the direction is not high enough.

This combined effect is compensated with the following approach. The following process is done separately for each coding band which has two directional spatial audio parameter sets, and the process affects the quantization spatial resolution of the direction parameters. First, the diff_index and dfRatio_index values are decoded to quantized diffuse-to-total ratio $\hat{r}_{diff}$ and quantized distribution factor $\hat{r}_{df}$ respectively. The decoding of the index values is described in (TODO:ref to dec) where the index values are decoded to give the decoded quantized values $\hat{r}_{diff}$ and $\hat{r}_{df}$.

The first quantized direct-to-total energy ratio is obtained as

$$\hat{r}_{dir1} = \hat{r}_{df}\left(1 - \hat{r}_{diff}\right)$$

The second quantized direct-to-total energy ratio is obtained as

$$\hat{r}_{dir2} = 1 - \hat{r}_{diff} - \hat{r}_{dir1}$$

Then, an inverted index is determined for both of the quantized direct-to-total energy ratios by re-quantizing values $1 - \hat{r}_{dir1}$ and $1 - \hat{r}_{dir2}$ using the diffuseness quantizer described in (TODO: add ref). This provides the indices index_dirRatio1Inv and index_dirRatio2Inv respectively.

Next, a combined ratio is determined by taking a sum of the quantized ratios:

$$\hat{r}_{comb} = \hat{r}_{dir1} + \hat{r}_{dir2}$$

Then, the larger of the two quantized direct-to-total energy ratios is selected. When $\hat{r}_{dir1}$ is larger than (or equal to) $\hat{r}_{dir2}$, $\hat{r}_{dir1}$ is selected and a set of modified direct-to-total energy ratios is obtained as follows:

$$r_{dir1-mod} = \hat{r}_{comb}$$
$$r_{dir2-mod} = \frac{\hat{r}_{dir2}}{\hat{r}_{dir1}}\hat{r}_{comb}$$

When $\hat{r}_{dir2}$ is larger and consequently selected, the indices in the two above equations are turned the other way round resulting in

$$r_{dir2-mod} = \hat{r}_{comb}$$
$$r_{dir1-mod} = \frac{\hat{r}_{dir1}}{\hat{r}_{dir2}}\hat{r}_{comb}$$

The modified direct-to-total energy ratios are then used to determine the modified inverted ratio indices index_dirRatio1Inv_mod and index_dirRatio2Inv_mod by using the diffuseness quantizer to quantize values $1 - r_{dir1-mod}$ and $1 - r_{dir2-mod}$ respectively.

Next a check is performed where, both modified inverted ratio indices are compared to their respective original indices as a difference in order to limit the maximum difference caused by the modification. In practice, this is expressed as

- If, index_dirRatio1Inv - index_dirRatio1Inv_mod > MASA_DIR_RATIO_COMP_MAX_IDX_STEPS, then index_dirRatio1Inv = ratio_index_1 - MASA_DIR_RATIO_COMP_MAX_IDX_STEPS
- Else, index_dirRatio1Inv is unmodified

Same is done for the second direction. Thus, the difference of the energy ratios is limited as well based on the difference between the modified and non-modified values. These modified energy ratio indices are then used in the algorithm presented in (TODO:ref Direction metadata quantization) to select bits used for quantizing the two spatial direction parameters associated with the time-frequency tile, and the bits in turn define the quantization spatial resolution for the two direction parameters of the time-frequency tile in the encoding process.

It should be noted that although Ho-DirAC mode encodes two directional parameter sets per frequency band, the modified ratio scheme is not used for it as the energy ratio model is different.

5.2.4.5 Direction metadata coding methods

5.2.4.5.1 Entropy coding 1 (EC1)

The Entropy Coding 1 (EC1) is the first method used to attempt to encode the direction metadata. It has two coding modes, a full raw coding mode and a frame-wise entropy coding mode, and then selects the most efficient mode producing the fewest bits. Unlike other entropy coding methods, EC1 does not re-quantize direction parameters. It losslessly codes the previously quantized direction indices, derived from the spherical quantization solely selected by the quantized diffuseness indices.

5.2.4.5.1.1 Full raw coding mode

The full raw coding mode sequentially code, without any modelling, each spherical index or azimuth index of the time-frequency grid. The spherical indices are coded from the lowest frequency band to the highest, and for each frequency band from the earliest time block to the latest.

In case of 3D audio scene, the spherical indices are coded within an integer number of bits, using a resolution depending on the diffuseness as described in clause xxx. The spherical indices are then written sequentially in the order as described above. The bit consumption is then easily estimated as:

$$direction\_bits\_raw = \sum_{b=start\_band}^{nbands} nblocks.spherical\_idx\_bits \quad (5.2\text{-}117)$$

In case of 2D audio scene, i.e. if the current configuration takes into account only the horizontal equatorial plane, the elevation is always set to 0, and only the azimuth indices have to be transmitted. It is achieved by using the *encode_quasi_uniform()* coding tool. Considering the azimuth alphabet size used for each frequency band, the bit consumption of the raw coding is then given by:

$$direction_{bits_{raw}} = \sum_{b=start\_band}^{nbands-1} \sum_{n=0}^{nblocks-1} encode\_quasi\_uniform\_length(az\_idx(b,n), az\_alph(b)) \quad (5.2\text{-}118)$$

Frame-wise entropy coding mode

The frame-wise entropy coding mode uses some modelling to code the quantized directions, which are grouped into two categories. The first group contains the quantized directions for diffuseness indices $diff\_idx(b) \leq ec\_max$ which are entropy coded, and the second contains the quantized directions for diffuseness indices $diff\_idx(b) > ec\_max$ which are raw coded, where ec_max is a threshold optimally chosen depending on $diff\_alph$. This approach implicitly excludes from entropy coding the frequency bands with high diffuseness, when frequency bands with low to medium diffuseness are also present in a frame, to avoid mixing statistics of the residuals. For a mixed diffuseness frame, raw coding is always used for the frequency bands with high diffuseness. However, if all frequency bands have high diffuseness, $diff\_idx(b) > ec\_max$, the threshold is set in advance to $ec\_max = diff\_alph$ in order to enable entropy coding for all frequency bands.

For the entropy-coded group, an average direction vector is first derived, by converting each quantized direction which is entropy coded back to a direction vector in cartesian coordinate and computing their sum. From the average direction an average elevation el_avg and azimuth az_avg are obtained. These two values are then quantized using the best angular precision deg_req used by the quantized directions which are entropy coded, denoted by deg_req_avg, which is the required angular precision corresponding to the smallest diffuseness index $diff\_\text{idx\_min} = min(diff\_idx(b))$, for $b \in \{0, \dots, nbands - 1\}$ and $diff\_idx(b) \leq ec\_max$. In case of High Resolution mode, used on certain MASA modes, $diff\_\text{min} \_idx$ is set to zero, for getting the maximum quantization accuracy possible to quantize the average direction. The average direction is then used as a predictor for entropy-coded group directions.

Average direction quantization and subsequent predictive coding are performed separately for elevation and azimuth.

5.2.4.5.1.2 Elevation entropy-coding

The average elevation alphabet and codebook are derived from the angular resolution and number of angles theta (no_theta), which are defined in Table 5.2-26 as a function of the diffuseness index:

Table 5.2-26: definition of the elevation angular resolution and no_theta in function of the diffuseness index and number of bits allocated to the spherical indexing.

Diff_idx	0	1	2	3	4	5	6	7
Spherical_bits	11	11	10	9	7	6	5	3
No_theta	19	19	14	19	6	5	4	2
Delta_theta (degree)	5	5	6.7	10.78	20	25.5	36	45

The elevation codebook is then built delta_theta[diff_idx].ele_idx, where ele_idx is defined as the integers between -no_theta[diff_idx] +1 and no_theta[diff_idx]-1. The alphabet size of the codebook is then avq_elevation_alphabet = 2no_theta[diff_idx]-1.

The absolute value of the average elevation is quantized with nearest neighbor from the elevation codebook, and the sign if finally coded using the following rule:

```
Avg_elevation_idx = ( avq_elevation_alphabet >> 1) – avg_abs_elevation_index if avg_elevation <0
Avg_elevation_idx = ( avq_elevation_alphabet >> 1) + avg_abs_elevation_index if avg_elevation >=0
```

For each direction to be entropy coded, the dequantized average elevation q_el_avg is projected using the precision of that elevation to code, to obtain predicted elevation indices. For an elevation index el_idx, its precision, which can be derived from el_alph, is used to compute the projected average elevation index $el\_avg\_idx\_p$. For the corresponding azimuth index az_idx, its precision on the horizontal circle situated at the q_el elevation, which can be derived from az_alph, is used to compute the projected average azimuth index $az\_avg\_idx\_p$.

The projection to obtain predicted elevation is computed as $el\_avg\_idx\_p = round\left(\frac{el\_avg\_idx}{el\_avg\_alph-1} \cdot (el\_alph - 1)\right)$. To facilitate bit-exact operation, the previous formula can be rewritten using integer only math, including division, as $el\_avg\_idx\_p = (2 \cdot el\_avg\_idx \cdot (el\_alph - 1) + (el\_avg\_alph - 1))\ div\ (2 \cdot (el\_avg\_alph - 1))$.

The signed distance el_idx_dist is computed as the difference between each elevation index l_idx and its corresponding $el\_avg\_idx\_p$. Additionally, because the difference produces values in the interval $\{-el\_alph + 1, \dots, el\_alph - 1\}$, they are reduced to the interval $\{-\lfloor el\_alph \div 2\rfloor, \dots, \lfloor el\_alph \div 2\rfloor\}$ by adding el_alph for values which are too small and subtracting el_alph for values which are too large, like in modular arithmetic. If this reduced distance relative to $el\_avg\_idx\_p$ is interpreted using wrap-around, it can produce all values from to the unsigned alphabet containing el_alph values.

The elevation part consists of three components: the average elevation index, a Golomb-Rice parameter, and the reduced signed elevation distances. The average elevation index el_avg_idx is converted to signed, so that the zero value is in the middle of the signed interval of possible values, the *ReorderGeneric* function is applied, and the result is coded using the *EncodeQuasiUniform* function. The Golomb-Rice parameter, having an alphabet size depending on the maximum of the alphabet sizes of the elevation indices, is coded using the *EncodeQuasiUniform* function. Finally, for each reduced signed elevation distance el_idx_dist, the *ReorderGeneric* function is applied to produce $el\_idx\_dist\_r$, and the result is coded using the Extended Golomb-Rice method with the parameter indicated above.

The operation is repeated by adding an offset to the average elevation index, offset selected in the following order {+1,-1,+2,-1, +3,-3…}, where the number of tested offsets is defined by search effort requested. By default, the search effort is set to 3, meaning that the best offset among 0, +1 and -1 is retained as the one engendering the lowest bit consumption for the elevation.

In case of McMasa, or when the directions are only defined in the north hemisphere for positions of positive or null elevation, the average elevation codebook is limited to an alphabet of *no_theta[diff_idx]* elements, discarding the negative elevation. The sign is then not needed to be transmitted since always positive, and the average elevation is simply coded by the *encode_quasi_uniform*() function. Moreover, during the search effort, only positive offsets are then tested.

5.2.4.5.1.3 Azimuth entropy-coding

Unlike the elevation, which has a single resolution possible for a given diffuseness index, the azimuth can be quantized on different angular resolution for the same spherical quantization, depending on the elevation plane on which it is quantized. For the entropy coding, the average azimuth is quantized using *quantize_phi()* on the equatorial plane using the maximal resolution of the given spherical quantization.

For each direction to be entropy coded, the dequantized average azimuth q_az_avg is projected using the precision of that the direction to be coded, to obtain predicted azimuth indices. For the corresponding azimuth index az_idx, its precision on the horizontal place situated at the associated q_el elevation, which can be derived from az_alph, is used to compute the projected average azimuth index $az\_avg\_idx\_p$.

The projection to obtain predicted azimuth indices is computed as $az\_avg\_idx\_p = round\left(\frac{q\_az\_avg}{360} \cdot az\_alph\right) \; mod \; az\_alph$, which can be easily simplified to $az\_avg\_idx\_p = round\left(\frac{az\_avg\_idx}{az\_avg\_alph} \cdot az\_alph\right) \; mod \; az\_alph$. To facilitate bit-exact operation, the previous formula can be rewritten using integer only math, including division, as $az\_avg\_idx\_p = \left((2 \cdot az\_avg\_idx \cdot az\_alph + az\_avg\_alph) \; div \; (2 \cdot az\_avg\_alph)\right) \; mod \; az\_alph$. At the poles, where $az\_alph = 1$, we always have $az\_idx = 0$ and set $az\_avg\_idx\_p = 0$ directly.

The signed distance az_idx_dist is computed as the difference between each azimuth index az_idx and its corresponding $az\_avg\_idx\_p$. The difference operation produces values in the interval $\{-az\_alph + 1, \ldots, az\_alph - 1\}$, which are reduced to the interval $\{-az\_alph \div 2, \ldots, az\_alph \div 2 - 1\}$ by adding az_alph for values which are too small and subtracting az_alph for values which are too large. When $az\_alph = 1$, the azimuth index is always $az\_idx = 0$ and nothing needs to be coded.

As for the elevation, the azimuth part also consists of three components: the average azimuth index, a Golomb-Rice parameter, and the reduced signed azimuth distances. The average azimuth index az_avg_idx is converted to signed, so that the zero value is in the middle of the signed interval of the possible values, the *ReorderGeneric* function is applied, and the result is coded using the *encode_quasi_uniform()* function. The Golomb-Rice parameter, having an alphabet size depending on the maximum of the alphabet sizes of the azimuth indices, is coded using the *encode_quasi_uniform()* function. Finally, for each reduced signed azimuth distance az_idx_dist, the *ReorderGeneric* function is applied to produce $az\_idx\_dist\_r$, and the result is coded using the Extended Golomb-Rice method with the parameter indicated above.

For example, if the best angular precision deg_req_min used is 5 degrees, then the maximum of the azimuth alphabet sizes az_alph will be $az\_alph\_max = 4 \cdot \lceil 90 \div deg\_req\_min \rceil = 72$. In this case, the Golomb-Rice parameter values (denoted as p in the description of the Golomb-Rice method below) are limited to the interval $\{0,1,2,3,4,5\}$. The optimal value of the Golomb-Rice parameter az_gr_param is chosen by efficiently computing, for each value in the interval above, the total size in bits for all the $az\_idx\_dist\_r$ values to be coded using the Extended Golomb-Rice method, and choosing the one which provides the smallest bit size.

As for the elevation, the operation is repeated by adding an offset to the average azimuth index, offset selected in the following order {+1,-1,+2,-1, +3,-3…}, where the number of tested offsets is defined by the search effort requested. By default, the search effort is set to 3, meaning that the best offset among 0, +1 and -1 is retained as the one engendering the lowest bit consumption for the azimuth.

5.2.4.5.1.4 Decision for fully raw coding or entropy-coding

Once the optimal code is found for the entropy coded elevation and azimuth, its bit consumption summed the bits used for coded the raw coding are compared to the full raw coding bit consumption. The method using the less bits is retained and signal by the bit flags in Table 5.2-27:

Table 5.2-27: Signalling bits for EC1

EC1 signaling 2-bit field	EC1 disable flag	Full raw coding disable flag
Raw coding	0	0
Frame-wise entropy coding	0	1

5.2.4.5.2 Entropy coding 2 (EC2)

5.2.4.5.3 Entropy coding 3 (EC3)

5.2.4.6 Coherence coding

5.2.4.7 DTX coding

DTX coding is common for SBA-DirAC and MASA. In the case of non-active frames and for the SID frames, it transmits a second sound field parameter representation, which is more compact than the parametric sound field representation used for active frames. The total payload allocated to the SID sound field parameters is a maximum of 2.8kbps, corresponding to the total SID bit rate of 5.2 kbps minus the 2.4 kbps SID of core-coding.

The number of parameter bands is limited to 5 for MASA and 2 for the DirAC part of SBA, representing the average of parameter bands expect the last one and the highest parameter band. The number of direction (DoA) is also limited to only one direction per parameter band. In case of MASA, the parameters are combined to one direction, 5 bands, and 1 time block.

First the diffuseness quantized on 8 levels is mapped to 4 levels on coded in raw coding on 2 bits:

$$diff\_index[b] \leftarrow \max\left(4, diff\_index[b]\right)$$

The associated spherical or azimuth coding resolution and bit demand is then computed for 3D or 2D audio scene, resolution. If the bit demand is greater than the bit allocated for the spatial parameter of the SID frame, then two different mechanisms are used, one for MASA and another for SBA.

In MASA and if the bit demand is too low compared to the bit-budget for the SID spatial parameters, the spherical or azimuthal resolution is modified greedily, from the lowest to the highest parameter band, in increments of +1, in order to meet the bit budget. If the bit demand is too high, the resolution is also modified greedily, but this time from the highest to the lowest band, and by a decrement of -1. The same logic has then to be applied at the decoder side.

In DirAC, the quantization resolutions are only changed if the bit demand is too high. It is achieved by increasing the diffuseness index by 1, from the highest to the lowest parameter bands. This is repeated until the bit budget is fulfilled. Since the diffuseness indices are transmitted, no extra logic must be applied at the decoder side.

The diffuseness indices for each parameter bands are then coded using the function *encode_quasi_uniform(),* for an alphabet of 4. The directions are averaged by averaging their cartesian coordinates through the time blocks for each parameter bands. The average directions are then transformed back to polar coordinates by quantizing on the azimuth in case of 2D scenes or the azimuth and elevation jointly by using the spherical quantization and indexing in case of 3D scenes. The spherical indices are coded in raw coding while the azimuth indices are coded using the *encoder_quasi_uniform()*, with the appropriate alphabet.

5.2.5 Modified Discrete Fourier Transform (MDFT) Analysis Filter Bank

The MDFT analysis filter bank is a 12-band convolution-type filter bank with a 1 ms signal delay. It is implemented by making use of 12 pre-computed bandpass filters, with the overall filter impulse response $h[n]$ formed from the weighted sum of the band-pass filter impulse responses $h_b[n]$ such that

$$h[n] = \sum_{b=1}^{B} h_b[n] w_b \qquad (5.2\text{-}119)$$

where the weights w_b are dynamic across blocks and are used to alter the gain of the audio signal at each frequency band. The filters are applied in the MDFT domain (subclause 5.2.5.1) at a 20ms block size with 1ms crossfade between blocks. The filter bank is in the audio signal path for SBA coding (subclause 5.4) and is used to generate an active downmix which is signal-dependent and time-varying.

5.2.5.1 Modified Discrete Fourier Transform (MDFT)

The N-point Modified Discrete Fourier Transform ($MDFT_N$) is the DFT of input signal $x[n]$ modulated by a complex exponential $e^{\frac{-j\pi n}{N}}$.

$$X[k] = \frac{1}{N}\sum_{n=0}^{N-1} e^{\frac{-j2\pi n(k+\frac{1}{2})}{N}} x[n]. \qquad (5.2\text{-}120)$$

The modulation corresponds to a half-bin shift in the DFT spectrum, which results in no bin centered at DC.

5.2.5.2 Filter bank responses

The pre-computed filter bank magnitude responses $|H_b(f)|$ for each of the 12 bands b for sampling rate $Fs = 48000$ Hz are shown in Figure 5.2-12 and the corresponding impulse responses $h_b[n] = MDFT_{960}^{-1}(H_b[k])$ are shown in Figure 5.2-13. It can be observed from Figure 5.2-12 that the filters are narrower at lower frequencies than at higher frequencies. Figure 5.2-13 shows that the filters are aligned to have peak energy at 1ms. These filters sum very closely to a unit impulse with a 1 ms delay:

$$\sum_{b=1}^{12} \mathrm{h_b}[n] \cong \delta[\mathrm{n} - \mathrm{L}], \qquad (5.2\text{-}121)$$

where $L = 0.001Fs$. This delay is equal to the audio latency of the filter bank. Note that $h_b[n] = 0, \forall n < 0$.

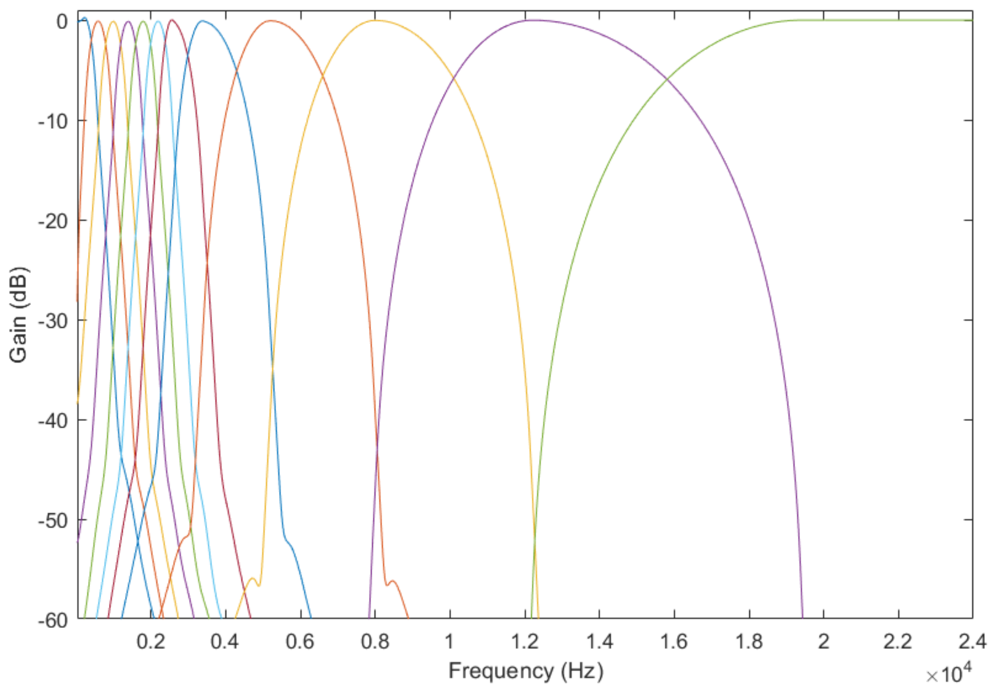


Figure 5.2-12: 48kHz MDFT filterbank magnitude responses

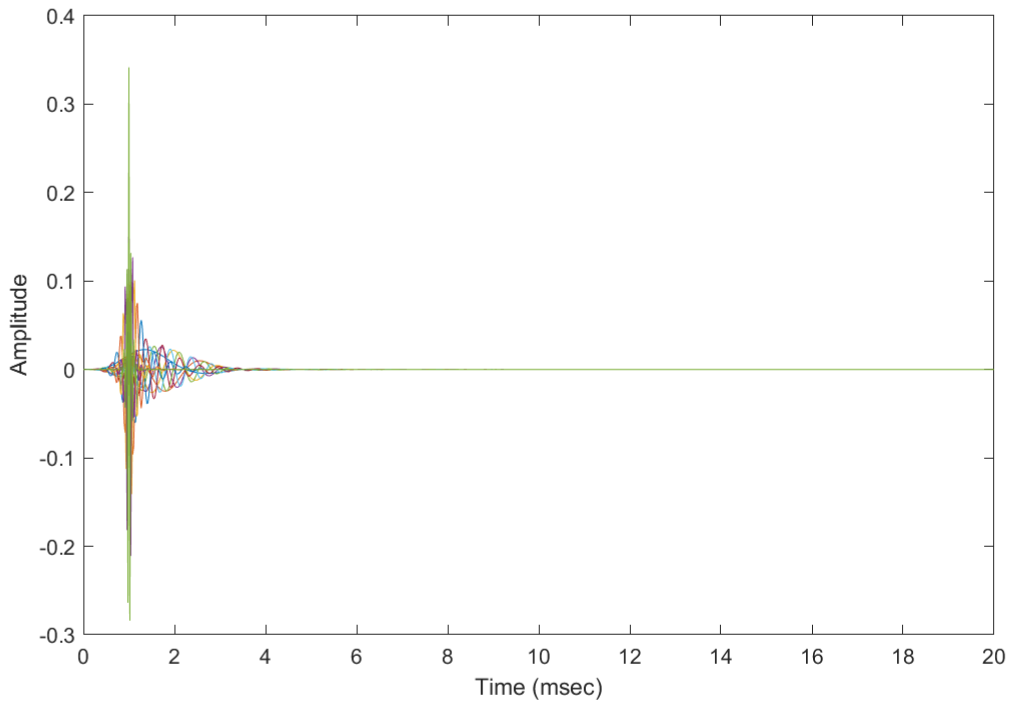


Figure 5.2-13: 48kHz MDFT filterbank impulse responses

For operation at 32 kHz and 16 kHz sampling rates, it is desirable to maintain the properties of the original 48 kHz design whilst also maintaining the same 1ms processing delay. To achieve this, one seeks to obtain a new set of filters that sum approximately to a unit impulse at 1ms, maintain the low-latency property that $h_b[n] = 0, \forall n < 0$ and closely approximate the banded frequency responses of the original design below the new (lower) Nyquist frequency. The following method is used to construct appropriate filters for the new sampling rates.

The starting point is to truncate the original frequency response at the new Nyquist frequency, then convert to the time domain to obtain an impulse response h'_b:

$$h'_b[n] = MDFT_M^{-1}(H'_b[k]) \tag{5.2-122}$$

where

$$H'_b[k] = \begin{cases} H_b[k], & k < M \\ 0, & otherwise \end{cases} \tag{5.2-123}$$

and the block size $M = 0.02Fs$ corresponds to the index of the Nyquist frequency at the new sampling rate Fs. This impulse response represents a filter that matches the original frequency response perfectly but will in general contain pre-ripple energy at $n < L$ that extends below n = 0 and would therefore require additional delay to implement. To address this, a fade function $s(n)$ is defined as

$$s[n] = \begin{cases} 1 & n < -L \\ 0 & n > L \\ f_{cheby}[n] & -L \le n \le L \end{cases} \tag{5.2-124}$$

where f_{cheby} is a pre-computed ramp function. A pre-ripple repsonse r_b^{pre} and post-ripple response r_b^{post} are obtained from the original impulse response $h'_b[n]$ as follows:

$$r_b^{pre}[n] = h'_b[n+L]s[n]\,h_{win}[n] \tag{5.2-125}$$

$$r_b^{post}[n] = -\,r_b^{pre}[-n], \tag{5.2-126}$$

where h_{win} is an M-point Hanning window and the post-ripple response is obtained by time-reversing the pre-ripple response. A final window is defined as

$$h_{fwin}[n] = \begin{cases} \sin(\pi\frac{n+1}{2L}), & n < L \\ 1, & L \le x \ge 2L \\ \sin(\pi\frac{M-n}{2(M+1-2L)}), & 2L \le x \le M \\ 0, & otherwise \end{cases} \tag{5.2-127}$$

The mofidied impulse response h_b^{ld} is then obtained by subtracting the pre-ripple response and adding the post-ripple response:

$$h_b^{ld}[n] = (h'_b[n] - r_b^{pre}[n-L] + r_b^{post}[n-L])\,h_{fwin}[n]. \tag{5.2-128}$$

As an optimisation, filters in the original 48 kHz design corresponding to bands where the energy above the Nyquist frequency is essentially zero are used as in eq. (5.2-122) whilst filters for bands having nearly all their energy above the Nyquist frequency are not used for processing. For bands 8 – 10 at 16 kHz sampling rate, and bands 10 – 12 at 32 kHz sampling rate, which contain energy above and below the Nyquist frequency, the filters as given by eq. (5.2-128) are used.

Editor's note: Complete?

Note: $\boldsymbol{Filt_{abs,5ms}^{mdft}(b,k)}$ is the short stride filterbank magnitude response referred in Eqn (5.4-16).

5.3 Stereo audio operation

5.3.1 Stereo format overview

The IVAS codec supports native stereo coding at bitrates from 13.2 kbps to 256 kbps. The stereo format encoder consists of stereo modules operating in time domain or frequency domain representing the following stereo modes:

- Time-domain (TD) stereo mode
- Discrete Fourier Transform (DFT) domain parametric stereo mode

- Modified Discrete Cosine Transform (MDCT) domain stereo mode

The use of stereo modes at different bitrates is summarized in Table 5.3-1.

Table 5.3-1: Overview of bitrates, bandwidths, and coding modes in Stereo format

Bitrate [kbps]	Maximum bandwidth	Stereo mode
13.2 – 24.4	SWB	DFT / TD stereo
32	FB	DFT / TD stereo
48 - 256	FB	MDCT stereo

At low bitrates from 13.2 kbps up to 32 kbps the IVAS stereo codec switches between TD stereo mode and DFT stereo mode in a so-called Unified stereo module (Clause 5.3.3.2). The selection between DFT stereo and TD stereo is done in each frame based on the stereo classifier decision (Clause 5.3.3.2.4). At higher bitrates from 48 kbps up to 256 kbps the IVAS stereo codec then operates in MDCT stereo mode (Clause 5.3.3.3).

In addition, the IVAS codec supports a stereo downmix processing to generate a mono signal for EVS-interoperable encoding (Clause 5.3.2).

5.3.2 Stereo downmix operation for EVS mono coding

5.3.2.1 Modes of downmix

This downmix process aims to produce a mono signal for keeping the interoperability with EVS mono coding without additional delay. Depending on the characteristics of the input signal, one operation mode out of two types of downmix tools is selected every frame. One mode is a simple weighted sum of two channels based on phase-only correlation (clause 5.3.2.2), the other is a sum of two channels modified by phase compensation (clause 5.3.2.3). The mode selection rule and the handling of mode switching are described in clauses Mode selection rule5.3.2.4 and 5.3.2.5, respectively.

5.3.2.2 Phase-only correlation (POC) mode

5.3.2.2.1 Overview

This tool generates a downmixed monaural signal by an adaptive weighted sum of two input channels. This is achieved by detecting the preceding channel based on an efficient calculation of ITD (inter-channel time difference) in the frequency domain. According to the detected time difference between the two channels, a mixed monaural signal is produced with a larger weight for the preceding channel signal.

5.3.2.2.2 ITD analysis

5.3.2.2.2.1 Cross talk adding procedure in the frequency domain

The polarity of the ITD, which can be efficiently derived through phase-only spectrum in the frequency domain shows the preceding channel. Channel 1 and 2 signals in the time domain are denoted as $s_1(n)$ and $s_2(n)$, and corresponding frequency domain spectrum are represented by $S_1^{FFT}(k)$, $S_2^{FFT}(k)$, with time domain sine window $w(n)$ and frame length L.

$$S_i^{FFT}(k) = \frac{1}{\sqrt{L}} \sum_{n=0}^{L-1} w(n+1) s_i(n+1) e^{-j\frac{2\pi kn}{L}} \qquad (5.3\text{-}1)$$

In order to obtain robust results for various input signals, an intentional cross talk process with one sample delay in the time domain is applied. The modified complex spectrums $S_1^{\#}(k)$, $S_2^{\#}(k)$ are equivalently generated by adding another channel spectrum multiplied by a small value constant $\varepsilon(= 2\pi/L)$ and phase rotation of $e^{-j\frac{2\pi}{T}k}$ as

$$S_1^{\#}(k) = S_1^{FFT}(k) + \varepsilon S_2^{FFT}(k) \times e^{-j\frac{2\pi}{T}k} \qquad (5.3\text{-}2)$$

$$S_2^{\#}(k) = S_2^{FFT}(k) + \varepsilon S_1^{FFT}(k) \times e^{-j\frac{2\pi}{T}k} \qquad (5.3\text{-}3)$$

Using modified Spectrum Phase only spectrum$S_1^{\#}(k)$ and $S_2^{\#}(k)$, inter-channel phase difference complex spectrum $\phi(k)$ is defined as,

$$\phi(k) = \frac{S_1^{\#}(\mathrm{k})/|S_1^{\#}(\mathrm{k})|}{S_2^{\#}(\mathrm{k})/|\,S_2^{\#}(\mathrm{k})|} \qquad (5.3\text{-}4)$$

5.3.2.2.2.2 Approximated phase spectrum

Calculation of $\phi(k)$ needs operations of square root and division for each spectral bin. Instead, $\phi(k)$ can be approximated by simple quantization of the angle of $\phi^*(k)$ by making use of the property that $\phi(k)$ has complex values on a unit circle. The angle $\phi^*(k)$ is same as that of inter-channel difference spectrum $Y(k)$ which is derived by multiplication of $S_1^{\#}(\mathrm{k})$ *and* conjugate of $S_2^{\#}(\mathrm{k})$, $\overline{S_2^{\#}(\mathrm{k})}$ as

$$Y(k) = S_1^{\#}(k) * \overline{S_2^{\#}(k)} \qquad (5.3\text{-}5)$$

$\phi(k)$ can be approximated by the polarity of $u(k) = real(Y(k))$ and $v(k) = imag(Y(k))$ and quantized values of $\phi^*(k)$, $|u_Q(k)|$ and $|v_Q(k)|$ shown in **Table 5.3-2**. The weighted phase spectrum is defined a*s* $\phi^*(k)$, where $w(k) = sin(k\pi/L)\,(1 - 0.75k/L)$.

$$\phi^*(k) = w(k)\phi_Q(k) = w(k)(sign(u(k)) \times |u_Q(k)|, sign(v(k)) \times |v_Q(k)|\,) \qquad (5.3\text{-}6)$$

The lower resolution is applied for lower frequency bins due to the small weight values, which allow bigger approximation errors due to quantization. For the lowest frequency bins up to the 10th, a smaller value is used for the quantized phase angle, while for other frequency regions, quantized angle values have equal spacing.

When Search intensity *R* equals 0 (0 bit in case of transmission of code), $(|u(k)|,\ |v(k)|)$ can be looked up from **Table 5.3-2** for the same quadrant of $Y(k)$. In case *R* is positive, the nearest angle region of $(|u_Q(k)|\ ,\ |v_Q(k)|\ )$ with that of $(|u(k)|,\ |v(k)|)$ is detected among 2^R regions by R times binary search with thresholds, *T,* such as $|u(k)| > |v(k)| \times T$.

Table 5.3-2 Quantized values $|u_Q(k)|$, $|v_Q(k)|$on a unit circle. (in case only the real and imaginary parts are positive)

Frequency bin index *k*	Search intensity *R*	Quantized phase spectrum $(|u_Q(k)|,\ |v_Q(k)|)$	Quantized phase In angle value [rad]
$k < 10$ or w(k) = 0	0	(0.866, 0.5)	0.5236
$10 \le k < L/16$	0	(0.7071, 0.7071)	$\pi/4$
$L/16 \le k < L/8$	1	(0.923880, 0.382683)	$\pi/8$
		(0.382683, 0.923880)	$3\pi/8$
others	2	(0.980785, 0.195090)	$\pi/16$
		(0.831470, 0.555570)	$3\pi/16$
		(0.555570, 0.831470)	$5\pi/16$
		(0.195090, 0.980785)	$7\pi/16$

5.3.2.2.2.3 Calculation of phase only correlation

Time domain inter-channel cross correlation between the channel one and two at time difference of τ is defined as $\psi(\tau)$ and derived by inverse Fourier transformation of the weighted phase spectrum $\phi^*(k)$. Squared inter-channel correlation $C(\tau)$ and filtered inter-channel correlation $P(\tau)$ is derived from $C(\tau)$ as follows,

$$C(\tau) = \psi^2(\tau) = \left(\frac{1}{\sqrt{L}}\sum_{k=0}^{T-1}\phi^*(k)e^{j\frac{2\pi k\tau}{L}}\right)^2 \qquad (5.3\text{-}7)$$

$$P(\tau) = 1.25C(\tau) - 0.125(C(\tau+1) + C(\tau-1)) \qquad (5.3\text{-}8)$$

$$\begin{cases} P(L/2+ITD_{1,prev}) = 0.79P_{prev}(L/2+ITD_{1,prev}) + 0.21P(L/2+ITD_{1,prev}) \\ P(L/2-ITD_{2,prev}) = 0.79P_{prev}(L/2-ITD_{2,prev}) + 0.21P(L/2-ITD_{2,prev}) \\ P(\tau) = 0.78P_{prev}(\tau) + 0.22P(\tau)\ when\ \tau \neq L/2+ITD_{1,prev}\ and\ \tau \neq L/2-ITD_{2,prev} \end{cases} \quad (5.3\text{-}9)$$

Here, $P_{prev}(\tau)$ is the filtered correlation of the previous frame. $ITD_{1,prev}$ (resp. $ITD_{2,prev}$) is the previous frame's candidate of *ITD,* assuming the channel one (resp. two) is preceding.

5.3.2.2.2.4 Calculation of ITD and weighting coefficients

To find the most preferable *ITD*, two candidates of ITD_1^{CAND} (negative, assuming the channel one is preceding) and ITD_2^{CAND} (positive, the channel two is preceding) are searched, which give the largest value of inter-channel correlation $P(L/2-ITD_1^{CAND})$ and $P(L/2+ITD_2^{CAND})$ within the rage of $0 \leq |ITD_1^{CAND}|, |ITD_2^{CAND}| \leq$ *limit.* The *limit* is the maximum time difference in samples corresponding to the time difference of 5.16 ms; namely *limit* is 247 when the sampling rate is 48 kHz. Then, the updating process decides whether to update the probable *ITD*s, ITD_1 and ITD_2 by $ITD_1^{CAND}$and ITD_2^{CAND}. *ITD* is finally selected from ITD_1 and ITD_2 based on the quality factors Q_1, Q_2 of the peaks of associated inter-channel correlation values, depending on the activities of the previous frame, such as ON_1 and OFF_1 . For the channel one, $Q_1$and ITD_1 is calculated as in the following process. For the channel two, the same process is used to obtain Q_2 and ITD_2

Peak range PR_1 is defined as $PR_1 = |ITD_1^{CAND}| + limit/32$. Peak width PW_1 and cumulative peak value Q_1^{CUMU} are defined as

$$PW_1 = \sum_{i=1}^{PR_1} \left(P(L/2 - ITD_1^{CAND} + i) > 0.38 \times P(L/2 - ITD_1^{CAND}) \right) + \sum_{i=1}^{PR_1} \left(P(L/2 - ITD_1^{CAND} - i) > 0.38 \times P(L/2 - ITD_1^{CAND}) \right) \quad (5.3\text{-}10)$$

$$Q_1^{CUMU} = P(L/2 - ITD_1^{CAND}) + \sum_{i=1}^{PR_1} \left(P(L/2 - ITD_1^{CAND} + i) + P(L/2 - ITD_1^{CAND} - i) \right), \quad (5.3\text{-}11)$$

using an updated PW_1 : $PW_1 = 0.875 \times PW_{1,prev} + 0.125 \times PW_1$, and $\varepsilon = 0.25 \times PW_1/L$,

Q_1 is derived as a relative difference between the correlation peak and the average correlation around it when the peak is large enough compared to the peak width.

$$Q_1 = (1 - (Q_1^{CUMU}/(2 \times PR_1 + 1) + (\varepsilon \times PW_1))/(P(L/2 - ITD_1^{CAND}) + (\varepsilon \times PW_1))) \quad (5.3\text{-}12)$$

The following process decides whether to update ITD_1 by ITD_1^{CAND}or leave it as ITD_1 used in the previous frame. If $ON_1\ equals\ 1$, namely the channel one was active (preceding) in the previous frame, ITD_1 and $peakQ_1$ is of the current frame is determined as follows;

```
if ($Q_1 > (0.3 - 0.2 \times |ITD_1^{CAND}|)/limit \times peakQ_1$) {$ITD_1 = 0,\ ON_1 = 0, Q_1 = 0,\ peakQ_1 = 0$}
else if ( ($Q_1 > 1.25 \times peakQ_1$) {$ITD_1 = ITD_1^{CAND}$ and $peakQ_1 = \max(peakQ_1, Q_1)$}
```

If $ON_1\ equals\ 0$, namely the channel one was not active (not preceding) in the previous frame, ITD_1 and $Q_1$is updated as follows.

```
if (($Q_1 < (0.75 - 0.2 \times |ITD_1^{CAND}|) / limit$) { $ITD_1 = 0$ and $Q_1 = 0$. }
else {$ITD_1 = ITD_1^{CAND}$ and $ON_1 = 1$}
```

Depending on Q_1, Q_2 and previous ITD_{prev}, *ITD* for this frame is set according to the rule as in Table 5.3-3.

Table 5.3-3 Determination of ITD

Primary condition	Secondary condition	decision
If(ON_1 & $OFF_{1,prev}$ & ON_2 & $OFF_{2,prev}$)	If $(Q_1 > Q_2)$	$ITD = ITD_1$
	else	$ITD = ITD_2$
else if (ON_1 & $OFF_{1,prev}$ & $(Q_1 > (Q_2 - 0.1))$)		$ITD = ITD_1$
else if (ON_2 & $OFF_{2,prev}$ & $(Q_2 > (Q_1 - 0.1))$)		$ITD = ITD_2$
else if (Q_1 > (Q_2 + 0.25))		$ITD = ITD_1$
else if (Q_2 > (Q_1 + 0.25))		$ITD = ITD_2$
else if (ITD_{prev} ==0)		$ITD = 0$
Else	If (ITD_{prev} > 0)	$ITD = ITD_1$
	Else	$ITD = ITD_2$

Downmix weight *h* is determined by a parameter defined as *conf* and *ITD*, depending on the preceding channel.

$$conf = \sqrt{(|Q_{1-} - Q_2|)} \tag{5.3-13}$$

$$conf = 0.78 \times conf_{prev} + 0.22 \times conf \tag{5.3-14}$$

$$h = \begin{cases} 0.5 \times (1 - conf) & when\ ITD > 0 \\ 0.5 & when\ ITD = 0 \\ 0.5 \times (1 + conf) & when\ ITD < 0 \end{cases} \tag{5.3-15}$$

We can see that if Q_1 is bigger than Q_2, *ITD* is negative, and the channel one signal is likely preceding to the channel two. As a result, the weighting to the channel one, h is bigger than 0.5, and a downmixed signal is generated with bigger weight for the preceding channel signal.

5.3.2.2.3 Downmix process

The first mixing process is executed as follows with the window $w(n) = sin(8\pi n/L)$ and $w(L/16) = 1$.

$$d(n) = \begin{cases} (h_{prev} + (h - h_{prev})w(n))s_1(n) + (1 - (h_{prev} + (h - h_{prev})w(n)))\, s_2(n) & 0 \le n < L/16 \\ hs_1(n) + (1 - h)s_2(n) & L/16 \le n < \boxed{?}\boxed{?} \end{cases} \tag{5.3-16}$$

$d(n)$ is modified by the weighting coefficients g_1 and g_2 derived from the energy E_1 for $S_1(n)$, E_2 for $S_2(n)$ and E_D for $d(n)$. The following trapezoidal windowed frame energy E_y for signal $y(n)$ is used.

$$E_y = \sum_{n=0}^{L/16-1} w(n)^2 y^2(n) + \sum_{i=L/16}^{L-L/16-1} y^2(n) + \sum_{i=L-L/16}^{L-1} w(n)^2 y^2(n) \tag{5.3-17}$$

$$E_y = E_{y.prev} * 0.25 + (E_y/L) * 0.75 \tag{5.3-18}$$

$$w(i) = \begin{cases} 16(0.5 + i)/L & 0 \le i < L/16 \\ 16(L - 0.5 - i)/L) & L - L/16 \le i < L \end{cases} \tag{5.3-19}$$

For enhancing the weight for the preceding channel, weighting coefficients g_1 for $s_1(n)$ and g_2 for $s_2(n)$ are derived from the frame energy as in Table 5.3-4.

Table 5.3-4 Determination of g_1 and g_2 with $\varepsilon = 1024$

State	energy	g_1	g_2
$g_{2,prev} = 0$	$(E_1 \times 4 > E_2)$	$max(1. - \sqrt{\frac{(E_D + \varepsilon)}{(E_1 + \varepsilon)}}, 0.0)$	0.0
	$(E_1 \times 4 \leq E_2)$	0.0	$max(1. - \sqrt{\frac{(E_D + \varepsilon)}{(E_2 + \varepsilon)}}, 0.0)$
$g_{2,prev} \neq 0$	$(E_1 > E_2 \times 4)$	$max(1. - \sqrt{\frac{(E_D + \varepsilon)}{(E_1 + \varepsilon)}}, 0.0)$	0.0
	$(E_1 \leq E_2 \times 4)$	00.	$max(1. - \sqrt{\frac{(E_D + \varepsilon)}{(E_1 + \varepsilon)}}, 0.0)$

Downmixed signal $d(n)$ is further modified as

$$d(n) = \begin{cases} d(n) + (g_{1,prev} + (g_1 - g_{1,prev})w(n))s_1(n) & \\ \qquad + (g_{2,prev} + (g_2 - g_{2,prev})w(n))s_2(n) & \text{when } 0 \leq n < L/16 \\ d(n) + g_1 s_1(n) + g_2 s_2(n) & \text{when } L/16 \leq n < L \end{cases} \quad (5.3\text{-}20)$$

Finally, $d(n)$is a monaural signal used to input the EVS encoder, which generates interoperable bit streams for the EVS decoder.

5.3.2.3 Phase compensation (PHA) mode

Phase compensation in time domain is used to modify the input stereo signal. Left and right channels are filtered adaptively by short impulse responses approximating phase differences; two phase compensation modes are defined (ICPD or +/-ICPD/2 or /4) and adaptively switched on a frame basis.

5.3.2.4 Mode selection rule

The selection rule depending on several criteria (transient detection, inter-channel level difference, realigned inter-channel correlation).

5.3.2.5 Handling of mode switching

In a transition frame, i.e. when the downmix mode in the current frame is not the same as in the previous frame, the downmix output is crossfaded between the output of the selected mode and the output of the other mode.

5.3.3 Stereo coding

5.3.3.1 Common stereo coding tools

5.3.3.1.1 Inter-channel Time Difference (ITD) detection

The inter-channel time difference (ITD) is derived using an adaptive partial whitening GCC-PHAT algorithm. A cross-correlation function $r_{01}(m, l)$ is defined as

$$r_{01}(m, l) = DFT^{-1}\left(\frac{S_{01,LP}(m,k)}{|S_{01,LP}(m,k)|^{\alpha}}\right) \quad (5.3\text{-}21)$$

where $S_{01,LP}(m, k)$ is an adaptively filtered cross-spectrum which is filtered using a low-pass filter coefficient β, α depends on the estimated background noise level of the current frame m and l is a lag describing the inter-channel time difference in samples.

$$S_{01}(m, k) = S_0(m, k)S_1(m, k)^* \quad (5.3\text{-}22)$$

$$S_{01,LP}(m, k) = \beta S_{01}(m, k) + (1 - \beta)S_{01,LP}(m - 1, k) \quad (5.3\text{-}23)$$

where $S_0(m,k)$ and $S_1(m,k)$ are the short time DFT spectra of the time domain input channels $s_0(m,n)$ and $s_1(m,n)$ as derived in [DFT STFT analysis] and '$*$' denotes complex conjugate. In case of active frame encoding, the low-pass filter coefficient $\beta = sfm$ is the spectral flatness measure of the frame, calculated according to

$$sfm = \max(sfm_0, sfm_1)$$
$$sfm_0 = \frac{\exp\left(\frac{1}{N_{FFT}}\sum_{k=0}^{N_{FFT}}\log|S_0(m,k)|\right)}{\frac{1}{N_{FFT}}\sum_{k=0}^{N_{FFT}}|S_0(m,k)|} \tag{5.3-24}$$
$$sfm_1 = \frac{\exp\left(\frac{1}{N_{FFT}}\sum_{k=0}^{N_{FFT}}\log|S_1(m,k)|\right)}{\frac{1}{N_{FFT}}\sum_{k=0}^{N_{FFT}}|S_1(m,k)|}$$

For noisy speech $\alpha = 0.8$ and for clean speech it is $\alpha = 1.0$. To handle the transitions from active segments to inactive segments, an additional filtered cross-spectrum $S_{01,LPCNG}(m,k)$ is maintained. It is evaluated during the hangover period in the start of an inactive segment.

$$S_{01,LPCNG}(m,k) = \beta_{CNG}S_{01}(m,k) + (1-\beta_{CNG})S_{01,LPCNG}(m-1,k) \tag{5.3-25}$$

where β_{CNG} is an adaptive filter coefficient. If we are in the first frame of a VAD hangover period, the filtered cross-spectrum memory is reset to zero, $S_{01,LPCNG}(m-1,k) := 0$, meaning that the filtered cross-spectrum for this frame is defined according to

$$S_{01,LPCNG}(m,k) = \beta_{CNG}S_{01}(m,k) \tag{5.3-26}$$

During DTX hangover a faster adaptation of $S_{01,LPCNG}(m,k)$ is achieved by increasing the low-pass filter coefficient β_{CNG} according to

$$\beta_{CNG} = \max\left(sfm, \min\left(0.8, \frac{8}{N_{exp}+N_{upd}}\right)\right) \tag{5.3-27}$$

where N_{exp} is the number of expected updates to $S_{01,LPCNG}(m,k)$ during the hangover period before it is used and N_{upd} is the actual number of updates since the first hangover frame where the filtered cross-spectrum memory was reset to zero. In the first hangover frame, N_{exp} is calculated as

$$N_{exp} = 1 + \min(N_{rem\_dtx\_ho0}, N_{rem\_dtx\_ho1}) \tag{5.3-28}$$

where $N_{rem\_dtx\_ho0}, N_{rem\_dtx\_ho1}$ are the remaining hangover frames as indicated by [front VAD 0 and 1] respectively. For the remaining hangover frames, N_{exp} is updated according to

$$N_{exp} = \begin{cases} N_{exp}^{[-1]} + 1 + \min(N_{rem\_dtx\_ho0}, N_{rem\_dtx\_ho1}), N_{upd} \geq N_{exp} \\ N_{exp}^{[-1]}, \qquad otherwise \end{cases} \tag{5.3-29}$$

In the first CNG frame after the hangover period, the low-pass filtered CNG cross-spectrum is copied to the low-pass filtered cross-spectrum

$$S_{01,LP}(m,k) = S_{01,LPCNG}(m,k) \tag{5.3-30}$$

and the GCC-PHAT algorithm is evaluated as in (5.3-21). When the CNG period has started, the purpose of $S_{01,LPCNG}(m,k)$ is shifted from fast ITD adaptation to instead be used for coherence estimation, as described in (5.3-21). During the following CNG frames, a slow adaptation of $S_{01,LPCNG}(m,k)$ is maintained by setting $\beta_{CNG} = 1/32$.

The ITD candidate $ITD_{cand}(m)$ of the current frame m is identified as the lag l which yields the maximum absolute value $r_{max}(m)$ of $r_{12}(m,l)$.

$$\begin{cases} ITD_{cand}(m) = \underset{l}{\operatorname{argmax}}|r_{12}(m,l)| \\ r_{max}(m) = |r_{12}(ITD_{cand}(m))| \end{cases} \tag{5.3-31}$$

However, background noises and reverberation may introduce noise in the cross-correlation, making the identification of a maximum lag more difficult. For this reason, $r_{max}(m)$ is compared to an adaptive threshold $r_{thr}(m)$. In case $r_{max}(m) > r_{thr}(m)$ the candidate ITD is selected as the valid ITD of the current frame, $ITD(m) = ITD_{cand}(m)$. In case $r_{max}(m)$ is hovering near the threshold $r_{thr}(m)$ for several consecutive frames m, the stability of the ITD may suffer. To stabilize the ITD the following measures are taken. $r_{max}(m)$ is low-pass filtered to form a stability estimate of $ITD_{cand}(m)$.

$$r_{stab}(m) = \begin{cases} \alpha_r r_{max}(m) + (1-\alpha_r) r_{stab}(m-1), & r_{max}(m) > r_{thr}(m) \\ 0, & r_{max}(m) \le r_{thr}(m) \end{cases} \qquad (5.3\text{-}32)$$

where the low-pass filter coefficient α_r is set using a fast-attach-slow-decay strategy according to

$$\alpha_r = \begin{cases} 0.9, & r_{max}(m) > r_{stab}(m-1) \\ 0.1, & r_{max}(m) \le r_{stab}(m-1) \end{cases} \qquad (5.3\text{-}33)$$

An ITD counter $ITD_{cnt}(m)$ is maintained to keep track of consecutive valid ITDs.

$$ITD_{cnt}(m) = \begin{cases} \max\left(ITD_{cnt,max}, ITD_{cnt}(m-1)+1\right), & r_{max}(m) > r_{thr}(m) \\ 0, & r_{max}(m) \le r_{thr}(m) \end{cases} \qquad (5.3\text{-}34)$$

where the maximum value of the ITD counter is $ITD_{cnt,max} = 2$. If an ITD is not found to be valid, i.e. $r_{max}(m) \le r_{thr}(m)$, and the maximum value of the ITD counter was reached in the preceding frame $ITD_{cnt}(m-1) = ITD_{cnt,max}$, an ITD hangover counter is calculated according to

$$ITD_{HO}(m) = \max\left(0, \min\left(6, 11 + r_{stab}(m-1)\frac{50}{3}\right)\right) \qquad (5.3\text{-}35)$$

If a valid ITD is not found $r_{max}(m) \le r_{thr}(m)$ and the hangover count exceeds zero $ITD_{HO}(m) > 0$, the previous ITD is selected for the current frame $ITD(m) = ITD(m-1)$. If the hangover count is at zero $ITD_{HO}(m) = 0$, the ITD of the current frame is zeroed $ITD(m) = 0$.

5.3.3.2 Unified stereo

5.3.3.2.1 TD-based stereo

The TD stereo coder is a low-rate low-complexity parametric stereo technology where the stereo input signal is encoded in time domain. It is especially effective for scenes where the correlation between the left and the right channel of the input audio signal is low, when the background noise is fluctuating a when an interfering talker is present. The classifier of uncorrelated input signals and the cross-talk detector, described in clause XX, are used to select frames suitable for the TD stereo coder. The TD stereo coder comprises a method for combining the the left and the right channel of the input stereo signal to obtain the primary and the secondary channel. The bitrate is adaptively distributed between the primary and the secondary channel to maximize the quality of the encoded stereo signal. The secondary channel usually requires lower amount of bits due to leveraging some parameters from the the primary channel.

The TD stereo encoder is schematically depicted in Fig. 5.3-1.

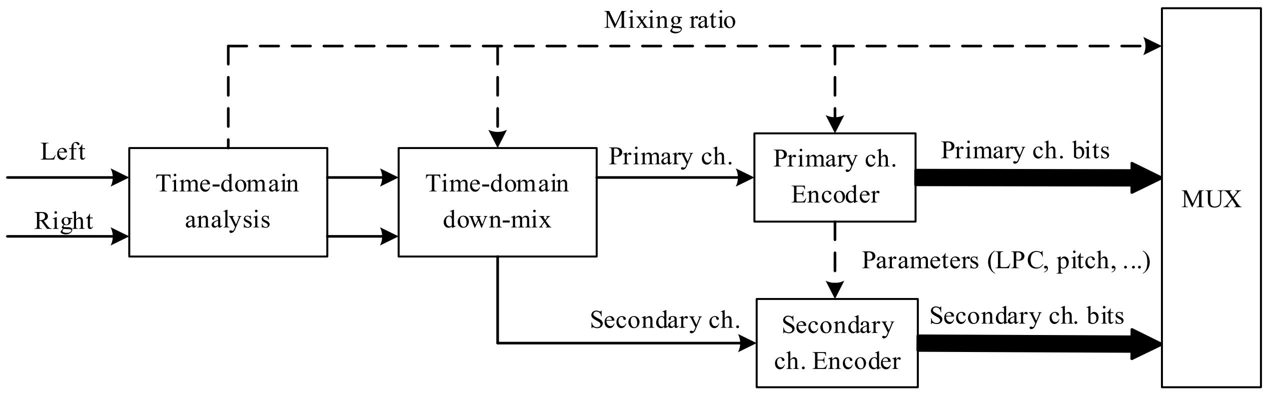


Figure 5.3-1: TD stereo encoder

The left channel and the right channel of the input stereo signal signal are mixed together such that a primary channel $Y(n)$ and a secondary channel $X(n)$ are formed in time domain. A minimum bitrate is allocaye for the encoding of the secondary channel and its associated bit budget varies on a frame-by-frame basis depending of the content. The bits that are not used in the encoding of the secondary channel are used in the encoding of the primary channel. The bit budget of the secondary channel is always less than the bit budget of the primary channel. The sum of the bitrates required by the primary and the secondary channel is equal to the total bitrate which is constant in every frame. The time-domain down-mixing mechanism may be skipped in situations where the left channel and the right channel are incoherent, resulting in a Left-Right TD stereo submode (LRTD). In the LRTD submode the amount of bits in the primary channel and the secondary channel are close to equal. On the contrary, when a maximum emphasis is put on the primary channel, the bit budget of the secondary channel is aggressively forced to a minimum.

5.3.3.2.1.1 Time-domain stereo downmix

The TD stereo downmix converts the signals in the left channel and the right channel into primary channel and secondary channel. The functionality of the TD stereo downmix is schematically desribed in Fig. XX.

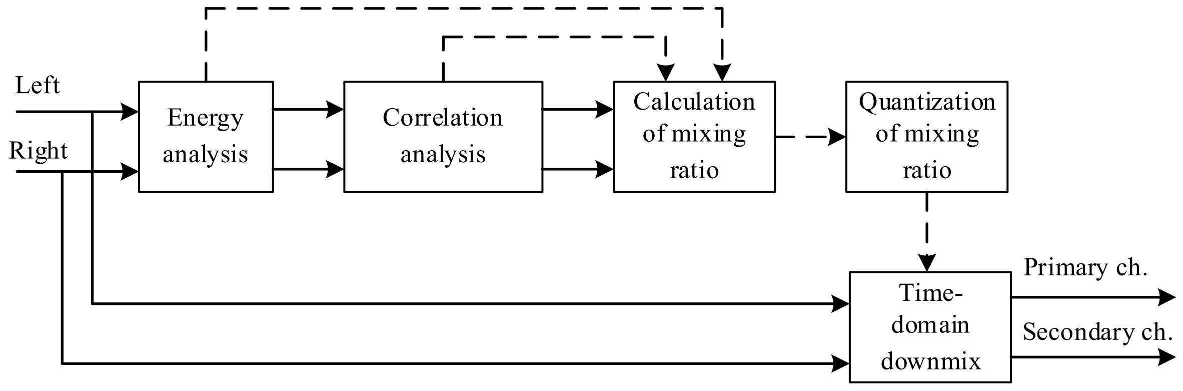


Figure 5.3-2: TD stereo downmix

The TD stereo downmix is designed in such a way that the generated signals maintain their speech character allowing the ACELP codec to be efficiently applied on them.

The TD stereo downmix is based on the inter-channel (left-right) energy analysis. First, the average energy of each input channel in the current frame is computed as

$$rms_L = \sqrt{\frac{\sum_{n=0}^{N-1} L(n)^2}{N}} \qquad rms_R = \sqrt{\frac{\sum_{n=0}^{N-1} R(n)^2}{N}} \tag{5.3-36}$$

where the subscripts L and R stand for the left channel and the right channel, respectively and N corresponds to the number of samples per frame. The RMS values are then used to compute the long-term RMS values of each channel as

$$\begin{aligned} \overline{rms}_L &= 0.6 \cdot \overline{rms}_L^{[-1]} + 0.4 \cdot rms_L \\ \overline{rms}_R &= 0.6 \cdot \overline{rms}_R^{[-1]} + 0.4 \cdot rms_R \end{aligned} \tag{5.3-37}$$

where the superscript $^{[-1]}$ denotes the previous frame. The long-term RMS values are then used to determine the energy trend in each channel as follows

$$\begin{aligned} \overline{rms\_dt}_L &= \overline{rms}_L - \overline{rms}_L^{[-1]} \\ \overline{rms\_dt}_R &= \overline{rms}_R - \overline{rms}_R^{[-1]} \end{aligned} \tag{5.3-38}$$

The energy trend contains the information whether temporal events captured in the input signals are fading out or if they are changing side. The long-term RMS values and the energy trend are also used to determine the convergence speed of the long-term correlation difference as will be shown later.

The correlations of the left channel and the right channel against the passive mono downmix in the current frame are computed with

$$G_L = \frac{\sum_{n=0}^{N-1}(L(n)\cdot M(n))}{\sum_{i=0}^{N-1} M(n)^2}$$
$$G_R = \frac{\sum_{n=0}^{N-1}(R(n)\cdot M(n))}{\sum_{i=0}^{N-1} M(n)^2} \quad (5.3\text{-}39)$$

where the passive mono downmix is calculated as

$$M(n) = \frac{L(n)+R(n)}{2} \quad (5.3\text{-}40)$$

The correlations G_L and G_R are then smoothed with

$$\overline{\mathrm{G}_L} = \alpha \cdot \bar{G}_L^{[-1]} + (1-\alpha)\cdot G_L$$
$$\overline{\mathrm{G}_R} = \alpha \cdot \bar{G}_R^{[-1]} + (1-\alpha)\cdot G_R \quad (5.3\text{-}41)$$

where α is the convergence speed and the superscript $^{[-1]}$ denotes the previous frame. Finally, the long-term correlation difference is calculated as

$$\overline{G_{LR}} = \overline{\mathrm{G}_L} - \overline{\mathrm{G}_R} \quad (5.3\text{-}42)$$

The convergence speed α can have the value of 0.8 or 0.5 depending of the long-term energies computed in eq. (5.2-37) and the energy trend computed in eq. (5.3-38). The convergence speed α is set to the value of 0.8 if the long-term energies are pointing in the same direction, the long-term correlation difference $\overline{G_{LR}}$ in the current frame and the long-term correlation difference $\bar{G}_{LR}^{[-1]}$ in the previous frame are both low and if at least one of the long-term energies has a value above certain pre-defined threshold. This basically ensures that the energies in both channels are evolving smoothly and that there is no fast change of energy from one channel to the complementary channel and that at least one of the channels contains a meaningful amount of energy. Otherwise, when the long-term energies are pointing in opposite directions or when the long-term correlation difference $\overline{G_{LR}}$ is too high or if both input channels have a low energy, then the converegence speed α is set to the value of 0.5 to increase the adaptation speed of the long-term correlation $\overline{G_{LR}}$.

Once the long-term correlation difference is properly estimated it is converted into a mixing factor β which is then quantized with a scalar quantizer and the index is transmitted to the decoder in the bitstream. The mixing factor β represents two aspects of the stereo input combined in a single parameter. The mixing factor β represents the proportion of the input channels that are combined together to create the primary channel. The mixing factor β also represents an energy-scaling factor that, when applied to the primary channel, would result in a signal energetically close to the passive mono downmix. Thus, it allows the primary channel to be decoded as a mono signal without the need of side information (auxiliary stereo parameters). The mixing factor β may also be used to rescale the energy of the secondary channel before its encoding such that its energy is closer to the optimal energy range of the underlying encoder.

The mapping between the long-term correlation difference $\overline{G_{LR}}$ and the mixing factor β is done by limiting the long-term correlation difference to the interval of -1.5 to 1.5 and linearly scaling in the interval of 0.0 to 2.0. This is done as follows

$$G'_{LR} = \begin{cases} 0, & \overline{G_{LR}} \le -1.5 \\ \frac{2}{3}\cdot\overline{G_{LR}} + 1.0, & -1.5 < \overline{G_{LR}} < 1.5 \\ 2, & \overline{G_{LR}} \ge 1.5 \end{cases} \quad (5.3\text{-}43)$$

After the linearization, the long-term correlation difference G'_{LR} is converted to the mixing factor β using the cosine function. That is

$$\beta = \frac{1}{2}\cdot\left(1 - cos\left(\pi\cdot\frac{G'_{LR}}{2}\right)\right) \quad (5.3\text{-}44)$$

The primary channel $Y(n)$ and the secondary channel $X(n)$ are then calculated as a weighted sum of the left channel and the right channel using the following expressions

$$Y(n) = R(n)\cdot(1-\beta) + L(n)\cdot\beta, \qquad \text{for } n = 0,\dots,N-1$$
$$X(n) = L(n)\cdot(1-\beta) - R(n)\cdot\beta, \qquad \text{for } n = 0,\dots,N-1 \quad (5.3\text{-}45)$$

For the purposes of the TD stereo coder the long-term correlation difference G'_{LR} is also converted to the energy-normalization factor ε using the following relation

$$\varepsilon = -\frac{1}{2}(G'_{LR} - 1)^2 + 1.0 \tag{5.3-46}$$

The mixing factor β is quantized with a 5-bit scalar quantizer using 31 possible quantization levels. An odd number of quantization levels is used to allow for a left/right and mono/side coding. The last bit combination in the scalar quantizer is reserved as an indicator of an out-of-phase stereo input signal. This is explained below.

The TD downmix may have a problem if the signal in the left channel and the signal in the right channel have opposite phase. In that case the passive mono downmix obtained by summing the left channel and the right channel together would cancel each other. To ensure that this problem does not occur in the TD downmix explained above the energy of the passive mono downmix must be higher than the energy of the left channel of the energy of the right channel. Otherwise, the TD downmix enter a special mode with out-of-phase input.

In the special mode with out-of-phase input the mixing factor β is forced to the value of 1.0 and the secondary channel is encoded using the GENERIC or the UNVOICED coder type preventing the INACTIVE coder type from being selected inside the core coder. Furthermore, the special mode with out-of-phase input is signaled to the decoder by using the last bit combination in the scalar quantizer of the mixing factor where no energy scaling is applied.

5.3.3.2.1.2 LP filter coherence

The ACELP encoder in the secondary channel is optimized in such a way that it requires only a minimal amount of bits. To reduce the required amount of bits in the secondary channel certain parameters from the primary channel may be reused such as the LP filter or the OL pitch lag. First, a low-complexity pre-processing is performed on the signal in the secondary channel including the VAD (clause XX), LP analysis (clause XX) and OL pitch analysis (clause XX). Then, the characteristics of the secondary channel are analyzed and the signal is classified as GENERIC, UNOVICED or INACTIVE. The signal classification is done using the mechanism described in clause XX.

An important part of the bit-rate consumption in the secondary channel is the quantization of LP filter coefficients. Given that the spectral content of the secondary channel is often close to that of the primary channel the TD stereo coder may decide on reusing the LP filter coefficients of the primary channel in the secondary channel. The TD stereo coder performs LP analysis as described in clause XX on both the primary channel $Y(n)$ and the seconadry channel $X(n)$. Let's denote the LP filter coefficients of the primary channel as $A_Y(i)$, where $i = 0, \dots, M$ and M is the order of the LP filter. Using the same symbols, let's denote the LP filter coefficients of the secondary channel as $A_X(i)$. The TD stereo coder calculates two versions of LP residuals for the secondary channel. The first version of the LP residual is calculated by filtering the signal in the secondary channel with the LP filter of the primary channel. That is

$$r_Y(n) = \mathrm{X}(n) + \sum_{i=1}^{\mathrm{M}}\big(A_Y(i) \cdot \mathrm{X}(n-i)\big), \qquad n = 0, \dots, N-1 \tag{5.3-47}$$

The scond version of the of the LP residual is calculated by filtering the signal in the secondary channel with the LP filter of the secondary channel. That is

$$r_X(n) = \mathrm{X}(n) + \sum_{i=1}^{\mathrm{M}}\big(A_X(i) \cdot \mathrm{X}(n-i)\big), \qquad n = 0, \dots, N-1 \tag{5.3-48}$$

The absolute energy of the secondary channel is calculated using the following relation

$$E_X = 10 \cdot log_{10}\left(\sum_{n=0}^{N-1} X(n)^2\right) \tag{5.3-49}$$

The TD stereo coder calculates the energy of the first version of the LP residual signal $r_Y(n)$ as

$$E_{rY} = 10 \cdot log_{10}\left(\sum_{n=0}^{N-1} r_Y(n)^2\right) \tag{5.3-50}$$

and the the energy of the second version of the LP residual signal $r_X(n)$ as

$$E_{rX} = 10 \cdot log_{10}\left(\sum_{n=0}^{N-1} r_X(n)^2\right) \tag{5.3-51}$$

The TD stereo coder then calculates the LP predicition gains for both the first and the second version of the LP residual signal in the secondary channel by subtracting the energies of LP residuals from the absolute energy of the secondary channel. That is

$$\begin{gathered} G_Y = E_X - E_{rY} \\ G_X = E_X - E_{rX} \end{gathered} \tag{5.3-52}$$

Finally, the TD stereo coder calculates a ratio of gains $G_{Y|X}$ using the following relation

$$G_{Y|X} = \frac{G_Y}{G_X} \tag{5.3-53}$$

The decision on reusing the LP filter coefficients from the primary channel is also based on LP filter similarity between the primary and the secondary channel. The TD stereo coder calculates the similarity of LP filters by measuring the Euclidean distance between the line spectral pairs (LSPs) of the primary and the secondary channel. The LSPs of the primary and the secondary channel are converted from the respective LP filter coefficients using the procedure described in clause XX. Let's denote the LSPs of the primary channel as $lsp_Y(i)$ where $i = 0, \dots, M$ and M is the order of the LP filter. Similarly, let's denote the LSPs of the secondary channel as $lsp_X(i)$. The LSPs $lsp_Y(i)$ and $lsp_X(i)$ are weighted in such a way that emphasis is put on certain critical parts of the frequency spectrum. Then, the Eucidean distance between the LSPs is calculated with

$$dist = \sum_{i=0}^{M-1} \left(lsp_Y(i) - lsp_X(i)\right)^2 \tag{5.3-54}$$

If the ratio $G_{Y|X}$ is lower than the pre-defined threshold τ and the LP filter similarity measured by the Euclidean distance $dist$ is lower than the threshold σ, then the LP filter from the primary channel will be reused in the encoder of the secondary channel. Otherwise, the encoder of the secondary channel will use the LP filter $A_X(i)$ for quantization and in further processing. As a consequence, the secondary channel will need to allocate more bits to transmit the information about the LP filter to the decoder. The decision logic on re-using the LP filter from the primary channel is is illustrated in Fig. 5.3-3.

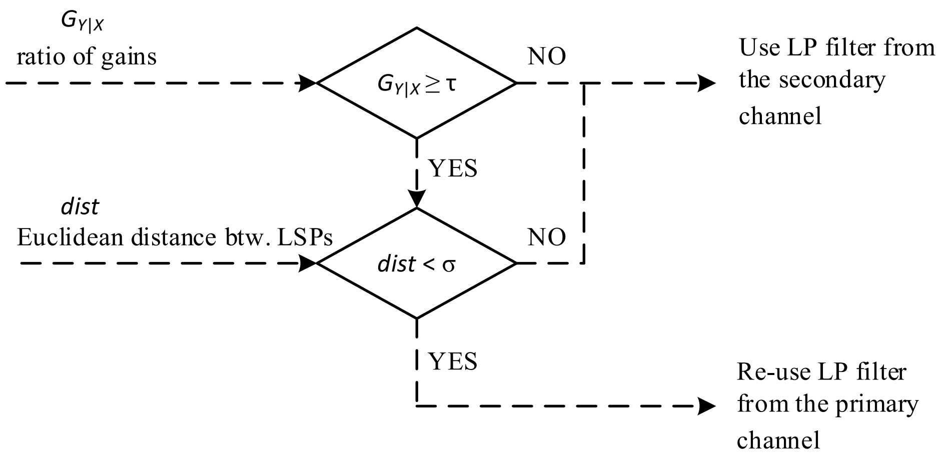

$G_{Y|X}$
ratio of gains
$G_{Y|X} \geq \tau$
NO
Use LP filter from the secondary channel
YES
$dist$
Euclidean distance btw. LSPs
$dist < \sigma$
NO
YES
Re-use LP filter from the primary channel

Figure 5.3-3: Re-using LP filter from the primary channel

Please, note that the TD stereo coder performs additional tests affecting the decision on reusing the LP filter from the primary channel. For example, when the secondary channel is assigned the UNVOICED coder type it is assumed that the spectral content is "easy" and there will be enough bits to encode the LP filter in the secondary channel. The LP filter from the primary channel is also reused when the energy of the LP residual signal in the secondary channel is very low or when the energy of the secondary channel itself is already too low.

5.3.3.2.1.3 Open-loop pitch coherence

The TD stereo coder analyzes the pitch coherence between the primary and the secondary channel. The coherence analysis is based on the open-loop pitch as described in clause XX. The pitch coherence is calculated with

$$S_{pc} = \left|\sum_{i=0}^{2} T_{OL\_Y}(i) - \sum_{i=0}^{2} T_{OL\_X}(i)\right| \tag{5.3-55}$$

where $T_{OL\_Y}(i)$ is the open-loop pitch of the primary channel in the ith half-frame.

If the pitch coherence is below a predetermined threshold Δ the OL pitch information from the primary channel may be re-used in the coding of the secondary channel. The re-using of the OL pitch is further conditioned on the available bit budget of the secondary channel and on the coder type of both the primary and the secondary channel. The decision upon re-using OL pitch information is described in Fig. 5.3-4.

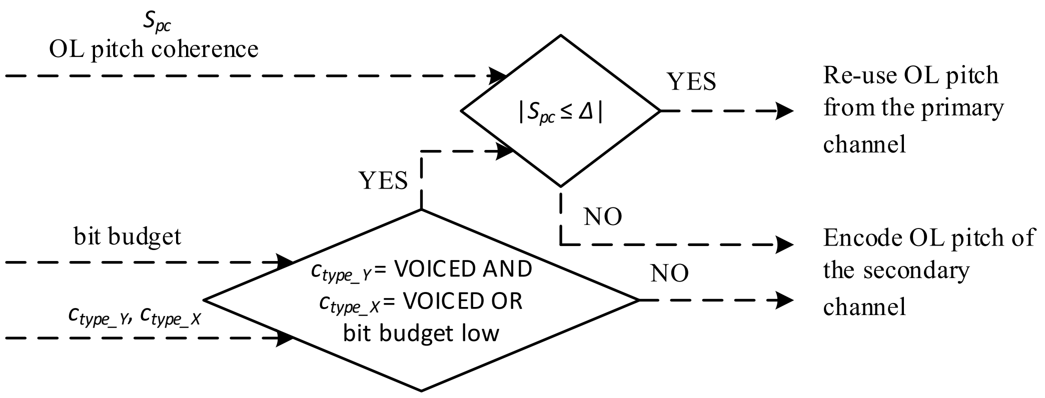

S_{pc}
OL pitch coherence
$|S_{pc} \leq \Delta|$
YES
Re-use OL pitch from the primary channel
YES
NO
bit budget
Encode OL pitch of the secondary channel
c_{type_Y}= VOICED AND c_{type_X}= VOICED OR bit budget low
NO
c_{type_Y}, c_{type_X}

Figure 5.3-4: Re-using OL pitch information from the primary channel

The condition in Fig. 5.3-4 relies on the bit budget available to the secondary channel. When the bit budget is below 14 kb/s and the pitch coherence S_{pc} is below or equal to 6 ($\Delta = 6$), the pitch information from the primary channel is re-used in the encoding of the secondary channel. When the bit budget available to the secondary channel is above 14 kb/s and below 26 kb/s, then both the primary channel and the secondary channel are considered as VOICED and the pitch coherence S_{pc} is compared to a lower threshold $\Delta = 3$. This leads to a smaller re-use rate of the pitch information of the primary channel at the bit-rate of 22 kb/s.

5.3.3.2.1.4 Excitation coding in the secondary channel using two or four subframes

The TD stereo coder adaptively allocates bits among the primary channel and the secondary channel based on the signal content. The schematic block diagram in Fig. 5.3-5 shows that the ACELP encoder in the secondary channel varies between a 4-subframe and 2-subframe scheme depending on the LP filter coherence between the primary channel and the secondary channel. The primary channel is encoded using the ACELP encoder as described in clause XX. Note, that all coder types can be selected in the ACELP encoder of the primary channel.

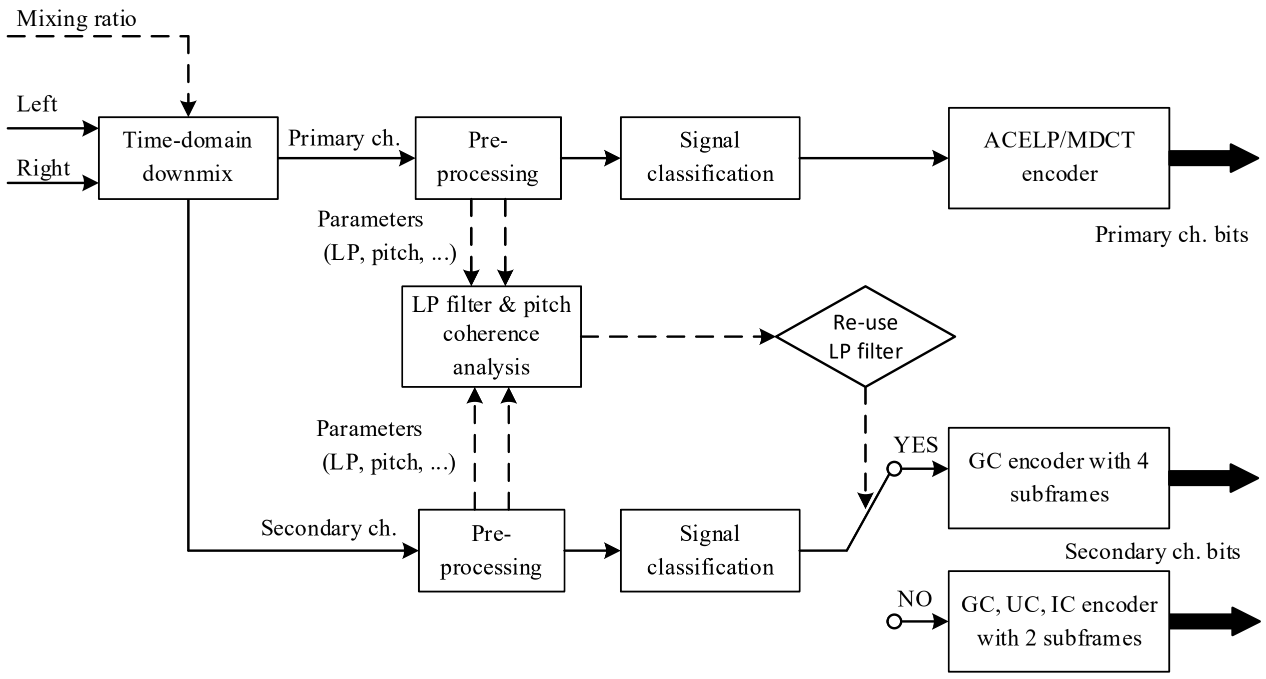


Figure 5.3-5: Dynamic encoding of the secondary channel in TD stereo mode

When the signal in the secondary channel is classified as GENERIC and the LP filter from the primary channel is reused, then the ACELP encoder as described in clause XX is applied with four subframes segmentation. In all other cases, the ACELP encoder as described in clause XX is applied for the GENERIC, UNVOICED and INACTIVE coder types using two subframes per frame. In certain specific situations, the bitrate required to encode the signal in the secondary channel may need to be reduced. For example, when the signal in the secondary channel is classified as

INACTIVE, the bitrate of the ACELP encoder might be reduced down to 1.5 kb/s. In that case, the INACTIVE encoder described in clause XX is used and modified in such a way that the signal in the secondary channel is converted to the frequency domain and only a half of the spectrum is encoded. The rest of the spectrum is not encoded, assuming that noise-filling mechanism will be applied at the decoder. Furthermore, in case the signal in the secondary channel is classified as UNVOICED a similar approach is taken to reduce the bitrate when needed. It is noteworthy to say that the LP filter quantization in the UNVOICED coder type takes more bits when compared to the INACTIVE coder type.

Finally, in case the signal in the secondary channel is classified as GENERIC, then GENERIC encoder as described in clause XX is used under the constraint that the frame to be encoded is split into two subframes. In this case the LP residual signal XX, the past adaptive excitation signal XX and the input signal in the secondary channel XX are down-sampled by a factor two. The LP filter coefficients are converted into a representation corresponding to the down-sampled domain instead of the 12.8 kHz sample rate in which they were initially calculated. After the excitation coding process described in clause XX bandwidth extension is performed in the frequency domain on the difference signal $f_d(k)$, defined in clause XX of Reference [EVS ref]. The bandwidth extension process replicates the spectral content from the lower spectral bands into the higher spectral bands. First, spectral energies of the difference signal $f_d(k)$ are calculated in the first nine spectral bands using the procedure described in clause 5.2.3.5.7 of Reference [EVS Ref]. Let's denote the spectral band energies as $G_{bd}(i)$, where $i = 0, \dots, 8$. The energies of the remaining bands are then filled as follows

$$G_{bd}(i) = G_{bd}(16 - i - 1), \qquad \text{for } i = 8, \dots, 15 \qquad (5.3\text{-}56)$$

The high-frequency content of the difference signal $f_d(k)$ is then populated using the low-frequency content with

$$f_d(k) = f_d(k - P_b), \qquad \text{for } k = 128, \dots, 255 \qquad (5.3\text{-}57)$$

where P_b is the pitch offset in frequency domain, calculated based on a multiple of the pitch information as described in clause 5.2.3.1.4.1 of Reference [EVS Ref]. The conversion of the pitch information into the pitch offset P_b is done as follows

$$P_b = \begin{cases} \frac{8 \cdot \left(\frac{F_s}{\bar{T}}\right)}{F_r}, & \bar{T} > 64 \\ \frac{4 \cdot \left(\frac{F_s}{\bar{T}}\right)}{F_r}, & \bar{T} \leq 64 \end{cases} \qquad (5.3\text{-}58)$$

where $\bar{T}$ represents an average of the decoded pitch information per subframe, and F_r is the frequency resolution.

In both, two-subframe and four-subframe excitation coders the indices of the quantized parameters in INACTIVE, UNVOICED or GENERIC coder types in the secondary channel are added to the bitstream of the secondary channel.

5.3.3.2.1.5 Bit allocation

The bit allocation mechanism uses the mixing ratio β calculated in Eq. (5.3-44) together with the decision whether or not to re-use the LP filter coefficients from the primary channel described in clause XX and the OL pitch coherence information S_{pc}, calculated in Eq. (5.3-55). Depending on the primary and the secondary channel encoding requirements the bit allocation mechanism calculates the bit budget for the primary and the secondary channel. For signals that are not classified as INACTIVE, only a fraction of the total available bit budget is allocated to the secondary channel. The bit budget of the secondary channel is then increased by an amount which is related to an energy normalization (rescaling) factor ε, calculated in Eq. (5.3-46), in the following way

$$B_x = B_M + (0.25 \cdot \varepsilon - 0.125) \cdot (B_t - 2 \cdot B_M) \qquad (5.3\text{-}59)$$

where B_x represents the bitrate allocated to the secondary channel, B_t represents the total bitrate available to the TD stereo coder, B_M represents the minimum bitrate allocated to the secondary channel which is usually around 20% of the bitrate available to the TD stereo coder. Hence, the bitrate allocated to the primary channel corresponds to the difference between the bitrate available to the TD stereo coder and the bitrates allocated to the secondary channel.

For INACTIVE content, the bitrate of the secondary channel is set to the minimum bitrate needed to encode the spectral shape of the secondary channel which is usually close to 2 kb/s.

In case the signal in the secondary channel is classified neither as INACTIVE nor as UNVOICED and using the bit budget allocated for the secondary channel, the TD stereo coder determines whether there is a sufficient amount of bits in the secondary channel for the encoding using four subframes scheme. If not, then the TD stereo coder will encode the signal in the secondary channel using two subframes scheme. To encode the signal with the four subframes scheme there must be at least 40 bits available in the secondary channel for the indices of algeraic codebooks. The number of bits available to the indices of algeraic codebooks are calculated by subtracting the bitrate of all other quantized parameters incl. e.g. the LP filter, pitch or gains from the bitrate allocated to the secondary channel.

5.3.3.2.2 DFT-based stereo

5.3.3.2.2.1 General

The DFT-based stereo is a joint Mid/Side (M/S) stereo coding exploiting some spatial cues, where the Mid-channel is coded by a primary mono core coder as described in 5.1.3. The Side-channel is predicted parametrically and optionally the residual of its estimation coded by a secondary core coder as described below.

The main encoding processing is depicted in Figure 5.3-6. The DFT stereo operates mainly in the frequency domain after a STFT analysis of the time-aligned or partly time-aligned stereo channels and producing for the Mid signal a downmix signal which is conveyed back to the time domain after STFT syntheses to the different core-coding schemes, that means either to ACELP and the associated TD-BWE, or to one of the two MDCT-based core-coders, namely TCX associated to IGF and HQ-Core. The Side-signal is usually solely modelled by the stereo parameters, but can be partly and discretely transmitted at higher bit-rates, 32kbps and above, through its prediction residual signal coding of the first 1kHz. The residual coding of the Side signal operates in a separated MDCT where a scalar quantization is followed by an entropy coding.

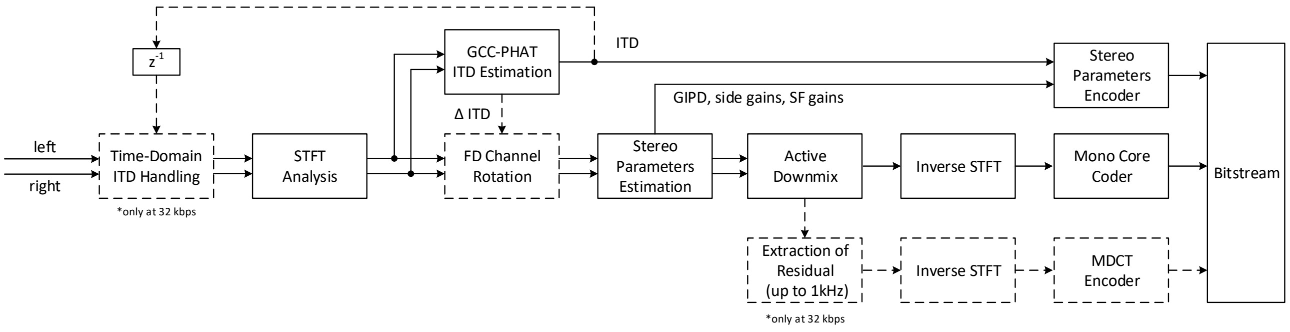


Figure 5.3-6: block diagram of DFT stereo encoder

The DFT stereo encoding scheme consists of the following steps: a time-domain Inter-channel Time Difference (ITD) – compensation, a STFT analysis, a GCC-PHAT ITD estimation, a stereo parameter estimation, and active downmixing, a stereo parameters encoder, an optional residual coding, and inverse STFTs providing the downmix channel at different sampling rates and in different frequency bands. The different steps are detailed in the following sections.

5.3.3.2.2.2 Time Domain ITD compensation

The time alignment of the two channels is achieved by compensating the ITD by time shifting one of the two channels at the input sampling rate and in time domain. The lagging channel is advanced, and for not engendering extra delay, the samples beyond the available samples are extrapolated.

Moreover, since the ITD is estimated in the frequency domain after the time-domain ITD compensation occurs, the ITD estimated at the previous frame is employed. The delta between the ITD used in TD and the current estimated ITD will be used in the frequency-domain ITD compensation.

5.3.3.2.2.3 STFT analysis

Most of stereo analysis and processing is performed in frequency domain after a Short-term Fourier Transform (STFT) analysis on the stereo channels of the input audio signal at the input sampling-rate. An analysis window is applied for allowing a good auditory scene analysis and stereo processing in the frequency domain. The windowed STFT (Short Time Fourier Transform) analysis formula can be expressed as:

$$S(c,m,\mathrm{k}) = \sum_{n=0}^{\mathrm{N}-1} s[c,n+m.M-zp]w_{ana}[n]e^{-j\frac{2\pi\mathrm{k}n}{N}}, \quad (5.3\text{-}60)$$

where $s[c,n]$ is the time-domain input signal for the channel index c, $w_{ana}[n]$ is the analysis window function, m is the index of the *M* stride size (stride parameter), zp is an offset corresponds to the zero padding of the analysis windows, k is the frequency index, N the DFT size corresponding to a total duration of 40ms at the signal sampling rate, and j is the imaginary unit.

For analysis the audio signal is divided into overlapping windows, wherein the stride size *M* corresponds to the frame duration of core-coder, i.e. 20 ms and the overlapping region to the encoder lookahead of the core-coders, i.e. 8.75 ms. For getting even finer frequency resolution and to counterbalance the circular shift when time aligning the channels in frequency domain, zero padding of 5.625 ms is added at both sides to the analysis windows for a total window length of 40ms as depicted in Figure 5.3-7.

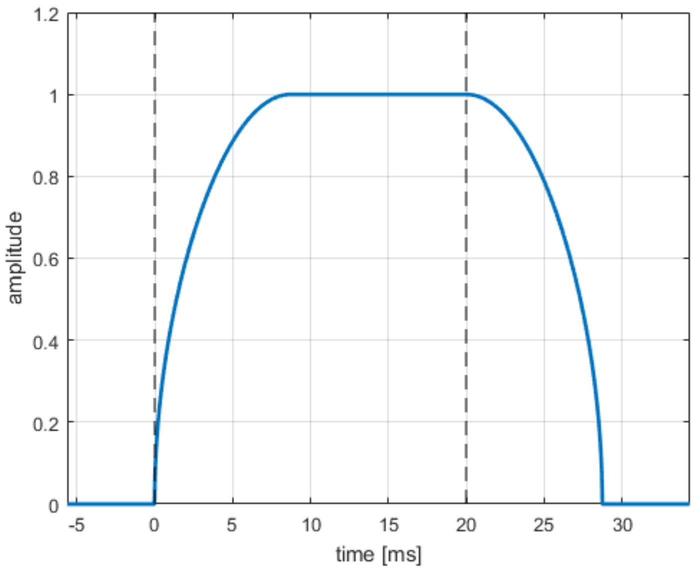


Figure 5.3-7: STFT analysis window with a stride of 20ms and an overlapping of 8.75ms.

The overlapping window part is 8.75ms and corresponds to the lookahead needed for the pre-processing and LP analysis used by the core-code. It also corresponds to overlapping region of the MDCT used in TCX and HQ-Core. The analysis window is defined as:

$$w_{ana}[n] = \begin{cases} \sqrt{sin(\frac{\pi.(n - zp+0.5)}{2.L})} \; for \; zp \le n < zp + L \\ 1, for \; zp \; + \; L \le n < M \\ \sqrt{\sin\left(\frac{\pi.(n- zp+L - M+0.5)}{2.L}\right)} for \; M \le n < M + L \end{cases} \qquad (5.3\text{-}61)$$

, and zero otherwise.

Hereinafter, or to simplify notation, the stride index *m* is replaced by the frame index *i*, the stride size being the same as the frame size, and the spectrum $S(c,m,\mathrm{k})$ is replaced by $L_i(k)$ for *c=0*, for the left channel, and $R_i(k)$ for *c=1*, the right channel.

5.3.3.2.2.4 GCC-PHAT ITD estimation

The inter-channel time difference (ITD) is commutated by estimating the Time Delay of Arrival (TDOA) using an adapted version Generalized Cross Correlation with Phase Transform (GCC-PHAT), where the inter-channel cross-correlation spectrum, the normalization as well as the maximum finding are adapted to the characteristic of the input signal. ITD estimation can summarized by the following equation:

$$ITD_i = argmax'(IDFT(\frac{Xcorr_i'}{|Xcorr_i'|^{\alpha}})) \qquad (5.3\text{-}62)$$

Where $Xcorr_i'$ is the smoothed version of the cross-correlation spectrum over time for the frame index *i*, derived from the cross-correlation spectrum computed in frequency domain, where α is a normalization factor depending on the noisiness characteristic of the signal, and $argmax'$ a peak picking algorithm, more evolved than a simple argmax operation. The spectrum of cross-correlation function is first derived between the left and right channels as:

$$Xcorr_i(k) = L_i(k)R_i^*(k) \qquad (5.3\text{-}63)$$

where L_i and R_i are the frequency spectra of the of the left and right channels respectively for the frame index *i*, derived from the STFT analysis described above, where *k* is the DFT frequency index, and where * is the complex conjugate operator.

The cross-correlation spectrum is then smoothed depending on the Spectral Flatness Measurement (SFM), a spectral characteristic of the input signal. The SFM is defined as the ratio between the geometric mean over the arithmetic mean of the power spectrum. The SFM is bounded between 0 and 1. In case of noise-like signals, the SFM will tend to be high (i.e. around 1) and the smoothing weak. On the other hand, in case of tone-like signal, SFM will tend to be low and the smoothing will become stronger. SFM is computed on the same DFT spectra as used for the DFT stereo processing and for the spectrum computation of the cross-correlation, and this for both the left and right channels. The SFM is given by:

$$SFM\_L = \frac{\exp\left(\sum_{k=1}^{N} \log(|L(k)|), 1/N\right)}{\left(\sum_{k=1}^{N} |L(k)|\right)/N} \qquad (5.3\text{-}64)$$

Where N is the number of DFT bins till a maximum of 8kHz of audio bandwidth. The SFM for the right channel SFM_R is computed the same way- The final *SFM* value used for the cross-correlation spectrum smoothing isas the maximum of the two SFM measures. The smoothed cross-correlation spectrum over time is then obtained with the following filtering:

$$Xcorr_i'(k) = Xcorr_i\ (k) * SFM + Xcorr_{i-1}'(k) * (1 - SFM) \qquad (5.3\text{-}65)$$

The smoothed cross-correlation is then normalized by its amplitude or one of its norm before being transformed back to time domain. The normalization corresponds to the Phase –transform of the cross-correlation, and is known to show better performance than the normal cross-correlation in low noise and relatively high reverberation environments. However, to make it even more robust the norm of the numerator is adapted depending of the nature of the signal, leading to the normalized cross-correlation function τ defined in time as:

$$\tau = IDFT\left(\frac{Xcorr_i'}{|Xcorr_i'|^{\alpha}}\right) \qquad (5.3\text{-}66)$$

where $\alpha = 1$ for clean speech and $\alpha = 0.8$ when noisy speech is detected in the pre-processing stage as described in 5.2.2.1.

The so-obtained time domain function is first filtered for achieving a more robust peak peaking. The index corresponding to the maximum amplitude corresponds to an estimate of the time difference between the Left and Right Channel (ITD). If the amplitude of the maximum is lower than a given threshold, then the estimated of ITD is not considered as reliable and is set to zero.

5.3.3.2.2.5 FD circular time-shift

The ITD compensation is achieved in both time and frequency domain. The application of time shift in FD is limited since it can only be achieved only up to a certain maximal lag without introducing strong window mismatch between channels. Nonetheless, the zero padding of the analysis window prevents any severe artefacts which could come from the circular shifting, since the maximum ITD of 5ms is less than the both-sided zero-padding guards of 5.625ms. Therefore, the channel time alignment is split in a way the Frequency Domain part is lower possible and correspond to a delta ITD. The Delta ITD is the difference between the ITD estimated at the previous frame and applied in TD at the present frame, and the estimated ITD at the current frame. The channels are time shifted in a way that the lagging channel is advanced in time, i.e. if the ITD is negative, the lagging left channel is advanced.

$$\begin{cases} L(k) = L(k)e^{-j2\pi k\frac{\Delta ITD}{NFFT}}\ if\ ITD < 0 \\ R(k) = R(k)e^{+j2\pi k\frac{\Delta ITD}{NFFT}}\ if\ ITD > 0 \end{cases} \qquad (5.3\text{-}67)$$

To prevent that an eventual too strong window mismatch between the channels occur after a circular time shifting in FD, the attenuation factor is computed if the ΔITD is above 2.5ms. This attenuation factor will be served to attenuate the residual signal if coded, since less relevant in case of strong window mismatch.

5.3.3.2.2.6 Stereo parameters Estimation

Encoding of stereo parameters and computation of the downmix signal is done in frequency domain on the time-frequency vectors L_i and R_i of the left and right channel. The DFT bins are then grouped into subbands $(L_t, k)_k \in I_b$ resp. $(R_t, k)_k \in I_b$, where I_b denotes the set of subbands indices.

Different subband partitioning schemes are used, depending on the bitrate, but also on the processing. The followed roughly a scale version of the equivalent rectangular bandwidth (ERB). For stereo parameter estimation, the subband partitioning B is used, which dependent on the bit-rate. For the downmix parameter estimation, the partitioning B2 is used, which is a superset of B with finer frequency resolution, further decomposing the subband partitioning B. Finally, the downmix equalization is performed on the highest subband resolution, which combines 2 consecutive frequency bins within a subband partitioning band B. If the number of frequency bins within a subband B is not a multiple of 2, then the last subband b3 consists of 3 frequency bins and is called triple.

Table 5.3-5: Subband portioning for stereo and downmix parameters.

Partitioning name	Used for	Subband upper limits in STFT frequency bin index
ERB 8	<=16.4kbps, stereo parameters (partitioning B)	1, 5, 18, 41, 84, 214, 470, 601
ERB 4	Downmix parameters (partitioning B2) and stereo parameters (partitioning B) for >16.4kbps	1, 3, 5, 10, 18, 26, 41, 56, 84, 132, 214, 342, 470, 601

5.3.3.2.2.7 IPD Calculation and stabilization

For the downmix, a single global inter-channel-phase-difference (IPD) *gIPD* is calculated over the first 8 subbands of the ERB 8 partitioning (up to DFT bin 84) as

$$gIPD = \arg\left(\sum_{k\in I_{1,..,8}} L_{t,k}\, R^*_{t,k}\right) \tag{5.3-68}$$

where $R^*_{t,k}$ denotes the complex conjugate of $R_{t,k}$.

To provide a more stable *gIPD* estimate, a stability mechanism is employed which is described in detail below and shown more comprehensively as a flow diagram in Figure 5.3-8.

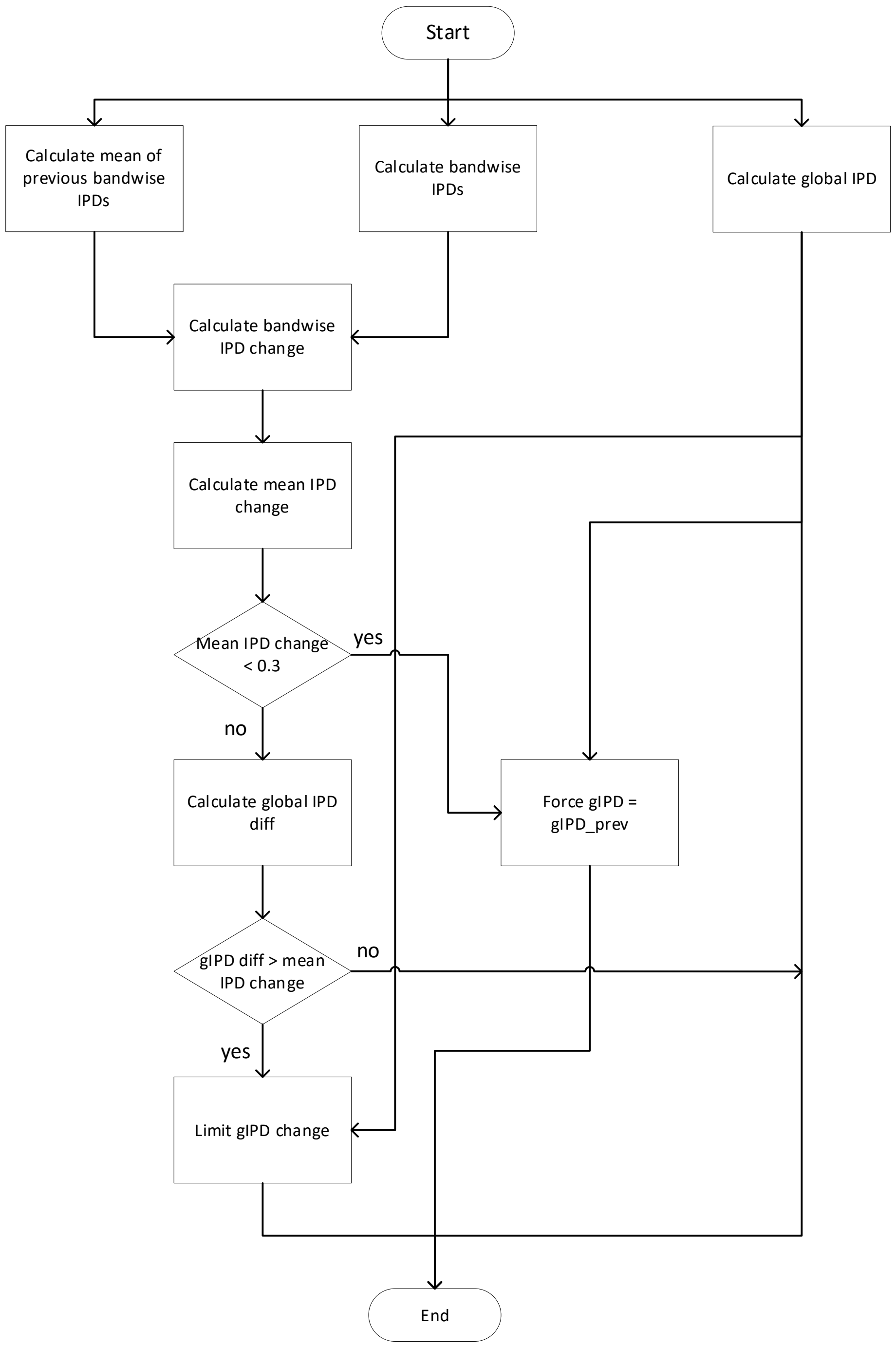

Start
Calculate mean of previous bandwise IPDs
Calculate bandwise IPDs
Calculate global IPD
Calculate bandwise IPD change
Calculate mean IPD change
Mean IPD change < 0.3
yes
no
Calculate global IPD diff
Force gIPD = gIPD_prev
gIPD diff > mean IPD change
no
yes
Limit gIPD change
End

Figure 5.3-8: Flow diagram of global IPD stabilization

The stabilization first requires the calculation of additional bandwise phase differences $IPD_{t,b}$ as

$$IPD_{t,b} = \arg(\sum_{k \in l_b} L_{t,k} R^*_{t,k}) \quad (5.3\text{-}69)$$

for each of the 8 subbands over which the global IPD is calculated. Note that also for bitrates <= 16.4 kbps the ERB 8 bands are used for IPD calculation.

Additionally, in each subband bandwise mean IPDs over the 5 previous frames are calculated. Since distances between phases are ambiguous (2 possible directions on a circle) a meaningful bandwise mean IPD cannot always be calculated by standard averaging (only if all phases are within the same semi-circle). Instead, the bandwise mean IPD of a subband, denoted as $IPD_{mean,b}$, may be initialized with 0 and then updated iteratively with

$$IPD_{mean,b} = \frac{i}{i+1} IPD_{mean,b} + \frac{1}{i+1} IPD_{prev,b}[i] \quad (5.3\text{-}70)$$

where $i = 0, \ldots, 4$ is the index over the previous IPD values of the band. After each iteration, the distance of the current result to the next value in the IPD_{b_prev} buffer is calculated:

$$IPD_{diff} = \left|IPD_{mean,b} - IPD_{prev,b}[i+1]\right| \tag{5.3-71}$$

If IPD_{diff} is greater than π, i.e. more than a half-circle rotation in the given direction, $IPD_{mean,b}$ needs to be temporarily shifted outside of the $[-\pi, \pi]$ range by adding or subtracting 2π depending on which side of the circle it lies on:

$$IPD_{mean,b} = IPD_{mean,b} + 2\pi,\ if\ IPD_{mean,b} < 0 \tag{5.3-72}$$

or

$$IPD_{mean,b} = IPD_{mean,b} - 2\pi,\ if\ IPD_{mean,b} > 0 \tag{5.3-73}$$

Then the mean will be updated using this shifted version which now has a distance of less than π to the next value in $IPD_{prev,b}$. If after the update $IPD_{mean,b}$ is still outside $[-\pi, \pi]$ the shift is reversed before the next iteration.

Now the bandwise IPD change, denoted as *IPD_{change,b}*, between the current bandwise $IPD_{t,b}$ and bandwise mean $IPD_{mean,b}$ is computed for each subband with

$$IPD_{change,b} = \left|IPD_{t,b} - IPD_{mean,b}\right| \tag{5.3-74}$$

with

$$IPD_{change,b} = 2\pi - IPD_{change,b},\ if\ IPD_{change,b} > \pi \tag{5.3-75}$$

From the individual bandwise IPD changes in each subband a mean bandwise change, denoted as IPD_{change}, over all subbands is computed:

$$IPD_{change} = \frac{\sum_{b=1}^{nBands} IPD_{change,b}}{nBands} \tag{5.3-76}$$

The mean bandwise IPD change is taken as an overall indication of the stability of the bandwise IPD in the current frame and is now used to force a similar level of stability on the global IPD estimate, i.e. to calculate a stabilized IPD estimate using the current global IPD estimate, the transmitted stabilized estimate of the previous frame and the mean bandwise IPD change. For small values of the mean bandwise IPD change (smaller than 0.3) the current global IPD is overwritten with the stabilized IPD estimate of the previous frame:

$$gIPD = gIPD_{prev},\quad if\ IPD_{change} < 0.3 \tag{5.3-77}$$

For larger values, a modulus of a difference between the transmitted IPD of the previous frame and the global IPD of the current frame, denoted as $gIPD_{diff}$, is computed:

$$gIPD_{diff} = \left|gIPD - gIPD_{prev}\right| \tag{5.3-78}$$

with

$$gIPD_{diff} = 2\pi - gIPD_{diff},\ if\ gIPD_{diff} > \pi \tag{5.3-79}$$

If

$$gIPD_{diff} > IPD_{change} \tag{5.3-80}$$

which means that the modulus of the difference between the transmitted IPD of the last previous frame and the global IPD estimate of the current frame is larger than the mean bandwise IPD change, the maximum allowed difference to the previously transmitted IPD is limited to the mean bandwise IPD change, so that the global IPD is calculated as:

$$gIPD = gIPD_{prev} + IPD_{change},\quad if\ gIPD > gIPD_{prev} \tag{5.3-81}$$

or

$$gIPD = gIPD_{prev} - IPD_{change},\quad if\ gIPD < gIPD_{prev} \tag{5.3-82}$$

If, however,

$$gIPD_{diff} \leq IPD_{change}, \tag{5.3-83}$$

which means that the modulus of the difference between the previously transmitted IPD and the global IPD estimate of the current frame is equal to or smaller than the mean bandwise IPD change, the original global IPD estimate *gIPD* is used for the current frame.

5.3.3.2.2.8 Calculation of the side and residual prediction gains

In case of the residual prediction gain is transmitted or if the residual is directly coded in the frequency band b currently considered, the side gain stereo parameter is computed as the optimal gain for predicting the side signal $S_i[k] = L_{i[k]} - R_i[k]$ by the mid signal $M[k] = L[k] + R[k]$, such that the energy of the remainder

$$p[k] = S[k] - g[b]M[k] \tag{5.3-84}$$

is minimal. The optimal prediction gain can be calculated from the energies in the subbands

$$E_L[\text{b}] = \sum_{k\in I_b} |\text{L[b]}\|^2 \quad \text{and} \quad E_R[\text{b}] = \sum_{k\in I_b} |\text{R[b]}|^2 \tag{5.3-85}$$

and the absolute value of the inner product of L and R

$$X_{L/R}[b] = \left| \sum_{k\in I_b} L[k]R^*[k] \right| \tag{5.3-86}$$

as

$$g[b] = \frac{E_L[\text{b}] - E_R[b]}{E_L[b] + E_R[b] + 2X_{L/R}[b]} \tag{5.3-87}$$

From this it follows that $g[b]$ lies in [-1,1].

In bands, where the residual gain is transmitted, the side gain is calculated similarly as previously defined. The residual gain is then computed form the side gain, the channel energies and the inner product of the channels as:

$$r[b] = \sqrt{\frac{(1-g[b])E_L[b] + (1+g[b])E_R[b] - 2X_{L/R}[b]}{reg[b] + E_L[b] + E_R[b] + 2X_{L/R}[b]}} \tag{5.3-88}$$

Where $reg[b]$ is regularization factor obtained by adding a coherent low energy contribution to avoid singularity for very low energy signals:

$$reg[b] = (n(I_b) * NFFT_{32})^2 \tag{5.3-89}$$

Where the $n(I_b)$ gives the cardinality of I_b, the set of frequency indexes laying in the frequency band b, and $NFFT_{32}$ is the size of the analysis DFT at 32kHz.

It implies that

$$0 \leq r[b] \leq \sqrt{1 - g[b]^2}. \tag{5.3-90}$$

In particular, this shows that $r[b] \in [0,1]$. This way, the stereo parameters can be calculated independently from the downmix by calculating the corresponding energies and the inner product.

In the case of non-zero ΔITD and time alignment in the frequency domain, there is a mismatch in the DFT analysis windows between the two channels, which biases the calculation of the residual prediction gain. An offset is therefore calculated, involving a normalized autocorrelation of the DFT analysis windows, modeling the mismatch of the two windows after a circular shift of ΔITD samples. The normalized autocorrelation function is precalculated and stored in the table given in Table 2, for a resolution of 8 samples at 32 kHz. The delta ITD is then first converted to 32kHz, and the corresponding offset is found by linear interpolation of the two Wn_table entries.

Based on this normalized autocorrelation function $\widehat{W}_X(n)$, the offset parameter $\hat{r}_t$ is calculated as:

$$\hat{r} = \frac{2\,c}{c+1} \sqrt{2 \frac{1 - \widehat{W}_X(\Delta ITD)}{1 + c^2 + 2\,c\,\widehat{W}_X(\Delta ITD)}}, \tag{5.3-91}$$

where

$$c = \frac{E_L[b]}{E_R[b]}. \tag{5.3-92}$$

The residual gains r[b] is then corrected as given in the equation:

$$r[b] \leftarrow \max\{0, r[b] - \hat{r} \ \}. \tag{5.3-93}$$

Table 5.3-6: Normalized cross-correlation function of the analysis window at a resolution of 8 samples at 32kHz

$\widehat{\boldsymbol{W}}_{\boldsymbol{X}}(\Delta \boldsymbol{ITD})$							
1.0000000f,	0.9992902f,	0.9975037f,	0.9948399f,	0.9914063f,	0.9872797f,	0.9825207f,	0.9771799f,
0.9713015f,	0.9649245f,	0.9580846f,	0.9508143f,	0.9431443f,	0.9351030f,	0.9267174f,	0.9180131f,
0.9090145f,	0.8997447f,	0.8902261f,	0.8804801f,	0.8705271f,	0.8603868f,	0.8500779f,	0.8396186f,
0.8290262f,	0.8183170f,	0.8075067f,	0.7966103f,	0.7856416f,	0.7746140f,	0.7635394f,	0.7524292f,
0.7412935f,	0.7301411f,	0.7189796f,	0.7078147f,	0.6966495f,	0.6854842f,	0.6743189f,	0.6631536f,
0.6519884f,	0.6408231f,	0.6296578f,	0.6184926f,	0.6073273f,	0.5961620f,	0.5849968f,	0.5738315f,
0.5626662f,	0.5515010f						

Finally, in case of active speech, which is detected if VAD flag is set and if speech is detected in the pre-processing step, the residual prediction gain is further limited for mitigating potential synthetic artefacts in the stereo upmix and ambience generation. First if ΔITD changed from the previous frame the residual prediction gain is replaced by the minimum between the present and previous gain:

$$r_i[b] \leftarrow \min\{r_{i-1}[b], r_i[b]\} \tag{5.3-94}$$

Then it is compared to an IIR filtered and smoothed gain threshold, used to clip the actual residual gain value to be used:

$$r'_i[b] = 0.9r'_{i-1}[b] + 0.1r_i[b] \tag{5.3-95}$$

$$r_i[b] \leftarrow \min\{1.1r'_i[b], r_i[b]\} \tag{5.3-96}$$

In case neither residual gain nor residual coding are used in the frequency band, the side gain is computed based the Inter-channel Level difference, which is more psycho-acoustically motivated and quasi optimal assuming that the channels are time and phase aligned. The side gain is then given by:

$$g[b] = \frac{c-1}{c+1}, \tag{5.3-97}$$

where

$$c = \frac{E_L[b]}{E_R[b]}. \tag{5.3-98}$$

Stereo parameter coding

The global IPD is quantized on 4 bits in active frames and on 2 bits on SID frames, by the following uniform quantization of the angle:

$$gIPD_{index} = \left\lfloor \frac{gIPD+\pi}{deltaIPD}' + 0.5 \right\rfloor \tag{5.3-99}$$

Where $deltaIPD = 2\pi / gIPD_{index\_max}$, where $gIPD_{index\_max} = 2^{gIPD_{bits}}$.

The index is coded in modulo 2π, and an index of 16 is then converted to index 0:

$$gIPD_{index} = 0 \text{ if } gIPD_{index} = gIPD_{index\_max} \tag{5.3-100}$$

The dequantized global IPD value is then used in the active downmix and computes as follows:

$$\widehat{gIPD} = gIPD_{index}.deltaIPD - \pi \tag{5.3-101}$$

The global IPD index $gIPD_{index}$ is written in raw coding in the bitstream.

The quantization of side and residual gains is done jointly, since there is a strong dependence of the residual gain on the side gain, since the latter determines the range of the first. Quantizing the side gain g and the residual gain r independently by choosing quantization points in [−1, 1] and [0, 1] is therefore inefficient, since the number of possible quantization points for r would decrease as g tends towards ±1. The joint quantization is done by first computing the inherent absolute ILD, using the following equality:

$$|ILD| = 10\log_{10}(\frac{(1+|g|)^2+r\prime^2}{(1-|g|)^2+r\prime^2}), \qquad (5.3\text{-}102)$$

Where is the bounded residual prediction gain r', is bounded between 0 and the theoretical maximum depending on the side gain:

$$r' = \min(r, \sqrt{1-g^2}) \qquad (5.3\text{-}103)$$

The resulting absolute ILD $|ILD|$ is bounded between 0 and 50dB, before being quantized on the 16 values defined as follows:

±{0,2,4,6,8,10,13,16,19,22,25,30,35,40,45,50},

Once the quantized level ILD_{level} of the absolute ILD is found, a 2-dimensional 8-points quantization is selected for quantization the absolute value g and the bounded value of r'. This gives rise to an overall codebook with 256 entries, which is organized as a 16 × 8 tables of quantization points holding the values corresponding to non-negative values of g and r' and a sign bit for the sign of g. This gives rise to a 8 bit integer representation of the quantization points (g, r') where the first bit specifies the sign of g, the next four bits holding the row index in the 16 × 8 table and the last three bits holding the column index. Quantization of (|g|, r'') is done by an exhaustive codebook search by minimizing the mean squared error:

$$gains_{index} = \underset{i}{\text{argmax}}\ ((|g| - gains_{cdbk}[8*ILD_{level}+i][0])^2 + (r' - gains_{cdbk}[8*ILD_{level}+i][1])^2) \quad (5.3\text{-}104)$$

Table 5.3-7: Codebook for the joint quantization of the absolute side gain and the residual prediction gain function of the absolute ILD level

$\vert ILD\vert$=0	**$\vert ILD\vert$=2**	**$\vert ILD\vert$=4**	**$\vert ILD\vert$=6**	**$\vert ILD\vert$=0**
0.000000,0.000000	0.114623,0.000000	0.226274,0.000000	0.332279,0.000000	0.430506,0.000000
0.000000,0.116982	0.116171,0.115424	0.229210,0.110915	0.336318,0.103909	0.435293,0.095060
0.000000,0.226991	0.120448,0.22385	0.237306,0.214802	0.347423,0.200786	0.448405,0.183172
0.000000,0.340693	0.127733,0.335704	0.251046,0.321340	0.366155,0.299242	0.470357,0.271705
0.000000,0.464549,	0.138966,0.45714	0.272098,0.435947	0.394585,0.403641	0.503282,0.363897
0.000000,0.605079	0.155840,0.59427	0.303429,0.563535	0.436296,0.517359	0.550751,0.461614
0.000000,0.776250	0.182248,0.760034	0.351766,0.714464	0.499282,0.647470	0.620606,0.568856
0.000000,1.000000	0.226274,0.974064	0.430506,0.902588	0.598480,0.801138	0.726386,0.687287
$\vert ILD\vert$=10	$\vert ILD\vert$=13	$\vert ILD\vert$=16	$\vert ILD\vert$=19	$\vert ILD\vert$=22
0.519494,0.000000	0.634158,0.000000	0.726386,0.000000	0.798235,0.000000	0.852825,0.000000
0.524665,0.085097	0.639318,0.069554	0.731048,0.054862	0.802164,0.042077	0.855976,0.031585
0.538769,0.163459	0.653296,0.132950	0.743597,0.104384	0.812683,0.079739	0.864374,0.059662
0.562182,0.241193	0.676201,0.194597	0.763914,0.151643	0.829536,0.115098	0.877717,0.085671
0.596843,0.320512	0.709442,0.255549	0.792863,0.197003	0.853180,0.148173	0.896203,0.109492
0.645875,0.402058	0.755149,0.315316	0.831670,0.239542	0.884212,0.178009	0.920064,0.130302
0.716076,0.487487	0.818046,0.373555	0.883261,0.278217	0.924333,0.203506	0.950256,0.147176
0.818182,0.574960	0.904547,0.426375	0.950993,0.309212	0.975135,0.221614	0.987460,0.157870
$\vert ILD\vert$=25	$\vert ILD\vert$=30	$\vert ILD\vert$=35	$\vert ILD\vert$=40	$\vert ILD\vert$=45
0.893520,0.000000	0.938693,0.000000	0.965056,0.000000	0.980198,0.000000	0.988816,0.000000
0.895958,0.023331	0.940202,0.013738	0.965951,0.007932	0.980717,0.004528	0.989113,0.002568
0.902435,0.043958	0.944192,0.025807	0.968314,0.014873	0.982085,0.008481	0.989895,0.004807
0.912657,0.062866	0.950441,0.036736	0.971997,0.021112	0.984212,0.012019	0.991109,0.006806
0.926684,0.079898	0.958923,0.046392	0.976963,0.026561	0.987068,0.015088	0.992735,0.008532
0.944559,0.094403	0.969577,0.054375	0.983149,0.030984	0.990608,0.017552	0.994746,0.009911
0.966814,0.105673	0.982605,0.060264	0.990633,0.034143	0.994866,0.019278	0.997156,0.010865
0.993695,0.112114	0.998002,0.063182	0.999368,0.035554	0.999800,0.019998	0.999937,0.011246
$\vert ILD\vert$=50				
0.993695,0.000000				
0.993864,0.001451				
0.994308,0.002715				

0.994996,0.003842 0.995917,0.004813 0.997054,0.005586 0.998414,0.006117 0.999980,0.006324				

The sign of side gain and ILD_{level} are combined in a 5-bit wise parameter. The parameter can be coded either in raw coding, or using an adaptive Golomb-rice coding up to order 3, and an inter-frame delta coding. The optimal coding is selected are signalling in the bitstream.

The 3-bit wise column index, is coded separately. The parameter can also be coded using different coding strategies: in raw coding, an adaptive Golomb-Rice coding up to order 2, or an inter-frame delta coding. Best and selected coding is signaled in the bitstream.

Since the side gain and residual prediction gain generally varies slowly between frames, the inter-frame delta coding mode usually yields the lowest bit rate. However, this predictive scheme suffers from error propagation in case of packet loss. For this reason, the number of consecutive predictive encoding modes is limited to $N_{PREDMAX}+1=11$. The predictive encoding mode is further disabled in case of speech signals and the decision is stored in a flag $M_{try-diff}$ according to

$$M_{try-diff}=\begin{cases}1, sp\_aud\_decision0=1 \wedge N_{gpred}<N_{PREDMAX} \wedge R_{core}^{[-1]}>2400 \\ 0, otherwise\end{cases} \tag{5.3-105}$$

Where $sp\_aud\_decision0$ is the primary speech/music flag as defined in 5.2.2.1.9, N_{gpred} is a counter of consecutive frames of predictive side gain encoding mode and $R_{core}^{[-1]}$ is the core bit rate for the previous frame where $R_{core}^{[-1]} \leq 2400$ indicates the last frame was either a NO_DATA frame or a SID frame. In case $M_{try-diff}=0$ the counter N_{gpred} is reset to zero and the absolute coding mode is selected and set to be used. The required number of bits for absolute encoding using the variable rate AGR B_{AGR} is calculated. The fallback solution is to encode each index using 5 bits, which yields $B_{ABS}=5N_{bands}$ bits. If the predictive coding mode is allowed, $M_{try-diff}=1$, the number of required bits for the predictive coding scheme B_{PRED} is obtained. A bit rate difference is then obtained according to

$$B_{diff}=\min(B_{ABS},B_{AGR})-B_{PRED} \tag{5.3-106}$$

The bit rate difference B_{diff} represents the additional number of bits to encode the side gain using an absolute coding scheme like the AGR or the fallback binary index encoding and is typically larger than zero. The bit rate difference is low-pass filtered across frames according to

$$B_{diff,LP}=0.06B_{diff}+0.94B_{diff,LP}^{[-1]} \tag{5.3-107}$$

Then, the predictive mode is selected if the following condition is met

$$B_{diff}>0.8B_{diff,LP}\,N_{gpred}/(N_{PREDMAX}+1) \tag{5.3-108}$$

where N_{gpred} is the number of consecutive frames using predictive coding mode for the side gain. Further, the predictive mode counter is incremented $N_{gpred}:=N_{gpred}+1$. If the condition in xxx is not met, the best performing absolute coding mode is selected and set to be used, meaning the one yielding the lowest number of bits $\min(B_{ABS},B_{AGR})$ out of the AGR and the fallback binary index coding. The predictive mode counter N_{gpred} is also reset to zero.

5.3.3.2.2.9 Active Downmix

The global IPD if used is first compensated by rotating the Left channel and leave untouched the right channel. It is achieved as follows:

$$\begin{cases}L'_i[\mathrm{k}]=L_i[k].e^{-j2\pi.\widehat{gIPD}} \\ R'_i[\mathrm{k}]=R_i[k]\end{cases} \tag{5.3-109}$$

The so-obtained globally phase-aligned channels are further processed, and depending on the frequency band and if the residual coding is used in this band. For frequency bands where the residual coding will be applied a passive downmix and a prediction is used to compute the Mid and Side channels respectively. A sum difference transformation is first performed on the time and phase aligned spectra of the two channels in a way that the energy is conserved in the Mid signal.

$$\begin{cases} M_i[k] = 0.5 * (L'_i[k] + R'_i[k]) \\ S_i[k] = 0.5 * (L'_i[k] - R'_i[k]) \end{cases} \tag{5.3-110}$$

The Side channel is then predicted from the Mid channel and the transmitted stereo parameter side gain:

$$Res_i[\mathrm{k}] = S_i[k] - g[b].M_i[k] \tag{5.3-111}$$

The residual signal is further attenuated in case a delta ITD compensation was performed in the frequency domain. This is done to counter the window mismatch engendered by the time shift.

$$Res'_i[\mathrm{k}] = Res_i[\mathrm{k}] * \min(1.0, (\max(0.2(2.6 - 0.02 * |\Delta ITD_{32}|)) \tag{5.3-112}$$

where ΔITD_{32} is the delta ITD applied in the current frame reported at 32kHz.

For frequency bands, where no residual coding is performed, an active downmixing is performed by a controlled energy-equalization of the sum of the left and right channels mixed with a complementary signal, which is selected as being the left channel. The complementary signal, i.e. left channel, is there to avoid singularities during the sum, when the left and right channels are almost coherent and out-of-phase. The energy-equalization of the sum is controlled for avoiding problems at the singularity point but also to minimize significantly signal impairments due to large fluctuations of the gain. The complementary signal is there to compensate the remaining energy loss or at least a part of it. The general form of the downmix for a given frequency band index *k* is be expressed as:

$$M[k] = W_1[b3](L[k] + R[k]) + W_2[b3]L[k] \tag{5.3-113}$$

where *b3* is the index of the frequency band of the partition used to compute the mixing factors and to which frequency index *k* belongs.

The downmixing generates the sum channel L+R as it is done in conventional passive and active downmixing approaches, where the gain $W_1[b3]$ aims at equalizing the energy of the sum of the channels for matching the average energy of the input channels or a regularization of it. However, unlike conventional active downmixing approaches, $W_1[b3]$ is limited to avoid instability problems and to avoid that the energy relations are restored based on an impaired sum of channels. The mixing with a complementary signal is there to avoid that the energy vanishes when $L[k]$ and $R[k]$ are out-of-phase. $W_2[b3]$ compensates the energy-equalization due to the limitation introduced in $W_1[b3]$. The downmixing can then be rewritten as:

$$M[b] = W_L[b3]L[k] + W_R[b3]R[k] \tag{5.3-114}$$

where

$$W_R[b3] = W_1[b3] = \frac{\sqrt{(\sum_{k\in Ib3}|L[k]|^2 + \sum_{k\in Ib3}|R[k]|^2)(1 + \alpha[b2])}}{2.(\sum_{k\in Ib3}|L[k]| + \sum_{k\in Ib3}|R[k]|)} \tag{5.3-115}$$

and

$$W_L[b3] = W_R[b3] + \left(1 - \frac{\sum_{k\in Ib3}|L[k] + R[k]|}{\sum_{k\in Ib3}|L[k]| + \sum_{k\in Ib3}|R[k]|}\right) \tag{5.3-116}$$

where $\alpha[b2]$ is a band-wise parameter running on the frequency band partition *b2*, and used to stabilize and regularize the target energy of the mixed signal. It is computed as:

$$\alpha[b2] = \frac{\sqrt{\sum_{i=0}^{-2}\sum_{k\in Ib2} L_i[k]R_i^*[k]}}{\sum_{i=0}^{-2}\sum_{k\in Ib2}|L_i[k]|^2 + \sum_{i=0}^{-2}\sum_{k\in Ib2}|R_i[k]|^2} \tag{5.3-117}$$

by average energies and inner product norm over last three frames.

5.3.3.2.2.10 STFT synthesis

From the downmixed spectrum M, a time domain signal is synthesized by an inverse DFT:

$$m_i[n] = \sum_{k=0}^{N-1} M_i[k] \cdot e^{\frac{2\pi jkn}{N}}, \text{ for } 0 \le n < N \tag{5.3-118}$$

Finally, an overlap-add operation allows reconstructing a frame of M samples:

$$m'[i \cdot \mathrm{M} + n - zp] = \begin{cases} m_{i-1}[M+n] \cdot w_{syn}[M+n] + m_i[n] \cdot w_{syn}[n], \text{for } zp \le n < zp + L \\ m_i[n] \qquad , \text{for } zp + L \le n < zp + M \end{cases} \tag{5.3-119}$$

The synthesis window is not the same as analysis window, but are together complementary in energy. The synthesis window depicted in Figure 5.3-9.

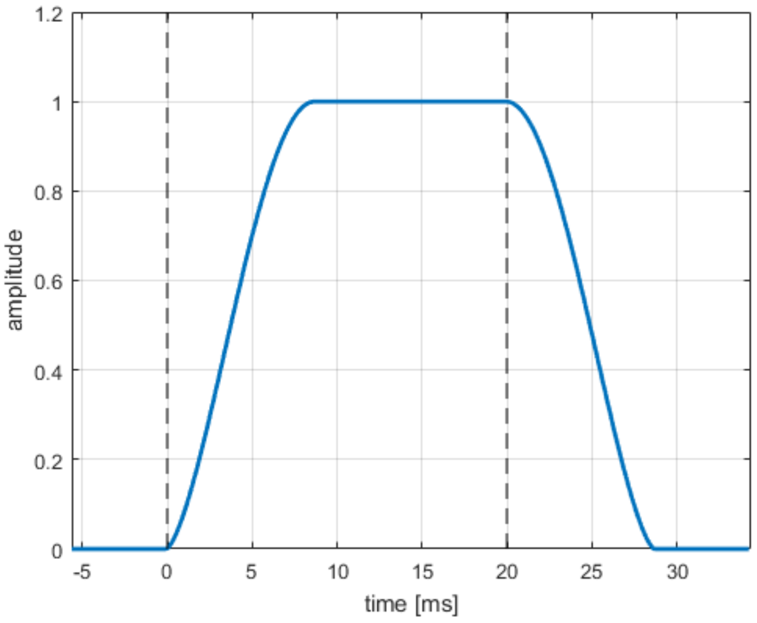


Figure 5.3-9: STFT synthesis window with a stride of 20ms and an overlapping of 8.75ms.

The overlapping window part is 8.75ms and corresponds to the lookahead needed for the pre-processing and LP analysis used by the core-code. It also corresponds to overlapping region of the MDCT used in TCX and HQ-Core. The synthesis window is defined as:

$$w_{syn}[n] = \begin{cases} \left(sin(\frac{\pi.(n - zp+0.5)}{2.L})\right)^{1.5} for\ zp \le n < zp + L \\ 1, for\ zp\ +\ L \le n < M \\ \left(\sin\left(\frac{\pi.(n - zp+L\ -\ M+0.5)}{2.L}\right)\right)^{1.5} for\ M \le n < M + L \end{cases} \tag{5.3-120}$$

and zero otherwise.

For reducing the delay engendered by overlapping region with the next frame, corresponding to the lookahead used in the pre-processing steps and exploited by the core-coders, is already estimated by an un-windowing of the solely current synthesized frame, without relying on the following frame. For this the inverse of the analysis is applied to the right part of the synthesis window, as follows:

$$m'[i \cdot \mathrm{M} + n - zp] = \frac{m_i[n] - d}{w_{ana}[n]} + d, n \in zp + [M, M + L[\tag{5.3-121}$$

where d is an approximated DC offset for avoiding instabilities and large values engendered by inverting the analysis window, especially at the right overlapping region bound, corresponding also to the lookahead bound. This un-windowing adaptation dependent on the characteristic of the processed synthesized signal m_i, which is used to approximate the DC offset by taking one of his values within the right zero padding range of the analysis window after processing and inverse DFT. The approximated DC offset is the expressed as:

$$d = m_i[zp + L + M] \tag{5.3-122}$$

The size of the inverse DFT and therefore the output sampling rate of the STFT synthesis is depending on the modules requiring the downmixed signal for the subsequent processing and the core-coding. By truncating or zero-padding the STFT spectrum, and using the appropriate normalization, a resampling is achieved allowing conveying the same downmixed signal at different modules and at different sampling-rates. The output sampling-rates can be different from the input sampling-rate of the input stereo signal used in the STFT analysis. The supported sampling-rates are 12.8 and 16 kHz when conveying and resampling the downmix signal to the core-coder ACELP. For the MDCT core-coders, the output sampling-rate can the 16,32 and 48kHz, while being at 16kHz for high-band downmixed signal used in the Time-Domain BWE of ACELP.

5.3.3.2.2.11 Residual coding

The residual coding is achieved after synthesis the signal Res'_i[k] back in time-domain at a sampling-rate of 8 kHz through the inverse DFT. No overlap-adding is realized since the windowed synthesized signal is directly transformed by a forward MDCT after applying the same windows as the analysis STFT window to achieve an analysis MDCT window equivalent to a sine window.

The residual coding operates in the MDCT domain at a target quantization SNR, specified together with the maximum number of bits that are allowable for each frame. If the target SNR requires a larger number of bits than the maximum specified, the SNR is gradually decreased so that the actual number of bits will satisfy the bit constraint.

The target SNR is derived from the psychoacoustic consideration that quantization errors are more perceptible if the restored stereo channels are out-of-phase. Therefore, the target SNR is made dependent on an out-of-phase estimator. Considering the stereo upmix, the left are right channels are simplistically generated by a mid signal, a side gain as well a residual signal,

$$\begin{cases} L[k] = M[k] + g[b]M[k] + R[k] \\ R[k] = M[k] - g[b]M[k] - R[k] \end{cases} \tag{5.3-123}$$

Since $|g[b]| < 1$, one can consider that the components $M[k] - g[b]M[k]$ and $M[k] + g[b]M[k]$ are always in-phase, while $+R[k]$ and $-R[k]$ are obviously out-of-phase. The out-of-phase ratio retained is the maximum out-of-phase ratio among the two channels:

$$oop_{ratio}[b] = \max\left(\frac{E_{R[b]}}{(1-g[b])^2 E_{M[b]} + E_{R[b]}}, \frac{E_{R[b]}}{(1+g[b])^2 E_{M[b]} + E_{R[b]}}\right) = \frac{E_{R[b]}}{(1-|g[b]|)^2 E_{M[b]} + E_{R[b]}} \tag{5.3-124}$$

given $|g[b]| < 1$. The in-phase ratio is the 1-complementary of oop_{ratio}, and can be expressed as:

$$ip_{ratio}[b] = \frac{(1-|g[b]|)^2 E_{M[b]}}{(1-|g[b]|)^2 E_{M[b]} + E_{R[b]}} \tag{5.3-125}$$

The target SNR is the maximum of the interpolations calculated for each frequency band between a SNR of 10dB for in-phase components and 40dB for out-of-phase components:

$$SNR_{target} = \max_b(10.\, ip_{ratio}[b] + 40.\, ip_{ratio}[b]) \tag{5.3-126}$$

The real-valued MDCT coefficients of the residual signal is truncated above a maximum frequency given as input configuration parameter, and form the $input$ vector. The output of the encoder is the global gain index $ggi \in \{0, \ldots, 127\}$ and the entropy coded bits generated by the arithmetic coder. The target SNR is achieved by choosing a suitable global gain index ggi, which is then used to quantize the $input$ vector into the integer q_input vector that will be entropy coded.

The global gain index ggi is dequantized to the global gain gg by the relation

$$gg = dequantize\_gain(gg) = 10^{\frac{90}{20 \cdot 127} ggi}, \text{ for } ggi \in \{0, \ldots, 126\} \tag{5.3-127}$$

and the global gain gg is quantized to the global gain index ggi by the relation

$$ggi = quantize\_gain(gg) = \left\lfloor \frac{20 \cdot 127}{90} \log_{10} gg + 0.4898 \right\rfloor, \text{ for } gg \in [1, 29145] \tag{5.3-128}$$

where 0.4898 is used for rounding instead of 0.5 to achieve optimal mean-squared-error reconstruction.

The special global gain index value 127 is used to indicate that all values in the q_input vector are zero. The global gain index is coded raw using 7 bits, and it is always placed before the entropy coded bits, therefore using the special value 127 signals there are no entropy coded bits to follow.

The $input$ vector is converted to the quantized q_input vector using the dequantized global gain gg derived from the chosen global gain index ggi by the relation

$$q\_input[i] = \left\lfloor \frac{input[i]}{gg} + 0.5 \right\rfloor, \text{for } i \in \{0, \ldots, N-1\} \qquad (5.3\text{-}129)$$

which represents scaling by gg and uniform scalar quantization with rounding to the nearest integer.

Let the block on position $k \in \{0, \ldots, \left\lceil \frac{N}{8} \right\rceil - 1\}$ be extracted as $block[k][i] = q\_input[8 \cdot k + i]$, for $i \in \{0, \ldots, blk\_length[k] - 1\}$, where the block sizes are $blk\_length[k] = \min(8, N - 8 \cdot k)$. For each block, a parameter $param[k] \in \{0, 15\}$ which identifies the model used for coding the block is selected and coded as side information. The value $param[k] = 0$, indicating the very low entropy case, uses a model which allows for coding of $nz\_count \in \{0, \ldots, 3\}$ nonzero values of ± 1, while the rest of the values are 0. This includes the case where all the values in the block are 0. The values $param[k] \geq 1$ use a model assuming the values are generated by a Laplace distribution with scale parameter $2^{param[k]-1}$.

The parameter value $param[k] = 0$ can be used to code only the blocks that satisfy the corresponding model constraints, the other parameter values can encode any arbitrary block, however with different number of bits. For a block, the encoder selects the optimal parameter from those that can be used to code it, such that the total number of bits for coding both the parameter and the block is minimized.

Entropy coding for $param[k] = 0$ of a block starts by coding nz_count, the number of nonzero values of ± 1, with raw coding using 2 bits. Then, the nonzero mask is coded, which contains nz_count ones and $blk\_length - nz\_count$ zeros, with raw coding of the sign bits of the nonzero positions.

```
encode_low_entropy_block(block, blk_length)
{
  nz_count = 0
  for (i = 0; i < blk_length; i++)
  {
    if (block[i] != 0)
    {
      nz_count++
    }
  }

  rc_uni_enc_encode_bits(nz_count, 2)
  left_1 = nz_count
  left_0 = blk_length - nz_count

  for (i = 0; i < blk_length; i++)
  {
    val = block[i]

    if ((left_0 == 0) || (left_1 == 0))
    {
      /* only ones left or only zeros left */
    }
    else
    {
      count_0 = left_0 * ECSQ_tab_inverse[left_0 + left_1]
      rc_uni_enc_encode_bit_prob_fast(abs(val), count0, 14)
    }

    if (val != 0)
    {
      rc_uni_enc_encode_bits(get_sign(val), 1)
      left_1--
    }
    else
    {
      left_0--
    }
  }
}
```

The precomputed table is computed as $ECSQ\_tab\_inverse[k] = \left\lfloor \frac{2^{14}}{k} + 0.5 \right\rfloor$, for $k \in \{1, \ldots, 8\}$. Also, a helper function is used to obtain the sign bit, $get\_sign(x) = 1$, if $x < 0$ and 0, if $x \geq 0$.

During coding of a block, if there are only ones or zeros left, the nonzero mask is already determined. Otherwise, the mask is coded with an adaptive probability model giving the probability of a zero as $p_0 = \frac{left_0}{left_0 + left_1}$. This probability is approximately mapped to a 14-bit frequency count, without using a division operation as $count_0 = \left\lfloor 2^{14} * \frac{left_0}{left_0 + left_1} + 0.5 \right\rfloor \approx left_0 * \left\lfloor \frac{2^{14}}{left_0 + left_1} + 0.5 \right\rfloor$, where both $left_0 \neq 0$ and $left_1 \neq 0$, and the second term in the approximation is available in a precomputed table. The value of $count_1$ is derived implicitly from the relation $count_0 + count_1 = 2^{14}$.

The obtained code length in bits of the nonzero mask is exactly the same as would be obtained by optimal combinatorial coding, which would use $\log_2 \left\langle {8 \atop nz\_count} \right\rangle = \log_2 \frac{8!}{nz\_count! * (8 - nz\_count)!}$ bits.

Entropy coding with $param[k] \geq 1$ of a block starts by computing $shift = \max(0, param[k] - 3)$, which represents the number of least significant bits of the absolute values that are coded approximately uniformly or with raw coding. The most significant bits of each absolute value are coded using a probability model selected by $param[k]$ together with escape coding. For $shift \leq 4$, coding of LSBs takes into account that for absolute values the probability of zero (0 maps to one value) is half of the probability of nonzeros (1 maps to two values, ± 1). For larger shifts, the length difference is negligible and raw coding is used using $shift$ bits. Finally, if the value is nonzero, the sign is coded raw.

```
encode_normal_block(block, blk_length, param)
{
  shift = max(0, param - 3)

  for (i = 0; i < blk_length; i++)
  {
    val = block[i]
    sym = abs(val)

    if (shift != 0)
    {
      lsbs = sym & ((1 << shift) - 1)
      sym = sym >> shift

      arith_encode_prob_escape(ECSQ_tab_vals[param - 1], 16, sym)

      if ((sym > 0) || (shift > 4))
      {
        rc_uni_enc_encode_bits(lsbs, shift)
      }
      else /* (sym == 0) && (shift <= 4) */
      {
        rc_uni_enc_encode_symbol_fast(lsbs, ECSQ_tab_abs_lsbs[shift], 14)
      }
    }
    else
    {
      arith_encode_prob_escape(ECSQ_tab_vals[param - 1], 16, sym)
    }

    if (val != 0)
    {
      rc_uni_enc_encode_bits(get_sign(val), 1)
    }
  }
}
```

The encoding of the entire quantized q_input vector can be expressed in terms of the previous two functions, which encode low entropy blocks and normal blocks, together with a helper function *find_optimal_parameter*, which computes for a block the optimal parameter to use for encoding.

```
encode_raw_vector(q_input, N)
{
  block_cnt = (N + 7) / 8
  for (k = 0; k < block_cnt; k++)
  {
    blk_length[k] = min(8, N - 8 * k)

    for (i = 0; i < blk_length[k]; i++)
    {
      block[k][i] = q_input[8 * k + i]
    }

    param[k] = find_optimal_parameter(block[k], blk_length[k])
    rc_uni_enc_encode_symbol_fast(param[k], ECSQ_tab_param, 14)

    if (param[k] == 0)
    {
      encode_low_entropy_block(block[k], blk_length[k])
    }
    else
    {
      encode_normal_block(block[k], blk_length[k], param[k])
    }
  }
}
```

5.3.3.2.3 Common unified stereo tools

5.3.3.2.3.1 Inter-channel Alignment (ICA)

The Inter-Channel Alignment module is used in based on different scenarios. Audio capture devices in teleconference rooms, or telepresence rooms, include multiple microphones that acquire spatial audio. The spatial audio may include speech as well as background audio that is encoded in a near-end device and transmitted to a far-end device. The speech/audio from a given source (e.g., a talker) arrives at the multiple microphones at different times depending on how the microphones are arranged as well as where the source (e.g., the talker) is located with respect to the microphones and room dimensions. For example, when the sound source is closer to one microphone, mic1, on the near-end device than another microphone, mic2, on the near-end device the sound emitted from the sound source reaches the first microphone earlier in time than the second microphone. The microphones can also receive/detect sounds from multiple sound sources. The multiple sound sources often include a dominant sound source (e.g., a talker) and one or more secondary sound sources (e.g., a passing car, traffic, background music, street noise). The sound emitted from the dominant sound source could reach the first microphone earlier in time than the second microphone.

To illustrate the concept of ICA, consider a teleconference room setting as shown below in Figure 5.3-1.

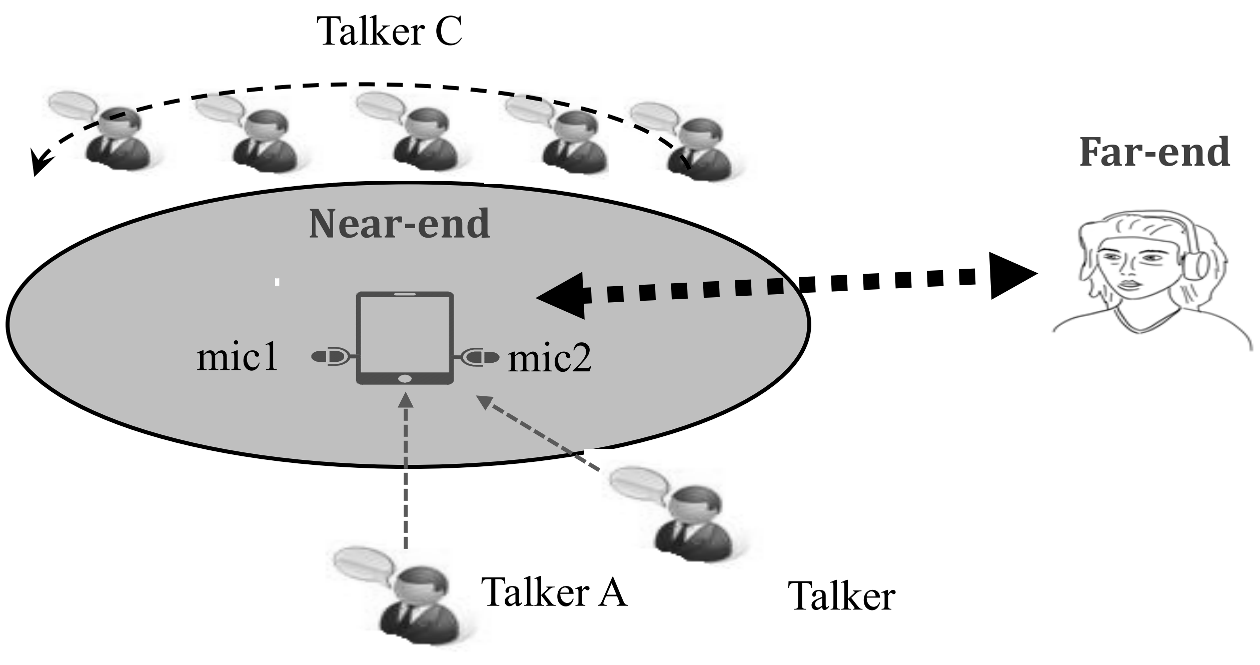


Figure 5.3-1: Teleconference Scenario

Case 1-Temporally Aligned Channels

If Talker A is the person speaking, the sound emitted by Talker A arrives at both near end device microphones, mic1 and mic2, at the same time. In such a case, the channels, Ch1 and Ch2, associated with microphones, m1 and m2, are temporally aligned, as shown in Figure 5.3-2 below.

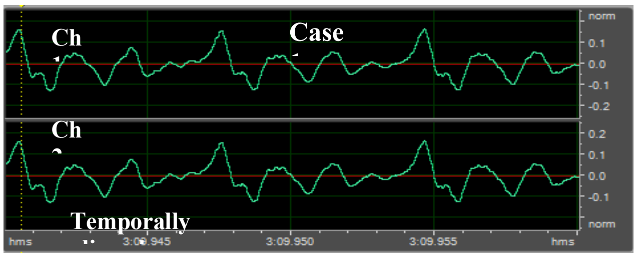


Figure 5.3-2: Temporally Aligned Audio Signals in Ch1 and Ch2

Case 2: Channels temporally NOT aligned

<u>Source to Microphones Distance Difference</u>
If Talker B is the person speaking, the sound emitted by Talker B arrives at both near end device microphones, mic1 and mic2, at different times. In such a case, the channels, Ch1 and Ch2, associated with microphones, m1 and m2, are temporally not aligned.

<u>Position of Source is Moving relative to Microphones</u>
If Talker C, whose position is moving relative to the microphones, mic1 and mic2, is the person speaking, the sound emitted by Talker C arrives at both near end device microphones, mic1 and mic2 at different times. In such a case, the channels, Ch1 and Ch2, associated with microphones, m1 and m2, are temporally not aligned.

<u>Overlapping Sources speaking simultaneously</u>
If Talkers B and C are speaking at the same time, speech is overlapping, and the sound emitted by both of them arrives at both near end device microphones, mic1 and mic2, at different times, which results in varying temporal shift values between the speech of Talker B and C depending on who is the loudest talker, closest to the microphone, etc.
In such scenarios, the channels, Ch1 and Ch2, associated with microphones, m1 and m2, are temporally not aligned, i.e., there is a temporal mismatch between the channels.

In each of these scenarios, the audio signals in the channels are shifted and temporally misaligned as shown in Figure 5.3-3 below.

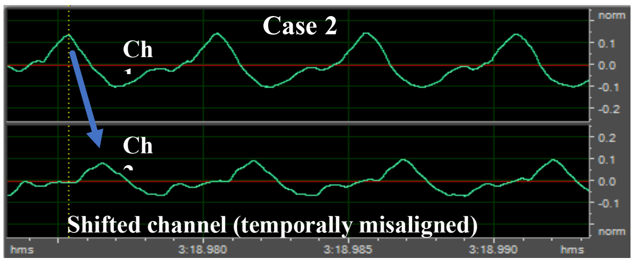


Figure 5.3-3: Temporally Misaligned Audio Signals in Ch1 and Ch2

The near-end device encodes the audio signal captured by the microphones, mic1 and mic2, in different frames that contain varying PCM samples depending on the sampling rate, e.g., 960 samples @ 48kHz. Mid-side (MS) coding and parametric stereo (PS) coding are stereo coding techniques that provide improved efficiency over the dual-mono coding techniques.

In dual-mono coding, the Left (L) channel (or signal) and the Right (R) channel (or signal) are independently coded without making use of inter-channel correlation. MS coding reduces the redundancy between a correlated L/R channel-pair by transforming the Left channel and the Right channel to a sum-channel and a difference-channel (e.g., a side channel) prior to coding. The sum signal and the difference signal are waveform coded in MS coding. Relatively more bits are spent on the sum signal than on the side signal.

Depending on a recording configuration, i.e., the placement of the microphones relative to one or more sound sources, there is a temporal shift between a Left channel and a Right channel, as well as other spatial effects such as echo and room reverberation. When the temporal shift and phase mismatch between the channels are not compensated, the sum channel and the difference channel contain comparable energies reducing the coding-gains associated with MS or PS techniques. The reduction in the coding-gains is based on the amount of temporal (or phase) shift. The comparable energies of the sum signal and the difference signal limits the usage of MS coding in certain frames where the channels are temporally shifted but are highly correlated. In stereo coding, a Mid channel (e.g., a sum channel) and a Side channel (e.g., a difference channel) is generated by downmixing M= (L+R)/2 and S= (L-R)/2, where M corresponds to the Mid channel, S corresponds to the Side channel, L corresponds to the Left channel, and R corresponds to the Right channel. The reverse process of generating the Left channel and the Right channel from the Mid channel and the Side channel is an upmixing algorithm.

When the sound source is closer to the first microphone, mic1, than to the second microphone, mic2, frames of the second audio signal are delayed relative to frames of the first audio signal. In this case, the first audio signal is referred to as the "reference audio signal" or "reference channel" and the delayed second audio signal is referred to as the "target audio signal" or "target channel". Alternatively, when the sound source is closer to the second microphone than to the first microphone, frames of the first audio signal is delayed relative to frames of the second audio signal. In this case, the second audio signal is referred to as the reference audio signal or reference channel and the delayed first audio signal is referred to as the target audio signal or target channel.

The encoder determines the target signal, or the reference signal based on a mismatch value (e.g., an estimated shift value or the final shift value) corresponding to a current frame to be encoded and mismatch (e.g., shift) values corresponding to previous frames.

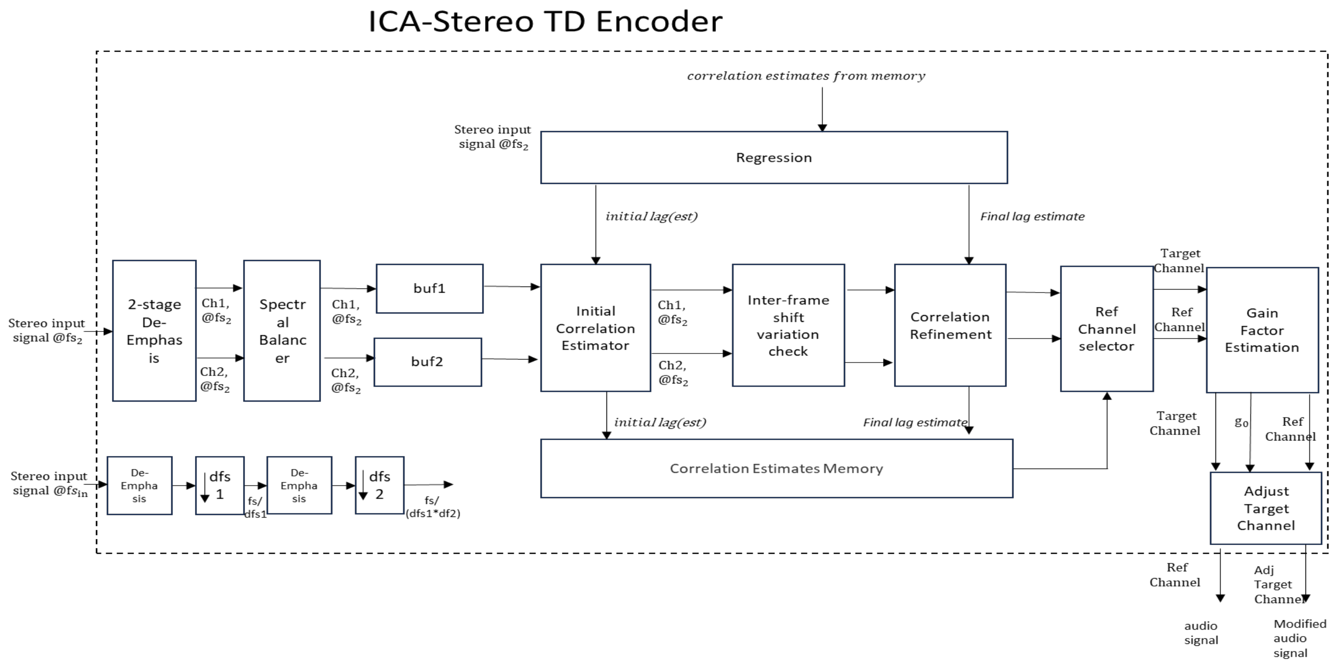


Figure 5.3-4: ICA block diagram, encoder part

<u>**Two-Stage Deemphasis/Spectral Balancer**</u>

The stereo signals are passed through a de-emphasis filter then down-sampled (dfs1), then passed through the de-emphasis filter again and down-sampled (fs2). The de-emphasis filters counteract the effect of boosting higher frequencies. The de-emphasis filter is 1/(1-μz^-1), where μ is the pre-emphasis factor. The de-emphasis on the stereo signals is performed twice to ensure the audio signals are properly equalized at the lower sampling rate. Application of the de-emphasis filters before each stage of down-sampling ensures that the high-frequency components of the audio signals are reduced to preserve the quality of the audio signal at the lower sampling rate, i.e., there is not any unwanted distortion introduced into the signal. Performing de-emphasis before down-sampling minimize aliasing, as it reduces the high-frequency content that could potentially be aliased to lower frequencies when the sampling rate is reduced. The stereo signals are then passed into a spectral balancer which applies a high-pass filter to the input signal to handle low-frequency rumble and compensate for pre-emphasis. The filter coefficients are defined based on the input sampling rate.

Initial Lag Estimator-Correlation

A sliding window cross-correlation calculation between two signals to measure the similarity between the left and right channels of a stereo audio signal at different time shifts (lags) over a lag range. If buf1 and buf2 are perfectly aligned and identical, the cross-correlation will be maximum at zero lag. If buf1 leads or lags behind buf2, the cross-correlation will be maximum at a positive or negative lag, respectively. By searching over a range of lags, the time shift that best aligns the two signals is estimated. The cross-correlation at a specific lag is calculated as the dot product of buf1 and buf2 shifted by the lag, and essentially measures how much the signals overlap for the given time shift. The result is then normalized by the total energy of the signals to give the final cross-correlation value.

The energy in each buffer (buf1 and buf2) is computed using the sum of the squares of each signal's amplitude values. For example, E1 = Σ [buf1(n) buf1(n)] and E2 = Σ [buf2(n) buf2(n)]. These energy values are used to normalize the cross-correlation values. The purpose of this normalization is to make the cross-correlation values independent of the absolute energy levels of the signals, allowing for a more meaningful comparison of their temporal alignment. The normalization factor is calculated as the inverse of the square root of the product of the energies, i.e., normalization factor = 1 / sqrt(E1 E2). The cross-correlation at a specific lag is calculated as the dot product of buf1 and buf2 shifted by the lag, and then multiplied by the normalization factor:

Normalized Rxy(*l*) = Rxy(*l*)*normalization factor, where R_xy(*l*) is the cross-correlation of buf1 and buf2 at lag *l*, where x = buf1, and y= bu2. This gives the final cross-correlation value, which is stored in the corrEst memory buffer. This value represents the similarity between buf1 and buf2 at a specific time shift, normalized by the energy of the signals. The process of measuring similarity continues for both negative and positive lags. The corrEst memory buffer is updated with the latest cross-correlation value for each lag. These lags define the range of shifts used to compare the two signals The lag search range includes a minimum and maximum lag to compute cross-correlation between the two input signals. The cross-correlation for negative lags is computed when the signal in buf1 lags behind the signal in buf2. The cross-correlation for positive lags is computed when the signal in buf1 leads the signal in buf2. The initial estimate of the correlation lag(corrLogStats[0]) is obtained by finding the maximum value in the cross-correlation between the two audio channels and the corresponding lag over the lag search range (minlag-maxlag).

Regression Analysis

A linear regression is performed to estimate the correlation of the previous frame. The regression is based on the delay_0_mem[], which stores the delay values for the previous MAX_DELAYREGLEN frames. The regression line is characterized by the slope beta_reg and the y-intercept alpha_reg. The slope beta_reg is calculated as the covariance of X and Y (where X is the frame index and Y is the delay value) divided by the variance of X. The y-intercept alpha_reg is then calculated as the mean of Y minus the slope beta_reg times the mean of X. The predicted correlation of the previous frame, reg_prv_corr, is then calculated as the slope beta_reg times the total number of frames MAX_DELAYREGLEN plus the y-intercept alpha_reg as shown in the equation below.

reg_prv_corr = beta_reg * MAX_DELAYREGLEN + alpha_reg

The prediction of the correlation of the previous frame, reg_prv_corr, serves as an estimate of the temporal shift that needs to be applied to the current frame to align it with the previous frame. This is useful to exploit the temporal redundancies between successive frames to achieve efficient compression. The predicted correlation from the past frame (reg_prv_corr) is used as a reference point to check and adjust the initial correlation lag estimate (corrLagStats[0]). If the initial lag estimate deviates significantly from the predicted correlation from the past frame, the initial estimate may not be accurate. In such a case, the initial lag estimate is updated to be closer to the predicted correlation from the past frame. This helps to ensure a more accurate temporal adjustment of the current frame, leading to better audio quality and coding efficiency. Furthermore, the predicted correlation of the previous frame is also used to calculate a bias and width for a local weighting window loc_weight_win[]. The predicted correlation of the previous frame is then used to adjust the initial correlation estimate (corrEst[i]) by applying the local weighting window (loc_weight_win[]).

```
// Adjust the correlation estimates[i]

for (i = 0, j = L_NCSHIFT_DS - TRUNC(reg_prv_corr); i < 2 * L_NCSHIFT_DS + 1; i++, j++)
  {corrEst[i] *= loc_weight_win[j]; }
```

The local weighting window has a bias towards the estimated lag, which means that correlation estimates around the estimated lag are given more weight. This can help to emphasize or de-emphasize certain lags in the correlation estimate based on the predicted correlation of the previous frame.

After this adjustment by applying the weighting window, the maximum value in the adjusted correlation estimate is found again (if applicable, see below) to compute corrLagStats[1], which is the refined estimate of the correlation lag. The final correlation lag estimate (corrLagStats[2]) is then computed by restricting the refined estimate within a certain range and applying another weighting window.

While corrLagStats[0] provides an initial estimate of the lag by finding the maximum value of the cross-correlation, it does not consider any temporal trends in the delay between the two audio channels. The regression analysis, on the other hand, fits a linear model to the delay values across different time points. This captures trends in the delay, such as if the delay is consistently increasing or decreasing over time. The resulting estimate (reg_prv_corr) could be more accurate if such trends are present. Using both the regression and the correlation provides for a more robust lag estimate.

Long -Term Correlation

A long-term correlation estimate is preformed (corrEstLT) to provide a more stable and reliable estimate of the inter-channel correlation over time. This is done by scaling the current long-term correlation estimate by alpha and the current correlation estimate by (1-alpha). Then add the scaled current correlation estimate to the scaled long-term correlation estimate to update the long-term correlation estimate. Here alpha is a smoothing factor and corrEst is the current estimate of the cross-correlation. This is a form of exponential smoothing, where the long-term estimate is a weighted combination of the previous long-term estimate and the current estimate, with weights alpha and 1-alpha. This smoothed version of the inter-channel correlation estimate is updated at every frame based on the current correlation estimate, corrEst.

Inter-Frame Lag Variation Check

Check the absolute value of the currentNCShift-pastNCshift > Threshold (N_MAX_SHIFT_CHANGE) TODO update description further

Lag Refinement of initial lag Estimate using cross-correlation

A refined estimate, corrLagStats[1] of the correlation lag is computed when the absolute value of the currentNCShift-pastNCshift > Threshold. The refined estimate of the correlation lag is obtained by interpolating the cross-correlation around the initial correlation lag estimate and finding the maximum of the interpolated cross-correlation. A sinc interpolation window is used and is based on the downsample factor dsFactor. The equation below scales the initial correlation lag estimate by the downsample factor to get an initial estimate of the correlation lag at the original sampling rate. corrLagStats[1] = corrLagStats[0] * dsFactor . For each lag in the interpolation range [interpMin and interpMax], an interpolation of the cross-correlation is computed a weighted sum of the cross-correlation values at nearby lags. The weights are given by the values of the sinc interpolation window. The maximum value of the interpolated cross-correlation is used to refine the intial estimate of the correlation lag, which is stored in corrLagStats[1], and also the computed correlation lag statistics are saved for the current frame.

Final Correlation Lag Estimate

A final correlation lag estimate, corrLagStats[2], is used for the adjustment of the channels in the next frame. It is obtained by restricting the refined correlation lag estimate within a certain range and applying a weighting window to bias (using the regression analysis discussed above) the correlation towards the estimated lag.

The currentNCShift is updated based on corrLagStats[2] and scaled by the downsample factor. The updated currentNCShift is then used in the temporal adjustment of the target signal and the downmix gain estimation.

Channel Selector

The reference channel is the channel that the other channel (the target channel) is compared to and adjusted based on. The sign of currentNCShift effectively determines which channel is leading or lagging in time, which is why it used to choose the reference channel. The reference channel index (refChanIndx) is determined based on whether currentNCShift is positive or negative.

If currentNCShift (which is updated to corrLagStats[2]) is greater than or equal to 0, the reference channel index (refChanIndx) is set to the left channel index (L_CH_INDX), i.e., the left channel is the reference channel. If currentNCShift is less than 0, the reference channel index(refChanIndx) is set to the right channel index (R_CH_INDX), i.e., the right channel is the reference channel.

```
/* reference channel index */
if (hStereoTCA->corrLagStats[2] >= 0) {
    hStereoTCA->refChanIndx = L_CH_INDX;
} else {
    hStereoTCA->refChanIndx = R_CH_INDX;
}
```

A reference channel selector determines which channel is the reference channel (ref) and which is the target channel (target) based on the refChanIndx. If the reference channel index (refChanIndx) is equal to the left channel index (L_CH_INDX), the left channel is designated as the reference audio signal and the right channel is designated as the target audio channel.

```
if ( hStereoTCA->refChanIndx == L_CH_INDX )
{
    ref = ptrChanL;
    target = ptrChanR;
}
else
{
    ref = ptrChanR;
    target = ptrChanL;
}
```

Downmix Gain Estimation

The gain factor is estimated based on the relative energy levels of the two channels. It's an estimate because it's based on the energy levels in the current frame of audio, and these energy levels can vary from frame to frame. The gain factor equalizes the energy levels of the two channels before they are mixed together. Adjusting the level of one channel (the target or non-reference channel) to match the level of the other channel (the reference channel) as closely as possible before downmixing helps maintain the perceived loudness and quality of the audio signal after downmixing.

The gain factor is estimated by calculating the sum of the absolute values of the samples in each channel (Ref and Targ), which serves (though not strictly energy – a square of amplitudes) as an estimate of the energy in that channel. The gain factor is then calculated as the ratio of these two energy estimates.

$$gain\_factor = \frac{\sum_{i=0}^{N-N1}|Ref(i)|}{\sum_{i=0}^{N-N1}|Targ(i+N_1)|}$$

There is also a check on the voice activity detection flags and adjustments are made to the gain factor in cases of silent or near-silent passages. The gain factor is quantized and later used to scale the target channel.

Target Signal Adjustment

The temporal alignment of the lookahead samples in the target channel to match the reference channel includes adjusting the level of the target channel samples based on the level of the reference channel samples, with a windowing function used to smoothly transition the level adjustment over the frame. The window length is determined by the frame length and the current non-causal shift. The window values are calculated using a sine function. Windowing and the gain adjustment to the target channel. For the first part of the window, it calculates a weighted average of the reference and target samples, with the weights determined by the windowing function. For the remaining samples, it applies the gain adjustment to the reference samples. This adjustment is applied in-place to the target channel, so the modified target channel samples replace the original samples. The goal of this process is to minimize the perceptual impact of the non-causal shift on the encoded audio signal. After the target channel has been adjusted the memory is updated with the last input and output samples.

As can be seen in Figure 5.3-5, the previous frame audio signals and current frame audio signals are stored in memory buffers, Buf1 and Buf2.

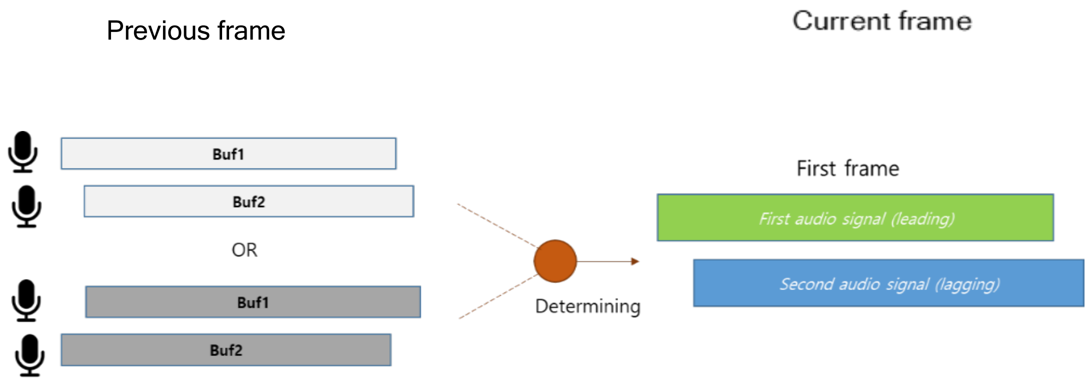


Figure 5.3-5: Reference and target signal determination

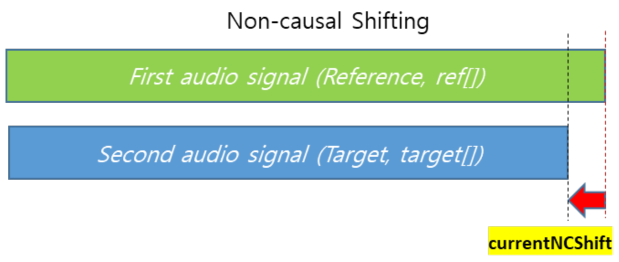

Non-causal Shifting
First audio signal (Reference, ref[])
Second audio signal (Target, target[])
currentNCShift

Figure 5.3-6: Non-causal shifting

To maximally align the first audio signal, i.e., reference signal in Figure 5.3-7, with the second audio signal, i.e., target signal in Figure 5.3-7, generation of new audio data is performed. New audio data samples are generated from the reference signal, Target [i + currentNCShift] = gAdj*ref[i] which is designated in the purple area below. New audio data is also generated based on both the reference signal (designated in the darker blue area below) and target signal, Target [i + currentNCShift] = (win[j]*gAdj*ref[i] + (1.0f-win[j])*target[I + currentNCShift).

The new audio samples are selected samples of the second audio signal (i.e., the target signal) based on the non-causal shift value (NCShift_ and used to generate a modified (e.g., time-shifted) audio signal.

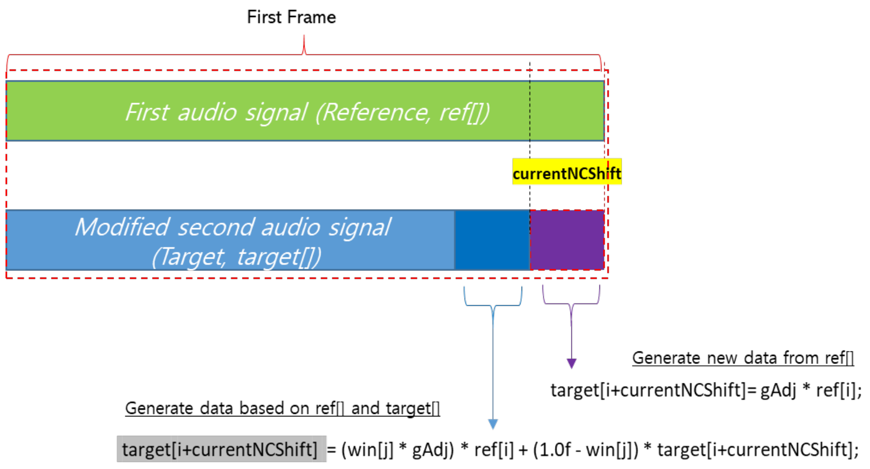

First Frame
First audio signal (Reference, ref[])
currentNCShift
Modified second audio signal (Target, target[])
Generate new data from ref[]
target[i+currentNCShift]= gAdj * ref[i];
Generate data based on ref[] and target[]
target[i+currentNCShift] = (win[j] * gAdj) * ref[i] + (1.0f - win[j]) * target[i+currentNCShift];

Figure 5.3-7: Generation of new audio data

Purple Area in Figure 5.3-7:

```
        for ( ; i < input_frame; i++, j++ )
            target[i + currentNCShift] = gAdj * ref[i];
```

Dark Blue Area in Figure 5.3-7:

```
        for ( i = ( input_frame - currentNCShift - tempS ); i < input_frame - currentNCShift; i++, j++ )
                    target[i + currentNCShift] = ( win[j] * gAdj ) * ref[i] + ( 1.0f - win[j] ) * target[i + c
urrentNCShift];
```

5.3.3.2.3.2 Inter-channel Bandwidth Extension (IC-BWE)

The IVAS encoder is part of a complex audio processing system that performs multiple operations to enhance audio signals. The core functionality revolves around an encoder that generates a high-band portion of a signal from a left and a right signal. A reference signal is designated from these two signals, which is then used to generate a set of adjustment gain parameters that determine the amplification or attenuation of the non-reference signal.

The system also performs a bandwidth extension on the encoded signal, increasing the frequency range and thus enhancing audio quality. It generates a mid-channel time-domain high-band signal and a first and second channel time-domain high-band signal. The system also generates a first and second high-band mixing gain based on a specific flag related to the non-harmonic high band.

The bitstream created based on the first and second high band mixing gains, can be used for further processing or storage. This compact representation of the high-band mixing gains captures essential information needed for reconstructing the audio signals.

In essence, this system provides potential benefits such as better voice audio quality and lower bitrate communications due to its ability to extend the bandwidth of the encoded signal, adjust the gain of the non-reference signal, and generate high-band mixing gains.

Core functionality

The system's core functionality revolves around an encoder that generates a first high-band portion of a first signal. This signal is essentially derived from two distinct signals. These are represented in the source code as shb_speech_ref and shb_speech_nonref.

The encoder generates a first high-band portion of a signal based on a left signal and a right signal. A target channel signal is produced by combining the first channel time-domain high-band signal with a first channel low-band signal. A reference channel signal comes from the combination of the second channel time-domain high-band signal and a second channel low-band signal. The target channel signal undergoes modification based on a temporal mismatch value to create a modified target channel signal. Identification of a non-reference target channel happens through temporal shift values in the current frame. A synthesized non-reference high-band channel is then generated from a non-reference high-band excitation corresponding to the non-reference target channel. Spectral mapping parameters are estimated through a maximum-likelihood measure applied to the synthesized non-reference high-band channel and a high-band portion of the non-reference target channel. The synthesized non-reference high-band channel is subjected to these spectral mapping parameters to create a spectrally shaped synthesized non-reference high-band channel. An encoded bitstream is generated from the spectral mapping parameters and the spectrally shaped synthesized non-reference high-band channel.

Reference signal designation

Once these signals are introduced into the system, the next step involves designating a reference signal from these two signals. This task of designation is performed by selecting either the left or right signal depending on certain conditions. This operation to designate a reference signal from the left and the right signal is executed in the source code using the refChanIndx_bwe variable. This variable essentially stores the index of the signal which is chosen as the reference.

The identification of the non-reference target channel is based on temporal mismatch (shift) values between the channels. The temporal mismatch values are used to modify the target channel.

From an algorithmic perspective, the findRefChanBWE function operates in the following way:

1. Copy the audio data from two input channels into two separate arrays, inp0 and inp1.
2. Apply a spectral balancing function to each channel. The spectral balancing function modifies the audio signal so that all frequency components have roughly the same power. Mathematically, spectral balancing can be represented as follows:

$Y(w) = X(w) * H(w)$
Here, $X(w)$ is the Fourier transform of the input signal, $H(w)$ is the frequency response of the spectral balancer, and $Y(w)$ is the Fourier transform of the output signal.

3. Calculate the energy of each channel. The energy E of a signal $x[n]$ of length N is defined as follows:

$E = \Sigma|x[n]|^2$ for $n = 0$ to $N-1$
Here, $|x[n]|^2$ is the square of the absolute value of the nth sample. This is a summation over all samples.

4. Compare the energy of the two channels. If the energy of channel 1 is less than 64% of the energy of channel 0, select channel 0 as the reference. If the energy of channel 0 is less than 64% of the energy of channel 1, select channel 1 as the reference. If neither condition is met, leave the reference channel unchanged.

Adjustment gain parameters

Having identified the reference signal, the encoder then proceeds to generate a set of adjustment gain parameters. These parameters are crucial as they determine the amplification or attenuation of the non-reference signal to match the reference. Spectral mapping parameters are based on a maximum-likelihood measure applied to the synthesized non-reference high-band channel and a high-band portion of the non-reference target channel.

After the reference signal is identified, the encoder begins creating a set of adjustment gain parameters. These parameters play a crucial role in adjusting the non-reference signal to correspond with the reference signal. The spectral mapping parameters are based on a maximum-likelihood measure applied to the synthesized non-reference high-band channel and a high-band portion of the non-reference target channel.

In the process, the encoder compares the energy of the reference and non-reference signals and adjusts the gain of the non-reference signal to match that of the reference. The overall absolute energy of the synthesized speech of the reference channel and the non-reference channel is initially calculated. This energy is then smoothed with the previous energy. If the smoothed energy of the non-reference channel is zero, the gain shape mapping factor remains unchanged. Otherwise, it's updated based on the relative target gain and the ratio of the smoothed energies of the reference and non-reference channels.

A quantization process is used to identify the index that results in the smallest error or distortion. The spectral parameters are then used to produce a spectrally shaped synthesized non-reference high-band channel and to generate an encoded bitstream based on the spectral mapping parameters and the spectrally shaped synthesized non-reference high-band channel. The high-band non-reference signal and a synthesized signal are among the several arguments that the function receives.

The updated gain shape mapping factor is quantized using an appropriate table, based on the mode. The quantization index is then returned. This process is utilized in the encoding process of inter-channel bandwidth extension for applying the gain shape mapping of a reference channel to the non-reference channel.

Bandwidth extension

Once the adjustment gain parameters are generated, the system performs a bandwidth extension operation on the encoded signal. The purpose of this operation is to increase the frequency range of the signal, thereby enhancing the audio quality. This operation is performed in the source code by calling the icbwe_dft_stereo_param function.

The icbwe_dft_stereo_param function takes several parameters, including the high-band non-reference signal (represented as shb_synth_nonref in the code), and performs a Discrete Fourier Transform (DFT) on it. The output of this function is a signal that has been extended to include higher frequency components.

The process also performs spectral smoothing, calculates and quantizes spectral and gain shape mapping, and applies these mappings to the non-reference SHB signal.

The spectral shape mapping adjusts the spectral shape of the non-reference channel to match that of the reference channel, while the gain shape mapping adjusts the gain of the non-reference channel to match that of the reference channel.

The process further also updates certain values in the ICBWE encoder structure based on the computed parameters and the current state of the audio encoder.

This process is used specifically when the Channel Pair Element (CPE) mode is set to IVAS*CPE*DFT, which signifies the Discrete Fourier Transform (DFT) stereo processing mode.

Time-domain high-band signal generation

Following the bandwidth extension operation, the system generates a mid-channel time-domain high-band signal, as well as a first and second channel time-domain high-band signal. This operation is performed by invoking the syn_filt function in the code. This function takes as input the high-band non-reference signal and a set of filter coefficients and generates the time-domain high-band signals.

The syn_filt function performs the operation of generating a mid-channel time-domain high-band signal, a first and second channel time-domain high-band signal. The function performs a filtering operation on the input signal to create these time-domain high-band signals, which are then used in further processing stages.

High-band mixing gain

Once the audio signals are received, the system then generates a first and second high-band mixing gain. These gains are computed based on a specific flag related to the non-harmonic high band. This flag is determined based on a voicing value and a gain value. This operation is performed by the ic_bwe_enc_gsMapping function in the code. This function calculates the mixing gains by comparing the energy of the reference and non-reference signals in such a way that the non-harmonic components are minimized.

The first and second high band mixing gains are parameters that are generated during the encoding process. They represent the adjustment factors applied to the high-band non-reference signal to match the energy level of the reference signal. The generation of these gains is accomplished through the application of a maximum likelihood measure to the synthesized non-reference high-band channel and a high-band portion of the non-reference target channel.

The first high band mixing gain is typically applied directly to the non-reference high-band signal, while the second high band mixing gain is used for subsequent adjustments based on the spectral parameters.

These mixing gains are used to produce a spectrally shaped synthesized non-reference high-band channel. The spectral parameters and the spectrally shaped synthesized non-reference high-band channel are then used to generate an encoded bitstream. This process ensures the non-reference signal aligns with the reference signal, which is crucial for the inter-channel bandwidth extension encoding process.

Bitstream output

The IVAS encoder generates a bitstream, which is based on the first and second high-band mixing gains.

5.3.3.2.3.3 Front VAD

The front VAD is applied on the input stereo signals for signal activity detection to aid stereo classification and selection between the TD-based stereo and DFT-based stereo, as well as coordinating the selection of CNG encoding mode for the input stereo channels.

The signal activity detection for each input stereo signal is done as for the IVAS core, see clause 5.2.2.1.4. The front VAD has independent states for the left and right channel and is independent from the IVAS core VAD(s). The speech/music classifier input for the front VAD is however shared with the core VAD (channel 0), obtained from the previous frame, to reduce complexity.

A joint VAD decision VAD_{01} for the input stereo signals is obtained as

$$VAD_{01} = VAD_0 \vee VAD_1 \quad (5.3\text{-}130)$$

where VAD_0 and VAD_1 are signal activity flags indicating activity for channel 0 and 1 respectively.

5.3.3.2.4 Stereo classifier

The stereo classifier selects between the TD stereo mode and the DFT stereo mode. The selection is done in each frame based on stereo cues and stereo parameters. As a general rule, TD stereo mode is mostly selected for the encoding of stereo signals with uncorrelated left and right channels, especially signals with overlapping speech (cross-talk). The DFT stereo mode is mostly selected for stereo signals with correlated left and right channels with single-talk speech, music and mixed content.

The stereo classifier is composed of the following modules: 1) classifier of uncorrelated content, 2) cross-talk detector, and 3) stereo mode selector.

5.3.3.2.4.1 Classification of uncorrelated content

The classification of uncorrelated content is done in both, the TD stereo mode and the DFT stereo mode. The classification is based on the Logistic Regression (LogReg) model. The LogReg model is trained individually for the TD stereo mode and for the DFT stereo mode on a large database of stereo cues extracted from the IVAS coder.

The classification of uncorrelated content in the TD stereo mode uses the stereo cues:

- position of the maximum inter-channel cross-correlation function, k_{max}
- instantaneous target gain, g_t
- unnormalized cross-channel correlation at lag 0, p_{LR}
- side-to-mono energy ratio, r_{SM}
- difference between the maximum and the minimum cross-channel correlation, d_{mmLR}
- absolute difference between the Left-to-Mono and the Right-to-Mono correlation, d_{LRM}
- zero-lag cross-channel correlation value, R_0
- position of the maximum windowed inter-channel cross-correlation function, k_{wmax}
- maximum of the windowed inter-channel cross-correlation function, R_{wmax}
- evolution of the inter-channel cross-correlation function, RR

In total, there are $F_{TD}^{UNCLR} = 10$ stereo cues used by the classifier of uncorrelated content in the TD stereo mode forming a feature vector $\mathbf{f}_{raw,TD}^{UNCLR}$. For the simplicity of notation the subscript "TD" and the superscript "UNCLR" will be ommitted from the variables unless specifically stated. Thus, the input feature vector will be denoted as $\mathbf{f}_{raw}$.

The feature vector is normalized by removing its mean and scaling it to unit variance. This is done as follows

$$f_i = \frac{f_{i,raw} - \overline{f_i}}{\sigma_{f_i}}, \quad i = 1, \dots, F \quad (5.3\text{-}131)$$

where $\overline{f_i}$ is the global mean of the ith feature across the training database and σ_{f_i} is the global variance of the ith feature across the training database. The global means $\overline{f_i}$ and the global variances σ_{f_i} for all $i = 1,\dots,F$ are defined as constants in the IVAS coder.

The LogReg model takes the real-valued features as an input vector and makes a prediction as to the probability of the input belonging to the uncorrelated class (class 0). The output of the LogReg model is real-valued and takes the form of linear regression. That is

$$y_p = b_0 + \sum_{i=1}^{F} b_i f_i \tag{5.3-132}$$

where b_i are the constant coefficients of the LogReg model. The real-valued output y_p is then transformed into a probability using the logistic function. This is done as

$$\mathrm{p}(\mathrm{class} = 0) = \frac{1}{1+e^{-y_p}} \tag{5.3-133}$$

The probability $\mathrm{p}(\mathrm{class} = 0)$ is a real value between 0 and 1 where higher value means that the content in the current frame is more uncorrelated. The output of the LogReg model y_p is normalized with the function shown in fig. 5.3-10.

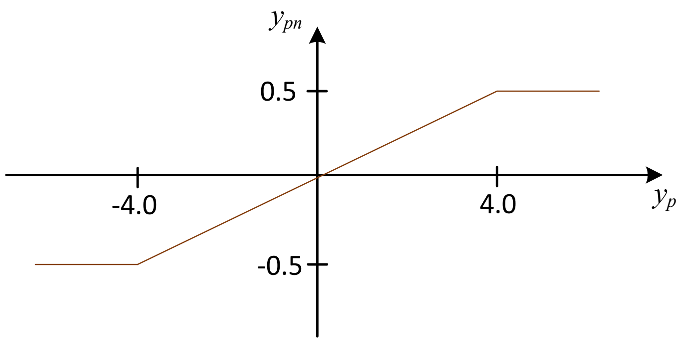


Figure 5.3-10: Normalization of the LogReg output of the classifier of uncorrelated content in the TD stereo mode

The normalization can be mathematically described as follows

$$y_{pn}(n) = \begin{cases} 0.5 & \text{if } y_p(n) \geq 4.0 \\ 0.125 y_p(n) & \text{if } -4.0 < y_p(n) < 4.0 \\ -0.5 & \text{if } y_p(n) \leq -4.0 \end{cases} \tag{5.3-134}$$

The normalized output of the LogReg model $y_{pn}(n)$ is then weighted with the relative frame energy as follows

$$scr_{UNCLR}(n) = y_{pn}(n) \cdot E_r(n) \tag{5.3-135}$$

where $E_r(n)$ is the relative frame energy defined in eq. XX. The normalized weighted output of the LogReg model $scr_{UNCLR}(n)$ is called the "non-correlation score".

The non-correlation score $scr_{UNCLR}(n)$ contains occasional short-term "peaks" resulting from imperfect statistical model. The peaks can be filtered out by a simple averaging filter such as the first-order IIR filter. Unfortunately, the application of such averaging filter usually results in smearing of the rising edges representing transitions between the correlated and uncorrelated content. To preserve the rising edges it is necessary to reduce or even stop the smoothing process when a rising edge is detected in the input signal. The detection of rising edges in the input signal is done by analyzing the evolution of the relative frame energy.

The rising edges of the relative frame energy are found by filtering the relative frame energy with a cascade of $P = 20$ identical first-order RC filters each of which has the following form

$$F_p(z) = \frac{b_1 z^{-1}}{a_0 + a_1 z^{-1}}, \quad p = 1,\dots,20 \tag{5.3-136}$$

where a_0, a_1 and b_1 are the constants of the RC filter. The filtering of the relative frame energy with the cascade of $P = 20$ RC filters is done as follows

$$\begin{aligned} E_f^{[0]}(n) &= t_{edge} \cdot E_f^{[0]}(n-1) + \left(1 - t_{edge}\right) \cdot E_{rl}(n) \\ E_f^{[1]}(n) &= t_{edge} \cdot E_f^{[0]}(n-1) + \left(1 - t_{edge}\right) \cdot E_f^{[0]}(n) \\ &\dots \\ E_f^{[p]}(n) &= t_{edge} \cdot E_f^{[p]}(n-1) + \left(1 - t_{edge}\right) \cdot E_f^{[p-1]}(n) \end{aligned} \qquad (5.3\text{-}137)$$

where

$$\frac{-a_1}{a_0} = \tau_{edge}, \quad \frac{b_1}{a_0} = 1 - \tau_{edge} \qquad (5.3\text{-}138)$$

The superscript $[p]$ has been added to denote the stage in the RC filter cascade. The output of the cascade of RC filters is equal to the output from the last stage, i.e.

$$E_f(n) = E_f^{[P-1]}(n) = E_f^{[19]}(n) \qquad (5.3\text{-}139)$$

The cascade of first-order RC filters acts as a low-pass filter with a relatively sharp step function. When used on the relative frame energy it tends to smear out occasional short-term spikes while preserving slower but important transitions such as onsets and offsets. The slope of the rising edges is calculated as the difference between the relative frame energy and the filtered output. That is

$$f_{edge}(n) = 0.95 - 0.05\left(E_{rl}(n) - E_f(n)\right) \qquad (5.3\text{-}140)$$

where $f_{edge}(n)$ is limited to the interval $\langle 0.95, \dots, 0.95 \rangle$. The score of the LogReg model scr is then smoothed with an IIR filter where $f_{edge}(n)$ is used as the forgetting factor. The smoothing is done as follows

$$wscr_{UNCLR}(n) = f_{edge}(n) \cdot wscr_{UNCLR}(n-1) + \left(1 - f_{edge}(n)\right) \cdot scr_{UNCLR}(n) \qquad (5.3\text{-}141)$$

The classification of uncorrelated content in the DFT stereo mode is done similarly as the classification of uncorrelated content in the TD stereo mode described in equations (5.3-131) to (5.3-141). However, there are some notable differences.

In the DFT stereo mode the feature vector $\mathbf{f}_{raw}$ consists of the following stereo cues:

- ILD gain, g_{ILD}
- IPD gain, g_{IPD}
- IPD rotation angle, φ_{rot}
- prediction gain, g_{pred}
- mean energy of the inter-channel coherence, E_{coh}
- ratio of maximum and minimum intra-channel amplitude products, r_{PP}
- overall cross-channel spectral magnitude, f_X
- the maximum value of the GCC-PHAT function, G_{ITD}

In total, there are $F = 8$ features used by the classifier of uncorrelated content in the DFT stereo encoder. The normalization of the feature vector is done as per eq. (5.3-131) where the global means $\overline{f_i}$ and the global variances σ_{f_i} for all $i = 1, \dots, F$ are defined specifically for the DFT stereo mode.

The LogReg model used in the DFT stereo mode is similar to the LogReg model in the TD stereo mode. The output of the LogReg model y_p is calculated with eq. (5.3-132) and the probability that the current frame is uncorrelated (class 0) is given by eq. (5.3-133). The raw output of the LogReg mode y_p is normalized similarly as in the TD stereo mode and according to the function in fig. 5.3-10. The normalization is mathematically described as follows

$$y_{pn}(n) = \begin{cases} 0.5 & \text{if } y_p(n) \geq 4.0 \\ 0.125 y_p(n) & \text{if } -4.0 < y_p(n) < 4.0 \\ -0.5 & \text{if } y_p(n) \leq -4.0 \end{cases} \qquad (5.3\text{-}142)$$

The normalized output of the LogReg model $y_{pn}(n)$ is weighted with the relative frame energy as follows

$$scr_{UNCLR}(n) = y_{pn}(n) \cdot E_r(n) \quad (5.3\text{-}143)$$

and the output value is called the "non-correlation score". The non-correlation score is reset to 0 when the alternative VAD flag $f_{xVAD}(n)$ defined in eq. XX is reset to 0. That is

$$scr_{UNCLR}(n) = 0, \quad \text{if } f_{xVAD}(n) = 0 \quad (5.3\text{-}144)$$

Finally, the non-correlation score $scr_{UNCLR}(n)$ is smoothed with the IIR filter using the rising-edge detection mechanism described in eq. (5.3-141). The output of the smoothing filter is denoted $wscr_{UNCLR}(n)$.

The classifier of uncorrelated content takes the smoothed non-correlation score $wscr_{UNCLR}(n)$ and makes a binary decision. Let $c_{UNCLR}(n)$ denote the binary output of the classifier with "1" representing the uncorrelated content and "0" representing the correlated content. The final output of the classifier is a state variable. It is initialized to 0. The classifier state is changed from the current class to the complementary class only in frames where certain conditions are met.

The mechanism for switching between the output states is depicted in fig. 5.3-11 in the form of a simple state machine.

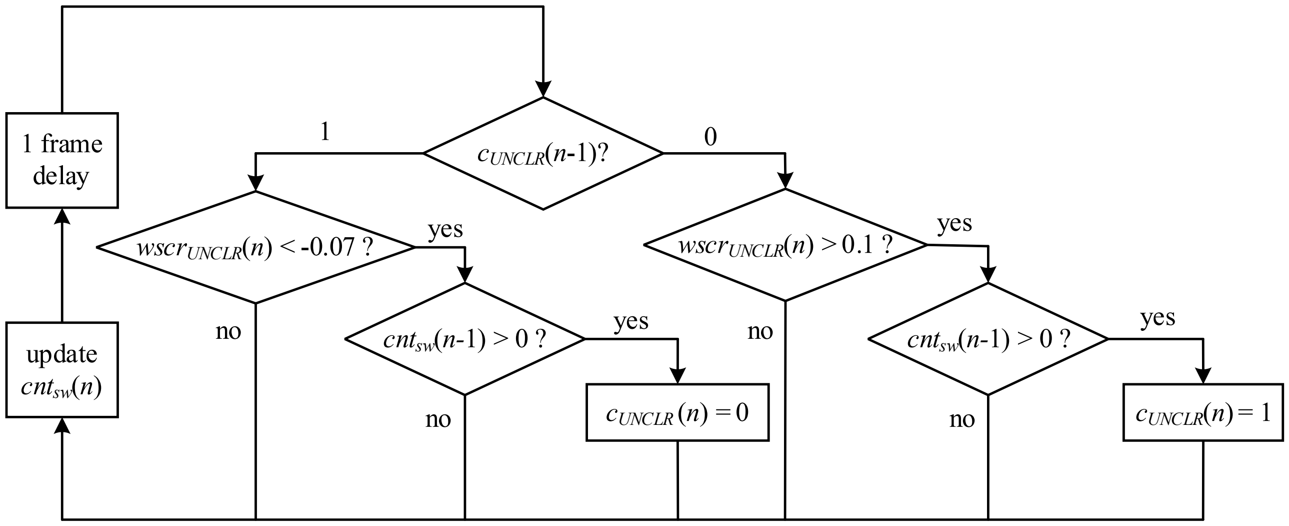


Figure 5.3-11: State-switching logic in the classifier of uncorrelated content

The counter $cnt_{sw}(n)$ indicates frames where it is possible to switch between the TD stereo mode and the DFT stereo mode. This counter is initialized to zero and updated in each frame with the following logic

$$cnt_{sw}(n) = \begin{cases} cnt_{sw}(n) + 1, & \text{if } c_{type} \in (\text{GENERIC, UNVOICED, INACTIVE}) \\ cnt_{sw}(n) + 1, & \text{if } c_{type} \notin (\text{GENERIC, UNVOICED, INACTIVE}) \wedge \text{VAD} = 0 \\ 0, & \text{otherwise} \end{cases} \quad (5.3\text{-}145)$$

where c_{type} is the frame type of the current frame in the encoder defined in eq. XX abd the VAD flag is defined in eq. XX. The counter $cnt_{sw}(n)$ is limited to the upper limit of 100.

5.3.3.2.4.2 Cross-talk detection using logistic regression

The cross-talk detector is based on the logistic regression statistical model trained individually for the TD stereo mode and for the DFT stereo mode. Both statistical models are trained on a large database of stereo cues collected by running the IVAS coder on real recordings as well as artificially-prepared stereo samples.

Cross-talk detection in the TD stereo mode is done similarly as the classification of uncorrelated content in the TD stereo mode described in clause 5.3.3.2.4.1. Due to the similarity between the classifiers the mathematical symbols from clause 5.3.3.2.4.1 will be re-used in the following text without specifically distinguishing them with the "XTALK" superscript.

The following features are used by the cross-talk detector in the TD stereo mode:

- L/R class difference, d_{clas}
- L/R difference of the maximum autocorrelation, d_v
- L/R difference of the average LSF, d_{LSF}
- L/R difference of the residual LP energy, d_{LPC13}
- L/R difference of the spectral correlation map, d_{cmap}
- L/R difference of noise characteristics, d_{nchar}
- L/R difference of the non-stationarity, d_{sta}
- L/R difference of the spectral diversity, d_{sdiv}
- un-normalized cross-channel correlation at lag 0, p_{LR}
- side-to-mono energy ratio, r_{SM}
- difference between the maximum and the minimum cross-channel correlation, d_{mmLR}
- zero-lag cross-channel correlation value, R_0
- evolution of the inter-channel cross-correlation function, RR
- position of the maximum inter-channel cross-correlation function, k_{max}
- maximum of the inter-channel correlation function, R_{max}
- difference between L/M and R/M correlation functions, Δ_{LRM}
- smoothed ratio of energies of the side signal and the mono signal, $\overline{r_{SM}}(n)$

In total, there are $F_{TD} = 17$ features used by the cross-talk detector in the TD stereo mode forming a feature vector $\mathbf{f}_{raw,TD}$. For the simplicity of notation the subscript "TD" will be dropped from the symbol names unless specifically stated.

Before the training process, the entire feature set is normalized by removing its global mean and scaling it to unit variance. This is done as follows

$$f_i = \frac{f_{i,raw} - \overline{f_i}}{\sigma_{f_i}}, \quad i = 1,\dots,F \qquad (5.3\text{-}146)$$

where $\overline{f_i}$ is the global mean of the *i*th feature across the training database and σ_{f_i} is the global variance of the *i*th feature across the training database. Please, note that the parameters $\overline{f_i}$ and σ_{f_i} used in the equation above have the same meaning but different value than in eq. (5.3-131).

The output of the LogReg model is calculated with eq. (5.3-132) and the probability that the current frame belongs to the cross-talk segment class (class 0) is given by eq. (5.3-133). The output of the LogReg model y_p is then normalized with the function shown in Fig. 5.3-12.

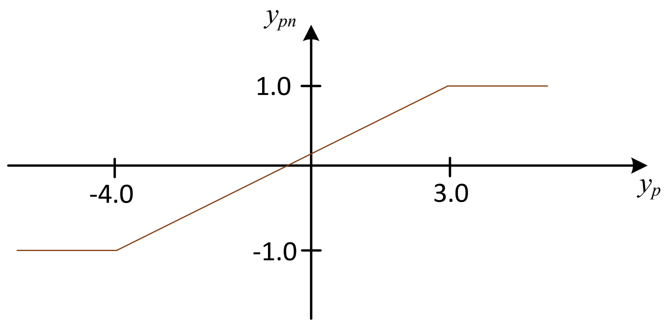


Figure 5.3-12: Normalization of the LogReg output of the cross-talk classifier in the TD stereo mode

The normalization can be mathematically described as follows

$$y_{pn}(n) = \begin{cases} 1.0 & \text{if } y_p(n) \geq 3.0 \\ 0.333 y_p(n) & \text{if -3.0} < y_p(n) < 3.0 \\ -1.0 & \text{if } y_p(n) \leq -3.0 \end{cases} \qquad (5.3\text{-}147)$$

The normalized output of the LogReg model $y_{pn}(n)$ is reset to 0 if the previous frame was encoded with the DFT stereo mode and the current frame is encoded with the TD stereo mode. This is done to prevent stereo mode switching artifacts.

The normalized output of the LogReg model $y_{pn}(n)$ is then weighted based on the relative frame energy $E_r(n)$ defined in eq. XX. The weighting scheme applied in the cross-talk classifier in TD stereo mode is similar to the weighting scheme applied in the classifier of uncorrelated content in the TD stereo mode described in clause 5.3.3.2.4.1. The main difference is that the relative frame energy $E_r(n)$ is not used as the multiplicative factor (5.3-143). Instead, the relative frame energy $E_r(n)$ is linearly mapped in the interval $\langle 0, \ldots, 0.95 \rangle$ with inverse proportion. That is

$$w_{relE}(n) = -2.375 E_r(n) + 2.1375 \qquad (5.3\text{-}148)$$

Thus, in frames with higher relative energy the weight is closer to 0 whereas in frames with low energy the weight is closer to the maximum value of 0.95. The weight $w_{relE}(n)$ is then used to filter the normalized output of the LogReg model $y_{pn}(n)$ as follows

$$scr_{XTALK}(n) = w_{relE} scr_{XTALK}(n-1) + (1 - w_{relE}) y_{pn}(n) \qquad (5.3\text{-}149)$$

The normalized weighted output of the cross-talk detector is called the “cross-talk score”.

Similarly as in the classification of uncorrelated content in the TD stereo mode it is necessary to filter out occasional short-term “peaks” and “dips” from $scr_{XTALK}(n)$ as they could lead to clasification errors or false alarms. The filtering is done in such a way that rising edges of the LogReg output are preserved. The rising edges represent important transitions between the cross-talk segments and single-talk segments. The mechanism for rising-edge detection in the cross-talk classifier in the TD stereo mode is different to the mechanism of rising-edge detection described in clause 5.3.3.2.4.1.

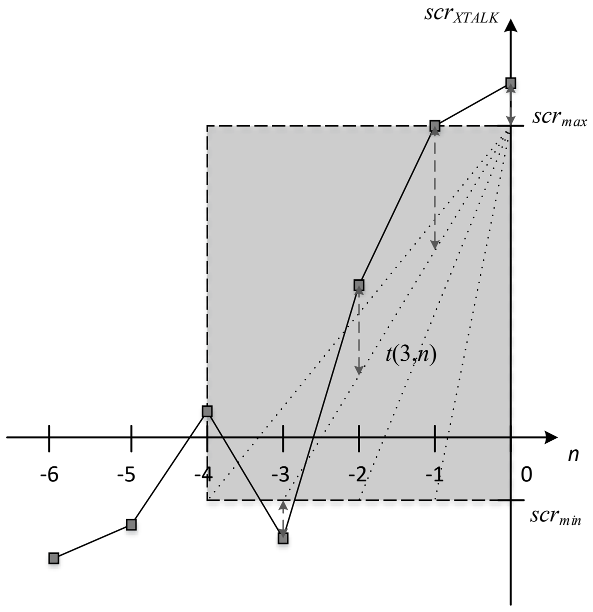


Figure 5.3-13: Rising-edge evaluation in the cross-talk detector in the TD stereo mode

The rising-edge detection algorithm analyzes the LogReg values from previous frames and compares them against a set of pre-calculated "ideal" edges with various slopes. The "ideal" edges are represented as linear functions of the frame index (n). The rising-edge detection mechanism is described by the schematic diagram in Fig. 5.3-15. The *x* axis has the frame indices of frames preceding the current frame 0. The small grey rectangles represent exemplary values of the cross-talk score $scr_{XTALK}(n)$ in the last six frames preceding the current frame. It is clear from the graph that there is a rising edge starting at frame (-3). The dotted lines illustrate a set of four "ideal" rising edges that $scr_{XTALK}(n)$ is compared to.

For each "ideal" rising edge the algorithm calculates the mean-square error between the dotted line and the cross-talk score $scr_{XTALK}(n)$. The output of the rising-edge detector is the minimum value among the mean-square errors. The coefficients of the linear functions represented by the dotted lines are pre-calculated based on the minimum value scr_{min} and the maximum value scr_{max}.

The rising-edge detection is performed only in frames meeting the following criterion:

$$\begin{gathered}\min_{k=0,..,K}(scr_{XTALK}(n-k)) < 0 \text{ AND} \\ \min_{k=0,..,K}(scr_{XTALK}(n-k)) < 0 \text{ AND} \\ \max_{k=0,..,\text{K}}(scr_{XTALK}(n-k)) - \min_{k=0,..,K}(scr_{XTALK}(n-k)) > 0.2\end{gathered} \quad (5.3\text{-}150)$$

where $K = 4$ is the maximum length of the tested edges.

Let the output value of the rising-edge detector be denoted ε_{0_1}. The "0_1" subscript underlines the fact that the output value of the rising-edge detector is limited in the interval $\langle 0,\dots,1\rangle$. For frames not meeting the criterion (5.3-35) the output value of the rising-edge detector is reset to 0, i.e. $\varepsilon_{0\_1} = 0$.

The set of linear functions representing the tested rising edges can be mathematically expressed with the following formula

$$t(l,n-k) = scr_{\max} - k\frac{scr_{\max}-scr_{\min}}{l}, \quad l = 1,\dots,K; k = 1,\dots,l \quad (5.3\text{-}151)$$

where the index l denotes the length of the tested edge and $n-k$ is the frame index. The slope of each linear function is determined by three parameters, the length of the tested edge l, the minimum threshold $scr_{\min}$ and the maximum threshold $scr_{\max}$. For the purpose of the cross-talk detector in the TD stereo mode the thresholds are set as follows

$$\begin{gathered}scr_{max} = 1.0 \\ scr_{\min} = -0.2\end{gathered} \quad (5.3\text{-}152)$$

The values of the minimum and the maximum threshold were found experimentally. For each length of the tested edge the algorithm calculates the mean-square error between the linear function t and the true cross-talk score $\mathrm{scr}_{\mathrm{XTALK}}$. That is

$$\varepsilon(l) = \frac{1}{l}\varepsilon_0 + \frac{1}{l}\sum_{k=1}^{l}[scr_{XTALK}(n-k) - t(\mathrm{l}, n-k)]^2, \quad l = 1, .., K \tag{5.3-153}$$

where ε_0 is the initial error given by

$$\varepsilon_0 = [scr_{XTALK}(n) - scr_{max}[]^2] \tag{5.3-154}$$

The minimum mean-square error is calculated by

$$\varepsilon_{\min} = \min_{l=1,..,K}(\varepsilon(l)) \tag{5.3-155}$$

The lower the minimum mean-square error the stronger the detected rising edge. If the minimum mean-square error is higher than 0.3 then the output value of the rising-edge detector is reset to 0. In all other cases the minimum mean-square error is mapped linearly in the interval $\langle 0, \dots, 1\rangle$. That is

$$\begin{aligned} \varepsilon_{0\_1} &= 0 && \text{if } \varepsilon_{min} > 0.3 \\ \varepsilon_{0\_1} &= 1 - 2.5\varepsilon_{\min} && \text{otherwise} \end{aligned} \tag{5.3-156}$$

Note, that the relationship between the output value of the rising-edge detector and the minimum mean-square error is inversely proportional.

The output value of the rising-edge detector is linearly mapped and limited to the interval of $\langle 0.5, \dots, 0.9\rangle$. The resulting value is denoted as the edge sharpness $f_{edge}(n)$. The edge sharpness is calculated as follows

$$f_{edge}(n) = 0.9 - 0.4\varepsilon_{0\_1} \tag{5.3-157}$$

Finally, the normalized weighted output of the LogReg model $scr_{XTALK}(n)$ is smoothed with an IIR filter where $f_{edge}(n)$ is used as the smoothing factor. The smoothing with the IIR filter is done as follows

$$wscr_{XTALK}(n) = f_{edge}(n) \cdot wscr_{XTALK}(n-1) + \left(1 - f_{edge}(n)\right) \cdot scr_{XTALK}(n) \tag{5.3-158}$$

The smoothed output $wscr_{XTALK}(n)$ is reset to 0 in frames where the alternative VAD flag $f_{xVAD}(n)$ from eq. XX is zero. That is

$$wscr_{XTALK}(n) = 0, \quad \text{if } f_{xVAD}(n) = 0 \tag{5.3-159}$$

Cross-talk detection in the DFT stereo mode is done similarly as cross-talk detection in the TD stereo mode which is described in eq. (5.3-146) - (5.3-159). The Logistic Regression (LogReg) model is used as the basis of binary classification of the input feature vector. Please note, that to avoid the duplication of symbols the notation used in eq. (5.3-146) - (5.3-159) will be reused in this section without specifically mentioning that it's related to the DFT stereo mode.

In the DFT stereo mode the feature vector $\mathbf{f}_{raw}$ of the cross-talk detector consists of the following stereo cues:

- ILD gain, g_{ILD}
- IPD gain, g_{IPD}
- IPD rotation angle, φ_{rot}
- prediction gain, g_{pred}
- mean energy of the inter-channel coherence, E_{coh}
- ratio of maximum and minimum intra-channel amplitude products, r_{PP}
- overall cross-channel spectral magnitude, f_X
- the maximum value of the GCC-PHAT function, G_{ITD}

- relationship between the amplitudes of the first and the second highest peak of the GCC-PHAT function, r_{GITD12}
- the amplitude of the second highest peak of the GCC-PHAT, m_{ITD2}
- the difference of the position of the second highest peak in the current frame with respect to the previous frame, Δ_{ITD2}

In total, there are $F = 11$ features used by the cross-talk detector in the DFT stereo mode.

The feature vector is normalized by removing its global mean and scaling it to unit variance. This is done as follows

$$f_i = \frac{f_{i,raw} - \bar{f}_i}{\sigma_{f_i}}, \quad i = 1, \dots, F \tag{5.3-160}$$

where $\bar{f}_i$ is the global mean of the *i*th feature across the training database and σ_{f_i} is the global variance of the *i*th feature across the training database. Please, note that the parameters $\bar{f}_i$ and σ_{f_i} used in the equation above have the same meaning but different value than in eq. (5.3-131).

The output of the LogReg model is described by eq. (5.3-132) and the probability that the current frame is part of cross-talk segment (class 0) is given by eq. (5.3-133). The raw output of the LogReg model y_p is normalized with the function shown in Fig. 5.3-10. The normalized output of the LogReg model is denoted y_{pn}. In the DFT stereo mode, no weighting based on relative frame energy is performed. Therefore, the normalized weighted output of the LogReg model (the cross-talk score) $scr_{XTALK}(n)$ is simply set as follows

$$scr_{XTALK}(n) = y_{pn} \tag{5.3-161}$$

The cross-talk score $scr_{XTALK}(n)$ is reset to 0 when the alternative VAD flag $f_{xVAD}(n)$ calcualted in eq. XX is set to 0. That is

$$scr_{XTALK}(n) = 0, \quad \text{if } f_{xVAD}(n) = 0 \tag{5.3-162}$$

Similarly as in the case of cross-talk detection the TD stereo mode it is necessary to smooth the cross-talk score $scr_{XTALK}(n)$ to remove short-term peaks. The smoothing is done by means of IIR filter using the rising-edge detection mechanism described in eq. (5.3-150) - (5.3-157). The cross-talk score $scr_{XTALK}(n)$ is smoothed with the IIR filter as follows

$$wscr_{XTALK}(n) = f_{edge}(n) \cdot wscr_{XTALK}(n-1) + \left(1 - f_{edge}(n)\right) \cdot scr_{XTALK}(n) \tag{5.3-163}$$

where $f_{edge}(n)$ is the edge sharpness calculated in eq. (5.3-157).

The classifier of uncorrelated content takes the smoothed score $wscr_{XTALK}(n)$ and makes a binary decision. Let $c_{UNCLR}(n)$ denote the binary output of the classifier with “1” representing the uncorrelated content and “0” representing the correlated content. The final output of the classifier is a state variable. It is initialized to 0. The classifier state is changed from the current class to the complementary class only in frames where certain conditions are met.

The cross-talk detector takes the smoothed cross-talk score $wscr_{XTALK}(n)$ and makes a binary decision. Let $c_{XTALK}(n)$ denote the binary output of the cross-talk detector with “1” representing the cross-talk class and “0” representing the single-talk class. The final output of the classifier is a state variable. It is initialized to 0. The classifier state is changed from the current class to the complementary class only in frames where certain conditions are met. The mechanism for cross-talk class switching is similar to the mechanism of class switching on uncorrelated content which has been described in detail in clause 5.3.3.2.4.1. However, there are some notable differences for both the TD stereo mode and the DFT stereo mode. These differences are discussed in more detail.

The state-switching mechanism in the cross-talk detector in the TD stereo mode is shown in fig. 5.3-15.

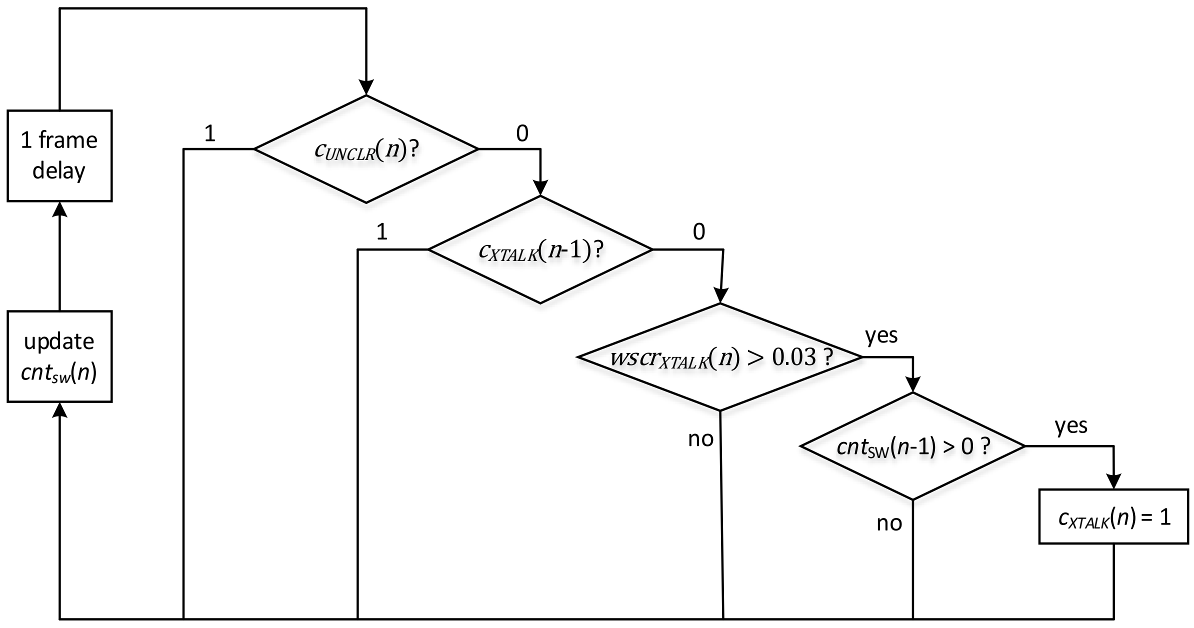


Figure 5.3-14: State-switching logic in the cross-talk detector in the TD stereo mode

The counter $cnt_{sw}(n)$ defined in eq. (5.3-145) is shared between the classifier of uncorrelated content and the cross-talk detector. A positive value of the counter $cnt_{sw}(n)$ indicates that switching of the binary output of the cross-talk detector $c_{XTALK}(n)$ is allowed. Note, that the state switching logic in the cross-talk detector uses the binary output of the uncorrelated classifier $c_{UNCLR}(n)$. Note, that the state-switching logic in fig. 5.3-15 is unidirectional in the sense that the output of the cross-talk detector $c_{XTALK}(n)$ can only be changed from “0” (single-talk) to “1” (cross-talk). The state-switching logic in the opposite direction, i.e. from “1” (cross-talk) to “0” (single-talk), is part of the DFT/TD stereo mode switching logic which is described in detail in clause XX.

The state-switching mechanism of the cross-talk detector in the DFT stereo mode is shown in fig. 5.3-15.

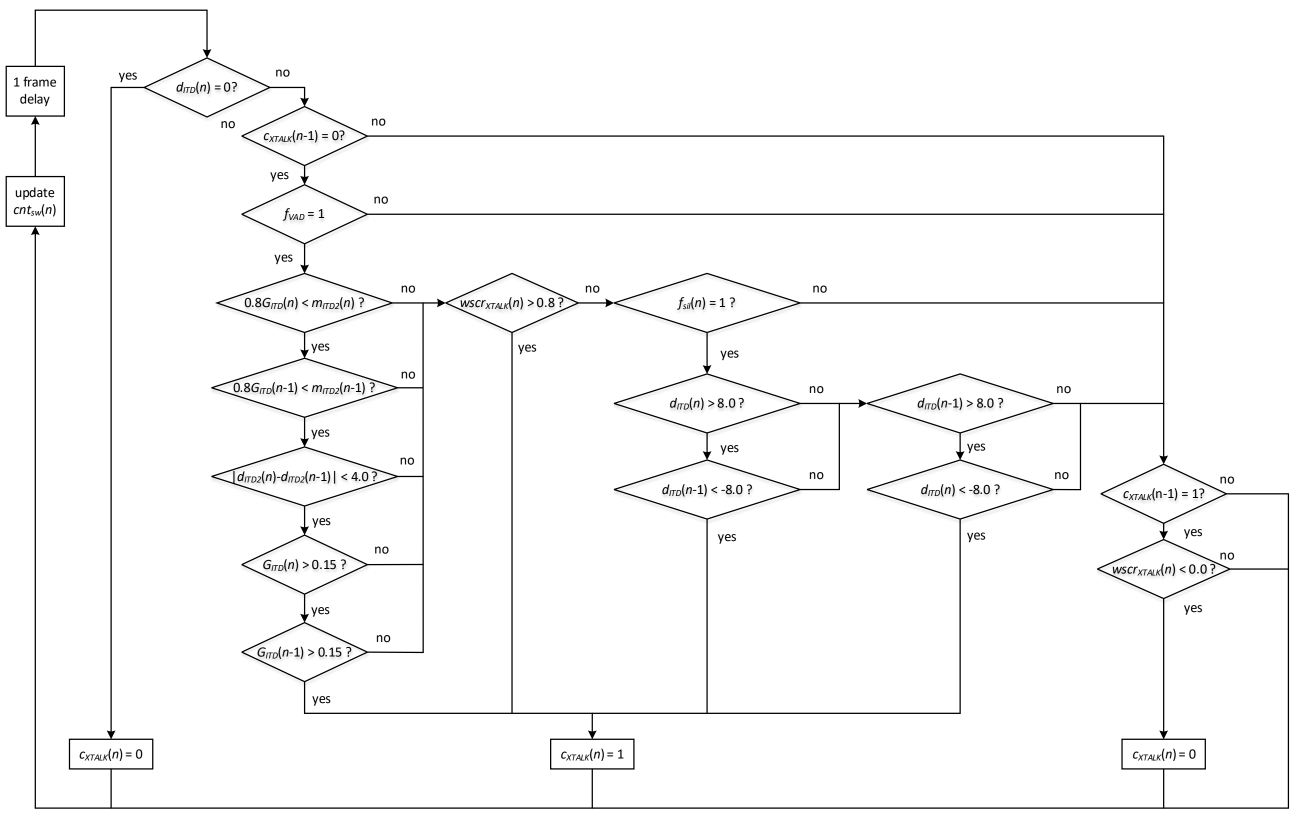


Figure 5.3-15: State-switching logic in the cross-talk detector in the DFT stereo mode

Similarly to the TD stereo mode, the counter $cnt_{sw}(n)$ is the counter of frames where it is possible to switch the binary output of the cross-talk detector. The counter $cnt_{sw}(n)$ is shared between the classifier of uncorrelated content and the cross-talk detector. The counter $cnt_{sw}(n)$ is initialized to zero and updated in each frame with eq. (5.3-145).

5.3.3.2.4.3 Cross-talk detection based on the GCC-PHAT

For the transmission of stereo speech signals captured with two microphones with a certain distance between them when low bitrate is required, DFT stereo as a parametric stereo technique described in 5.3.3.2.2 is more suitable.

For the case where we have two or more talkers around this microphone setup and more than one talker is talking simultaneously during the same time period a parametric stereo system performs adequately for most situations. However, there are some special cases where the parametric model fails to reproduce the stereo image and deliver intelligible speech output for interfering talker scenarios. That happens when each of the two talkers are captured with a different ITD (Inter-channel Time Difference), the ITD values are large (large distance between the microphones) and the talkers are sitting in opposite positions around the microphone setup axis.

On the other hand, in a parametric stereo scheme like DFT stereo described in clause 5.3.3.2.2 some parameters are extracted to reproduce the spatial stereo scene and the stereo signal is deduced to a single-channel downmix that is further coded. In the case of interfering talkers, the downmix signal would be coded with ACELP as described in 5.2.2.2.2.1. However, such coding schemes are source-filter models of speech production, designed to represent single talker speech. For interfering talkers, it may be that the core coding model is being violated and perceptual quality is degraded.

For these particular cases discrete coding of the two stereo channels is preferred for best performance and therefore the stereo classifier switches to the TD stereo coding mode where discrete coding of the two channels is possible as described in 5.3.3.2.1.

In this clause the critical case is described when the two talkers have different ITDs and the difference between the two ITDs is large.

The stereo classifier utilizes the existing ITD estimator described in 5.3.3.2.2.4 which is based on the GCC-PHAT (Generalized Cross-Correlation Phase Transform). The basic principle of such an estimator is to detect a peak in the GCC-PHAT and this peak corresponds to the ITD of the stereo signal. However, when two talkers are speaking at the same time and they have two different ITDs, there are typically two peaks in the GCC-PHAT as can be seen in an example in Figure 5.3-16 . The logic is then to detect whether there is only one peak which would imply remaining in the parametric coding mode of the DFT stereo or two peaks far from each other in the GCC-PHAT which would imply switching to the TD-stereo mode.

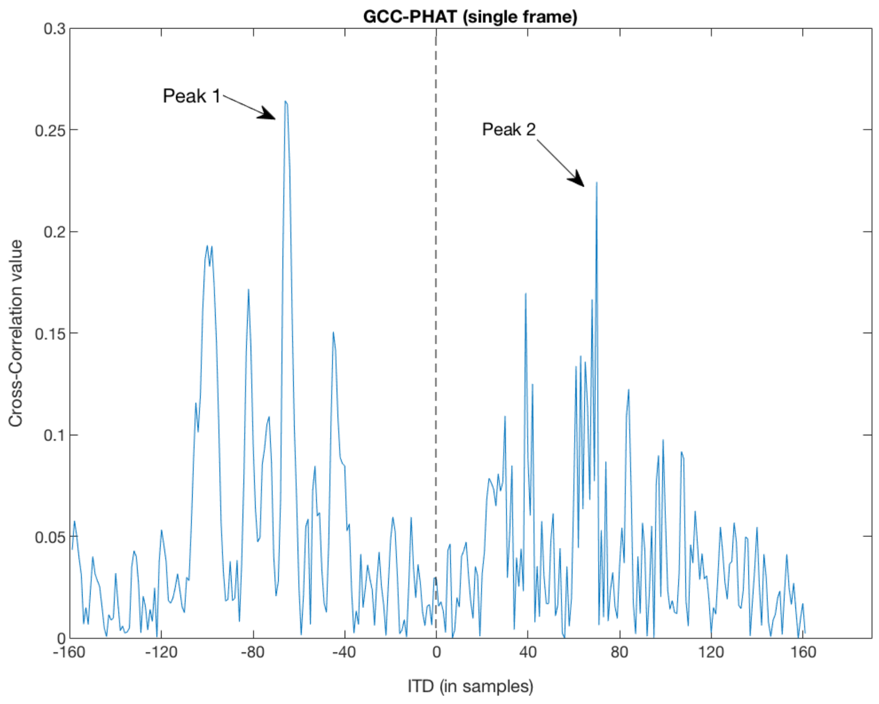


Figure 5.3-16: Example of the GCC-PHAT at a single frame with overlapping talkers

The algorithm is described in the following steps:

- Codec operates by default in DFT stereo mode
- Compute the GCC-PHAT of the stereo signal using a smoothed version of the cross-spectrum as described in 5.3.3.2.2.4 and quation (5.3-66)
- Estimate the main peak m_1 of the GCC-PHAT. This should correspond to the maximum of the absolute value of the GCC-PHAT. But it can also be different if some hysteresis is applied from the hangover mechanism described in XXX to have a more stable ITD estimation.
- Select a portion of the GCC-PHAT which is far from the main peak: the distance between the main peak and the border of the portion is above a certain threshold. The threshold is bitrate-dependent due to the fact that starting from 32 kbps, DFT stereo operates with residual coding, which to some extent cans still accommodate coding of cross-talk segments. The thresold is defined as follows in Table

Table 5.3-8: Threshold to enable switching per bitrate

Total bitrate	Threshold thr
32kbps	16
24.4 kbps	8

- Find a second peak in the selected portion: this is for example the maximum of the absolute value of the GCC-PHAT. The steps to find the second peak are described in the following pseudocode:

```
m1 = 0.0f;
m2 = 0.0f;

itd2 = 0;

 if ( itd > thr )
{
        m1 = fabsf( gcc_phat[itd + XTALK_PHAT_LEN] );
        m2 = fabsf( gcc_phat[0] );
        itd2 = -XTALK_PHAT_LEN;
        for ( i = 1; i < XTALK_PHAT_LEN - thr; i++ )
        {
            if ( fabsf( gcc_phat[i] ) > m2 )
            {
                itd2 = -XTALK_PHAT_LEN + i;
                m2 = fabsf( gcc_phat[i] );
            }
        }
    }
    else if ( itd < -thr )
    {
        m1 = fabsf( gcc_phat[itd + XTALK_PHAT_LEN] );
        m2 = fabsf( gcc_phat[XTALK_PHAT_LEN + thr + 1] );
        itd2 = thr + 1;
        for ( i = XTALK_PHAT_LEN + thr + 2; i < 2 * XTALK_PHAT_LEN + 1; i++ )
        {
            if ( fabsf( gcc_phat[i] ) > m2 )
            {
                itd2 = -XTALK_PHAT_LEN + i;
                m2 = fabsf( gcc_phat[i] );
            }
        }
    }
```

where XTALK_PHAT_LEN=200.
If the value of the second peak m_2 is higher of the threshold which is defined as the 80% of the value of the first peak m_1, i.e.:

$$m_2 > 0.8\, m_1 \tag{5.3-164}$$

then we can consider that the GCC-PHAT contains two significant peaks and we switch to mode TD stereo. Otherwise, there is no significant second peak, and we stay in DFT stereo.

Apart from the basic algorithm and the decision based on Equation XXX there are considerations to be made specific to the signal properties. The conditions that need to be additionally evaluated are summarized below:

Condition 1: check that $m_1 > 0.15$ to avoid switching on noisy frames, meaning the normalized cross-correlation is rather low and the detected peak may be rather random

Condition 2: both conditions of Equation XXX and Condition 1 have to be verified on two consecutive frames. This is to avoid switching on unstable signals. This translates to:

$$m_2 > 0.8 m_1 \text{ and } m_2^{[-1]} > 0.8 m_1^{[-1]} \text{ and } m_1 > 0.15 \text{ and } m_1^{[-1]} > 0.15 \tag{5.3-165}$$

Condition 3: itd_2 of two consecutive frames have to be close to each other and their difference has to be below 4 samples. This is to avoid switching frequently on unstable signals, but to switch if the peak is toggling around a certain ITD value .

$$\left| itd_2 - itd_2^{[-1]} \right| < 4 \tag{5.3-166}$$

Condition 4: the SAD flag of the previous frame has to be 1 (meaning it is an active signal). This is to avoid switching at an onset of a scpeech segment.

Condition 5: itd_1 changes abruptly from one frame to the next by a big difference. In that case, we don't need to check for a second peak, we consider that a second speaker started talking and we switch to TD stereo:

$$itd_1 > thr \text{ and } itd_1^{[-1]} < -thr \tag{5.3-167}$$

or

$$itd_1 > thr \text{ and } itd_1 < -thr \tag{5.3-168}$$

where thr is the threshold defined in Table 5.3-8

5.3.3.2.4.4 Stereo mode selection

The classifier of uncorrelated content and the cross-talk detector are essential parts of the DFT/TD stereo mode selection technology. The IVAS coder selects the appropriate stereo mode based on the binary outputs of both stereo classifiers. The stereo mode selection logic also takes into account some auxiliary parameters. These auxiliary parameters are used to prevent stereo mode switching in perceptually sensitive segments or to prevent frequent switching in segments where either the classifier of uncorrelated content or the cross-talk detector provide inaccurate outputs.

Stereo mode selection is done before down-mixing and encoding of the input stereo signal. Since both stereo classifiers are using parameters extracted from the IVAS encoder the stereo mode selector uses the outputs of the stereo classifiers from the previous frame. The stereo mode selection logic is described at the schematic diagram in fig. 5.3-15.

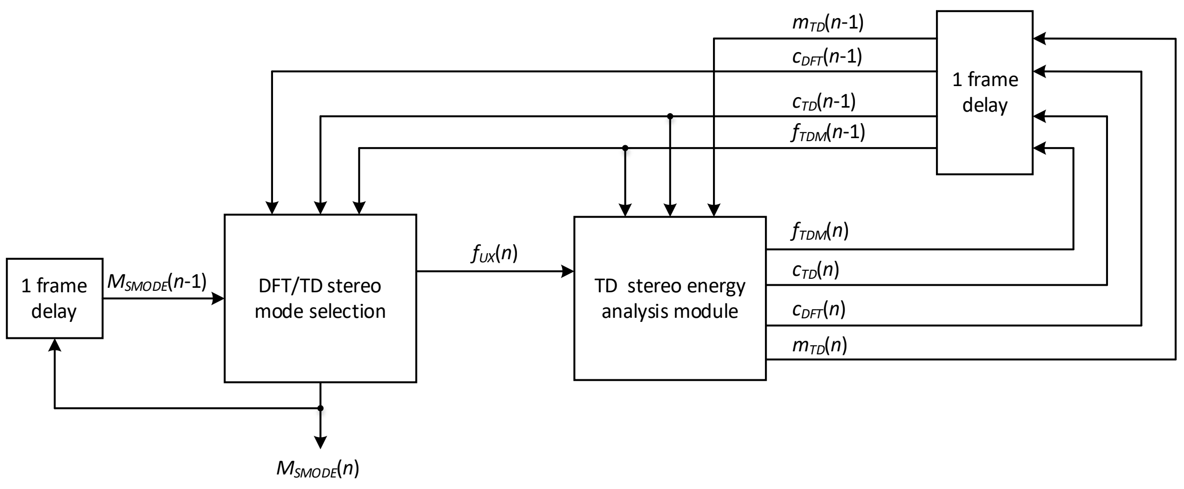


Figure 5.3-17: DFT/TD stereo mode selection

The DFT/TD stereo mode selection mechanism consists of the following functional blocks:

- Initial DFT/TD stereo mode selection
- TD to DFT stereo mode switching logic on cross-talk content

These functional blocks are described in detail in the following text.

The DFT stereo mode is the preferred mode for encoding single-talk speech with high inter-channel correlation between the left and the right channel of the input stereo signal. The initial selection of the stereo mode starts by checking whether the previous frame processed by the IVAS coder was "likely a speech frame". This is done by examining the log-likelihood ratio between the "speech" class and the "music" class of the speech/music classifier, described in clause XX. The log-likelihood ratio is the difference between the log-likelihood of "music", defined in eq. XX, and the log-likelihood of "speech", defined in eq. XX. That is

$$dL_{SM}(n) = L_M(n) - L_S(n) \qquad (5.3\text{-}169)$$

where $L_S(n)$ is the log-likelihood of the "speech" class and $L_M(n)$ the log-likelihood of the "music" class. The log-likelihood ratio is smoothed with two IIR filters each of which has its own forgetting factor. The IIR filtering is done as follows

$$\begin{aligned} wdL_{SM}^{(1)}(n) &= 0.97 \cdot wdL_{SM}^{(1)}(n-1) + 0.03 \cdot dL_{SM}(n-1) \\ dL_{SM}^{(2)}(n) &= 0.995 \cdot wdL_{SM}^{(2)}(n-1) + 0.005 \cdot dL_{SM}(n-1) \end{aligned} \qquad (5.3\text{-}170)$$

where the superscript (1) indicates the first IIR filter and the superscript (2) indicates the second IIR filter, respectively.

The smoothed outputs $wdL_{SM}^{(1)}(n)$ and $wdL_{SM}^{(2)}(n)$ are then compared with predefined thresholds and a binary flag $f_{SM}(n)$ is set to 1 if the following condition is met

$$f_{SM}(n) = \begin{cases} 1 & \text{if } wdL_{SM}^{(1)}(n) < 1.0 \text{ AND } wdL_{SM}^{(2)}(n) < 0.0 \\ 0 & \text{otherwise} \end{cases} \tag{5.3-171}$$

The binary flag $f_{SM}(n)$ is an indicator that the previous frame was likely a speech frame.

The initial stereo mode selection mechanism then sets anotherbinary flag $f_{UX}(n)$ to 1 if the binary output of the uncorrelated classifier $c_{UNCLR}(n-1)$ or the binary output of the cross-talk detector $c_{XTALK}(n-1)$ are set to 1 and if the previous frame was likely a speech frame. That is

$$f_{UX}(n) = \begin{cases} 1 & \text{if } f_{SM}(n) = 1 \text{ AND } (c_{UNCLR}(n-1) = 1 \text{ OR } c_{XTALK}(n-1) = 1) \\ 0 & \text{otherwise} \end{cases} \tag{5.3-172}$$

Let $M_{SMODE}(n) \in (\text{TD}=3, \text{DFT}=2)$ be a discrete variable denoting the selected stereo mode in the current frame. The numeric constants "TD" and "DFT" are used to denote the TD stereo mode and the DFT stereo mode, respectively. The stereo mode is initialized in each frame with the value from the previous frame, i.e.

$$M_{SMODE}(n) = M_{SMODE}(n-1) \tag{5.3-173}$$

where $M_{SMODE}(0)$ is initialized to XX. If the binary flag $f_{UX}(n)$ is set to 1, then the TD stereo mode is selected for the encoding of the input signal in the current frame. That is

$$M_{SMODE}(n) \leftarrow \text{TD if } f_{UX}(n) = 1 \tag{5.3-174}$$

If the binary flag $f_{UX}(n)$ is set to 0 and TD stereo mode was selected in the previous frame, then the auxiliary switching flag from the TD energy analysis module $f_{TDM}(n-1)$ (see clause XX) is analyzed to select the stereo mode in the current frame. This is done as follows

$$M_{SMODE}(n) \leftarrow \begin{cases} \text{TD} & \text{if } f_{SM}(n) = 1 \text{ AND } f_{TDM}(n-1) = 1 \\ \text{DFT} & \text{otherwise} \end{cases} \tag{5.3-175}$$

The parameter $f_{TDM}(n)$ is updated in every frame in the TD stereo mode only. The updating of this parameter is described in detail in clause XX.

If the binary flag $f_{UX}(n)$ is set to 0 and the stereo mode in the previous frame was the DFT sttereo mode, no stereo mode switching is done and the DFT stereo mode is selected in the current frame as well.

As can be seen from fig. xxx the binary output of the cross-talk detector in the TD stereo mode can only be set to 1 when cross-talk content is detected in the current frame. As a consequence, the initial stereo mode selection logic described above cannot select the DFT stereo mode when the cross-talk detector indicates single-talk content. An additional mechanism has been implemented for switching back from the TD stereo mode to the DFT stereo mode when single-talk content is detected. The mechanism is described in the following text.

If TD stereo mode has been selected in the previous frame and the initialization logic described above selected TD stereo mode in the current frame as well, then the stereo mode may be changed from the TD stereo mode to the DFT stereo mode. The change of stereo mode is conditioned on the binary output of the cross-talk detector $c_{XTALK}(n)$ as well as other auxiliary parameters. The stereo mode is changed if the following complex condition is fulfilled

$$M_{SMODE}(n) \leftarrow \text{DFT if } \begin{bmatrix} M_{SMODE}(n) = \text{TD AND} \\ M_{SMODE}(n-1) = \text{TD AND} \\ c_{XTALK}(n) = 1 \text{ AND} \\ f_{UX}(n-1) = 1 \text{ AND} \\ c_{TD}(n-1) > 15 \text{ AND} \\ c_{DFT}(n-1) > 3 \text{ AND} \\ clas(n-1) \in \begin{pmatrix} \text{UNVOICED\_CLAS} \\ \text{UNVOICED\_TRANSITION} \\ \text{VOICED\_TRANSITION} \end{pmatrix} \text{ AND} \\ (R_{total} \geq 16400 \text{ OR } wscr_{XTALK}(n-1) < 0.01) \end{bmatrix} \tag{5.3-176}$$

where R_{total} is the total bitrate of the IVAS encoder, defined in eq. XX and $clas(n)$ is the the frame type of the Frame Erasure Concealment (FEC) module of the primary channel in the TD stereo mode, defined in eq. XX. The parameters $c_{TD}(n)$ and $c_{DFT}(n)$ are the counters of frames in the TD stereo mode and the DFT stereo mode, respectively. The counters $c_{TD}(n)$ and $c_{DFT}(n)$ are updated in each frame as part of the TD energy analysis module. The updating of these counters is described in detail in the following text.

When the IVAS encoder runs in the TD stereo mode several auxiliary parameters are calculated as part of the TD energy analysis module. These auxiliary parameters are used to improve the stability of the DFT/TD stereo mode selection mechanism.

For certain special types of frames the TD stereo mode runs in s so-called "regular sub-mode". The regular sub-mode is usually applied for short transition periods before switching from the TD stereo mode to the DFT stereo mode. Whether or not the TD stereo mode is runing in the regular sub-mode is indicated by the binary flag $m_{TD}(n)$. The binary flag $m_{TD}(n)$ is first initialized in each frame as follows

$$m_{TD}(n) = f_{TDM}(n-1) \qquad (5.3\text{-}177)$$

where $f_{TDM}(n)$ is the auxiliary switching flag described in eq XX later in this clause. The binary flag $m_{TD}(n)$ is set to 0 or 1 in frames where the binary flag $f_{UX}(n) = 1$. The condition for setting $m_{TD}(n)$ is defined as follows

$$\mathrm{m_{TD}(n)} \leftarrow \begin{cases} 1 & \text{if } f_{\mathrm{TDM}}(\mathrm{n}-1) = 1 \text{ OR } M_{\mathrm{SMODE}}(\mathrm{n}-1) \neq \mathrm{TD} \text{ OR } c_{\mathrm{TD}}(\mathrm{n}-1) < 5 \\ 0 & \text{otherwise} \end{cases} \qquad (5.3\text{-}178)$$

If the binary flag $f_{UX}(n) = 0$ then the binary flag $m_{TD}(n)$ is not changed.

The TD energy analysis module contains two counters, $c_{TD}(n)$ and $c_{DFT}(n)$. The counter $c_{TD}(n)$ counts the number of consecutive frames when the IVAS coder runs in the TD stereo mode. Thus, the counter $c_{TD}(n)$ is set to 0 in frames where the IVAS coder runs in the DFT stereo mode. The counter $c_{TD}(n)$ is incremented by 1 in eeach frame where the IVAS coder runs in the TD stereo mode. That is

$$c_{TD}(n) = \begin{cases} c_{TD}(n-1) + 1 & \text{if } f_{UX}(n) = 1 \\ 0 & \text{otherwise} \end{cases} \qquad (5.3\text{-}179)$$

Essentially, $c_{TD}(n)$ contains the number of frames since the last DFT to TD stereo mode switching point. Note, that the counter $c_{TD}(n)$ is limited by the threshold of 100. The counter $c_{DFT}(n)$ counts the number of consecutive frames when the IVAS coder runs in the DFT stereo mode. Thus, the counter $c_{DFT}(n)$ is set to 0 in every frame where TD stereo mode has been selected in the encoder and is incremented by 1 in every frame where DFT stereo mode has been selected. That is

$$c_{DFT}(n) = \begin{cases} c_{DFT}(n-1) + 1 & \text{if } f_{TDM}(n) = 0 \\ 0 & \text{otherwise} \end{cases} \qquad (5.3\text{-}180)$$

Essentially, $c_{DFT}(n)$ contains the number of frames since the last TD to DFT stereo mode switching. Note, that the counter $c_{DFT}(n)$ is limited by the threshold of 100.

The last auxiliary parameter calculated in the TD energy analysis module is the auxiliary switching flag $f_{TDM}(n)$. This parameter is initialized with the binary flag $f_{UX}(n)$. That is

$$f_{TDM}(n) = f_{UX}(n) \qquad (5.3\text{-}181)$$

The auxiliary switching flag is set to 0 when the left and the right channel of the input stereo signal are out-of-phase (OOP). The details of OOP signal detection are described in detail in clause XX. In case of OOP it is desirable to invert the meaning of the primary and the secondary channel in the TD stereo mode. When OOP is detected the binary flag $s2m(n)$ in the current frame is set to 1, otherwise it is set to zero. The auxiliary switching flag in the TD stereo mode is set to zero when the parameter $s2m(n)$ is set to 1. That is

$$f_{TDM}(n) \leftarrow 0 \quad \text{if } s2m(n) = 1 \qquad (5.3\text{-}182)$$

If the parameter $s2m(n)$ is zero, then the auxiliary switching flag can be reset to zero based on the following set of conditions

$$f_{TDM}(n) \leftarrow 0 \ \text{ if } \begin{bmatrix} m_{TD}(n) = 1 \text{ AND} \\ c_{LRTD}(n) > 10 \text{ AND} \\ \begin{bmatrix} \text{VAD} = 0 \text{ OR} \\ c_{UNCLR}(n) = 0 \text{ AND } (scr_{XTALK}(n) < -0.8 \text{ OR } wscr_{XTALK}(n) < -0.13) \text{ OR} \\ c_{UNCLR}(n) = 1 \text{ AND } clas(n-1) = \text{UNVOICED\_CLAS AND } |wscr_{UNCLR}(n)| < 0.005 \end{bmatrix} \end{bmatrix} \qquad (5.3\text{-}183)$$

Furthermore, the auxiliary switching flag $f_{TDM}(n)$ can also be reset to 0 based on the following set of conditions

$$f_{TDM}(n) \leftarrow 0 \text{ if } \begin{bmatrix} m_{TD}(n) = 0 \text{ AND} \\ \begin{bmatrix} \text{VAD} = 0 \text{ OR} \\ c_{UNCLR}(n) = 0 \text{ AND } (scr_{XTALK}(n) \leq 0 \text{ OR } wscr_{XTALK}(n) \leq 0.1) \text{ OR} \\ c_{UNCLR}(n) = 1 \text{ AND } clas(n-1) = \text{UNVOICED\_CLAS AND } |wscr_{UNCLR}(n)| < 0.025 \end{bmatrix} \end{bmatrix} \quad (5.3\text{-}184)$$

Note, that in the two sets of conditions defined above the condition

$$clas(n-1) = \text{UNVOICED\_CLAS}$$

refers to the $clas$ parameter calculated in the preprocessing module of the primary channel when the codec is in the DFT stereo mode. In case the codec is in the TD stereo mode, the condition is replaced with

$$clas_P(n-1) = \text{UNVOICED\_CLAS AND } clas_S(n-1) = \text{UNVOICED\_CLAS}$$

where the subscripts “P” and “S” have been added as a reference to the primary and the secondary channel in the TD stereo mode, respectively.

5.3.3.3 MDCT-based stereo

5.3.3.3.1 General Overview

For bitrates starting from 48kbps up to 256kbp, the MDCT-based stereo coding mode is used. MDCT stereo is based on the EVS high-rate MDCT-based TCX coder, described in Clause 5.3.3 of [3]. This stereo mode offers discrete coding due to the higher bitrate compared to the unified stereo coding mode where the coding is parametric.

In Figure 5.3-18 a block diagram of the MDCT-based encoder is depicted. The input audio signal of each stereo channel is first transformed to the MCLT domain from which the MDCT-transform is further pre-processed to be perceptually flat (whitened) before any stereo processing is applied. The details of this pre-processing are described in 5.3.3.3.3. On the perceptually whitened signals the stereo processing is applied. The details of the stereo processing are described in clause 5.3.3.3.4. Finally, stereo IGF encoding is applied, and the obtained signals of each channel are quantized and encoded as described in clause 5.3.3.3.8.

5.3.3.3.2 ITD compensation

For the mid bitrates 48kbps and 64kbps the incoming signals are time aligned with respect to the inter-channel time difference (ITD) to achieve optimal energy compaction and larger coding gain. The time alignment and the ITD estimation are described in detail in clause xxx.

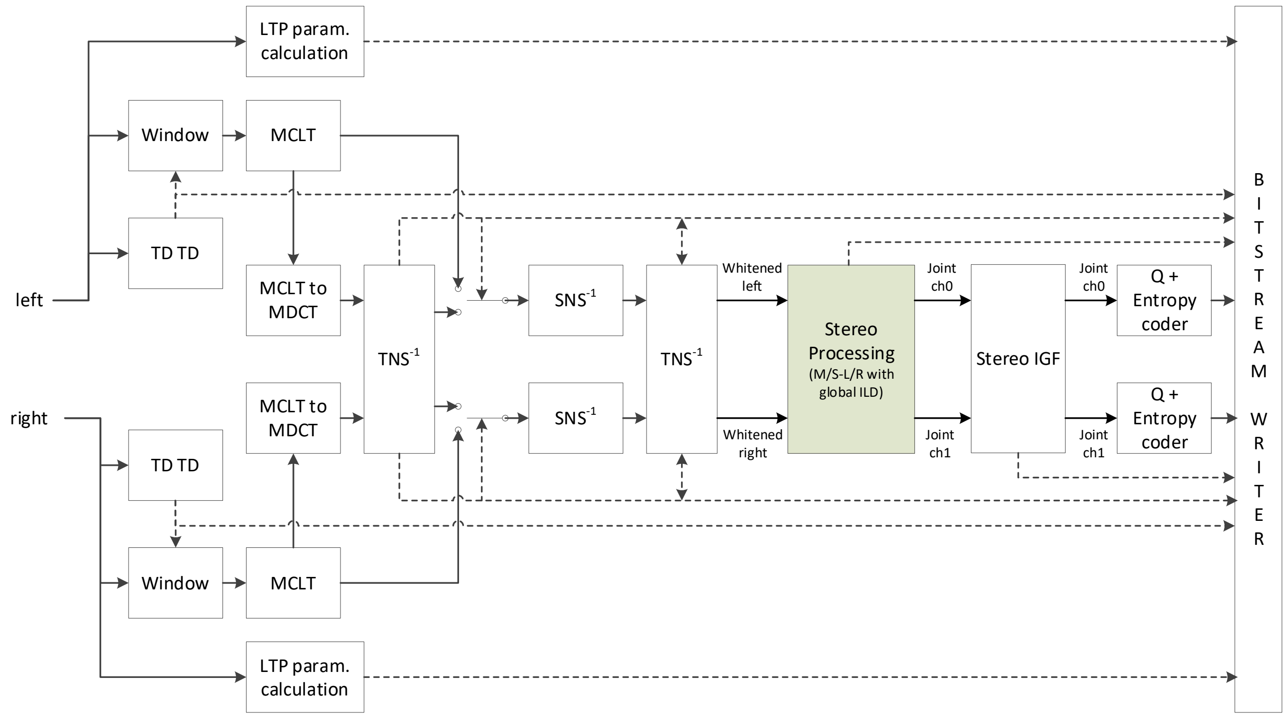


Figure 5.3-18: Block diagram of the MDCT-based stereo encoder

5.3.3.3.3 MDCT core pre-processing

5.3.3.3.3.1 General

The pre-processing of the input time-domain audio signal prior to any stereo processing comprises of the transform from time-domain to frequency-domain, more precisely to the MDCT-domain, and further processing of the first channel and second channel by applying Spectral Noise Shaping (SNS) operation on the transformed audio signal of each channel. SNS on the encoder side in fact performing flattening of the spectrum. We may refer to SNS also as Frequency-Domain Noise Shaping. Furthermore, depending on the input signal characteristics, temporal-noise shaping may be applied either prior or after SNS on the transformed signal.

5.3.3.3.3.2 TCX-LTP parameter estimation

The LTP filter parameters (pitch lag and gain) are first estimated and encoded into the bitstream such that the TCX LTP postfilter at the decoder side described in clause 6.9.2.2 of [3] can use them. The detailed description of the LTP filter parameter estimation can be found in clause 5.1.14.1.1.1 of [3].

5.3.3.3.3.3 Time-to-frequency transformations

The time-domain input audio signals of the first channel and second channel are then transformed to the frequency-domain by performing an MDCT transform. The MDST transform is also performed, which is needed to compute the power spectra used for SNS and IGF.

The time-to-frequency domain transform is the same as for the EVS TCX high rate MDCT-based coding mode. The details regarding the windowing, long and short block transforms, transient-depended overlap and transform length, transient detection, window transitions and the MDST transform can be found in clauses 5.3.2.1, 5.3.2.2, 5.3.2.3, 5.3.2.4.2 and 5.3.2.6 of [3]. Different to clause 5.3.2.5 of [3], transition between long and short windows is implemented as described in clause 5.3.3.3.3.3.1.

5.3.3.3.3.3.1 Transition between long and short blocks with double overlap

5.3.3.3.3.4 Mid-high bitrate pre-processing

5.3.3.3.3.4.1 General

For the mid-high stereo bitrates, namely 48 and 64kbps, some additional pre-processing steps are applied in order to improve energy compaction of the stereo processing and increase perceptual coding gain. The first step is kernel switching, allowing to switch between different MDCT and MDST based transform types depending on the frame-wise overall phase difference between the two input channel signals. The second step is stereo pre-processing, where the channel signals are phase aligned in the higher bands depending on frame-wise inter-channel correlation statistics. A third step, where the channel signals are additionally level fine-aligned in the higher bands after stereo pre-processing, is applied as part of the global inter-channel level difference (ILD) normalization during the stereo processing.

In the following, the term "mid-high stereo" shall indicate the operating mode defined in the previous paragraph, i.e., MDCT-based two-channel stereo coding at 48 and 64kbps.

5.3.3.3.3.4.2 Kernel switching

5.3.3.3.3.4.3 Stereo pre-processing

5.3.3.3.3.5 Temporal noise shaping

5.3.3.3.3.5.1 General overview of algorithm

Temporal Noise Shaping (TNS) is used to control the temporal shape of the quantization noise within each window of the transform. For each channel, if TNS is set to be active, up to two filters per MDCT-spectrum will be applied. TNS is always calculated on a per sub-window basis, so in case of a 4 TCX5 window sequence the filtering steps must be applied once for each of the 4 sub-windows. The algorithm for the TNS detection, filtering and coding of the parameters are described in detail in clause 5.3.3.2.2 of [3].

For the stereo audio input case there are some stereo specific considerations that apply. The first aspect concerns the case where the TNS algorithm applies similar filters to both channels. In that case, a common filter is applied in order to save on bitrate since only one set of parameters would need to be encoded. Furthermore, it has been found that in certain cases it is favorable, in terms of audio quality performance, to apply the temporal noise shaping prior to the spectral noise shaping (SNS) and for other cases it is better to apply it afterwards. The stereo-specific aspects of the TNS analysis are described in detail in the clause below.

5.3.3.3.3.5.2 TNS for stereo

At first the TNS analysis is applied to each channel individually to determine the filter configuration per channel. The TNS filtering can be synchronized between the channels only when the spectral resolution is the same, that means that the transform type must be the same for both channels. Specifically for the short blocks, the transform type for each subframe should be also in sync. If that is not the case than the TNS decision is applied for each channel separately as done in EVS and described in clause 5.3.3.2.2 of [3].

If the transform types are in sync for both channels than the similarity of the filters are examined and if they are similar than the TNS decision is forced to be the same for both channels. Depending on the similarity it may be that the same filters are used to save on TNS parameters that need to be transmitted to the decoder.

At first the mean prediction gain $\tilde{\eta}_{tns}$ is calculated as in:

$$\tilde{\eta}_{tns} = \frac{\eta_{tns_1} + \eta_{tns_2}}{2}$$

where η_{tns_1} and η_{tns_2} are the TNS prediction gains for the first and the second channel respectively.

For the mid-high bitrates of the MDCT-based stereo, namely 48 and 64 kbps the prediction gains n_{tns_1} and n_{tns_2} are set to the mean prediction gain $\tilde{\eta}_{tns}$, if the individual gains are larger than a minimum threshold $n_{tns_1} > \eta_{min}$ and $n_{tns_2} > \eta_{min}$ and if the number of TNS filters detected for both channels are the same.

To determine whether the TNS decision can be synced, given that the number of filters detected are the same, the difference of the prediction gains to a specific threshold are examined as follows:

$$\left|\eta_{\text{tns}_1} - \eta_{\text{tns}_2}\right| < 0.04 \cdot \tilde{\eta}_{tns}$$

Then if one of the conditions described in clause 5.3.3.2.2.1 of [3] for one of the channels is met then both channels are synced to enable TNS filtering, otherwise they are both kept to the same state as for the previous frame - keep it turned off or on.

Furthermore, for the mid-high bitrates 48 and 64kbps where TNS is detected to be applied, the filters applied are synchronized to save on bitrate for transmitting the TNS parameters. This is described in the steps below:

- Determine the maximum absolute difference between the respective filter coefficients of the two channels.
- If the maximum absolute difference equals 1, then the TNS coefficients are sufficiently similar to equalize the two filters.
- The common filter coefficients to be used for each tap by both channels are the coefficients with the smallest absolute value between the two channels.

If the above conditions do not apply, then the TNS detection and decision is applied individually for each channel as done in EVS [3] and described in clause 5.3.3.2.2.1.

Additionally, if the maximum prediction gain of both channels is lower than a threshold of 3 and the MDCT transform type is either TCX20 or TCX10, then the steps described above are repeated on the frequency-domain signal after the spectral noise shaping (whitened signal). In this case, all filters are reset to zero and TNS is turned off and is not applied on the MDCT spectrum of the original input signal.

Finally, after the procedure of determining whether TNS will be applied (before or after the whitening of the spectrum) and the filters to be applied, the actual TNS filtering occurs either on the MDCT spectrum of the original audio input or the whitened MDCT spectrum after the SNS. The TNS filtering is described in detail in clause 5.3.3.2.2.2 of [3].

5.3.3.3.3.6 Spectral Noise Shaping (SNS)

Spectral Noise Shaping (SNS) applies a set of scale parameters as factors to the signal after being transformed to a spectral representation (MDCT transform, see clause 5.3.3.3.3.3). These scale parameters shape the quantization noise introduced in the frequency domain by the spectral quantization. The noise shaping is performed in such a way that the quantization noise is minimally perceived by the human ear, maximizing the perceptual quality of the decoded output.

The SNS encoder consists of a scale parameter calculator, a downsampler and a scale parameter encoder. After encoding the SNS scale parameters, a signal processor interpolates the downsampled scale parameters to obtain scale factors and process the signal spectrum by applying these factors to both channels of the signal in the frequency domain.

5.3.3.3.3.6.1 SNS Frequency bands

The SNS scale parameters are calculated and applied in 64 frequency bands which approximately follow the Bark scale. These psychoacoustic bands are used in every frame except in transition frames when the previous frame was coded using ACELP. For such frames, equally sized bands are used and the length of each band is calculated by dividing the length of the spectrum by the number of bands (64). The bands are defined in Table 5.3-9with b being the band index, L_b the length of the band in frequency bins, $i_{low,b}$ indicating the first index of the band and $i_{high,b}$ indicating the last index of the band.

Table 5.3-9: Definition of frequency bands for SNS

	Core Fs = 16kHz						Core Fs = 25.6kHz						Core Fs = 32kHz					
	TCX20			TCX10			TCX20			TCX10			TCX20			TCX10		
b	L_b	$l_{low,b}$	$l_{high,b}$	L_b	$l_{low,b}$	$l_{high,b}$	L_b	$l_{low,b}$	$l_{high,b}$	L_b	$l_{low,b}$	$l_{high,b}$	L_b	$l_{low,b}$	$l_{high,b}$	L_b	$l_{low,b}$	$l_{high,b}$
0	2	0	1	1	0	0	2	0	1	1	0	0	2	0	1	1	0	0
1	2	2	3	1	1	1	2	2	3	1	1	1	2	2	3	1	1	1
2	2	4	5	1	2	2	2	4	5	1	2	2	2	4	5	1	2	2
3	2	6	7	1	3	3	2	6	7	1	3	3	2	6	7	1	3	3
4	2	8	9	1	4	4	2	8	9	1	4	4	2	8	9	1	4	4
5	2	10	11	1	5	5	2	10	11	1	5	5	2	10	11	1	5	5
6	2	12	13	1	6	6	2	12	13	1	6	6	2	12	13	1	6	6
7	2	14	15	1	7	7	2	14	15	1	7	7	2	14	15	1	7	7
8	2	16	17	1	8	8	2	16	17	1	8	8	2	16	17	1	8	8
9	2	18	19	1	9	9	2	18	19	1	9	9	2	18	19	1	9	9
10	2	20	21	1	10	10	2	20	21	2	10	11	2	20	21	2	10	11
11	2	22	23	1	11	11	4	22	25	2	12	13	4	22	25	2	12	13
12	2	24	25	1	12	12	4	26	29	2	14	15	4	26	29	2	14	15
13	2	26	27	1	13	13	4	30	33	2	16	17	4	30	33	2	16	17
14	2	28	29	1	14	14	4	34	37	2	18	19	4	34	37	2	18	19
15	2	30	31	1	15	15	4	38	41	2	20	21	4	38	41	2	20	21
16	4	32	35	2	16	17	4	42	45	2	22	23	4	42	45	2	22	23
17	4	36	39	2	18	19	4	46	49	2	24	25	4	46	49	2	24	25
18	4	40	43	2	20	21	4	50	53	3	26	28	4	50	53	3	26	28
19	4	44	47	2	22	23	4	54	57	3	29	31	4	54	57	3	29	31
20	4	48	51	2	24	25	4	58	61	3	32	34	4	58	61	3	32	34
21	4	52	55	2	26	27	4	62	65	3	35	37	4	62	65	3	35	37
22	4	56	59	2	28	29	4	66	69	3	38	40	4	66	69	3	38	40
23	4	60	63	2	30	31	4	70	73	3	41	43	4	70	73	3	41	43
24	4	64	67	2	32	33	4	74	77	3	44	46	4	74	77	4	44	47
25	4	68	71	2	34	35	4	78	81	3	47	49	4	78	81	4	48	51
26	4	72	75	2	36	37	4	82	85	4	50	53	4	82	85	4	52	55
27	4	76	79	2	38	39	6	86	91	4	54	57	6	86	91	4	56	59
28	4	80	83	2	40	41	6	92	97	4	58	61	6	92	97	4	60	63
29	4	84	87	2	42	43	6	98	103	4	62	65	6	98	103	4	64	67
30	6	88	93	2	44	45	6	104	109	4	66	69	6	104	109	5	68	72
31	6	94	99	2	46	47	6	110	115	4	70	73	6	110	115	5	73	77
32	6	100	105	3	48	50	6	116	121	4	74	77	6	116	121	5	78	82
33	6	106	111	3	51	53	6	122	127	4	78	81	8	122	129	5	83	87
34	6	112	117	3	54	56	6	128	133	4	82	85	8	130	137	5	88	92
35	6	118	123	3	57	59	8	134	141	4	86	89	8	138	145	5	93	97
36	6	124	129	3	60	62	8	142	149	5	90	94	8	146	153	6	98	103
37	6	130	135	3	63	65	8	150	157	5	95	99	8	154	161	6	104	109
38	6	136	141	3	66	68	8	158	165	5	100	104	10	162	171	6	110	115
39	6	142	147	3	69	71	8	166	173	5	105	109	10	172	181	6	116	121
40	6	148	153	3	72	74	8	174	181	5	110	114	10	182	191	6	122	127
41	6	154	159	3	75	77	8	182	189	5	115	119	10	192	201	6	128	133
42	6	160	165	3	78	80	8	190	197	5	120	124	12	202	213	7	134	140
43	6	166	171	3	81	83	10	198	207	5	125	129	12	214	225	7	141	147
44	6	172	177	3	84	86	12	208	219	5	130	134	12	226	237	7	148	154
45	6	178	183	3	87	89	12	220	231	5	135	139	12	238	249	7	155	161
46	6	184	189	3	90	92	12	232	243	6	140	145	14	250	263	7	162	168
47	6	190	195	3	93	95	12	244	255	6	146	151	14	264	277	7	169	175
48	6	196	201	4	96	99	12	256	267	6	152	157	14	278	291	8	176	183
49	6	202	207	4	100	103	12	268	279	6	158	163	16	292	307	8	184	191
50	8	208	215	4	104	107	14	280	293	6	164	169	18	308	325	8	192	199
51	8	216	223	4	108	111	14	294	307	6	170	175	18	326	343	8	200	207
52	8	224	231	4	112	115	14	308	321	6	176	181	18	344	361	8	208	215
53	8	232	239	4	116	119	14	322	335	6	182	187	20	362	381	8	216	223
54	8	240	247	4	120	123	14	336	349	6	188	193	22	382	403	9	224	232
55	8	248	255	4	124	127	16	350	365	6	194	199	22	404	425	9	233	241
56	8	256	263	4	128	131	16	366	381	7	200	206	22	426	447	9	242	250
57	8	264	271	4	132	135	18	382	399	7	207	213	24	448	471	9	251	259
58	8	272	279	4	136	139	18	400	417	7	214	220	26	472	497	10	260	269
59	8	280	287	4	140	143	18	418	435	7	221	227	26	498	523	10	270	279
60	8	288	295	4	144	147	18	436	453	7	228	234	26	524	549	10	280	289
61	8	296	303	4	148	151	18	454	471	7	235	241	28	550	577	10	290	299

62	8	304	311	4	152	155	20	472	491	7	242	248	30	578	607	10	300	309
63	8	312	319	4	156	159	20	492	511	7	249	255	32	608	639	10	310	319

5.3.3.3.3.6.2 SNS scale factor calculation

The scale parameter calculator first calculates the energy in each of the non-uniform bands as given in Table 5.3-9The energy is calculated as the squared amplitude of the complex signal spectrum. To calculate the complex amplitude of the signal spectrum, both the MDCT and the MDST spectral values are needed. If TNS has already been applied on the MDCT spectrum, the signal energy is only estimated from the MDCT spectrum as the TNS processing is not applied on the MDST spectrum. The energy per band is normalized by dividing it by the respective band length.

$$E(b) = \begin{cases} \sum_{i=i_{low,b}}^{i_{high,b}} \frac{(X^{MDCT}(i))^2}{L_b}, if\ TNS\ has\ been\ applied \\ \sum_{i=i_{low,b}}^{i_{high,b}} \frac{(X^{MDCT}(i))^2+(X^{MDST}(i))^2}{L_b}, otherwise \end{cases}, b = 0, \ldots, 63. \quad (5.3\text{-}185)$$

The energy per band is smoothed according to

$$E_s(b) = \begin{cases} 0.75\,E(0) + 0.25\,E(1), if\ b = 0 \\ 0.75\,E(63) + 0.25\,E(62), if\ b = 63 \\ 0.25\,E(b-1) + 0.5\,E(b) + 0.25\,E(b+1), otherwise \end{cases}, b = 0, \ldots, 63. \quad (5.3\text{-}186)$$

A pre-emphasis operation is performed on the smoothed energy values so that low frequency amplitudes are emphasized with respect to high frequency amplitudes:

$$E_p(b) = E_s(b) \cdot 10^{\frac{b \cdot g_{tilt}}{(63/10)}}, b = 0, \ldots, 63 \quad (5.3\text{-}187)$$

where g_{tilt} controls the pre-emphasis tilt and depends on the core sampling rate as specified in Table 5.3-10.

Table 5.3-10: Pre-emphasis tilt values for SNS parameter calculation

	Fs = 16kHz	**Fs = 25.6kHz**	**Fs = 32kHz**
g_{tilt}	19	22	23.5

Next, a noise floor at -40dB is added to the pre-emphasized energy values:

$$E_n(b) = max\left(E_p(b), nf\right), b = 0,1, \ldots, 63, where\ nf = max\left(\frac{\sum_{b=0}^{63} E_p(b)}{63} \cdot 10^{-4}, 2^{-32}\right). \quad (5.3\text{-}188)$$

The energy values are then transformed into a logarithmic domain according to:

$$E_L(b) = \frac{\log_2(E_n(b))}{2}. \quad (5.3\text{-}189)$$

After this first set of scale parameters in the logarithmic domain is computed, the downsampler calculates a second set of scale parameters by downsampling the first set.

The vector $E_L(b)$ is downsampled by a factor of 4 according to:

$$E_4(b) = \begin{cases} w(0)E_L(0) + \sum_{k=1}^{5} w(k)E_L(4b+k-1), if\ b = 0 \\ \sum_{k=0}^{4} w(k)E_L(4b+k-1) + w(5)E_L(63), if\ b = 15,\ b = 0, \ldots, 15 \\ \sum_{k=0}^{5} w(k)E_L(4b+k-1), otherwise \end{cases} \quad (5.3\text{-}190)$$

With

$$w(k) = \left\{\frac{1}{12}, \frac{2}{12}, \frac{3}{12}, \frac{3}{12}, \frac{2}{12}, \frac{1}{12}\right\} \quad (5.3\text{-}191)$$

This step applies a low pass filter before decimation which is equivalent to a weighted average operation performed on the values of the parameter vector weighting closer scale parameters stronger than more distant ones. Subsequently, a mean removal is performed so that the second set of scale parameters calculated by the downsampler is mean free and the parameter vector is scaled by a constant:

$$scf(n) = 0.85\left(E_4(n) - \frac{\sum_{b=0}^{15} E_4(b)}{16}\right), n = 0, \dots, 15 \qquad (5.3\text{-}192)$$

The resulting set of scale parameters is then passed to the scale parameter encoder that quantizes the set using one of two different vector quantizers depending on the operating point.

5.3.3.3.3.6.3 SNS scale factor quantization

The quantization mechanism for the SNS scale parameters depends on the codec bitrate and the sampling rate at which the core coder operates. For mid bitrates at a core sampling rate above 16kHz, a multi-stage stochastic VQ with a variable number of stages is used. At higher bitrates, a 2-stage vector quantizer structure is used consisting of a stochastic split VQ and an Algebraic Vector Quantizer (AVQ). Both SNS quantization modes employ a signal-adaptive joint coding mechanism. Based on a similarity measure between the channels, either joint (mid/side) or separate coding of the scale parameters is used. In the joint case, the side representation of the SNS scale parameters is treated differently than the other representations by quantizing it with a reduced number of stages compared to the number of stages used for mid, left and right representations of the SNS scale parameters. Also, dedicated signaling is used to communicate to the decoder if the side representation is very close to a zero vector, in which case it is encoded in the bitstream by just the single signaling bit.

Selection of the quantizer type is done according to Table 5.3-11 below.

Table 5.3-11: Assignment of SNS VQ types to operating points

Core F_s	<=64kbps	>80kbps
16 kHz	Split/AVQ	Split/AVQ
25.6 kHz	MSVQ	Split/AVQ
32 kHz	N/A	Split/AVQ

5.3.3.3.3.6.3.1 Determining the encoding mode

Using the joint encoding mode is only possible if both channels of a frame use the same transformation block length. If the transformation block length differs between the channels, the separate encoding mode is used. If both channels use short blocks, joint encoding is only possible at operating points where the MSVQ is used.

To determine the encoding mode, a similarity measure E is calculated as the squared difference between the SNS scale parameters for the left and the SNS scale parameters of the right channel:

$$E = \sum_{i=0}^{15}\left(SNS_{left}(i) - SNS_{right}(i)\right)^2. \qquad (5.3\text{-}193)$$

E is then compared to a threshold value $\varepsilon = 12$. If E is bigger than ε, the separate encoding mode is chosen, otherwise the joint encoding mode is chosen. In the joint encoding mode, the SNS scale parameters are transformed into a mid/side representation before quantization:

$$SNS_{mid}(i) = \frac{SNS_{left}(i) + SNS_{right}(i)}{\gamma_{enc}} \qquad (5.3\text{-}194)$$

$$SNS_{side}(i) = \frac{SNS_{left}(i) - SNS_{right}(i)}{\gamma_{enc}}, \qquad (5.3\text{-}195)$$

where $\gamma_{enc} = 1$ when the Split/AVQ quantizer is used and $\gamma_{enc} = 2$ when the MSVQ is used.

In operating points that use the MSVQ, the process is done once for each subframe (1 for long blocks, 2 for short blocks). Thus, it can be the case that the encoding mode is different in each subframe and the SNS scale parameters are encoded in different representation between the subframes.

The encoding mode is transmitted in the bitstream using a single bit (if the MSVQ is used, one bit per subframe is used). This bit is only written if both channels use the same transform block length in the current frame. If the block lengths differ between channels, this is used as an implicit signaling for the separate encoding mode as using the joint encoding mode is not possible in that case.

5.3.3.3.3.6.3.2 2-Stage Split/AVQ Quantizer

The 2-stage Split/AVQ quantizer is used at bitrates above 64kbps and in any bitrate if the core sampling rate is 16kHz. It consists of two stages with separate vector quantizers. The first stage vector quantizer is a stochastic split VQ which performs a fixed rate quantization. After the first stage, a residual vector is calculated, and the residual vector is further quantized using an Algebraic Vector Quantizer (AVQ) in the second stage which performs a variable rate quantization. In the joint encoding mode, the first stage is skipped for the side SNS parameters, and the residual is set to zero which results in SNS_{side}being directly quantized by the second stage AVQ only, as is depicted in Figure 5.3-19.

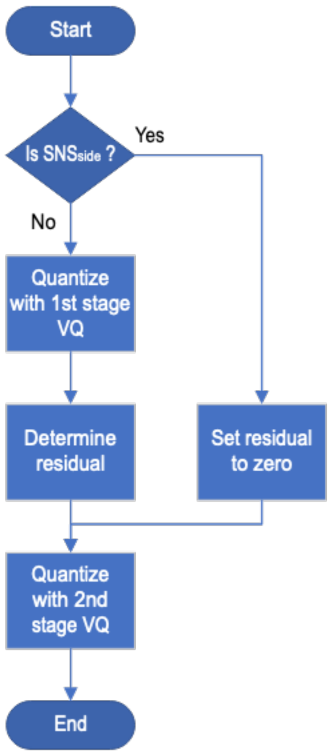


Figure 5.3-19: SNS parameter VQ encoding for long blocks in both channels

The 1st stage split VQ uses off-line trained stochastic codebooks of dimension 8 with 32 code vectors per codebook. Codebook selection depends on the split number (first split, second split) and the transform block size. To reduce the overall size of the needed codebooks, dependency on the core sampling rate is removed by normalizing the input vectors to the first stage VQ. Normalization is done by subtracting a mean value from each scale parameter in the target vector:

$$\overline{SNS}(i) = SNS(i) - \mu(i)\ \kappa_{means},\ i \in \{0, \ldots, 15\}. \qquad (5.3\text{-}196)$$

As the first stage VQ is skipped for side SNS scale parameter quantization, SNS can be one of SNS_{left}, SNS_{right} or SNS_{mid}. The values for $\mu(i)$ were computed as the means of each SNS scale parameter in the training data used for creating the codebooks for different core sampling rates and are denote in Table 5.3-12 below. The values are stored in a 16-bit Q format with 14 bits used for the fractional part. $\kappa_{means} = \frac{1}{2^{14}}$ is a conversion constant to convert the values back from the fixed-point number format.

Table 5.3-12: SNS parameter mean normalization values for 1st stage split VQ

Core sampling rate	16 kHz	16 kHz	25.6 kHz	25.6 kHz	32 kHz	32 kHz
block size	**Long**	**Short**	**Long**	**Short**	**Long**	**short**
μ(0)	14210	12018	14973	15560	15041	16510
μ(1)	19017	15915	20323	19489	20603	20660
μ(2)	14362	11089	16461	14623	16969	16025
μ(3)	9309	6015	9554	5595	10289	7224
μ(4)	5385	847	4017	2084	4973	3921
μ(5)	2674	-705	3103	1699	4283	3868
μ(6)	1055	-539	1602	775	3003	2623
μ(7)	-211	-1548	1694	-1312	3316	742
μ(8)	-1407	-893	-221	-2195	1684	-1316
μ(9)	-3059	-2163	-1401	-6101	-259	-6269
μ(10)	-4393	-1806	-6817	-9078	-6614	-8284
μ(11)	-8597	-4189	-10071	-9465	-9535	-7288
μ(12)	-11180	-7017	-11503	-7825	-10363	-6380
μ(13)	-11756	-8670	-11805	-6603	-11834	-8410
μ(14)	-12131	-8874	-13158	-7281	-16625	-13351
μ(15)	-13281	-9480	-16749	-9960	-24930	-20277

For both splits, a full search of the respective codebook is carried out. The code vector that minimizes the Mean Squared Difference to the target vector is chosen, i.e the codebook index for split $n \in [0, 1]$ is chosen according to:

$$ind_n = argmin_{i=0,\ldots,31}\left\{\sum_{j=0}^{7}\left(\widehat{SNS}(j+8\text{n}) - V_{n,m}(i)(j)\kappa_{codebook}\right)\right\}, for\ j = 0,\ldots,31 \quad (5.3\text{-}197)$$

where m is 0 for long transform blocks and 1 for short transform blocks and $V_{n,m}(i)$ denotes the i-th code vector for split n and the respective transform length. The codebooks are stored in 16-bit Q format using 12 bits for the fractional part with $\kappa_{codebook} = \frac{1}{2^{12}}$ being the conversion constant. The found two optimal indices are represented as two 5 bit values and get combined into a single 10-bit index as $ind_{SNS,1} = ind_0 + 2^5 \times ind_1$.

The output of the 1st stage VQ is then an intermediate quantized vector according to:

$$\widehat{SNS}_{VQ,1}(i) = \begin{cases} V_{0,m}(ind_0)\,(i)\kappa_{codebook} + \mu(i)\ \kappa_{means}, for\ i\ \in \{0,\ldots,7\} \\ V_{1,m}(ind_1)(i-8)\kappa_{codebook} + \mu(i)\ \kappa_{means}, for\ i\ \in \{8,\ldots,15\} \end{cases}. \quad (5.3\text{-}198)$$

For side SNS scale parameter quantization, the first stage is skipped and $\widehat{SNS}_{VQ,1}(i)$ is set to zero instead.

The residual vector is obtained by calculating the difference between each SNS scale parameter and the corresponding quantized parameter in the intermediate quantized vector and weighting the difference with the factor $1/_{0.4}$.

$$SNS_{res}(i) = \frac{SNS(i) - \widehat{SNS}_{VQ,1}(i)}{0.4}. \quad (5.3\text{-}199)$$

The residual vector is further quantized by the second stage VQ which consists of the AVQ as described in EVS [3] in clause 5.2.3.1.6.9. The output of the second stage consists of a base codebook index and a Voronoi extension index.

After the quantization indices have been calculated, the quantized SNS scale parameter vector is calculated as

$$\widehat{SNS}(i) = \widehat{SNS}_{VQ,1}(i) + 0.4\,\widehat{SNS}_{VQ,2}(i), \quad (5.3\text{-}200)$$

where $\widehat{SNS}_{VQ,2}$ denotes the quantized residual vector as calculated by the AVQ.

For scale parameters in side representation, a *zero_side* flag is set according to

$$zero\_side = \begin{cases} 1, if\ \sum_{i=0}^{15}\widehat{SNS}_{side}(i) = 0 \\ 0, otherwise \end{cases}. \quad (5.3\text{-}201)$$

The flag is encoded in the bitstream using a single bit. If the *zero_side* flag is 1, the side SNS scale parameter vector is not encoded in the bitstream.

If at least one of the channels uses short transform blocks, no joint coding of the SNS scale parameters with the split/AVQ quantizer is supported. Both channel's SNS scale parameters are then separately encoded using both stages.

At 48 kbps only, a special low-bitrate mode can be used additionally. The low-bitrate mode is activated globally for a frame (not on a per-channel basis) and is signaled by a single low-bitrate-mode indicator bit which is written only at 48 kbps if at least one channel uses short blocks. The indicator bit is set to one if the speech/music classifier (Clause 5.3.3.2.4) classified the first channel as speech, otherwise it is set to zero. If it is one, only the first stage split VQ is used to quantize all SNS scale parameter vectors in this frame and the second stage VQ is skipped.

If the low-bitrate mode is not used (i.e. at 48kbps if the frame is not classified as speech and at all other bitrates – regardless of signal classification), an inter-subframe coding mechanism is used for channels with short transform block lengths. For short transform block lengths, two sets of SNS scale parameters need to be quantized and encoded – one for each subframe. The first set is always encoded as described before, using both VQ stages. For the second set, one of two encoding strategies is chosen, based on which one needs less bits to encode the second parameter vector. The first strategy is simply to encode the second subframe's SNS vector with both stages of the VQ as done for the first subframe's vector. The number of bits needed for that is $nbits_1 = 10 + nbits_{AVQ,1}$, where 10 is the fixed number of bits used by stage 1 and $nbits_{AVQ,1}$ is the number of bits needed by stage 2. The second strategy employs the 2nd stage only, using the difference between the quantized first subframe's vector and the unquantized second subframe's vector as the residual to be quantized. The number of bits needed for this strategy is only the number of bits needed by the AVQ to quantize this residual, denoted as $nbits_{AVQ,2}$. If $nbits_{AVQ,2} < nbits_1$, the second encoding strategy is chosen and only the AVQ indices for quantizing the difference vector are transmitted. Otherwise, 1st and 2nd stage indices are transmitted for the second subframe, too. This encoding process is depicted in figure Figure 5.3-20. The encoding strategy is signaled in the bitstream using a single bit.

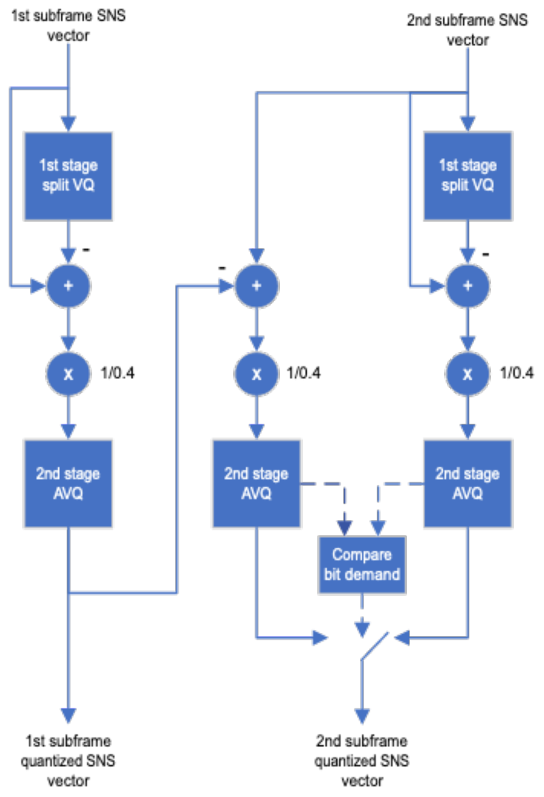


Figure 5.3-20: Inter-subframe encoding of SNS scale parameter vectors

5.3.3.3.3.6.3.3 MSVQ

At bitrates 48 and 64 kbps if the core sampling rate is higher than 16kHz, the SNS scale parameters are quantized using a stochastic MSVQ with offline-trained codebooks. Codebook selection differentiates between the kind of representation the respective SNS parameter vector is in. SNS_{left}, SNS_{right} and SNS_{mid} share codebooks while there is a separate codebook used for SNS_{side} vectors. In between these two groups, there are separate codebooks used depending on the transform block length of the current frame (long or short). Therefore, there are four codebooks in total. The number of stages and numbers of bits per stage for each codebook are shown in Table 5.3-13.

Table 5.3-13: Codebook parameters for SNS MSVQ

Codebook	Used for		Number of stages	Bits per stage
V_A	SNS_{left}, SNS_{right} , SNS_{mid}	Long blocks	4	7, 6, 5, 5
V_B	SNS_{left}, SNS_{right} , SNS_{mid}	Short blocks	3	7, 5, 3
V_C	SNS_{side}	Long blocks	2	5, 5
V_D	SNS_{side}	Short blocks	2	5, 3

Encoding of an SNS scale parameter vector is done as described in EVS [3] clause 5.6.3.5 but using the respective codebooks and parameters. For scale parameters in side representation, a reduced number of stages is used compared to what is used for the other representations and the quantization is skipped completely if the side representation is determined to be near zero. This is done by examining the similarity measure E as defined in Equation (5.3-193) and setting a *zero_side* flag according to

$$zero\_side = \begin{cases} 1, if\ E < 1 \\ 0, otherwise \end{cases}. \tag{5.3-202}$$

The flag is encoded in the bitstream using a single bit. If *zero_side* is 1, the values in the side SNS scale parameter vector are set to zero and quantization is skipped. Otherwise, the parameter vector is quantized using the respective codebook and MSVQ parameters as given in Table 5.3-13.

5.3.3.3.3.6.3.4 Restoring left/right representation of jointly coded SNS scale parameters

If the joint encoding mode was chosen, the SNS scale parameters were encoded in a mid/side representation and need to be decoded back to a left/right representation so they can be used to shape the channel signals. The restored left/right SNS scale parameter vectors are calculated according to:

$$\widehat{SNS}_{left}(i) = \begin{cases} \widehat{SNS}_{mid}(i), if\ zero\_side\ is\ 1 \\ \widehat{SNS}_{mid}(i) + \frac{\widehat{SNS}_{side}(i)}{\gamma_{dec}}, otherwise \end{cases} \tag{5.3-203}$$

$$\widehat{SNS}_{right}(i) = \begin{cases} \widehat{SNS}_{mid}(i), if\ zero\_side\ is\ 1 \\ \widehat{SNS}_{mid}(i) - \frac{\widehat{SNS}_{side}(i)}{\gamma_{dec}}, otherwise \end{cases} \tag{5.3-204}$$

Where $\gamma_{dec} = 2$ if the Split/AVQ quantizer was used and $\gamma_{dec} = 1$ if the MSVQ was used.

5.3.3.3.3.6.4 Spectral Shaping

After quantizing the SNS scale parameters, the MDCT spectrum is scaled using scale factors derived from the quantized scale parameters. The scale factors are obtained by interpolation of the scale parameters. Shaping of the MDCT spectrum is done separately for both channels.

5.3.3.3.3.6.4.1 SNS scale factor interpolation

The 16 quantized SNS scale parameters are interpolated to obtain 64 scale factors according to

$$\begin{aligned} scf(0) &= \widehat{SNS}(0) \\ scf(1) &= \widehat{SNS}(0) \\ scf(4n+2) &= \widehat{SNS}(n) + \tfrac{1}{8}\left(\widehat{SNS}(n+1) - \widehat{SNS}(n)\right), n = 0, \dots, 14 \\ scf(4n+3) &= \widehat{SNS}(n) + \tfrac{3}{8}\left(\widehat{SNS}(n+1) - \widehat{SNS}(n)\right), n = 0, \dots, 14 \\ scf(4n+4) &= \widehat{SNS}(n) + \tfrac{5}{8}\left(\widehat{SNS}(n+1) - \widehat{SNS}(n)\right), n = 0, \dots, 14 \\ scf(4n+5) &= \widehat{SNS}(n) + \tfrac{7}{8}\left(\widehat{SNS}(n+1) - \widehat{SNS}(n)\right), n = 0, \dots, 14 \end{aligned} \tag{5.3-205}$$

$$scf(62) = \widehat{SNS}(15) + \frac{1}{8}\left(\widehat{SNS}(15) - \widehat{SNS}(14)\right)$$

$$scf(63) = \widehat{SNS}(15) + \frac{3}{8}\left(\widehat{SNS}(15) - \widehat{SNS}(14)\right).$$

The scale factors are then transformed back into the linear domain according to

$$g_{SNS}(b) = 2^{-scf(b)}, b = 0, \ldots, 63. \qquad (5.3\text{-}206)$$

5.3.3.3.3.6.4.2 Scaling the MDCT spectrum

The interpolated scale factors are applied to the MDCT spectrum in the same bands as used for calculating the SNS scale parameters as described in clause 5.3.3.3.3.6.1 according to the following pseudocode:

```
For b = 0 to 63 do
    For k = I_low(b) to I_high(b) do
        X_s(k) = X(k)*g_SNS(b)
```

5.3.3.3.4 Stereo processing

5.3.3.3.4.1 General

The stereo encoding processing is applied band-wise on the frequency-domain noise-shaped signals. First, a global ILD normalization value is determined, based on the energy values of the first channel and the second channel inputs. Depending on the normalization value, at least one of the two channels is normalized to achieve similar levels for both channels. Then, the encoding of the normalized audio signals follows, by determining for each spectral band whether the output spectral band remains the same for the first and second channel (dual-mono representation) or whether the output spectral band of the first channels is a mid signal of the first and second channel and the output spectral band of the second channel is a side signal of the first and second channel (mid-side representation), depending on the spectral band-wise criteria described in detail in clause 5.3.3.3.4.5. For many signals significant coding gain is achieved by having some bands in L/R (dual-mono) and some bands in M/S (mid-side) representation. Finally, for each spectral band the decided encoding is performed outputting the encoded audio signal for each channel.

5.3.3.3.4.2 Stereo spectral bands

Most of the stereo processing is done in spectral bands configured specifically for the stereo processing. The stereo spectral bands are configured following mainly the ERB scale with some modifications to accommodate the long frames for the TCX20 coding mode and the short frames for the TCX10 coding mode and to match the IGF spectral bands for the bitrates where IGF is active.

In Table 5.3-14 the grouping of frequency bins to spectral bands and the number of bands that are assigned to each bandwidth for TCX20 and TCX10 coding mode is shown. If the total spectral offset exceeds the maximum available bandwidth length, then the last band is corrected to the maximum bandwidth length.

$$b_{offset[b_{last}]} = \min\left(b_{offset}, L_{frameTCX}\right) \qquad (5.3\text{-}207)$$

Table 5.3-14: Core stereo spectral bands per coding mode

	TCX20			TCX10		
Band	b_{offset}	N_{bins}	**Bwidth**	b_{offset}	N_{bins}	**Bwidth**
0	0	0	WB	0	0	WB
1	4	4	WB	4	4	WB
2	8	4	WB	8	4	WB
3	12	4	WB	12	4	WB
4	16	4	WB	16	4	WB
5	20	4	WB	20	4	WB
6	24	4	WB	28	8	WB
7	28	4	WB	36	8	WB
8	32	4	WB	44	8	WB
9	36	4	WB	52	8	WB
10	40	4	WB	64	12	WB
11	48	8	WB	76	12	WB
12	56	8	WB	88	12	WB
13	64	8	WB	100	12	WB
14	72	8	WB	112	12	WB
15	80	8	WB	124	12	WB
16	88	8	WB	136	12	WB
17	96	8	WB	148	12	WB
18	108	8	WB	160	12	WB
19	120	12	WB	172	12	SWB
20	132	12	WB	184	12	SWB
21	144	12	WB	196	12	SWB
22	164	12	WB	208	12	SWB
23	184	20	WB	220	12	SWB
24	204	20	WB	232	12	SWB
25	226	20	WB	244	12	SWB
26	248	22	WB	256	12	SWB
27	270	22	WB	268	12	SWB
28	292	22	WB	288	20	SWB
29	314	22	WB	320	32	SWB
30	336	22	SWB	352	32	FB
31	358	22	SWB	384	32	FB
32	360	22	SWB	432	48	FB
33	392	22	SWB	512	80	FB
34	424	22	SWB			
35	446	22	SWB			
36	468	22	SWB			
37	490	22	SWB			
38	512	22	SWB			
39	534	22	SWB			
40	576	22	SWB			
41	640	42	SWB			
42	704	64	FB			
43	800	96	FB			
44	960	160	FB			

The aforementioned stereo bands are applied for the configurations shown in Table 5.3-15.

Table 5.3-15: Configurations where the core stereo bands are used

F_s **Core (in kHz)**	**Bitrates (in kbps)**
48	all
32	up to 96
25.6	up to 96
16	up to 96

For all other configurations i.e. 16-32KHz F_s and bitrate > 96kbps the spectral bands defined in Table 5.3-9 are used.

For the bitrates where IGF is enabled the cross-over bands are adjusted to represent the cross-over frequency between the core encoding and the IGF encoding. Also, the bands over the IGF cross-over frequency are adjusted to be identical to the IGF spectral bands, defined depending on the configuration, as depicted in XXX. For this case, the spectral bands are divided to core spectral bands and IGF spectral bands.

5.3.3.3.4.3 Global ILD normalization

The single global ILD normalization value is determined based on the energies of the first and second channel as follows:

$$E_1 = \sqrt{\sum_{k=0}^{L} X_1^{MDCT}[k]^2} \quad (5.3\text{-}208)$$

$$E_2 = \sqrt{\sum_{k=0}^{L} X_2^{MDCT}[k]^2} \quad (5.3\text{-}209)$$

$$ILD = \frac{E_1}{E_1+E_2} \quad (5.3\text{-}210)$$

where: $X_1^{MDCT}[k]$ is the k-th coefficient of the MDCT spectrum of the first channel, $X_2^{MDCT}[k]$ is the k-th coefficient of the MDCT spectrum of the second channel respectively, and L is the frame length as in number of MDCT coefficients.

The final normalization value is calculated using the quantized ILD value, calculated as:

$$\widehat{ILD} = max\left(1, \min\left(2^{b_{ILD}} - 1, \lfloor 2^{b_{ILD}} * ILD + 0.5 \rfloor\right)\right) \quad (5.3\text{-}211)$$

$$g_{ILD} = \frac{2^{b_{ILD}}}{\widehat{ILD}} - 1 \quad (5.3\text{-}212)$$

where $\widehat{ILD}$ is the quantized ILD, b_{ILD} is the number of bits selected for quantization, and g_{ILD} is the normalization value.

Then the spectrum of either the first channel or second channel is normalized, depending on the normalization value, by applying g_{ILD} to the spectral coefficients, to achieve similar energy levels as follows:

$$\bar{X}_1^{MDCT}[k] = g_{ILD} \cdot X_1^{MDCT}[k], \quad \text{for } k = 0 \ldots L_{frameTCX} - 1, \quad \text{if } g_{ILD} < 1$$

(5.3-213)

$$\bar{X}_2^{MDCT}[k] = 1/g_{ILD} \cdot X_2^{MDCT}[k], \quad \text{for } k = 0 \ldots L_{frameTCX} - 1, \quad \text{if } g_{ILD} > 1$$

where $L_{frameTCX}$ represents the full bandwidth frame length in bins and $\bar{X}_1^{MDCT}, \bar{X}_2^{MDCT}$ represent the respective normalized signals.

The MDST spectrum coefficients are also normalized with the same normalization value:

$$\bar{X}_1^{MDST}[k] = g_{ILD} \cdot X_1^{MDST}[k], \quad \text{for } k = 0 \ldots L_{frameTCX} - 1, \quad \text{if } g_{ILD} < 1 \quad (5.3\text{-}214)$$

$$\bar{X}_2^{MDST}[k] = 1/g_{ILD} \cdot X_2^{MDST}[k], \quad \text{for } k = 0 \ldots L_{frameTCX} - 1, \quad \text{if } g_{ILD} > 1$$

5.3.3.3.4.4 Conditional ILD correction

5.3.3.3.4.5 Band-wise M/S decision

The stereo encoder selects between three stereo coding modes namely full mid-side encoding (full M/S), full dual mono encoding (dual-mono) and the band-wise encoding mode (band-wise M/S). To determine which mode should be selected for encoding, the total number of bits needed for each of the stereo coding modes is estimated, given the available number of bits for encoding using perceptual entropy coding. The stereo encoding mode with the smallest estimated number of bits is selected.

Stereo encoding may only be applied when the coding mode for both channels is identical (either TCX20 or TCX10) and therefore, spectral resolution is the same. Otherwise, the stereo encoding mode is forced to dual-mono and the steps described below are skipped.

The detailed steps for the M/S decision and determination of the stereo encoding mode are described below.

The SQ gain G is first estimated given the normalized spectra of the first channel and the second channel and the number of bits available for encoding.

The full M/S transform of the first channel and second channel spectra is calculated as:

$$X_M[k] = \frac{\sqrt{(2)}}{2}(X_1[k] + X_2[k]), \quad for\ k = 0, \dots, L_{frame} - 1 \tag{5.3-215}$$

$$X_s[k] = \frac{\sqrt{(2)}}{2}(X_1[k] - X_2[k]), \quad for\ k = 0, \dots, L_{frame} - 1$$

Then the respective number of bits, b_{LR} for the dual-mono mode and b_{MS} for the full M/S mode, are estimated by quantizing the respective spectra and applying a perceptual entropy coder described in clause 5.3.3.3.8.3 given the SQ gain G and the number of bits available for the encoding.

The number of bits b_{BW} needed for the band-wise M/S encoding mode are estimated using the following formula:

$$\mathrm{b_{BW}} = \mathrm{n_{Bands}} + \sum_{\mathrm{i=0}}^{\mathrm{n_{Bands,core}}-1} \min\left(b_{bwLR}^{i}, b_{bwMS}^{i}\right) \tag{5.3-216}$$

where: $n_{Bands,core}$ is the number of the core spectral bands of the normalized audio signal, one bit is needed for each band to signal the mode for the particular band;

b_{bwMS}^{i} is an estimation for a number of bits that are needed for encoding an i'-th spectral band of the mid signal and for encoding the i-th spectral band of the side signal;

b_{bwLR}^{i} is an estimation for a number of bits that are needed for encoding an i-th spectral band of the first signal and for encoding the i-th spectral band of the second signal.

The stereo encoding mode decision for each spectral band is stored in a $n_{Bands,core}$ long array, the so-called M/S mask, where for encoding the i-th spectral band M/S is signalled with 1 and dual-mono is signaled with 0.

If $b_{LR} < b_{BW}$ then the stereo mode is DUAL_MONO and the M/S mask is an n_{Bands} array of 0s.

If $b_{MS} < b_{BW}$ then the stereo mode is MS_FULL and the M/S mask is an n_{Bands} array of 1s.

Otherwise, the stereo encoding mode is BW_MS and the M/S mask is populated accordingly with decision m_i depending on the respective estimated number of bits for M/S b_{bwMS}^{i} and dual-mono b_{bwLR}^{i} for the i-th spectral band as follows:

$$\mathrm{m_i} = \begin{cases} 1, & if\ b_{bwMS}^{i} < b_{bwLR}^{i} \\ 0, & if\ b_{bwLR}^{i} < b_{bwMS}^{i} \end{cases} \quad i = 0 \dots n_{Bands,core} - 1 \tag{5.3-217}$$

If IGF is enabled for the given configuration, the M/S decision for the IGF spectral bands is determined by the algorithm described in clause 5.3.3.3.4.6.

5.3.3.3.4.6 M/S decision for IGF

5.3.3.3.4.7 M/S Transformation

Depending on the stereo encoding mode decision described in clause 5.3.3.3.4.5 either full-mid-side (MS_FULL), full dual-mono (DUAL_MONO) or band-wise encoding (BW_MS) is applied.

- If the selected mode is full mid-side (MS_FULL) : a mid signal from the first channel and from the second channel of the normalized audio signal is obtained as a first channel as shown in Equation (5.3-218) and a side signal from the first channel and from the second channel of the normalized audio signal as a second channel as shown in Equation (5.3-219).

$$\tilde{X}_1[k] = \frac{\sqrt{(2)}}{2}(\bar{X}_1[k] + \bar{X}_2[k]), \quad for\ \mathrm{k} = 0, \dots, \mathrm{L_{frame}} - 1 \tag{5.3-218}$$

$$\tilde{X}_2[k] = \frac{\sqrt{(2)}}{2}(\bar{X}_1[k] - \bar{X}_2[k]), \quad for\ k = 0, \dots, L_{frame} - 1 \qquad (5.3\text{-}219)$$

where $\bar{X}_1, \bar{X}_2$ are the normalized inputs and $\tilde{X}_1, \tilde{X}_2$ represent the processed outputs of the first and second channel. The factor $\frac{\sqrt{(2)}}{2}$ is employed to preserve the energy level of the signal after decoding and inverse transformation.

- If the full dual-mono (DUAL_MONO) mode is selected then no processing occurs and the normalized audio signals of the two channels are encoded as is.
- If the band-wise stereo encoding mode (BW_MS) is selected, then the M/S mask described in clause 5.3.3.3.4.5 which represents the decision for each spectral band determines whether mid-side encoding or whether dual-mono encoding is employed in a given spectral band.
    - If mid-side encoding is employed, for a given spectral band the respective processed output spectral band of the first channel is a mid-signal based on the respective spectral bands of the normalized audio signals of the first channel and the second channel. Accordingly, the respective processed output spectral band of the second channel is a side signal based on the respective spectral bands of the normalized audio signals of the first channel and the second channel.
    - If dual-mono encoding is employed for a given spectral band, then no processing occurs, and the output spectral band for the first channel is the respective spectral band of the input normalized audio signal and the output spectral band of the second channel is the respective spectral band of the second channel normalized audio signal.

This is depicted in Equations (5.3-220) and (5.3-221).

For each spectral band $i = 0 \dots n_{Bands,core} - 1$:

$$\tilde{X}_1[b_{offset}(i) + k] = \begin{cases} \frac{\sqrt{2}}{2}(\bar{X}_1[b_{offset}(i) + k] + \bar{X}_2[b_{offset}(i) + k]), & \text{if } m_i = 1 \\ \bar{X}_1[b_{offset}(i) + k], & \text{otherwise} \end{cases} \qquad (5.3\text{-}220)$$

$$\tilde{X}_2[b_{offset}(i) + k] = \begin{cases} \frac{\sqrt{2}}{2}(\bar{X}_1[b_{offset}(i) + k] - \bar{X}_2[b_{offset}(i) + k]), & \text{if } m_i = 1 \\ \bar{X}_2[b_{offset}(i) + k], & \text{otherwise} \end{cases} \qquad (5.3\text{-}221)$$

where: $k = 0 \dots N_{bins}(i) - 1$;

$\bar{X}_1[b_{offset}(i) + k]$, $\bar{X}_2[b_{offset}(i) + k]$ is the k-th bin of the *i*-th spectral band of the normalized first and second channel respectively;

$\tilde{X}_1[b_{offset}(i) + k], \tilde{X}_2[b_{offset}(i) + k]$ is the k-th bin of the *i*-th spectral band of the processed first and second channel output;

$b_{offset}(i)$ is the spectral offset of the i -th band from Table 5.3-14′

m_i is the MS mask decision of the the *i*-th spectral band set in equation (5.3-217);

$N_{bins}(i)$ is the width of the *i*-th spectral band as in number of spectral bins in Table 5.3-14;

$n_{Bands,core}$ is the total number of core spectral bands.

5.3.3.3.4.8 Encoding of the stereo parameters

The stereo parameters that are encoded to the bitstream to revert the stereo processing on the decoder side are the following:

- The stereo encoding mode
- The quantized ILD $\widehat{ILD}$
- If the stereo encoding mode is band-wise M/S the ms mask
- The stereo mode for the IGF spectral bands (if applicable)
- If the IGF stereo encoding mode is band-wise M/S the ms mask for the IGF spectral bands

5.3.3.3.5 Power spectrum calculation and spectrum noise measure

After the modified spectra from the stereo processing for the first and second channel are derived, the power spectrum is newly calculated. That is needed for the IGF analysis and the calculation of the spectrum noise measure for each channel. The procedure is identical as in the EVS MDCT based TCX mode and is described in detail in clause 5.3.3.2.5 of [3]. As mentioned in clause 5.3.3.3.3.6.2 if TNS is applied, then the MDST is not used for the calculation of the power spectrum, instead an estimate from the MDCT spectrum is used.

5.3.3.3.6 Intelligent gap filling (IGF)

5.3.3.3.6.1 General overview

The intelligent gap filling bandwidth extension is carried out in general for each channel separately on the modified spectra from the stereo processing. The details of the IGF algorithm are described in clause 5.3.3.2.11 of [3]. The modifications that apply in the IVAS codec are described in clause XXX.

However, if certain conditions apply, special considerations for the stereo IGF encoding takes place, the details of which are described in clause 5.3.3.3.6.2. Stereo IGF encoding is used when the spectral resolution for both channels is the same, meaning the transform type is identical and either when the stereo mode is band-wise M/S or when the stereo mode for the core spectral bands is different from the stereo mode for the IGF spectral bands.

5.3.3.3.6.2 Stereo IGF encoding

5.3.3.3.7 Bitrate distribution

The numbers of bits for quantization and encoding are assigned adaptively to each channel based on the channel energies. First the ratio of the energy of the first channel to total energy of both channels is computed in the same fashion as is done for the initial ILD calculation described with equations (5.3-208)-(5.3-210). The quantized ratio is signaled in the bitstream and is then used for splitting the bits to be assigned to each channel in order to be in sync with the decoder. The quantized ratio is calculated:

$$\hat{r}_{split} = \left\lfloor 2^{b_{\hat{r}_{split}}} * r_{split} + 0.5 \right\rfloor \quad (5.3\text{-}222)$$

where r_{split} is the split ratio derived from equation (5.3-210) applied on the stereo processed channels and $b_{\hat{r}_{split}}$ is the number of bits to code the split ratio.

The bits assigned to each channel for encoding the final spectra are divided among the total number of bits that remain after encoding the parameters of the processing steps namely TCX-LTP, TNS, SNS, IGF, the stereo processing parameters and the split ratio $\hat{r}_{split}$. Additionally, the number of bits reserved for the noise filling levels after quantization are also subtracted:

$$b_{available} = b_{total} - b_{tcxltp} - b_{TNS} - b_{SNS} - b_{IGF} - b_{stereo} - b_{NF} - b_{\hat{r}_{split}}$$

Then the bitrate assigned to each channel for the encoding of the spectrum is derived as follows:

$$b_{ch0} = \frac{\hat{r}_{split}}{2^{b_{\hat{r}_{split}}}} \cdot b_{available} \quad (5.3\text{-}223)$$

and

$$b_{ch1} = b_{available} - b_{ch0} \quad (5.3\text{-}224)$$

5.3.3.3.8 Quantization and encoding

5.3.3.3.8.1 Quantization

5.3.3.3.8.2 Stereo noise level estimation

5.3.3.3.8.3 Range coder

5.3.3.4 Switching between stereo modes

5.3.3.4.1 Switching between DFT and TD stereo modes

5.3.3.4.1.1 Pre-preprocessing steps

5.3.3.4.1.2 Switching from the TD stereo to the DTF stereo

5.3.3.4.1.3 Switching from the DFT stereo to the TD stereo

5.3.3.4.2 Switching between DFT and MDCT stereo modes

5.3.3.4.3 Switching between TD and MDCT stereo modes

5.3.3.5 DTX operation

5.3.3.5.1 DTX in Unified stereo

5.3.3.5.1.1 Signal activity detection in Unified stereo

The signal activity detection is run on the down-mix signal as described in 5.2.2.1.4. To aid in the stereo classification and selection between the TD-based stereo and DFT-based stereo, as well as activating the Stereo CNG mode, an additional signal activity detection is run on the input stereo signals coordinating the selection of encoding mode, see clause 5.3.3.2.3.3. Based on the selected active/inactive coding mode decision for each channel (VAD_0 and VAD_1), CNG encoding is applied based on the joint VAD decision VAD_{01} as obtained from equation (5.3-130). If a speech pause is detected, a transition period called the VAD hangover is entered, and after the hangover period CNG frames are transmitted.

5.3.3.5.1.2 General

The unified stereo support Comfort Noise Generation (CNG) operation with a stereo CNG mode based on the DFT stereo. If the encoder selects a CNG frame to be encoded, the DFT-based stereo analysis is performed in the same way as described in 5.3.3.2.2. A down-mix signal $s_M(n)$ is prepared according to 5.3.3.2.2.9 and is input to the IVAS core encoder. However, the setting of the core bit rate indicates a CNG frame is to be encoded and transmitted to the decoder, generally following the EVS CNG operation as described in [3] 5.6. There are however four differences compared to the EVS CNG encoding:

- Internal sampling frequency configuration
- LP-CNG high band analysis and quantization
- Core codec reset in active frame onset
- Energy scaling of CNG

An internal sampling frequency of 12.8 kHz is used for 13.2 kbps and 16.4 kbps LP-CNG WB, matching the internal sampling frequency of the active frame encoding. For all other configurations, 16 kHz internal sampling frequency is used.

The LP-CNG high band in Unified stereo CNG operating at SWB or FB bandwidth always uses SWB SID frames. WB SID frames are not used for SWB or FB operation.

The Unified stereo CNG decoder synthesizes the comfort noise in DFT domain and does not maintain the states of the core encoder and decoder. For this reason, the core encoder and decoder states are reset in the first active frame after an inactive segment.

The energy attenuation scheme of [3] 5.6 intends to match the level of the CNG with the level of the background noise when encoded with the active frame mode. For Unified Stereo CNG the same energy attenuation scheme is used both for LP-CNG and FD-CNG. This means that the energy offset selection for LP-CNG as described in [3] section 5.6.2.1.5 and table 151a is not used. Instead, the gain scaling described in [3] section 5.6.2.3.5 is used to calculate the attenuation for both types of CNG. The attenuation values have been updated for Unified stereo CNG operation as shown in Table 5.3-16.

Table 5.3-16: Energy scaling values in Unified stereo CNG

Bandwidth	**WB**				**SWB & FB**			
Bitrate [kbps]	13.2	16.4	24.4	32	13.2	16.4	24.4	32
Scaling factor [dB]	-4	-3	-1.6	0.2	-2	-3	-0.8	-0.25

In Unified stereo CNG operation, the DFT-based stereo parameters $ITD(m), gIPD(m), g_S[m,b]$ are obtained and quantized in the same way as for active frames as described in 5.3.3.2.2. These parameters are used in the preparation of the down-mix $s_M(n)$. The resolution of the stereo parameters is however reduced for CNG operation. The side gain prediction parameters as described in [ref to stereo filling] and the side gain prediction residual [ref to stereo residual] are however not used in CNG operation. Instead, a coherence parameter $C_{01}[b,m]$ is derived and encoded.

While the $ITD(m)$ and $gIPD(m)$ are applied on the full frequency band, $g_S[m,b]$ is applied on a band structure. The band resolution is generally lower than in active coding and the same band resolution is used for $g_S[m,b]$ and $C_{01}[m,b]$. The band resolutions are listed in comparison to the active encoding band resolution in Table 5.3-17.

Table 5.3-17: Number of frequency bands in Unified stereo CNG

Bitrate [kbps]	**WB**		**SWB**		**FB**	
	Active	**CNG**	**Active**	**CNG**	**Active**	**CNG**
13.2	10	5	12	6	12	6
16.4	10	5	12	6	12	6
24.4	10	5	12	6	12	6
32	10	5	12	6	13	6

The following clauses describe the encoding of the Unified stereo CNG parameters in more detail.

5.3.3.5.1.3 Stereo CNG side gain encoding

The side gain parameter $g_S[m,b]$ is averaged over the SID interval, i.e., side gain values calculated in the NO_DATA frames are included in the average to provide a more stable estimate for CNG frames. For the very first SID frame after starting the codec, or SID update frames during an inactive period, the average side gain $\tilde{g}_S[m,b]$ is defined according to

$$\tilde{g}_S[m,b] = \frac{\sum_{i=0}^{N_{curr}-1} g_{curr}[i,b]}{N_{curr}} \tag{5.3-225}$$

where N_{curr} denotes the number of frames in the current inactive segment including the current frame, $g_{curr}[i,b]$ is the side gain value for frame i of the current inactive segment and $b = 0,1,\dots,N_{bands}-1$ is the band index corresponding to the band resolution of the active frames. For the very first SID frame, $N_{curr} = 1$ and the average corresponds to the current side gain. In the first SID frame of an inactive segment, excluding the very first SID frame since startup, the side gain average $\tilde{g}_S[m,b]$ is computed according to

$$\tilde{g}_S[m,b] = \frac{\sum_{i=0}^{N_{curr}-1} g_{curr}[i,b] + W(N_f)\sum_{j=0}^{N_{prev}-1} g_{prev}[j,b]}{N_{curr} + W(N_f)N_{prev}} \tag{5.3-226}$$

where $g_{prev}[j,b]$ is the side gain value for frame j of the previous inactive segment, N_{prev} is the number of frames in the previous inactive segment and $W(N_f)$ is a weighting function defined as

$$W(N_f) = \begin{cases} \frac{0.8(1500-N_f)}{1500} + 0.2, & N_f < 1500 \\ 0.2, & N_f \geq 1500 \end{cases} \tag{5.3-227}$$

where N_f is the number of frames in the active segment that is just ending. In this case N_{curr} corresponds to the hangover period and the first SID frame. If the number of bands used in the SID frames $N_{SIDbands}$ is lower than the number of bands in active frames N_{bands} as shown in Table 5.3-17, the band values are combined according to

$$g_{S,SID}[m,b] = \begin{cases} \frac{\tilde{g}_S[m,2b]BW[2b] + \tilde{g}_S[m,2b+1]BW[2b+1]}{BW[2b]+BW[2b+1]}, & 2b+1 < N_{bands} \\ \tilde{g}_S[m,2b], & otherwise \end{cases} \tag{5.3-228}$$

where $BW[b]$ is the bandwidth of band b as defined in [Ref to DFT band description], used in calculation of $\tilde{g}_S[m,b]$ and $b = 0,1,\dots,N_{SIDbands}-1$. If the number of bands in active and inactive frames is the same, i.e. $N_{bands} = N_{SIDbands}$, the SID side gains are simply set to the averaged side gains $g_{S,SID}[m,b] = \tilde{g}_S[m,b]$. The side gain for the SID $g_{S,SID}[m,b]$ is encoded using the same encoding routine as for the active frames as described in xxx.

5.3.3.5.1.4 Stereo CNG ITD and IPD encoding

The ITD value $ITD(m)$ and IPD value $gIPD(m)$ is calculated as in XX and applied in the down-mix creation as described in 5.3.3.2.2 even before the decision is made to activate the Unified stereo CNG. In SID frames a coarser quantization step and narrower range is applied on the ITD in comparison to the active frames. The SID ITD values are calculated as follows,

$$ITD_{SID}(m) = \begin{cases} \max\left(15, \left\lfloor \frac{ITD(m)-256}{2} \right\rfloor\right), & ITD(m) > 255 \\ \max\left(15, \left\lfloor \frac{ITD(m)}{2} \right\rfloor\right), & ITD(m) \leq 255 \end{cases} \tag{5.3-229}$$

where $\lfloor\cdot\rfloor$ denotes a round-down or truncate operation. Additionally, a sign bit $ITD_{SID,sign}(m)$ is derived according to

$$ITD_{SID,sign}(m) = \begin{cases} 0, & ITD(m) > 255 \\ 1, & ITD(m) \leq 255 \end{cases} \tag{5.3-230}$$

The ITD parameters $ITD_{SID}(m)$ and $ITD_{SID,sign}(m)$ are encoded using the same scheme as the active ITD values, including a flag for non-zero ITD. If the non-zero ITD is set, only this flag is included in the bitstream. The resolution of the IPD value is reduced in a similar way in CNG operation. The index of the IPD value $gIPD_{SID,ind}(m)$ is calculated as

$$gIPD_{SID,ind}(m) = \begin{cases} I_{SID}, & I_{SID} < 4 \\ 0, & I_{SID} \geq 4 \end{cases}$$
$$I_{SID} = \left\lfloor \frac{2(gIPD(m)+\pi)}{\pi} + 0.5 \right\rfloor \quad (5.3\text{-}231)$$

Like the ITD parameter, the SID IPD index $gIPD_{SID,ind}(m)$ is included in the bit stream if the encoded IPD parameter non-zero flag is set.

5.3.3.5.1.5 Stereo CNG coherence encoding

The coherence $C_{band}[m,b]$ controls the perceived spatial width of the rendered comfort noise per band b in frame m. The range of $C_{band}[m,b]$ is $[0,1]$. The coherence $C_{01}(m,k)$ per bin k is defined as

$$C_{01}(m,k) = \frac{|S_{01,LP}(m,k)|^2}{P_{0,LP}(m,k)P_{1,LP}(m,k)} \quad (5.3\text{-}232)$$

where $S_{01,LP}(m,k)$ is the adaptively low-pass filtered cross-spectrum as defined in (5.3-23) and $P_{0,LP}(m,k)$ and $P_{1,LP}(m,k)$ are the low-pass filtered power spectra of channels 0 and 1 according to

$$\begin{cases} P_{0,LP}(m,k) = \beta|S_0(m,k)|^2 + (1-\beta)P_{0,LP}(m-1,k) \\ P_{1,LP}(m,k) = \beta|S_1(m,k)|^2 + (1-\beta)P_{1,LP}(m-1,k) \end{cases} \quad (5.3\text{-}233)$$

where β is the adaptive low-pass filter coefficient defined as in (5.3-23). The coherence per band $C_{band}[m,b]$ is a weighted average of $C_{01}(m,k)$, using the power spectrum of the down-mix channel $S_M(m,k)$ as a perceptual weighting according to

$$C_{band}[b,m] = \frac{\sum_{k=k_{start}(b)}^{k_{start}(b+1)-1} C_{01}(m,k)|S_M(m,k)|^2}{\sum_{k=k_{start}(b)}^{k_{start}(b+1)-1} |S_M(m,k)|^2} \quad (5.3\text{-}234)$$

where $b = 0,1,\dots,N_{bands}-1$, $k_{start}(b)$ is the start index of frequency band b and N_{bands} is the number of bands as defined in 5.3.3.2.2. For the first CNG frame in an inactive period, after being copied to $S_{01,LP}(m,k)$, $S_{01,LPCNG}(m,k)$ is reinitialized according to

$$S_{01,LPCNG}(m,k) := \sqrt{\hat{C}_{band}[m-2,b]P_{0,LP}(m,k)P_{1,LP}(m,k)}\frac{S_{01,LPCNG}(m,k)}{|S_{01,LPCNG}(m,k)|} \quad (5.3\text{-}235)$$

where the $\hat{C}_{band}[m-2,b]$ denotes the reconstructed coherence for band b for the second to last SID frame of the previous inactive period. The reconstructed coherence values $\hat{C}_{band}[m,b]$ are explained in the coming details leading up to equation (5.3-241).

The coherence $C_{band}[m,b]$ is encoded using a combination of intra-frame and inter-frame prediction and a residual encoding. An intra-frame prediction based on the target $C_{band}[m,b]$ is used to select a predictor index q according to

$$q = \underset{z}{\operatorname{argmin}} \sum_{b=0}^{N_{bands}-1} \left| C_{intra}^{(z)}[m,b] - C_{band}[m,b] \right|^2 \quad (5.3\text{-}236)$$

where the intra-frame prediction $C_{intra}^{(z)}[m,b]$ with predictor index z may be written

$$C_{intra}^{(z)}[m,b] = \begin{cases} 0, & b = 0 \\ \sum_{i=0}^{b-1} p^{(z)}[b,i]C_{band}[m,b], & b = 1,\dots,N_{bands}-1 \end{cases} \quad (5.3\text{-}237)$$

where the intra-frame predictor $p^{(z)}[b,i]$ are a set of predictor coefficients that predicts the next coherence value $C_{band}[m,b]$ based on the previous band values. There are 4 different intra-frame predictors $p^{(z)}[b,i]$, $z = 0,1,2,3$. The selected intra-frame predictor q is encoded using 2 bits. The combined prediction $\hat{C}_{pred}[m,b]$, now using reconstructed values, can be written

$$\hat{C}_{pred}[m,b] = \alpha\hat{C}_{intra}^{(q)}[m,b] + (1-\alpha)\hat{C}_{band}[m-1,b] \quad (5.3\text{-}238)$$

where the intra-frame prediction $\hat{C}_{intra}^{(q)}[m,b]$ now uses the selected predictor q and the previously reconstructed coherence values $\hat{C}_{band}[m,b]$.

$$\hat{C}_{intra}^{(q)}[m,b] = \begin{cases} 0, & b = 0 \\ \sum_{i=0}^{b-1} p^{(q)}[b,i]\hat{C}_{band}[m,b], & b = 1, \dots, N_{bands} - 1 \end{cases} \quad (5.3\text{-}239)$$

The locally decoded coherence values $\hat{C}_{band}[m,b]$ are reconstructed in the encoder to match the decoder state as basis for the prediction. Further, α is a weighting factor that combines the intra-frame prediction $\hat{C}_{intra}^{(q)}[m,b]$ with the inter-frame prediction $\hat{C}_{band}[m-1,b]$ which corresponds to the coherence value for band b in the previous frame $m-1$. The weighting factor α is selected from two candidate weighting factors $\alpha_{low}, \alpha_{high}$ that depend on the available bit budget for coherence encoding B_C in the current frame m. The selection of the candidate weighting factors is implemented using a lookup table, as illustrated in xxx.

Table 5.3-18: Candidate weighting factors based on coherence bit budget

Number of available bits	α_{low}	α_{high}
$B_C \leq 15$	0.1	0.6
$15 < B_C \leq 16$	0.1	0.6
$16 < B_C \leq 17$	0.1	0.7
$17 < B_C \leq 18$	0.1	0.9
$18 < B_C$	0.2	0.9

The principle of selecting the weighting factor α is to select the largest value which still fits within the bit budget B_C, since a larger α gives less inter-frame prediction and hence reduces error propagation. A trial encoding is run with each weighting factor $\alpha_{low}, \alpha_{high}$ and a selection is done according to

$$\alpha = \begin{cases} \alpha_{high}, & B_{high} \leq B_C \\ \alpha_{low}, & B_{low} \leq B_C < B_{high} \\ \underset{\alpha}{\operatorname{argmin}}\big((B(\alpha)\big), \min\big(B_{low}, B_{high}\big) > B_C \end{cases} \quad (5.3\text{-}240)$$

where $B(\alpha)$ denotes the number of bits to encode the coherence vector using the weighting factor α such that $B_{low} = B(\alpha_{low})$ and $B_{high} = B(\alpha_{high})$. The bottom line of (5.3-240) may be read as if both α yields a number of bits that exceeds the coherence bit budget B_C, then the candidate α giving the lowest number of bits is selected. The selection of the candidate α is encoded using one bit. Note that this bit may be subtracted in the comparison with the bit budget B_C, in which case the conditions may be written $B(\alpha) < B_C - 1$. A prediction error is formed by a prediction residual $C_{res}[m,b]$, calculated according to

$$C_{res}[m,b] = C_{band}[m,b] - \hat{C}_{pred}[m,b] \quad (5.3\text{-}241)$$

The prediction error is uniformly scalar quantized to 9 levels between -0.4 and 0.4 in steps of 0.1 and rearranged as in Table 5.3-19.

Table 5.3-19: Coherence prediction residual codebook

Index	Codeword	Value
0	0	0.0
1	10	0.1
2	110	-0.1
3	1110	0.2
4	11110	-0.2
5	111110	0.3
6	1111110	-0.3
7	11111110	0.4
8	111111110	-0.4

The residual $C_{res}[m,b]$ is quantized and encoded using a variable rate unary code in Table 5.3-19. If this is a trial encoding, the full number of bits is accumulated in $B(\alpha)$, but if it is the final encoding run a remaining number of bits is calculated as $B_C - B(\alpha)$ and the encoding is truncated if the bits run out, i.e. the length of the codeword for band b is larger than $B_C - B(\alpha)$. If the required number of bits to encode $C_{res}[m,b]$ for a certain band b, the index is reduced by 2 until it fits within the bit budget or reaches zero. If the bits run out, all the remaining residual indices are excluded from encoding and the remaining codewords are assumed to be zero. The truncated residual encoding can be described with the following pseudocode:

```
• $B(\alpha) \coloneqq 0$
• For all bands $b$
    o Quantize and encode $C_{res}[m,b]$ to obtain index $I_{res}$ with length $I_{res} + 1$ in bits
    o While $I_{res} + 1 > B_C - B(\alpha)$
        ▪ Reduce index by two $I_{res} \coloneqq \max(0, I_{res} - 2)$
    o If $I_{res} + 1 \leq B_C - B(\alpha)$
        ▪ Include $I_{res}$ in encoded bits using codeword representation
        ▪ Increment used bits $B(\alpha) \coloneqq B(\alpha) + I_{res} + 1$
```

The encoded quantization indices are included in the bitstream to be sent to the decoder. The reconstructed coherence values are formed according to

$$\hat{C}_{band}[m,b] = \alpha \hat{C}_{intra}^{(q)}[m,b] + (1-\alpha)\hat{C}_{band}[m-1,b] + \hat{C}_{res}[m,b] \qquad (5.3\text{-}242)$$

where α now denotes the selected weighting factor. This reconstruction is done for each band and is used for encoding the next band. Note that this matches the reconstruction that the decoder performs, with the exception that $\hat{C}_{band}[m-1,b]$ may diverge in case of transmission errors. If there are no bit at all in the bit budget for the stereo coherence coding, $B_C = 0$, then α and $\hat{C}_{res}[m,b]$ is set to zero and the previous coherence is used, i.e. $\hat{C}_{band}[m,b] = \hat{C}_{band}[m-1,b]$.

5.3.3.5.2 DTX in MDCT-based stereo

5.3.3.5.2.1 General overview

DTX operation in MDCT-based stereo is based on the FD-CNG functionality from EVS. It is extended by encoding information about the coherence between the two channels in the SID frame and an efficient way of transmitting the parametric background noise description for both channels.

5.3.3.5.2.2 VAD in MDCT-based Stereo

The stereo signal is analyzed by a voice activity detector that determines a frame to be an inactive frame or an active frame. The activity detector analyzes the two channels separately to classify them as either active or inactive. Then, it determines the whole frame to be inactive if both the first channel and the second channel are classified as inactive. Otherwise, the frame is classified as active. The decision to classify a single channel as active or inactive is done the same as described in clause 5.2.2.1.4.

The VAD functionalities in each of the channels operate completely independent of each other. This also includes the mechanism for determining when to send the next SID frame in variable-DTX-update-interval operation (see EVS [3], clauses 5.6.1.1 and 5.6.1.2). To prevent the two channel VADs to go out of synch, a synchronization mechanism is employed as depicted in Figure 5.3-21. If any of the two channel VADs decides on sending an SID in the current frame, the current frame will be an SID frame, even if the other channel's VAD decides that based on the other channel, the current frame would be a NO_DATA frame.

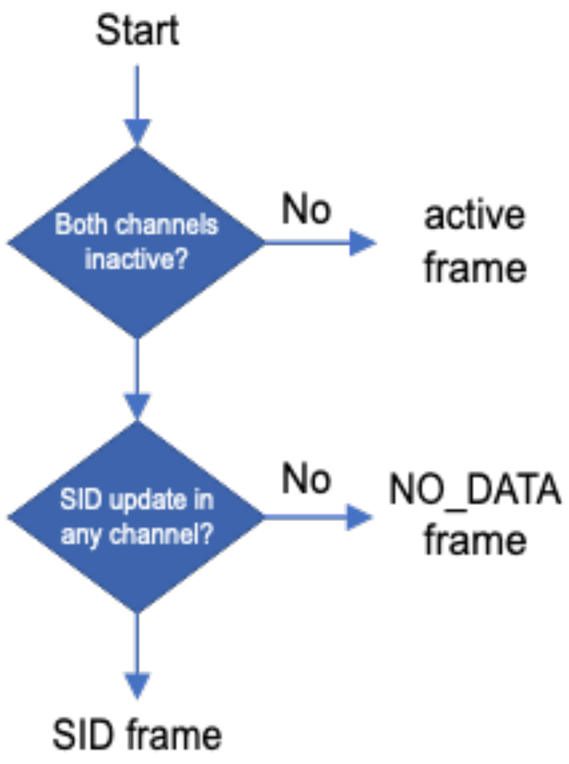


Figure 5.3-21: Frame type decision in MDCT-based Stereo DTX

5.3.3.5.2.3 Coherence estimation

A coherence calculator is run on the two input channel spectra in every frame to calculate a coherence value. In each of two subframes, four intermediate values (a real intermediate value, c_{real}, and an imaginary intermediate value, c_{imag}, as well as a first energy value for the first channel, e_L, and a second energy value for the second channel, e_R) are calculated as

$$c_{real} = \sum_{i=0}^{M-1} \Re\{L_i \times R_i^*\} \quad (5.3\text{-}243)$$

$$c_{imag} = \sum_{i=0}^{M-1} \Im\{L_i \times R_i^*\} \quad (5.3\text{-}244)$$

$$e_L = \langle L, L\rangle = \sum_{i=0}^{M-1} L_i \times L_i^* \quad (5.3\text{-}245)$$

$$e_R = \langle R, R\rangle = \sum_{i=0}^{M-1} R_i \times R_i^* \quad (5.3\text{-}246)$$

where L and R denote the subframe's FFT spectrum as calculated in EVS [3], clause 5.1.5. The final coherence value is calculated using the real intermediate value, the imaginary intermediate value, the first energy value and the second energy value after smoothing all the intermediate values. The smoothed values are calculated as

$$\overline{c}_{real} = 0.95\,\overline{c}_{real,previous} + 0.05\,c_{real} \quad (5.3\text{-}247)$$

$$\overline{c}_{imag} = 0.95\,\overline{c}_{imag,previous} + 0.05\,c_{imag} \quad (5.3\text{-}248)$$

$$\overline{e}_L = 0.95\,\overline{e}_{L,previous} + 0.05\,e_L \quad (5.3\text{-}249)$$

$$\overline{e}_R = 0.95\,\overline{e}_{R,previous} + 0.05\,e_R \quad (5.3\text{-}250)$$

where $\overline{c}_{real,previous}$, $\overline{c}_{imag,previous}$, $\overline{e}_{L,previous}$ and $\overline{e}_{R,previous}$ denote the respective intermediate values as calculated for the previous frame. If the current frame is the first frame to be encoded or a rate switch has happened from unified stereo, all the previous-frame values are initialized to 10^{-15}. The coherence value is calculated using the smoothed values as

$$c = \sqrt{\frac{\bar{c}_{real}^2 + \bar{c}_{imag}^2}{\bar{e}_L \times \bar{e}_R}} \tag{5.3-251}$$

The coherence value is quantized as

$$c_{ind} = \min(15, \lfloor 15 \times c + 0.5 \rfloor) \in [0,15] \tag{5.3-252}$$

and encoded in the SID bitstream using four bits.

5.3.3.5.2.4 Noise parameter estimation and encoding

Information about the spectral shape of the background noise is derived by a noise parameter calculator that calculates first parametric noise data for the first channel of the stereo signal and second parametric noise data for the second channel of the stereo signal. The parameter calculator consists of the noise estimation algorithm as described in EVS [3] clauses 5.6.3.2. which is run separately on the two channels of the stereo signal to generate two sets of parametric noise data - $N_{L,FD\text{-}CNG}$ for the left channel and $N_{R,FD\text{-}CNG}$ for the right channel. Additionally, the noise parameter calculator is configured to convert the first parametric noise data and second parametric noise data from a left/right representation to a mid/side representation. This conversion is applied after converting the parametric noise data values to dB as

$$N_{L,FD-CNG}^{dB}(i) = 10 \log_{10}\left(N_{L,FD-CNG}(i) + 0.0001\right) \tag{5.3-253}$$

$$N_{R,FD-CNG}^{dB}(i) = 10 \log_{10}\left(N_{R,FD-CNG}(i) + 0.0001\right) \tag{5.3-254}$$

From these, parametric noise data in a mid/side representation are calculated:

$$N_{M,FD-CNG}^{dB} = 0.5\left(N_{L,FD-CNG}^{dB}(i) + N_{R,FD-CNg}^{dB}(i)\right) \tag{5.3-255}$$

$$N_{S,FD-CNG}^{dB} = 0.5\left(N_{L,FD-CNG}^{dB}(i) - N_{R,FD-CNg}^{dB}(i)\right) \tag{5.3-256}$$

This parametric noise data is encoded using the MSVQ described in EVS [3] clause 5.6.3.5 with a modified first stage. For encoding the mid noise data, all 6 stages of the modified MSVQ are used while for the side noise data only the first four are used. The modified first stage MSVQ operation for FD-CNG is described in clause 5.2.2.2.5.1 .

The output of the MSVQ decoder is denoted as $N^{SID}_{M,FD\text{-}CNG}(i)$ and $N^{SID}_{S,FD\text{-}CNG}(i)$, respectively. The MSVQ output is reconverted from the mid/side representation back into a left/right representation. If the energy of the unquantized side noise data $N^{dB}_{S,FD\text{-}CNG}$ is smaller than 0.1, all values of $N^{SID}_{S,FD\text{-}CNG}$ are set to zero prior to reconverting to the left/right representation. A flag (the "no-side" flag) is encoded in the SID bitstream to signal this to the decoder. If the energy of the unquantized side noise data $N_{S,FD\text{-}CNG}$ is smaller than 0.1, the no-side flag is one, otherwise it is zero. The value is encoded using a single bit. Thus, the reconverted parametric noise data for each channel in left/right representation is calculated as

$$N_{L,FD-CNG}^{SID}(i) = \begin{cases} N_{M,FD-CNG}^{SID}(i)\ if\ no-side\ is\ 1, \\ N_{M,FD-CNG}^{SID}(i) + N_{S,FD-CNG}^{SID}(i)\ if\ no-side\ is\ 0. \end{cases} \tag{5.3-257}$$

$$N_{R,FD-CNG}^{SID}(i) = \begin{cases} N_{M,FD-CNG}^{SID}(i)\ if\ no-side\ is\ 1, \\ N_{M,FD-CNG}^{SID}(i) - N_{S,FD-CNG}^{SID}(i)\ if\ no-side\ is\ 0. \end{cases} \tag{5.3-258}$$

From the reconverted left/right representation of the parametric noise data, a first global gain value for the first channel and a second global gain value for the second channel are calculated and quantized as described in <reference to DFT-Stereo DTX part, difference already need to be described there>.

To align the SID frame size to that of other modes, the SID bitstream is padded with 16 zeros after completing the previously described encoding steps.

5.4 Scene-based audio (SBA) operation

5.4.1 SBA format overview

IVAS supports coding of ambisonics up to order 3 at bitrates 13.2 kbps to 512 kbps. At bitrates below 32kbps, the codec supports coding up to SWB bandwidth whereas for bitrates at or above 32kbps, the codec supports coding up to

FB bandwidth. An optional support for planar SBA coding is also provided which limits the coding to only planar SBA channels. IVAS expects SBA input to be SN3D normalized and follow the ACN ordering convention.

5.4.2 Combined DirAC and SPAR based SBA coding overview

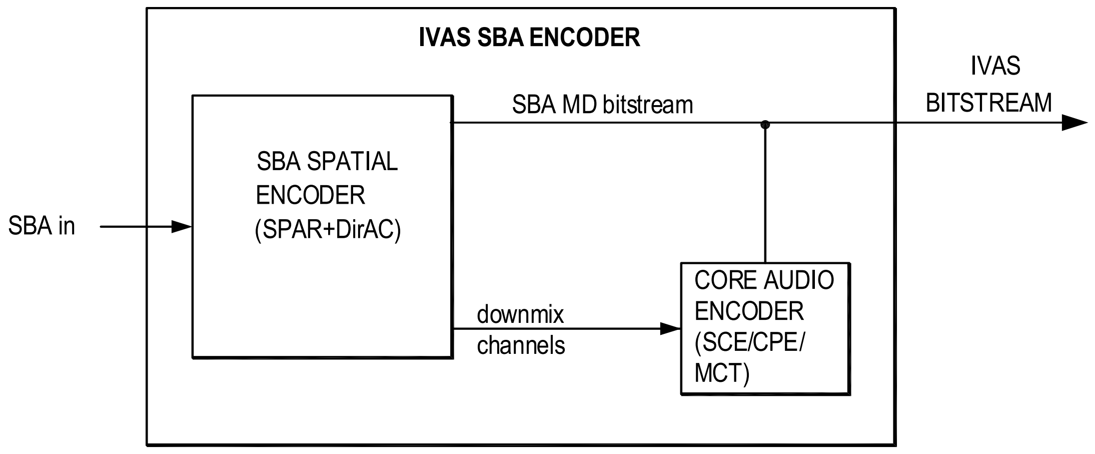


Figure 5.4-1: SBA coding architecture in IVAS

Figure 5.4-1 is a block diagram of SBA coding architecture. The N_{inp} channel high pass filtered (see subclause 5.2.1.1) SBA signal $s^{HP20}(n), n$ being the sample index, is analysed by the SBA spatial encoder block which outputs an SBA metadata (MD) bitstream and N_{dmx} downmix channels. The SBA spatial encoder block uses a combination of SPAR and DirAC spatial analysis methods (as described in subclause 5.4.3) to perform parameter estimation and downmixing.

SPAR seeks to maximize perceived audio quality while minimizing bitrate by reducing the energy of the transmitted audio data while still allowing the second-order statistics of the Ambisonics audio scene (i.e., the covariance) to be reconstructed at the decoder side using transmitted metadata. SPAR seeks to faithfully reconstruct the input Ambisonics scene at the output of the decoder.

Uses DirAC (mention sub section/sub clause) and SPAR (mention sub section/sub clause) approaches to compute spatial metadata.

N channel input is converted to N_{dmx} channel downmix such that N_dmx <= N. This conversion is done using MDFT filterbank.

In order to generate the audio streams for the transport channels over one SCE (Clause 5.2.3.1), one channel CPE (Clause 5.2.3.2), or the MCT (Clause 5.2.3.3), these must be transformed to a suitable representation on the encoder side.

The input ambisonics signal is processed with the MDFT analysis filterbank (see clause 5.2.5) prior to the parameter estimation. The calculation of the downmix matrix and the active downmix are performed individually on each band of a filterbank-domain (TF) representation of the signal. After the downmix, the transport channels (first set) are synthesized to the time domain.

The downmix matrix for the high frequency bands is estimated based on sound field parameter(s). Inter-channel prediction is applied in the downmix based on an inter-channel covariance matrix. This covariance matrix is derived from the sound field parameters. These include a DoA and a global diffuseness parameter.

SPAR and DirAC parameters are computed in the banded domain, as per the frequency banding in the MDFT filterbank (described in subclause 5.2.5). These parameters are then converted into a banded downmix matrix (described in subclause 5.4.4). The SBA spatial encoder block converts the N_{inp} channel SBA signal $s^{HP20}(n)$ into an N_{dmx} channel downmix signal $s^{DMX}(n)$, such that $N_{dmx} \leq N_{inp}$, using the banded downmix matrix. Downmixing occurs in filterbank domain using the MDFT filterbank and the downmixing process is described in subclause 5.4.5. The downmix signal $s^{DMX}(n)$ is coded using the core coding tools described in subclause 5.2.3. The core-coded bitstream and MD bitstream are multiplexed to generate the IVAS bitstream in SBA coding mode. Table 5.4-1 shows the mapping between IVAS bitrate in SBA format, number of downmix channels (N_{dmx}) and core coding tool that is used to code the downmix audio signal $s^{DMX}(n)$. The downmix audio signal $s^{DMX}(n)$ consists of a primary downmix channel W' and zero or more residual channels depending on the value of N_{dmx}.

The SBA MD analysis audio signal and downmix audio signal consists of channels that are selected based on perceptual relevance. Perceptual relevance is ranked based on the following principles:

- First-order channels are ranked more relevant than higher-order channels.
- For a given order l,
    - Channels corresponding to a spherical harmonic Y_l^m (θ, φ) having a larger overlap with the left-right-front-rear plane are ranked more relevant than channels having a larger overlap with the height direction.
    - Channels corresponding to spherical harmonic Y_l^m (θ, φ) having a larger overlap with the left-right direction are ranked more relevant than channels having a larger overlap with the front-rear direction.
- Higher-order planar channels of the same order, where planar channels for a given order l are the channels corresponding to spherical harmonics $Y_l^m(\theta, \varphi)$ for a given order l with $|m| = l$, are ranked more relevant than the non-planar channels of the same order. Moreover, the planar channels of order l are ranked more relevant than the non-planar channels of order l-1.

Table 5.4-1: Mapping between IVAS bitrate, number of downmix channels and core coding tools

Bitrate [kbps]	Number of downmix channels	Core-coding tool
13.2	1	SCE
16.4	1	SCE
24.4	1	SCE
32	1	SCE
48	2	CPE
64	2	CPE
80	2	CPE
96	3	MCT
128	3	MCT
160	3	MCT
192	3	MCT
256	4	MCT
384	4	MCT
512	4	MCT

5.4.2.1 FOA signal coding at all bitrates and HOA2, HOA3 signal coding at bitrates below 256 kbps

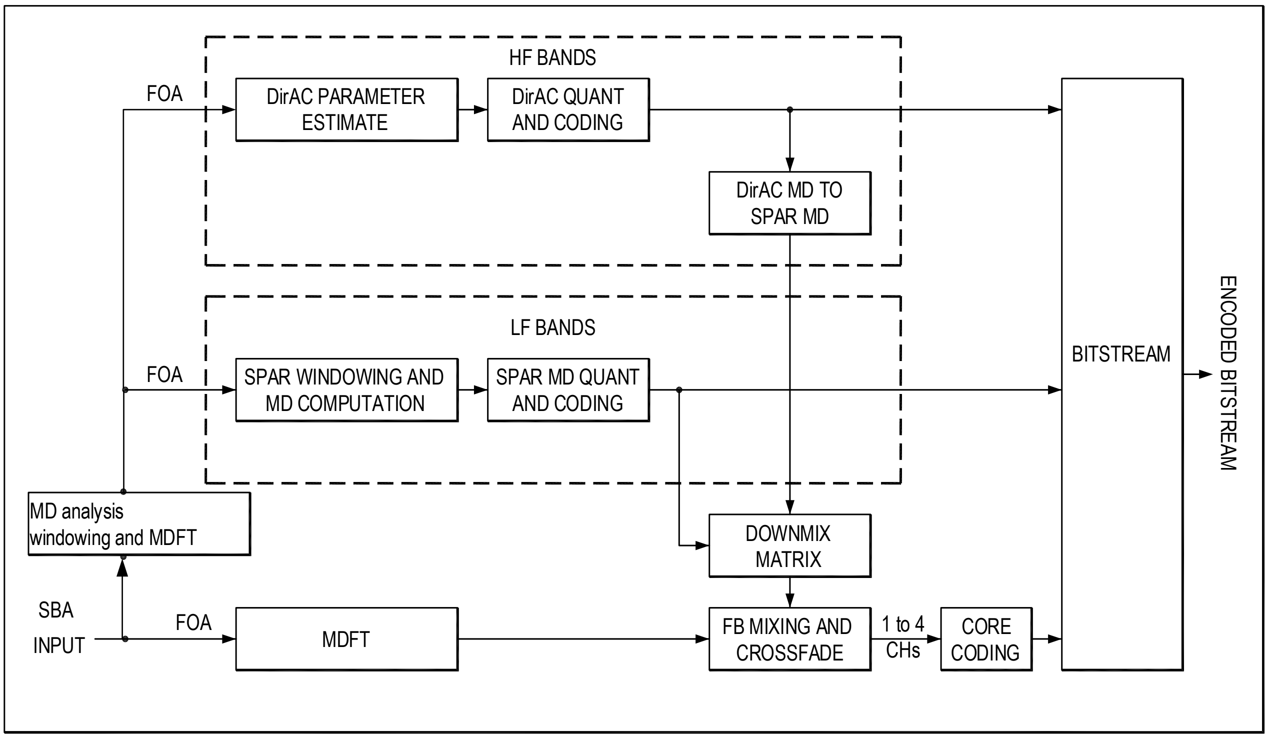


Figure 5.4-2: FOA signal coding at all bitrates and HOA2, HOA3 signal coding at 256 kbps bitrate

Figure 5.4-2 shows the block diagram of SBA coding architecture with FOA input signal at all bitrates and HOA2, HOA3 signal coding at bitrates below 256 kbps. It has two major components, that is, Metadata (MD) parameter analysis and downmixing and core coding. Metadata (MD) parameter analysis includes MD analysis windowing and MDFT, DirAC parameter estimation, DirAC quantization and coding, SPAR MD computation, SPAR MD quantization and coding, DirAC MD to SPAR MD conversion and generation of downmix matrix. Downmixing and core coding includes MDFT on FOA signal, filterbank mixing and crossfading to generate downmix signal and core coding of downmix signals. The signal chain for Figure 5.4-2 is described below.

5.4.2.1.1 Metadata parameter analysis

The FOA component of $s^{HP20}(n)$ is windowed using the metadata analysis window (as described in subclause 5.4.3.3) and transformed to a frequency domain signal S'^{MDFT}_i, where $i = 1,..,4$ is the channel index, using MDFT (as described in subclause 5.4.3.3). The high frequency bins, in the frequency range f^{DirAC}, such that f^{DirAC} > 4.4 KHz, of S'^{MDFT}_i are processed by DirAC parameter estimation block (described in subclause 5.4.3.4) which generates DirAC MD parameters in the f^{DirAC} frequency range, quantizes and codes these parameters into DirAC MD bitstream. The low frequency bins, in the frequency range f^{SPAR}, such that f^{SPAR} <= 4.4 KHz, of S'^{MDFT}_i are processed by SPAR parameter estimation block (described in subclause 0) which generates SPAR MD parameters in the f^{SPAR} frequency range, quantizes and codes these parameters into SPAR MD bitstream. The DirAC MD parameters in f^{DirAC} frequency range are converted into SPAR parameters MD using the DirAC MD to SPAR MD converter (described in subclause 5.4.3.7.1). SPAR MD in f^{SPAR} and f^{DirAC} frequency ranges is combined and is converted into a downmix matrix $DMX_{[N_{dmx}\,x\,4]}$, as described in subclause 5.4.4.

5.4.2.1.2 Downmixing and core coding

The FOA component of $s^{HP20}(n)$ is windowed using a 40ms window, as described in subclause 5.4.5.1, and transformed into a frequency domain signal S_i^{MDFT}, where $1 \leq i \leq 4$ is the channel index, using the MDFT. The frequency-domain signal S_i^{MDFT} is then processed by the filter bank (FB) mixing and crossfade block which generates the downmix signal $s_i^{DMX}(n), 1 \leq i \leq N_{dmx}$, by applying the downmix matrix $DMX_{[N_{dmx}\,x\,4]}$ to S_i^{MDFT} in the filter bank domain. The downmix generation process is explained in detail in subclause 5.4.5. The downmix signal $s_i^{DMX}(n)$

is coded with either SCE, CPE or MCT core coding tool as per Table 5.4-1. IVAS bitstream is generated with coded SPAR metadata, coded DirAC metadata and coded downmix signal.

5.4.2.2 HOA2 and HOA3 signal coding overview at 256 kbps

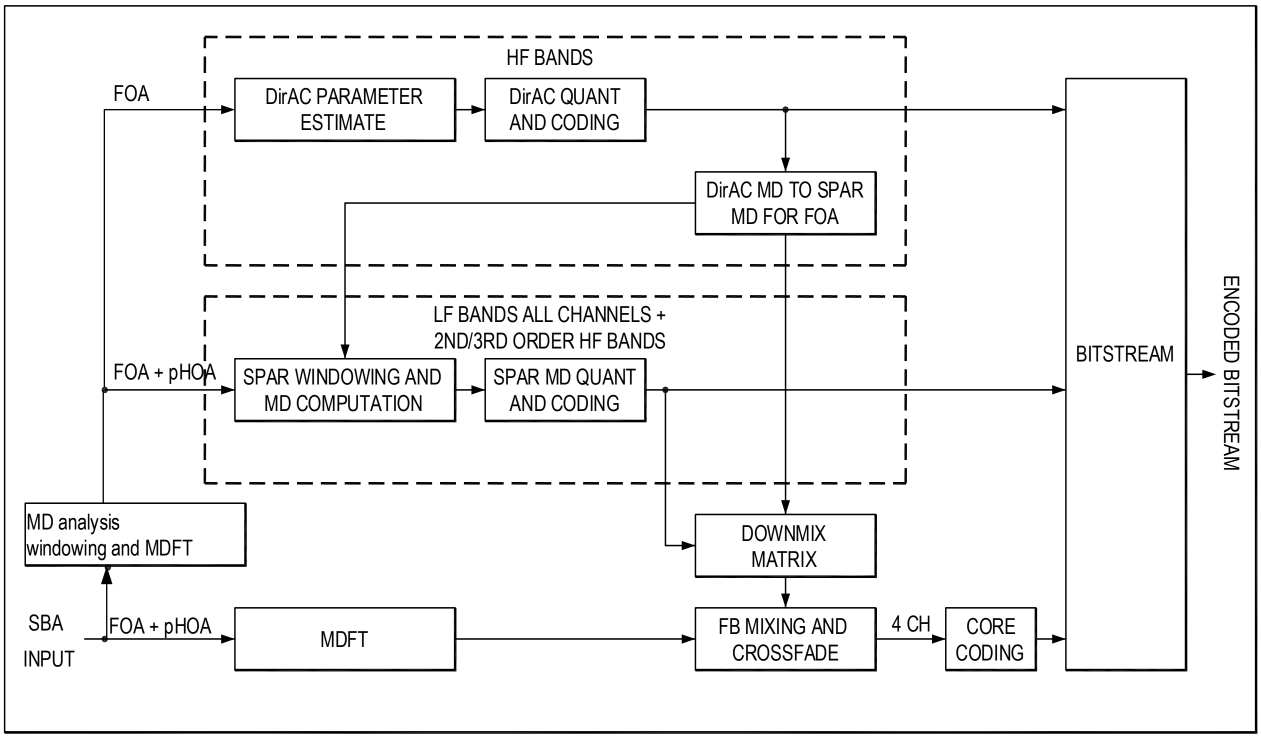


Figure 5.4-3: HOA2 and HOA3 signal coding at 256 kbps bitrate

Figure 5.4-3 shows the block diagram of SBA coding architecture with HOA2 and HOA3 signal coding at 256 kbps.

5.4.2.2.1 Metadata parameter analysis

The FOA and planar HOA components of $s^{HP20}(n)$ are windowed using the metadata analysis window (see subclause 5.4.3.3) and transformed to a frequency domain signal S'^{MDFT}_{i} using the MDFT, where i is the channel index such that $1 \leq i \leq N_{sba\_ana}$ and N_{sba_ana} is the number of FOA + planar HOA channels in the input SBA signal. $N_{sba\_ana} = 6$ for HOA2 input and $N_{sba\_ana} = 8$ for HOA3 input. Herein, planar channels for a given ambisonics order l are defined as channels corresponding to spherical harmonics $Y_l^m(\theta, \varphi)$ for a given order l with $|m| = l$. The high frequency bins in the frequency range f^{DirAC} > 4.4 KHz of the FOA component of S'^{MDFT}_{i} are processed by the DirAC parameter estimation block (described in subclause 5.4.3.4), which generates DirAC MD parameters in the f^{DirAC} frequency range, quantizes and codes these parameters into DirAC MD bitstream. All frequency bins of S'^{MDFT}_{i} are processed by the SPAR parameter estimation block (described in subclause 0) which generates FOA SPAR MD parameters in the frequency range f^{SPAR} <= 4.4 KHz, HOA SPAR parameters in both f^{SPAR} and f^{DirAC} frequency ranges, quantizes and codes these parameters into SPAR MD bitstream. The DirAC MD parameters in f^{DirAC} frequency range are converted into FOA SPAR parameters using the DirAC MD to SPAR MD converter (described in subclause 5.4.3.7.1). SPAR MD, corresponding to FOA and HOA channels, in f^{SPAR} and f^{DirAC} frequency ranges is combined and is converted into a downmix matrix $DMX_{[N_{dmx}\, x\, N_{sba\_ana}]}$ as described in subclause 5.4.4.

5.4.2.2.2 Downmixing and core coding

The FOA and pHOA component of $s^{HP20}(n)$ is windowed using a 40ms window as described in subclause 5.4.5.1 and transformed into a frequency domain signal S_i^{MDFT}, where $1 \leq i \leq N_{sba_{ana}}$ is the channel index, using the MDFT. S_i^{MDFT} is then processed by the filterbank (FB) mixing and crossfade block which generates the downmix signal $s_i^{DMX}(n), 1 \leq i \leq N_{dmx}$, by applying the downmix matrix $DMX_{[N_{dmx}\, x\, N_{sba\_ana}]}$ to S_i^{MDFT} in the filterbank domain. The downmix generation process is explained in detail in subclause 5.4.5. The downmix signal $s_i^{DMX}(n)$ is coded with either SCE, CPE or MCT core coding tool depending on the value of N_{dmx} as per Table 5.4-1. . IVAS bitstream is generated with coded SPAR metadata, coded DirAC metadata and coded downmix signal.

5.4.2.3 HOA2 and HOA3 signal coding overview at 384 kbps

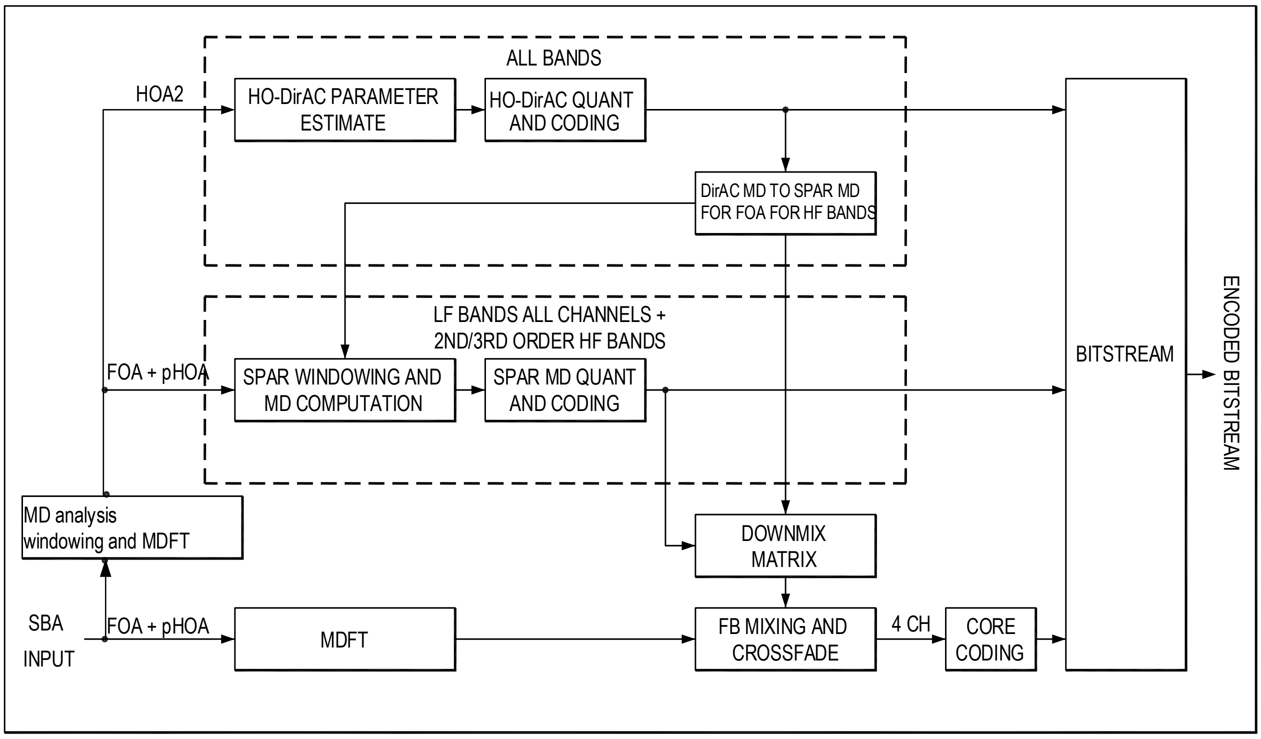

ALL BANDS
HOA2
HO-DirAC PARAMETER ESTIMATE
HO-DirAC QUANT AND CODING
DirAC MD TO SPAR MD FOR FOA FOR HF BANDS
LF BANDS ALL CHANNELS + 2ND/3RD ORDER HF BANDS
FOA + pHOA
SPAR WINDOWING AND MD COMPUTATION
SPAR MD QUANT AND CODING
BITSTREAM
ENCODED BITSTREAM
MD analysis windowing and MDFT
DOWNMIX MATRIX
SBA INPUT
FOA + pHOA
MDFT
FB MIXING AND CROSSFADE
4 CH
CORE CODING

Figure 5.4-4: HOA2 and HOA3 signal coding at 384 kbps bitrate

Figure 5.4-4shows the block diagram of SBA coding architecture with HOA2 and HOA3 signal coding at 384 kbps.

In this mode, SPAR analysis is done on FOA and planar HOA channels and HO-DirAC analysis is done on HOA2 channels. The DirAC to SPAR MD conversion is limited to FOA channels in this mode. The signal chain for Figure 5.4-4 is described below.

5.4.2.3.1 Metadata parameter generation

The HOA2 and planar HOA3 components of $s^{HP20}(n)$ are windowed using the metadata analysis window and transformed to a frequency domain signal S'^{MDFT}_{i} using the MDFT, where i is the channel index such that $1 \leq i \leq N_{sba\_ana}$, and N_{sba_ana} is the number of HOA2 + planar HOA3 channels in the input SBA signal. The number of analysis channels $N_{sba\_ana} = 9$ for HOA2 input and $N_{sba\_ana} = 11$ for HOA3 input. Herein, planar channels for a given ambisonics order l are defined as channels corresponding to spherical harmonics $Y_l^m(\theta, \varphi)$ for a given order l with $|m| = l$. All frequency bins in the HOA2 component of S'^{MDFT}_{i} signal is processed by DirAC parameter estimator block which generates DirAC MD parameters, quantizes and codes these parameters into DirAC MD bitstream. All frequency bins of the FOA and planar HOA component of S'^{MDFT}_{i} are processed by SPAR parameter estimator block which generates FOA SPAR MD parameters in the f^{SPAR} frequency range (f^{SPAR} <= 4.4 KHz), HOA SPAR parameters in both f^{SPAR} and f^{DirAC} frequency ranges, quantizes and codes these parameters into SPAR MD bitstream. The DirAC MD parameters in f^{DirAC} frequency range are converted into FOA SPAR parameters MD using DirAC MD to SPAR MD converter. SPAR MD, corresponding to FOA and HOA channels, in f^{SPAR} and f^{DirAC} frequency ranges is combined and is converted into a downmix matrix $DMX_{[N_{dmx}\, x\, N_{spar\_ana}]}$, where N_{spar_ana} is the number of FOA + planar HOA channels in the input SBA signal, $N_{spar\_ana} = 6$ for HOA2 input and $N_{spar\_ana} = 8$ for HOA3 input.

5.4.2.3.2 Downmixing and core coding

The FOA and pHOA components of $s^{HP20}(n)$ are windowed using a 40ms window and transformed into a frequency domain signal S_i^{MDFT}, where $1 \leq i \leq N_{spar\_ana}$ is the channel index, using the MDFT. $S_i^{MDFT}$is then processed by the filter bank (FB) mixing and crossfade block which generates the downmix signal $s_i^{DMX}(n), 1 \leq i \leq N_{dmx}$, by applying the downmix matrix $DMX_{[N_{dmx}\, x\, N_{spar\_ana}]}$ to S_i^{MDFT} in the filter bank domain. The downmix signal $s_i^{DMX}(n)$ is coded with either SCE, CPE or MCT core coding tool depending on the value of N_{dmx}, as per Table 5.4-1. IVAS bitstream is generated with coded SPAR metadata, coded DirAC metadata and coded downmix signal.

5.4.2.4 HOA2 and HOA3 signal coding overview at 512 kbps

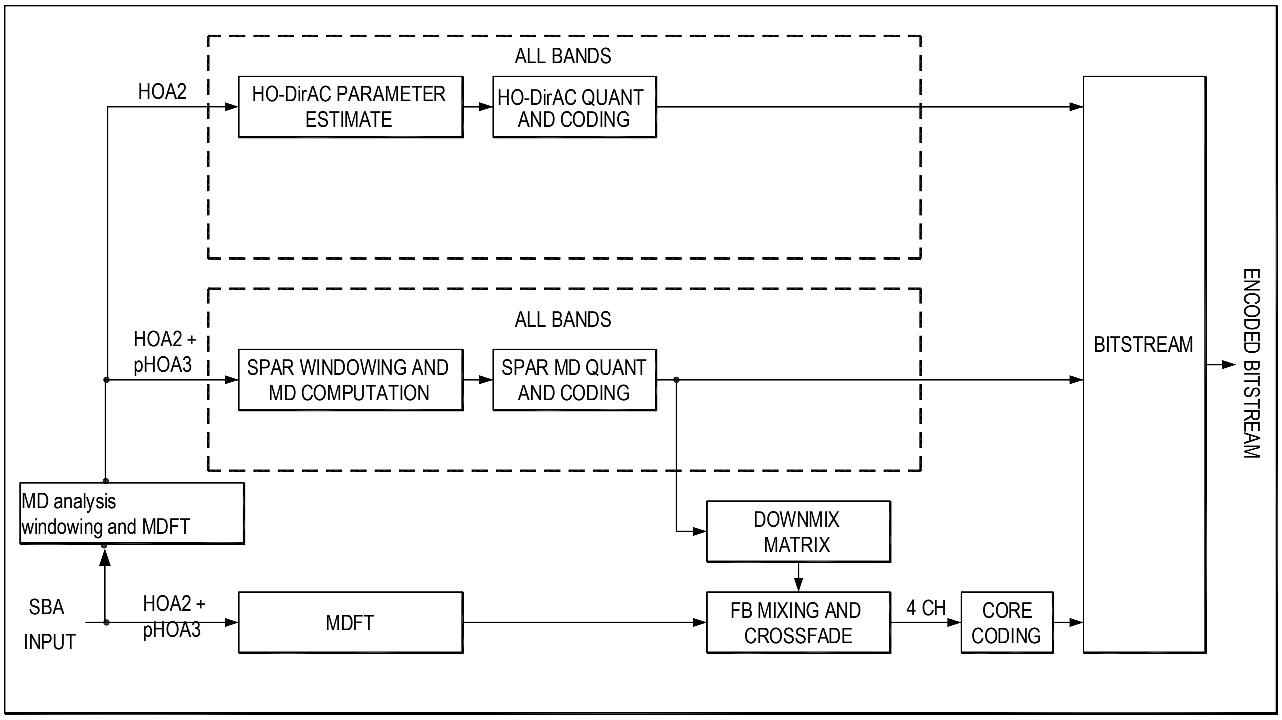


Figure 5.4-5: HOA2 and HOA3 signal coding at 512 kbps bitrate

Figure 5.4-5 shows the block diagram of SBA coding architecture with HOA2 and HOA3 signal coding at 512 kbps. In this mode, SPAR analysis is done on HOA2 and planar HOA3 channels and HO-DirAC analysis is done on HOA2 channels. The DirAC to SPAR MD conversion is not done in this mode. The signal chain is described below.

5.4.2.4.1 Metadata parameter generation

The HOA2 and planar HOA3 components of $s^{HP20}(n)$ are windowed using the metadata analysis and transformed to a frequency domain signal S'^{MDFT}_i using the MDFT, where i is the channel index such that $1 \leq i \leq N_{sba\_ana}$, and N_{sba_ana} is the number of HOA2 + planar HOA3 channels in the input SBA signal, $N_{sba\_ana} = 9$ for HOA2 input and $N_{sba\_ana} = 11$ for HOA3 input. Herein, planar channels for a given ambisonics order l are defined as channels corresponding to spherical harmonics $Y_l^m(\theta, \varphi)$ for a given order l with $|m| = l$. All frequency bins in the HOA2 component of S'^{MDFT}_i signal is processed by DirAC parameter estimator block which generates DirAC MD parameters, quantizes and codes these parameters into DirAC MD bitstream. All frequency bins of S'^{MDFT}_i are processed by SPAR parameter estimator block which generates SPAR MD parameters in both f^{SPAR} and f^{DirAC} frequency ranges, quantizes and codes these parameters into SPAR MD bitstream. DirAC MD to SPAR MD conversion is not needed at this bitrate as SPAR parameters corresponding to all analysis channels and entire frequency spectrum are generated by SPAR parameter estimator block. SPAR MD is then converted into a downmix matrix $DMX_{[N_{dmx}\, x\, N_{sba\_ana}]}$.

5.4.2.4.2 Downmixing and core coding

The HOA2 and planar HOA3 components of $s^{HP20}(n)$ are windowed using a 40 ms and transformed into a frequency domain signal S_i^{MDFT}, where $1 \leq i \leq N_{sba\_ana}$ is the channel index, using the MDFT. S_i^{MDFT} is then processed by the filterbank (FB) mixing and crossfade block which generates the downmix signal $s_i^{DMX}(n), 1 \leq i \leq N_{dmx}$, by applying the downmix matrix $DMX_{[N_{dmx}\, x\, N_{sba\_ana}]}$ to S_i^{MDFT} in the filterbank domain. The downmix signal $s_i^{DMX}(n)$ contains the primary downmix channel. The downmix signal $s_i^{DMX}(n)$ is coded with either SCE, CPE or MCT core coding tool depending on the value of N_{dmx}, as per Table 5.4-1. IVAS bitstream is generated with coded SPAR metadata, coded DirAC metadata and coded downmix signal.

5.4.3 SBA parameter estimation

5.4.3.1 SBA front VAD

5.4.3.2 Transient Detection

Transient detection makes use of the transient processor that is part of the time-domain decorrelator described in subclause 6.2.1.2.1. The transient processor outputs a time-varying envelope $e_{in,k}[n]$ for the current audio frame k of length N that is used by the decorrelator to separate the transient and non-transient components of the input signal. From this envelope, transients are detected by comparing changes in this envelope against a fixed threshold. Two boolean signals for transient detection, T_0 and T_1 are defined, operating on different time scales.

$$\mathrm{T}_0 = \begin{cases} 1, & \mathrm{e}_{in,k-1}[N-1] - \mathrm{e}_{in,k}[N-1] > 0.1 \\ 0, & \text{otherwise} \end{cases} \tag{5.4-1}$$

To compute T_1, the frame k is subdivided into 16 subframes $m = 0\,..\,15$, such that $e_{in,k,m}[n] = e_{in,k}[\frac{Nm}{16} + n]$.

$$\mathrm{T}_1 = \begin{cases} 1, & \mathrm{e}_{in,k,m-1}\left[\frac{N}{16} - 1\right] - \mathrm{e}_{in,k,m}[\frac{N}{16} - 1] > 0.11, \mathrm{m} \in \{1\,..\,15\} \\ 1, & e_{in,k-1,15}\left[\frac{N}{16} - 1\right] - e_{in,k,0}\left[\frac{N}{16} - 1\right] > 0.11 \\ 0, & \text{otherwise} \end{cases} \tag{5.4-2}$$

The main difference between these two transient detection signals is that T_0 samples the transient processor envelope once per 20 ms frame whilst T_1 samples the envelope 16 times per frame (every 1.25 ms). By sampling more often, T_1 is less sensitive to weak transients than T_0.

5.4.3.3 Analysis windowing and MDFT

The SBA metadata analysis signal $s_j^{sba\_ana}(n)$, where $1 \le j \le N_{sba\_ana}$ and N_{sba_ana} is the total number of SBA analysis channels, is formed by selecting a subset of channels from $s_i^{HP20}(n)$ signal, where $1 \le i \le (ambisonics_{order} + 1)^2$. Table 5.4-2 shows the subset of channel indices i that forms the signal $s_j^{sba\_ana}(n)$ and the subset of channel indices i that are used in DirAC and SPAR metadata analysis.

Table 5.4-2: SBA metadata analysis channels at various IVAS bitrates

IVAS bitrate [(kbps]	Input ambisonics order	SBA analysis order	SBA Metadata analysis channel indices	DirAC Metadata analysis channel indices	SPAR Metadata analysis channel indices
13.2 – 192	1	1	1, 2, 3, 4	1, 2, 3, 4	1, 2, 3, 4
	2	1	1, 2, 3, 4	1, 2, 3, 4	1, 2, 3, 4
	3	1	1, 2, 3, 4	1, 2, 3, 4	1, 2, 3, 4
256	1	1	1, 2, 3, 4	1, 2, 3, 4	1, 2, 3, 4
	2	2	1, 2, 3, 4, 5, 9	1, 2, 3, 4	1, 2, 3, 4, 5, 9
	3	3	1, 2, 3, 4, 5, 9, 10, 16	1, 2, 3, 4	1, 2, 3, 4, 5, 9, 10, 16
384	1	1	1, 2, 3, 4	1, 2, 3, 4	1, 2, 3, 4
	2	2	1, 2, 3, 4, 5, 6, 7, 8, 9	1, 2, 3, 4, 5, 6, 7, 8, 9	1, 2, 3, 4, 5, 9
	3	3	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 16	1, 2, 3, 4, 5, 6, 7, 8, 9	1, 2, 3, 4, 5, 9, 10, 16
512	1	1	1, 2, 3, 4	1, 2, 3, 4	1, 2, 3, 4
	2	2	1, 2, 3, 4, 5, 6, 7, 8, 9	1, 2, 3, 4, 5, 6, 7, 8, 9	1, 2, 3, 4, 5, 6, 7, 8, 9
	3	3	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 16	1, 2, 3, 4, 5, 6, 7, 8, 9	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 16

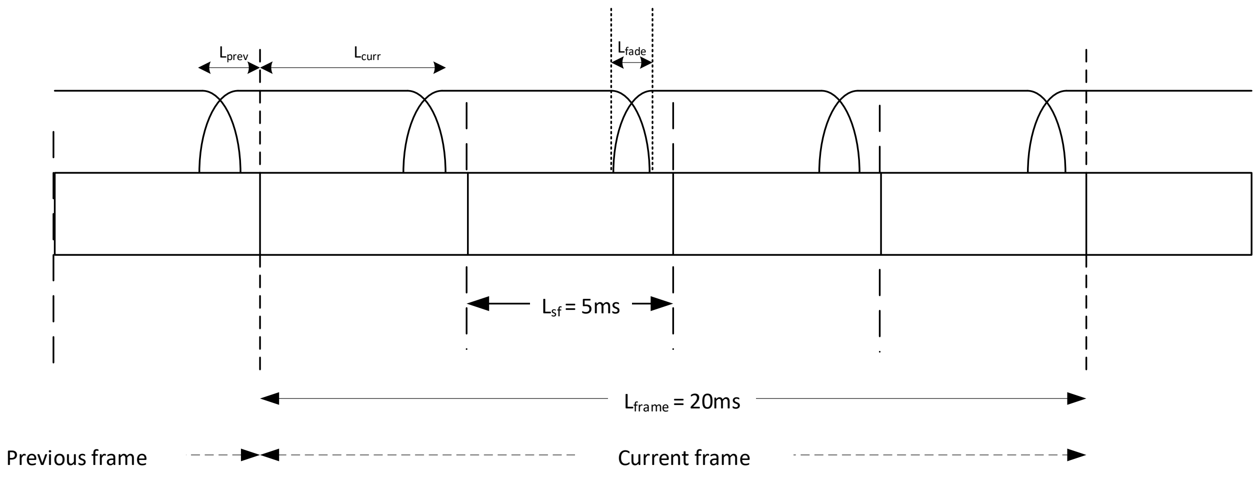
Every frame of $s_j^{sba\_ana}(n)$ is divided into 4 subframes and each subframe is windowed with a sinusoidal window. Figure
5.4-6 shows the subframe windowing and MDFT. The input to MDFT is a 10ms long time domain signal $s_j^{sba_{ana},mdft}(m,n)$, where m is the subframe index with $1 \le m \le 4$ and n is the sample index with $1 \le n \le \frac{10\,samplingfrequency}{1000}$, that is formed by concatenating two consecutive subframes of $s_j^{sba\_ana}$ and then applying the sin window function as shown in eq. (5.4-3), (5.4-4) and (5.4-5).

$$s_j^{sba_{ana},mdft}(m,n) = s'^{sba_{ana}}_j(m,n)Win^{sba_{ana}}(n) \ , \tag{5.4-3}$$

$$s'^{sba_{ana}}_j(m,n) = \begin{cases} 0, & 1 \le n \le (L_{sf} - L_{prev} + L_{offset}) \\ s_j^{sba_{ana}}(m-1, n - L_{offset}), & (L_{sf} - L_{prev} + L_{offset}) < n \le (L_{sf} + L_{offset}) \\ s_j^{sba_{ana}}(m, n - L_{sf} - L_{offset}), & (L_{sf} + L_{offset}) < n \le 2L_{sf} \end{cases} \ , \tag{5.4-4}$$

$$Win^{sba_{ana}}(n) = \begin{cases} 0, & 1 \le n \le (L_{sf} - L_{prev} + L_{offset}) \\ \sin^2\left(\frac{n-(L_{sf}-L_{prev}+L_{offset})}{L_{fade}} - \frac{1}{2L_{fade}}\right), & (L_{sf} - L_{prev} + L_{offset}) < n \le (L_{sf} - L_{prev} + L_{offset} + L_{\square\square ade}) \\ 1, & (L_{sf} - L_{prev} + L_{offset} + L_{fade}) < n \le (2L_{sf} - L_{fade}) \\ 1 - \sin^2\left(\frac{n-(2L_{sf}-L_{fade})}{L_{fade}} - \frac{1}{2L_{fade}}\right), & (2L_{sf} - L_{fade}) < n \le 2L_{sf} \end{cases} \tag{5.4-5}$$

where $L_{frame} = 20L_{ms}$, $L_{sf} = 5L_{ms}$, $L_{prev} = 1.5L_{ms}$, $L_{curr} = 4.5L_{ms}$, $L_{fade} = L_{ms}$, $L_{offset} = 0.5L_{ms}$, $L_{ms} = \frac{samplingfrequency}{1000}$, m is the subframe index with $1 \le m \le 4$ and $s_j^{sba_{ana}}(0,n)$ is the 4th subframe of previous frame.



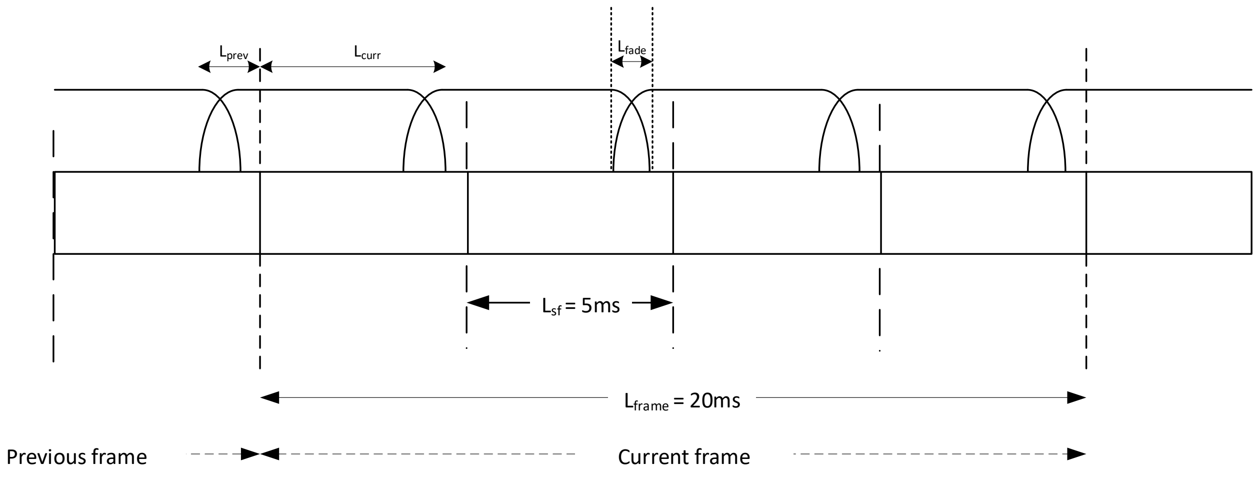


Figure 5.4-6: Subframe windowing and MDFT

For each m, $s_j^{sba_{ana},mdft}(m,n)$ is converted to frequency domain signal $S_j^{sba_{ana},mdft}(m,k)$ by applying the MDFT (as described in subclause 5.2.5.1) to $s_j^{sba_{ana},mdft}(m,n)$, where m is the subframe index and k is the frequency bin index.

5.4.3.4 DirAC parameter estimation

On the encoder side, psycho-acoustic sound field parameters (DirAC) are estimated from the input ambisonics audio channels in the time-frequency (TF) domain of the MDFT filter bank in the parameter estimation path (cf. clause xxx) These parameters comprise one or more direction(s) of arrival (DoA) and one or more diffuseness parameters per frequency sub-band. The time resolution depends on the parameters: for DoAs a time resolution of 5ms is employed, while for the diffuseness parameters a time resolution of 20ms is used instead. The frequency resolution is defined by 12 parameter groups (see below).

Using these parameters, an ambisonics input signal (of order 1, 2 or 3) is encoded into the compressed representation comprising audio downmix channels and side information. The audio downmix channels (the first set of transport channels as decoded in clause xxx) are generated using the SPAR downmix according to clause xxx based on the SPAR and DirAC metadata.

The encoder unit supports a high-order and a low-order operation mode. It can switch between these modes dynamically based on the bitrate. For the bitrates 384 and 512 kbps, the high-order operation mode is enabled. At all other bitrates the low-order operation mode is active.

The difference between these modes is the number of spatial sectors in the DirAC parameter estimation. In the high-order mode, both sector parameter estimation paths are activated. In the low-order operation mode, one of them is deactivated. The other one processes the input signal with an omni-directional sector filter to cover the sound field on the whole sphere. The high-order mode requires an input signal of at least order 2. Order 3 can be configured as input, but the DirAC encoder does not evaluate the channels of the third order. If a lower input order is configured, the encoder falls back to the low-order mode, even at the two aforementioned high bitrates. This SBA analysis order is transmitted to the decoder in the bitstream to configure the correct decoding method.

5.4.3.4.1 High-order operation mode

The side information derived by the encoder includes sound field parameters representing the scene described by the second- or third-order ambisonics input signal. These sound field parameters comprise a DoA and a sector diffuseness parameter for each of the two spatial sectors.

For the estimation of the parameters, for each spatial sector, the sector signal is generated by means of spatial filtering (segmentation) of the input ambisonic signal $\boldsymbol{x}_{HOA}$ (cf. Figure 5.4-7):

$$\boldsymbol{x_S} = \boldsymbol{w}_s^T\, \boldsymbol{x}_H \qquad (5.4\text{-}6)$$

Where $\boldsymbol{x_S}$ is the vector of the spatially filtered (segmental) first-order ambisonics signal in sector s. $\boldsymbol{w}_s^T$ is the matrix projecting the vector on the sector beamforming weights with the entries according to Table 5.4-3 for sector 0 and Table 5.4-4 for sector 1, respectively. It can be viewed as the projector onto the beams describing the sector $\boldsymbol{b}_s$. The beams $\boldsymbol{b}_s$ describe the directivity pattern of the spatial filter that belongs to the sector s.

Table 5.4-3: XX

	w (0,0)	x (1, -1)	y (1,0)	z (1,1)	(2, -2)	(2, -1)	(2,0)	(2,1)	(2,2)
w (0,0)	1.772454	1.772454	0	0	0	0	0	0	0
x (1, -1)	0	0	1.772454	0	1.023326	0	0	0	0
y (1,0)	0.590818	1.772454	0	0	0	0	- 0.590817	0	- 1.023326
z (1,1)	0	0	0	1.772454	0	1.023326	0	0	0

Table 5.4-4: XX

	w (0,0)	x (1, -1)	y (1,0)	z (1,1)	(2, -2)	(2, -1)	(2,0)	(2,1)	(2,2)
w (0,0)	1.772454	-1.772454	0	0	0	0	0	0	0
x (1, -1)	0	0	1.772454	0	-1.023326	0	0	0	0
y (1,0)	-0.590818	1.772454	0	0	0	0	0.590817	0	1.023326
z (1,1)	0	0	0	1.772454	0	-1.023326	0	0	0

To take full advantage of the sparse structure of the matrix $\boldsymbol{w}_s^T$ for complexity reduction, in the reference implementation, the matrix multiplication is explicitly written out as the individual multiplications and additions with the non-zero matrix elements.

Each of these sector signals is then fed into a sector parameter estimator (parameter generator, see Figure 5.4-7: SBA encoder block diagram in high-order operation mode), which derives the parameters for the sector, a sector DoA and a sector diffuseness.

The DoA is derived based on the pseudo-intensity vector, whose components are given by

$$\mathrm{I}_x^{\mathrm{j}} = \sum_{i=0}^{N_{\mathrm{MDFT}}} w_i^j \{\mathrm{Re}(\mathrm{x}_w^i) * \mathrm{Re}(\mathrm{x}_\mathrm{x}^i) + \mathrm{Im}(\mathrm{x}_w^i) * \mathrm{Im}(\mathrm{x}_\mathrm{x}^i)\}, \quad (5.4\text{-}7)$$

$$\mathrm{I}_\mathrm{y}^{j} = \sum_{i=0}^{N_{\mathrm{MDFT}}} w_i^j \{\mathrm{Re}(\mathrm{x}_w^i) * \mathrm{Re}(\mathrm{x}_\mathrm{y}^i) + \mathrm{Im}(\mathrm{x}_w^i) * \mathrm{Im}(\mathrm{x}_\mathrm{y}^i)\}, \quad (5.4\text{-}8)$$

$$\mathrm{I}_\mathrm{z}^{\mathrm{j}} = \sum_{i=0}^{N_{\mathrm{MDFT}}} w_i^j \{\mathrm{Re}(\mathrm{x}_w^i) * \mathrm{Re}(\mathrm{x}_\mathrm{z}^i) + \mathrm{Im}(\mathrm{x}_w^i) * \mathrm{Im}(\mathrm{x}_\mathrm{z}^i)\} \quad (5.4\text{-}9)$$

The $\mathrm{x}_{w;x;y;z}^i$ are the w;x;y;z components of the sector signal $\boldsymbol{x}_s$ in the MDFT bin with index i. The contributions of the MDFT bins are weighted by the numbers w_i^j to achieve a grouping of the bins into parameter bands. w_i^j is the weight of the bin i in the group j. This results in one intensity vector for each parameter band indexed by j.

The choice of the groups is perceptually motivated. The frequency resolution of the model parameters decreases at higher frequencies. The choice of the groups is aligned with the grouping of the CLDFB bands on the decoder side and the frequency responses of the bands of the MDFT-based filter bank in the signal path (cf. xxx and xxx).

The groups are defined according to Table 5.4-5.

Table 5.4-5: XX

Group index	0	1	2	3	4	5	6	7	8	9	10	11
Start bin	0	4	8	12	16	20	24	28	44	68	100	240

The weights w_i^j are non-zero only where the bin index i falls into the range of the group j. As each bin falls into exactly one group, there is exactly one weight to store in a table for each bin. The values are listed in Table 5.4-6.

Table 5.4-6: XX

9.962447e-02 9.627997e-01 9.926667e-01 9.981028e-01 9.996648e-01 1.000000e+00 9.997692e-01 9.992002e-01 9.983890e-01

9.973818e-01 9.962037e-01 9.948692e-01 9.933876e-01 9.917654e-01 9.900073e-01 9.881169e-01 9.860975e-01 9.839516e-01

9.816818e-01 9.792906e-01 9.767801e-01 9.741527e-01 9.714106e-01 9.685560e-01 9.655913e-01 9.625187e-01 9.593406e-01

9.560594e-01 9.526774e-01 9.491970e-01 9.456208e-01 9.419512e-01 9.381908e-01 9.343420e-01 9.304075e-01 9.263898e-01

9.222915e-01 9.181152e-01 9.138636e-01 9.095392e-01 9.051447e-01 9.006827e-01 8.961559e-01 8.915668e-01 8.869181e-01

8.822123e-01 8.774521e-01 8.726400e-01 8.677785e-01 8.628702e-01 8.579176e-01 8.529231e-01 8.478893e-01 8.428184e-01

8.377130e-01 8.325753e-01 8.274077e-01 8.222124e-01 8.169917e-01 8.117478e-01 8.064829e-01 8.011990e-01 7.958982e-01

7.905827e-01 7.852543e-01 7.799150e-01 7.745667e-01 7.692112e-01 7.638505e-01 7.584861e-01 7.531199e-01 7.477535e-01

7.423885e-01 7.370265e-01 7.316691e-01 7.263176e-01 7.209736e-01 7.156384e-01 7.103134e-01 7.049999e-01 6.996990e-01

6.944121e-01 6.891403e-01 6.838847e-01 6.786464e-01 6.734265e-01 6.682258e-01 6.630455e-01 6.578863e-01 6.527492e-01

6.476350e-01 6.425445e-01 6.374784e-01 6.324376e-01 6.274226e-01 6.224341e-01 6.174729e-01 6.125393e-01 6.076341e-01

6.027577e-01 5.979106e-01 5.930932e-01 5.883061e-01 5.835497e-01 5.788242e-01 5.741301e-01 5.694676e-01 5.648372e-01

5.602390e-01 5.556734e-01 5.511404e-01 5.466405e-01 5.421737e-01 5.377402e-01 5.333402e-01 5.289738e-01 5.246411e-01

5.203422e-01 5.160771e-01 5.118460e-01 5.076489e-01 5.034858e-01 4.993567e-01 4.952616e-01 4.912005e-01 4.871734e-01

4.831802e-01 4.792209e-01 4.752955e-01 4.714037e-01 4.675457e-01 4.637212e-01 4.599302e-01 4.561725e-01 4.524481e-01

4.487567e-01 4.450983e-01 4.414728e-01 4.378799e-01 4.343195e-01 4.307915e-01 4.272956e-01 4.238318e-01 4.203997e-01

4.169993e-01 4.136303e-01 4.102926e-01 4.069859e-01 4.037101e-01 4.004649e-01 3.972501e-01 3.940655e-01 3.909109e-01

3.877861e-01 3.846909e-01 3.816250e-01 3.785882e-01 3.755803e-01 3.726010e-01 3.696501e-01 3.667275e-01 3.638328e-01

3.609658e-01 3.581263e-01 3.553141e-01 3.525289e-01 3.497705e-01 3.470387e-01 3.443331e-01 3.416537e-01 3.390001e-01

3.363720e-01 3.337694e-01 3.311919e-01 3.286393e-01 3.261114e-01 3.236079e-01 3.211286e-01 3.186733e-01 3.162418e-01

3.138337e-01 3.114490e-01 3.090872e-01 3.067484e-01 3.044321e-01 3.021382e-01 2.998664e-01 2.976166e-01 2.953885e-01

2.931819e-01 2.909966e-01 2.888323e-01 2.866889e-01 2.845661e-01 2.824637e-01 2.803816e-01 2.783194e-01 2.762770e-01

2.742543e-01 2.722509e-01 2.702667e-01 2.683014e-01 2.663550e-01 2.644271e-01 2.625177e-01 2.606264e-01 2.587531e-01

2.568977e-01 2.550599e-01 2.532395e-01 2.514364e-01 2.496503e-01 2.478811e-01 2.461287e-01 2.443928e-01 2.426732e-01

2.409698e-01 2.392824e-01 2.376109e-01 2.359550e-01 2.343146e-01 2.326895e-01 2.310797e-01 2.294848e-01 2.279047e-01

2.263394e-01 2.247886e-01 2.232521e-01 2.217299e-01 2.202217e-01 2.187274e-01 2.172469e-01 2.157800e-01 2.143266e-01

2.128865e-01 2.114596e-01 2.100457e-01 2.086447e-01 2.072564e-01 2.058808e-01

The fast-varying vector components in Equation (5.6-5) are then smoothened with an averaging filter with an update rate of $\beta = 0.2$ to obtain the smooth versions $\bar{I}^{j}_{w;x;y;z}$. The application of the inverse trigonometric function then yields the DoA angles azimuth

$$\phi^{j} = \arctan\left(\frac{\bar{I}^{j}{}_{y}}{\bar{I}_{w}}\right), \qquad (5.4\text{-}10)$$

and elevation

$$\theta^{j} = \arcsin\left(\frac{\bar{I}^{j}_{z}}{\|\boldsymbol{I}+\epsilon\|}\right) \; . \qquad (5.4\text{-}11)$$

The sector diffuseness is then given by

$$\Psi = 1 - \left\|\frac{\boldsymbol{I}}{\bar{E}+\epsilon}\right\|, \qquad (5.4\text{-}12)$$

where ϵ is a small parameter for numerical stability and $\bar{E}$ is the smoothened version of the energy defined as

$$E = \frac{1}{2}(\boldsymbol{x}_{s} \cdot \boldsymbol{x}_{s}^{*}) \qquad (5.4\text{-}13)$$

The asterisk * denotes the complex conjugate and $\cdot$ denotes the inner product of the vectors. If E is less than ϵ in one subframe, then the azimuth and elevation angles are set to zero and the diffuseness Ψ is set to 1.

This calculation is repeated in each sector parameter estimator to obtain a sector DoA and a sector diffuseness parameter.

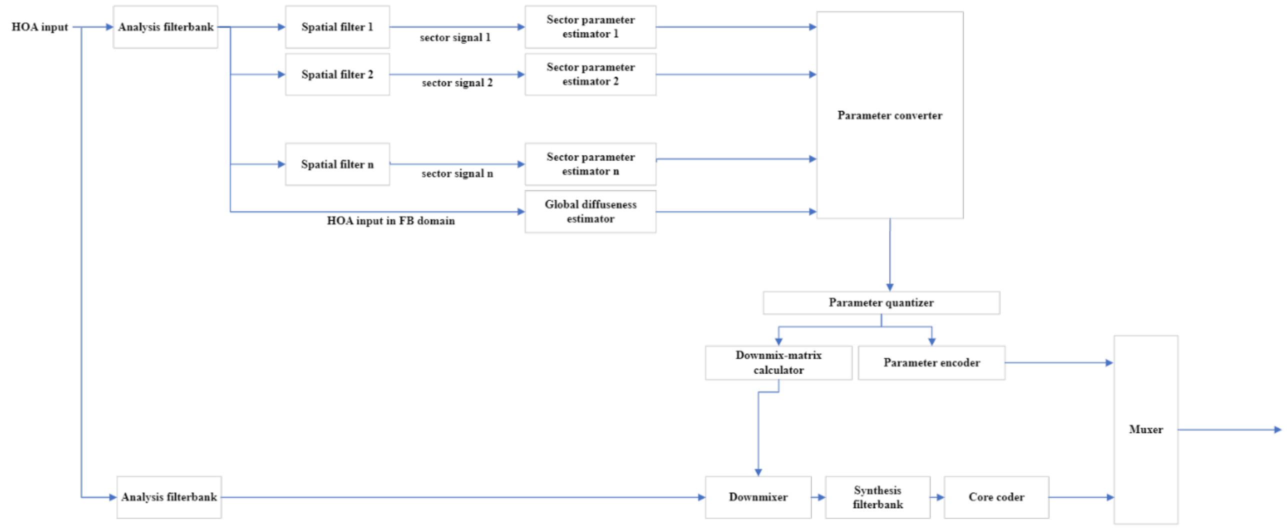


Figure 5.4-7: SBA encoder block diagram in high-order operation mode

In addition to the sector parameter estimators operating on the spatially filtered sector signals $\boldsymbol{x}_s$, there is a global diffuseness path. The global diffuseness estimator operates on the spatially unfiltered first-order input channels and returns a global diffuseness parameter Ψ. This global diffuseness parameter indicates how diffuse the whole audio scene is. It takes values between 0 and 1, where 0 means that the scene has no diffuse energy and 1 means that the scene has exclusively diffuse energy. Therefore, the general nature of the scene is always captured correctly, irrespective of the number of sector paths activated.

The smoothing of the intensity vector I and the Energy E are realized by storing the values of the last 40 ms and averaging over them. Moreover, and for reducing the number of parameters, the global diffuseness computed in each frequency subband and for each time slot of 5ms is averaged along the time axis to achieve a transmitted global diffuseness parameter of 20ms. Therefore, the time resolution is decoupled between the global diffuseness parameter and the DoAs, exploiting the fact that diffuseness retains a longer-term characteristic of the sound field than the directions, which is are more reactive spatial cue.

This global diffuseness parameter is then quantized and encoded into the bitstream according to clause xxx. In the reference implementation, the global diffuseness estimator is the same code as for the low-order mode.

The global diffuseness parameter is computed as:

$$\overline{\Psi}_b = \frac{\sum_{n=0}^{3} \Psi(\mathrm{n,b}).E(n,b)}{\sum_{n=0}^{3} E(n,b)}$$

where b is parameter band index, n is the stride index of the MDFT corresponding to the block index in the spatial metadata coding clause 5.1.5.

The sector diffuseness parameters are converted to relative directionalities of each of the two spatial sectors with respect to both spatial sectors by a parameter converter (cf. Figure 5.4-7). These relative directionalities are indicative of the relative diffuseness of one sector to all other sectors and are encoded and written to the bitstream as parameters.

The relative directionalities of the two spatial sectors are defined as

$$\mathrm{a}_1 = \frac{1-\Psi_1}{(1-\Psi_1)+(1-\Psi_2)} \qquad (5.4\text{-}9)$$

and

$$\mathrm{a}_2 = 1 - \mathrm{a}_1, \qquad (5.4\text{-}10)$$

where Ψ_1 is the sector diffuseness parameter for the first spatial sector and Ψ_2 is, or is obtained from, the sector diffuseness information for the second spatial sector.

This is a special case of the general formula

$$\mathrm{a_i} = \frac{1-\Psi_\mathrm{i}}{\sum_\mathrm{j}(1-\Psi_\mathrm{j})} \qquad (5.4\text{-}9)$$

with

$$\sum_j a_j = 1 \qquad (5.4\text{-}10)$$

for more than two sectors. It is important to note that the relative directionalities always add up to 1. The relative directionality of a spatial sector indicates the ratio of the directional energy in that sector to the sum of the directional energies of all sectors. The relative directionalities are quantized and encoded in the bitstream (cf. xxx).

5.4.3.4.2 Low-order operation mode

In the low-order operation mode, only one sector parameter estimator path is activated, and the other is deactivated. The spatial filtering according to Equation (5.4-6) becomes trivial as the matrix $\boldsymbol{w}_0^T$ is the unit matrix. The activated sector parameter estimator effectively operates on the first-order ambisonics input signal.

The global diffuseness and sector diffuseness are the same quantity in this case. Hence, only the global diffuseness estimator runs.

The DoA estimation in this single-sector case is modified as compared to the high-order mode. The smoothing of the intensity vector is omitted. The azimuth and elevation angles are calculated as

$$\phi^b = \mathrm{atan}\left(\frac{I_y}{I_x}\right),$$

and

$$\theta^b = \mathrm{atan}\left(\frac{I_z}{\sqrt{I_x^2+I_y^2}}\right),$$

respectively.

In the reference code, the DoA calculation is, for efficiency and simplicity, partly implemented in different functions than for the high-order operation modes. In this implementation, the smoothing of the intensity vector is achieved by storing the values of the last 40 ms and averaging over them.

The DirAC parameter estimation runs only for the 4 highest parameters bands (bin groups) according to Table 5.4-5. The bins of the two highest bands are grouped together into one parameter band, effectively encoding 3 parameter bands. The DirAC parameters for the lower parameter bands are estimated for the upmixed signal on the decoder side.

For the two lowest bitrates, 13.2 and 16.4 kbps, the time resolution of the direction parameters is reduced to one frame (20 ms). This is achieved by a weighted average of the intensity vector over all subframes of the frame. The weights are given by the associated directional energy. The frame-wise direction vector in cartesian coordinates is computed as:

$$\boldsymbol{r}_{DoA}^b = \frac{\sum_{n=0}^{3}\left(1-\Psi(\mathrm{n,b})\right).E(n,b).\mathrm{I}^b(n)/\|\mathrm{I}^b(n)\|}{\sum_{n=0}^{3}\left(1-\Psi(\mathrm{n,b})\right).E(n,b)}$$

where $E(n,b)$ and $\Psi(\mathrm{n,b})$ are the energy and the diffuseness, respectively, of the input signal measured in the parameter band b and in the time block n, and $\mathrm{I}^b(n)$ is the intensity vector in three-dimensional Cartesian coordinates for the same parameter band *b* and time block *n*. After averaging, the resulting directional vector $\boldsymbol{r}_{DoA}^b$ can no longer be a unit vector and normalization is therefore necessary:

$$\boldsymbol{r}_{DoA}^b \leftarrow \frac{r_{DoA}^b}{\|r_{DoA}^b\|}$$

5.4.3.5 Mono signal detection

During SBA operation, there are situations where a single Mono channel is sent as the W channel of an Ambisonics input, with silence or near silence in the other channels.

SBA mono handling ensures proper behaviour of the codec with mono input whilst operating in SBA mode. The algorithm consists of a mono detector in the encoder, signalling of mono through the bitstream and a mono detector/preserver to ensure the W channel is treated as mono in the decoder.

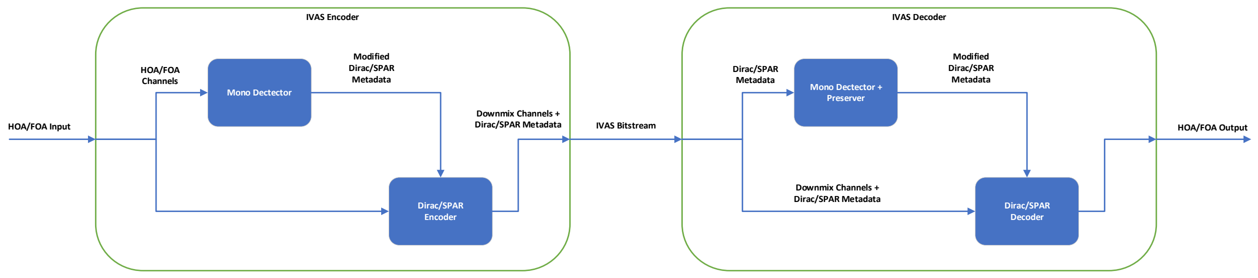

IVAS Encoder
HOA/FOA Input
HOA/FOA Channels
Mono Dectector
Modified Dirac/SPAR Metadata
Downmix Channels + Dirac/SPAR Metadata
Dirac/SPAR Encoder
IVAS Bitstream
IVAS Decoder
Dirac/SPAR Metadata
Mono Dectector + Preserver
Modified Dirac/SPAR Metadata
Downmix Channels + Dirac/SPAR Metadata
Dirac/SPAR Decoder
HOA/FOA Output

Figure 5.4.3-3Mono Detector Algorithm

Encoder Mono Detector

The encoder mono detector determines when an Ambisonics input has content in the W channel only. For each audio block the detector analyses the content in all the channels and decides whether the block of audio can be declared mono or Ambisonics.

The detector works in the MDFT domain by looping over each frequency band, calculating the power of the W channel W_p and the sum of the power of the other channels $\Sigma_p$in the input for each band. The detector uses the W_p, Σ_p and their ratio $\frac{W_p}{\Sigma_p}$ to determine if a band is mono, Ambisonics or silence. The classification decision (mono, multi-channel or silence) is based on two thresholds: a static noise threshold N_{thr} and a dynamic ratio threshold R_{thr}.

The noise threshold N_{thr} is used to determine whether a band is silent/near silent or has content. The ratio threshold R_{thr} is dynamic such that it scales with the signal as its amplitude decreases. It is used to classify a band as either Ambisonics or mono. The dynamic threshold is calculated using a linear function with W_p as input and is also clipped at a max and min threshold.

After determining the state of all the frequency bands, the detector checks if any band was classified as Ambisonics. If any band was Ambisonics the entire audio frame is declared as non-mono. If none of the bands were classified as Ambisonics, then all the bands must be either silence or mono. If there is at least one band that is mono then the audio frame is declared as mono.

There is further processing to ensure that the mono flag cannot toggle on and off quickly causing instability. The processing looks at the history of block mono flags and only after there have been consecutive mono flags for 30 ms does it classify the signal as mono and communicate this decision to the decoder as described in section 5.4.3.5.1.

The process for detecting mono in the encoder is as follows:

- For each frame of the encoder, loop over each MDFT frequency band
- For each frequency band calculate:
  - W channel power (W_p)
  - Sum of the power of each individual non-W channel (Σ_p)
- Check if the sum power is close to zero to prevent divide by zero errors.
  - If the sum power is close to zero check if the W_p is greater than the noise threshold N_{thr} and declaring the band as mono if it is.
  - Otherwise start the loop iteration again.
- Calculate the $\frac{W_p}{\Sigma_p}$.

- Calculate the ratio threshold R_{thr}.
    - Normalise W_p to be between 0 and 1
    - Scale MAX_TRESHOLD by $norm(W_p)$
    - Clip the calculated threshold to ensure the range is within MIN_THRESHOLD and MAX_TRESHOLD.
    - Check if the calculated ratio is above R_{thr}. If $\frac{W_p}{\Sigma_p}$ is above R_{thr} declare the band as mono
    - Otherwise declare the band as Ambisonics
- If none of these checks pass, then the band is declared as silence.
- After the frequency band loop, check if any band was declared as Ambisonics
    - If true, the encoder frame is non-mono
    - Otherwise, check if any band was declared as mono
        - If true, the encoder frame is mono
        - Otherwise, encoder frame is non-mono
- If the frame is mono increment the mono frame counter.
    - If the mono frame counter is > 30ms, then set the global mono flag to 1.
- If the frame is non-mono, reset the mono frame counter to zero.

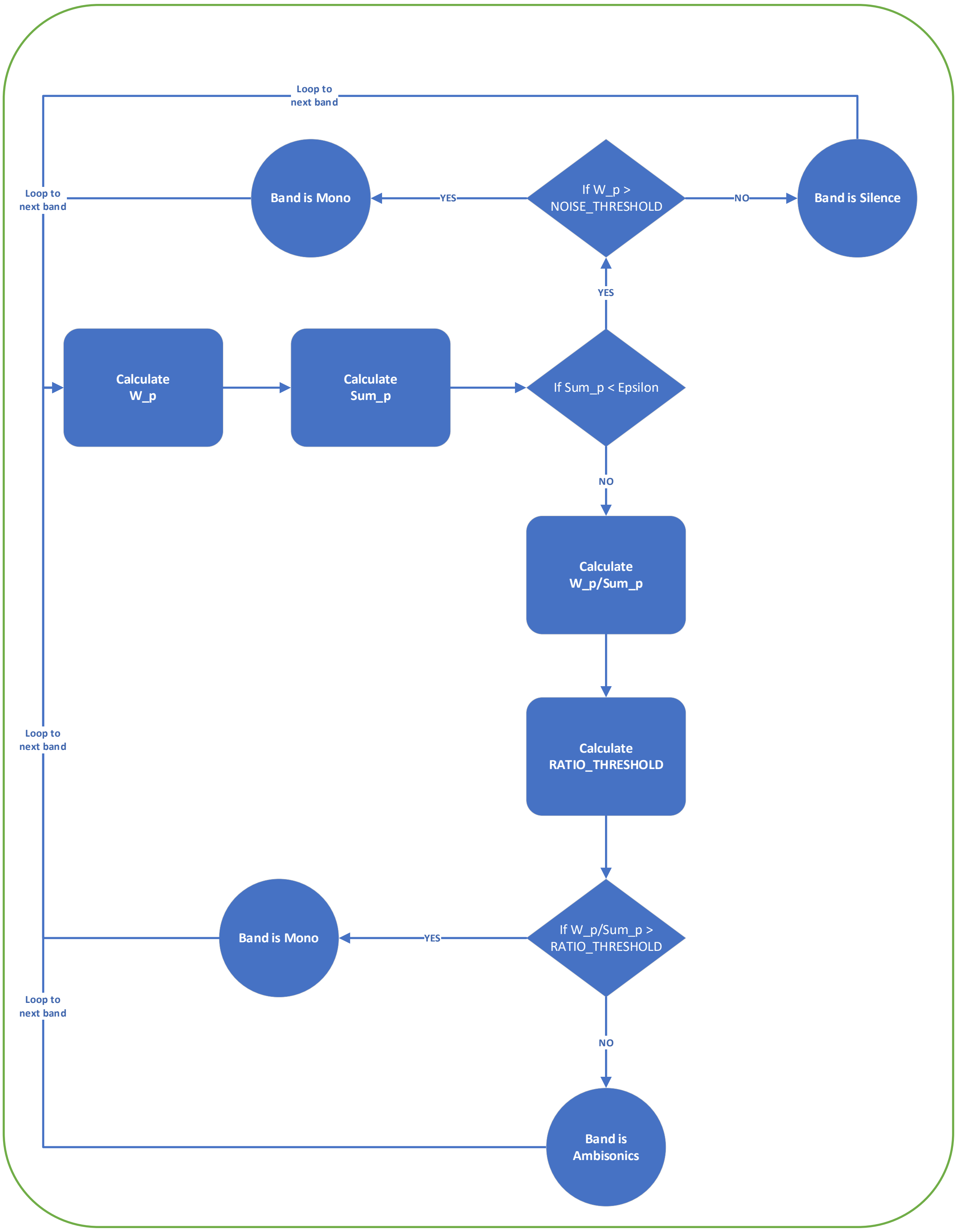

Loop to next band
Band is Mono
YES
If W_p > NOISE_THRESHOLD
NO
Band is Silence
Loop to next band
YES
Calculate W_p
Calculate Sum_p
If Sum_p < Epsilon
NO
Calculate W_p/Sum_p
Calculate RATIO_THRESHOLD
Loop to next band
Band is Mono
YES
If W_p/Sum_p > RATIO_THRESHOLD
Loop to next band
NO
Band is Ambisonics

Figure 5.4.3-4Mono Detector Band Classifier Loop

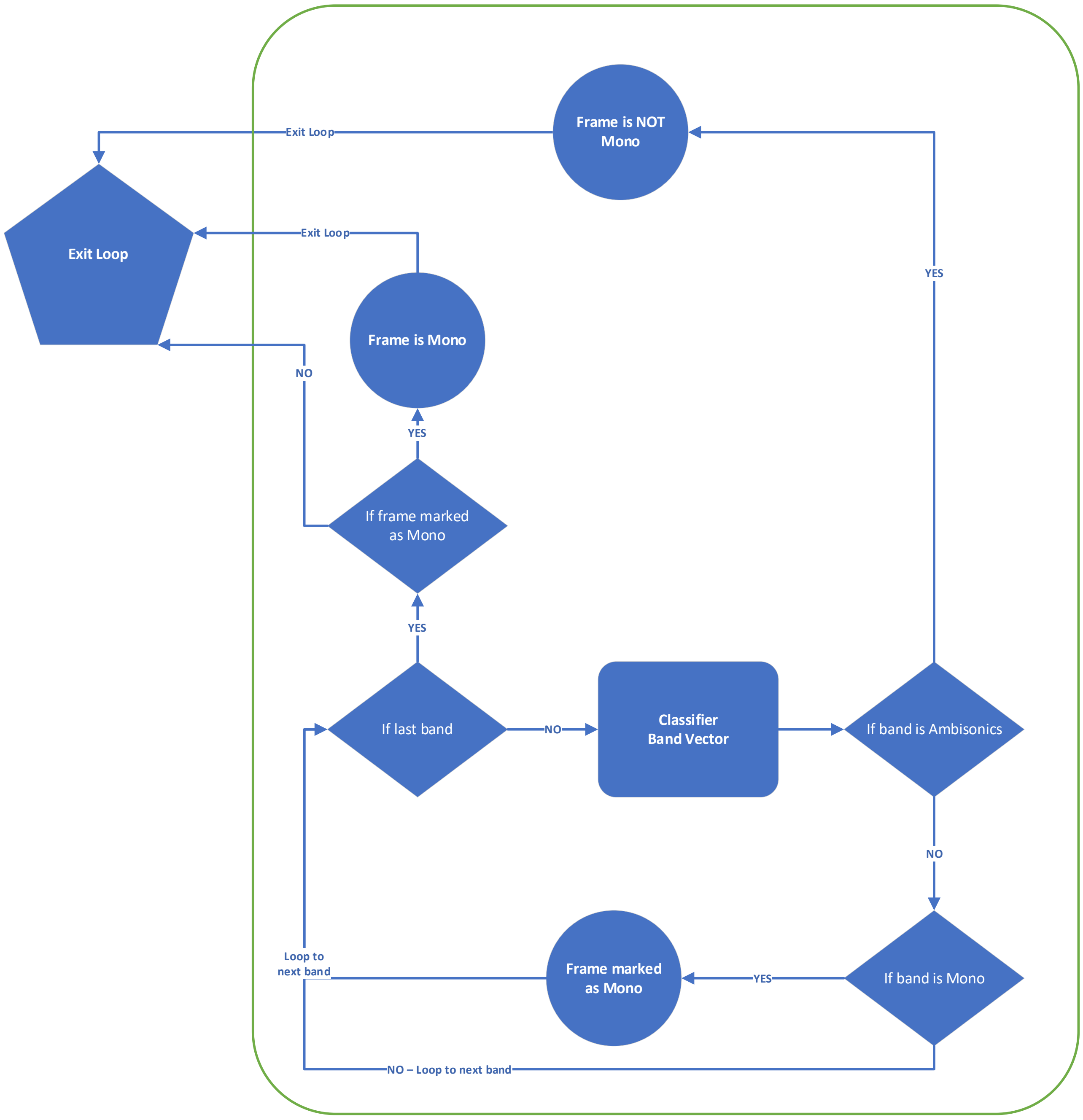


Figure 5.4.3-5 Mono Detector Frame Decision Loop

5.4.3.5.1 Signalling of Mono through the bitstream

The mono detected information is transmitted from the encoder to the decoder implicitly by modifying existing IVAS metadata to specific values. This ensures no update is required to the metadata specification, and no extra bandwidth is used for encoding this bit.

This implicit method works by setting existing SBA metadata set to specific values. In this case (which helps maintain bit-exactness on existing tests), 6 different SPAR and DIRAC metadata fields are set to zero at the encoder and then checked at the decoder. If the 6 metadata fields are zero in the decoder then it will process the frame as mono. The signalling also considers quantisation of the metadata values through the codec. Due to quantisation certain values cannot be encoded in the bitstream. The detector in the decoder takes into account this quantisation to detect mono correctly.

The metadata values are:

- SPAR metadata
  - pred_re
  - C_re
  - P_re
- DiRAC metadata
  - energy_ratio
  - azimuth
  - elevation

They are all set to zero when mono is detected. The SPAR values are related to extracting higher-order channels out of the downmixed channels sent in the bitstream, when they are set to zero, the extraction is not completed. The DiRAC values are related to diffuseness and position. When they are set to zero the mono channel is not spread across other channels in the Ambisonics output.

5.4.3.6 SPAR Metadata computation

SPAR metadata consist of quantized and coded prediction (PR) coefficients, cross-prediction (CP) coefficients and decorrelation (D) coefficients. PR coefficients are always coded in SPAR metadata bitstream whereas presence of C and D coefficients depends on number of downmix channels N_{dmx}. Table 5.4-7 shows which SPAR metadata parameters are computed and coded at a given downmix configuration.

Table 5.4-7: SBA metadata parameters at various downmix configurations

N_{dmx}	Metadata parameters
1	PR, D
1< N_{dmx} **<** $N_{spar_{ana}}$	PR, C and D
$N_{spar_{ana}}$	PR

Prediction coefficients indicate the correlation between W channel and other channels of SBA input and are computed such that SBA input can be converted into a downmix signal that has lower energy than input signal by subtracting the prediction component from the side channels as follows.

$$S' = S - pr_s W'$$

Where, W' is the primary downmix channel, S is the side channel and S' is the side channel prediction error OR residual side channel. With PR coefficients, N_{spar_ana} channel input is converted to N_{dmx} channel downmix signal.

At decoder, side channel is reconstructed as follows:

$$S = S' + pr_s W'.$$

At low bitrates, the number of downmix channels that are coded in IVAS bitstream is less than the number of input channels to SPAR and additional parameters CP and D coefficients are computed to reconstruct the ambisonics signal from the downmix signal preserving the second-order statistics of the Ambisonics audio scene (i.e. the covariance) after decoding. The CP coefficients indicate the correlation between residual side channels that are coded in the bitstream and the parametric channels, that is, the residual side channels that are not coded in the bitstream.

The decorrelation coefficients indicate the residual energy in the parametric channels that cannot be predicted using either prediction or cross-prediction. At the SPAR decoder the time domain decorrelator described in subclause 6.4.6.4.1 is used to separate the transient component of primary downmix channel W' and generate a slowly-fluctuating decorrelated output, corresponding to each parametric channel. The D coefficients are then used to scale the decorrelator output signal, which is then combined with the prediction signal, where prediction signal is generated based on the PR, CP coefficients and W' channel and includes a transient component, to reconstruct the side channels.

Detailed description of computation of SPAR metadata parameters is provided in the following subclauses.

5.4.3.6.1 Banded covariance computation

SPAR metadata analysis channels from MDFT signal $S_j^{sba_{ana},mdft}(\mathrm{m},k)$ are processed through magnitude response of MDFT filterbank to generate banded covariance matrix $C^{in}_{[N_{spar_{ana}} x N_{spar_{ana}}]}(b)$, here b is the band index such that $0 \le b \le nB$, where $nB = 12$ for 32kHz and 48kHz sampling rates and 10 for 16kHz sampling rate..

When sampling frequency is 16kHz, number of MDFT filterbank bands nB are set to 10. However, to avoid additional signalling bit in the bitstream, SPAR metadata computation is still done on 12 bands by appending 2 bands with magnitude response set to zero.

5.4.3.6.1.1 Band covariance

The channels of signal $S_j^{sba_{ana},mdft}$ are grouped into a column vector $V(m,k)$ as given in Eqn (5.4-14).

$$V(m,k) = \begin{bmatrix} S_1^{sba\_ana,mdft}(\mathrm{m},k) \\ S_2^{sba\_ana,mdft}(\mathrm{m},k) \\ \cdot \\ \cdot \\ \cdot \\ S_{\boldsymbol{N_{spar\_ana}}}^{sba\_ana,mdft}(\mathrm{m},k) \end{bmatrix} \tag{5.4-14}$$

Then covariance for each MDFT bin k is computed as follows:

$$C^{in,mdft}_{[\boldsymbol{N_{spar\_ana}} x \boldsymbol{N_{spar\_ana}}]}(m,k) = re(V(m,k)V^H(m,k)) \tag{5.4-15}$$

where V^H is the conjugate transpose of matrix V and $re(x)$ returns real value of complex variable x.

Then banded covariance is computed as follows:

$$C^{in,banded}_{[i,j]}(b) = \sum_{m=1}^{m=4} \sum_{\boldsymbol{k=bin_{start}}(b)}^{\boldsymbol{k=bin_{end}}(b)} (C^{in,mdft}_{[\mathrm{i,j}]}(m,k)\, H^{mdft}_{abs,5ms}(b,k)) \tag{5.4-16}$$

where i and j are channel indices with $1 \le i \le N_{spar\_ana}$, $1 \le j \le N_{spar\_ana}$, $H^{mdft}_{abs,5ms}(b,k)$ is the magnitude response of MDFT filterbank (described in subclause 5.1.5) for band index b and frequency bin index k, $bin_{start}(b)$ is the bin index of first non-zero magnitude response bin of MDFT filter bank and $bin_{end}(b)$ is the bin index of last non-zero magnitude response bin of MDFT filter bank.

5.4.3.6.1.2 Covariance smoothing

In SPAR, the banded covariance is smoothed across multiple frames by performing long-term averaging of the banded covariance using a first-order low-pass filter. The smoothing process includes comparing the number of bins in the band to the desired number of bins threshold and then computing a forgetting factor based on the ratio of number of bins in the band to the threshold value, where the number of bins are computed by summing the magnitude filter bank response in that band, as given in Eqn (5.4-21). When DTX is OFF or when VAD detector output W_{vad} is set to 1, as described in subclause 5.3.3.1, the covariance smoothing is done based on the previous frame's smoothed covariance matrix, forgetting factor and banded covariance in the current frame, $C^{in,banded}_{[i,j]}(b)$, as follows.

When DTX is OFF or when VAD detector output W_{vad} is set to 1, as described in subclause 5.4.3.1, the covariance matrix in each band $C^{in,banded}_{[i,j]}(b)$ is smoothed as per eq. (5.4-17)

$$C^{in}_{[i,j]}(b) = \alpha(b) C^{in,banded}_{[i,j]}(b) + (1-\alpha(b)) C^{in,prev}_{[i,j]}(b) + \alpha(b) fac \tag{5.4-17}$$

where

$$fac = \begin{cases} 0, & if\ i\ != j \\ \varepsilon, & if\ i == j \end{cases} \quad (5.4\text{-}18)$$

$C_{[i,j]}^{in,prev}(b)$ is the value of $C_{[i,j]}^{in}(b)$ in previous frame, $\alpha(b)$ is the frequency varying smoothing factor that depends on frame index of previous frame ($frameIdx_{prev}$), band index b and transient detector output T_0 and T_1 as described in subclause 5.4.3.2. Covariance smoothing gets reset when a transient is detected except at low frequency bands where resetting is bitrate dependent. The purpose of resetting covariance smoothing on transients is to preserve the transient components of the audio signal in the downmix. Additional smoothing is applied at low frequency bands as compared to high frequency bands as shown in eq. (5.4-19), (5.4-20), and (5.4-21).

$$\alpha(b) = \begin{cases} 1, & if\ frameIdx_{prev} = -1\ OR\ T_1 = 1 \\ 1, & if\ T_0 = 1\ , b > 2 \\ \alpha_{sm}(b), & otherwise \end{cases} \quad (5.4\text{-}19)$$

where

$$\alpha_{sm}(b) = \min\left(0.8, \frac{\left(\sum_{k=\boldsymbol{bin_{start}}(b)}^{k=\boldsymbol{bin_{end}}(b)} H_{abs,5ms}^{mdft}(b,k)\right) b\ R_{sm_{fact}}}{24}\right) \quad (5.4\text{-}20)$$

and

$$R_{sm_{fact}} = \begin{cases} 0.5, & if\ ivas\ bit\ rate < 24.4kbps \\ 0.75, & if\ ivas\ bit\ rate \geq 24.4kbps \end{cases} \quad (5.4\text{-}21)$$

When DTX is ON and VAD detector output W_{vad} is set to 0, then covariance matrix in each band $C_{[i,j]}^{in,banded}(b)$ is smoothened, as described in subclause xxx, to generate $C_{[i,j]}^{in}(b)$.

5.4.3.6.2 Active W downmix detector

SPAR uses active W downmixing to mix contents of X, Y, and Z channels into the W channel in the downmixing process described in subclause 5.4.5. Active W downmixing is activated based on the output of active W downmix detector $Flag_{activeW}$ given in eq. (5.4-22).

$$Flag_{activeW} = \begin{cases} 1 & , if\ IVAS\ bitrate \leq 32kbps \\ Dyn_{activeW} & , if\ 48kbps \leq IVAS\ bitrate \leq 192kbps \\ 0 & , if\ IVAS\ bitrate \geq 256kbps \end{cases} \quad (5.4\text{-}22)$$

Computation of $Dyn_{activeW}$ is as follows. First, broadband variance for all FOA channels is computed

$$Var_{[i]}^{in} = \sum_{b=1}^{b=nB} C_{[i,i]}^{in}(b) \quad (5.4\text{-}23)$$

where nB is the number of MDFT filter bank bands, i is the channel index with $1 \leq i \leq 4$.

Then parametric channel variance is computed by summing up the variance of FOA channels that are not present in the downmix signal as shown below.

$$Var_p^{in} = \sum_{i=\boldsymbol{N_{dmx}}+1}^{i=4} Var_{[i]}^{in} \quad (5.4\text{-}24)$$

where N_{dmx} is the number of downmix channels at a given bitrate as given in Table 5.4-1.

Then ratio of parametric channel variance to W channel variance is computed as follows.

$$En_{ratio} = \frac{Var_p^{in}}{Var_{[1]}^{in}} \quad (5.4\text{-}25)$$

Then $Dyn_{activeW}$ is computed by comparing Var_p^{in} and En_{ratio} against fix threshold values as follows.

$$Dyn_{activeW} = \begin{cases} 0 & , if\ Var_p^{in} < 1024 \\ 0 & , if\ En_{ratio} \geq 0.00001521 \\ 1 & , if\ En_{ratio} < 0.00001521 \end{cases} \quad (5.4\text{-}26)$$

At bitrates where the number of downmix channels is equal to 2 or 3 (as given in Table 5.4-1), a channel index, Res_{ind} as given in Eqn 5.4-27), is also computed which indicates the side channel index that is to be mixed into the W channel in the downmixing process described in subclause 5.4.5.

$$Res_{ind} = \begin{cases} 3 & , if\ N_{dmx} = 2\ , Var_{[3]}^{in} \geq Var_{[4]}^{in} \\ 4 & , if\ N_{dmx} = 2\ , Var_{[4]}^{in} < Var_{[3]}^{in} \\ 3 & , if\ N_{dmx} = 3 \end{cases} \quad (5.4\text{-}27)$$

Note that channel index 3 corresponds to the Z channel and channel index 4 corresponds to X channel of the ambisonics signal with ACN ordering.

Dynamic active W flag ($Dyn_{activeW}$) is coded in metadata bitstream when number of downmix channels N_{dmx} is set to 2 or 3. Active W residual index (Res_{ind}) is coded in metadata bitstream when number of downmix channels N_{dmx} is set to 2 as described in subclause 8.3.1.

5.4.3.6.3 Banded covariance re-mixing

SPAR (Spatial Reconstruction) computes parameters for band indices 1 to nB_{md}. Where nB_{md} is computed as follows.

First number of SPAR bands are computed:

$$nB_{spar} = \begin{cases} 8 & , if\ N_{spar\_ana} = 4 \\ nB & , otherwise \end{cases} \quad (5.4\text{-}28)$$

Then nB_{md} is computed as follows:

$$nB_{md} = \frac{nB_{spar}}{bw_{bands}} \quad (5.4\text{-}29)$$

where

$$bw_{bands} = \begin{cases} 4 & , if\ W_{vad} = 0 \\ 2 & , if\ W_{vad} = 1\ , ivas\ bitrate \leq 16.4\ kbps \\ 1 & , otherwise \end{cases} \quad (5.4\text{-}30)$$

The frequency resolution of the covariance matrix $C_{[i,j]}^{in}$ computed in Eq. (5.4-17) is modified as per the value of bw_{bands} and a new covariance matrix $C_{[i,j]}^{md}$ is computed as given below.

$$C_{[i,j]}^{md}(b) = \sum_{b1=1+(b-1)*bw_{bands}}^{b1=\ b*bw_{bands}} C_{[i,j]}^{in}(b1) \quad (5.4\text{-}31)$$

where b is the band index with $1 \leq b \leq nB_{md}$.

5.4.3.6.4 SPAR parameter estimation

SPAR parameters consist of prediction (PR) coefficients, cross-prediction (CP) coefficients and decorrelation (D) coefficients. The PR coefficients are used to create the downmix matrix as described in subclause 5.4.4 and can be computed in passive W mode or active W mode.

5.4.3.6.5 Passive-W channel prediction coefficients computation

Passive-W channel prediction mode is set when $Flag_{activeW}$ is 0, as described in subclause 5.4.3.6.5. In passive-W channel prediction mode, the primary downmix channel *W'* is simply a delayed version of W-channel input to SPAR and the computation of PR coefficients is as follows.

$$PR_{j-1}(b) = \frac{C_{[1,j]}^{md}(b)}{\max\left(C_{[1,1]}^{md}(b), \sqrt{\sum_{k=2}^{k=N_{spar\_ana}} \left|C_{[1,k]}^{md}(b)\right|^2}, \varepsilon\right)} \quad (5.4\text{-}32)$$

where b is the band index with $1 \leq b \leq nB_{md}$ and j is the channel index with $2 \leq j \leq N_{spar\_ana}$.

5.4.3.6.6 Active-W channel prediction coefficients computation

Active-W channel prediction mode is set when $Flag_{activeW}$, described in subclause 5.4.3.6.6, is 1. In active-W channel prediction mode, encoding downmix that is applied at the encoder is different than the decoding re-mix or upmix applied at the decoder. The primary downmix channel W' in active W mode is formed by

(a) Determining input gains for the FOA signal.
(b) Determining a set of prediction coefficients for the FOA signal.
(c) Scaling the FOA channels to SPAR and adding them together.

Active W mode gets enabled at IVAS bitrates <= 192 kbps. At these bitrates the analysis signal $s_j^{sba\_ana}(n)$ is the FOA component of the $s_i^{HP20}(n)$ signal and the computation of W' in a given time-frequency tile is given as

$$W' = s_w W + s_y Y + s_z Z + s_x X, \tag{5.4-33}$$

where $s_{w,} s_y, s_z, s_x$ are the gains applied to the FOA channels. These gains are determined such that overall prediction error on the side channels is minimized and the energy difference between the up-mixed signal at the decoder and input signal at the encoder is minimized. Computation of PR coefficients and gains ($s_{w,} s_y, s_z, s_x$) depend on the number of downmix channels. The following subclauses contain a more detailed description of the computation of PR coefficients in active W mode.

5.4.3.6.6.1 Active W prediction coefficients in 1 channel downmix mode

PR coefficients $PR_{[3x1]}(b)$ with $1 \le b \le nB_{md}$ are given in eq 5.4-47. The following steps are performed to compute $PR_{[3x1]}(b)$ from covariance matrix $C_{[4x4]}^{md}(b)$.

First, the covariance matrix that is computed in Eq. (5.4-31) is represented as follows.

$$C_{[4x4]}^{md}(b) = \begin{pmatrix} w & \alpha\hat{u}^* \\ \alpha\hat{u} & U \end{pmatrix} \tag{5.4-34}$$

where $\hat{u}$ *is* $3x1$ unit vector, $\alpha\hat{u} = \begin{pmatrix} C_{[2x1]}^{md}(b) \\ C_{[3x1]}^{md}(b) \\ C_{[4x1]}^{md}(b) \end{pmatrix}$, $w = C_{[1x1]}^{md}(b), \alpha = \sqrt{\sum_{k=2}^{k=N_{spar\_ana}} \left|C_{[k,1]}^{md}(b)\right|^2}$, $U = \begin{pmatrix} C_{[2x2]}^{md}(b) & C_{[2x3]}^{md}(b) & C_{[2x4]}^{md}(b) \\ C_{[3x2]}^{md}(b) & C_{[3x3]}^{md}(b) & C_{[3x4]}^{md}(b) \\ C_{[4x2]}^{md}(b) & C_{[4x3]}^{md}(b) & C_{[4x4]}^{md}(b) \end{pmatrix}$.

Then, a passive prediction factor $g_{passive}$ is computed as given below:

$$g_{passive} = \frac{\alpha}{w}. \tag{5.4-35}$$

Computation of active W prediction coefficients depend on the value of $g_{passive}$ as given below.

When $g_{passive} < 3$, the downmix matrix with active W prediction coefficients is computed as

$$PRdmx_{[4x4]} = \begin{pmatrix} 1 & (0,0,0) \\ -g\hat{u} & I_3 \end{pmatrix} \begin{pmatrix} 1 & F\hat{u}^H \\ \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} & I_3 \end{pmatrix} = \begin{pmatrix} 1 & F\hat{u}^H \\ -g\hat{u} & I_3 - gF\hat{u}\hat{u}^H \end{pmatrix} \tag{5.4-36}$$

where $g\hat{u}$ are the prediction coefficients, $F = \min\left(1, \frac{f\alpha}{max\left(mw, \frac{trace(U)}{1}\right)}\right)$ $I_3 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$,

$$f = \begin{cases} 0.25 & if\ ivas\ bitrate \le 16.4\ kbps\ OR\ W_{vad} == 0 \\ 1 & otherwise \end{cases} \tag{5.4-37}$$

$$m = \begin{cases} 1 & if\ b > 8, N_{spar\_ana} = 4 \\ 3 & otherwise \end{cases} \tag{5.4-38}$$

To compute g, the post prediction matrix is computed and then the covariance between primary downmix channel and predicted channels is minimized which results in a linear equation in g as given in eq (5.4-39).

$$Post\_prediction_{[4x4]} = PRdmx_{[4x4]} * C^{md}_{[4x4]}(b) * {PRdmx_{[4x4]}}^{H} \quad = \begin{pmatrix} \cdot & \cdot \\ \hat{r} & \cdot \end{pmatrix} \tag{5.4-39}$$

In eq (5.4-39), $\hat{r}$ is the indicator of overall prediction error on the side channels and is minimized by setting $\hat{u}^{*}\,\hat{r} = 0$. This results in linear equation in g as given below.

$$g\beta F^2 + 2\alpha gF + wg - \beta F - \alpha = 0 \tag{5.4-40}$$

$$g = \frac{\alpha + \beta F}{\beta F^2 + 2\alpha F + w} \tag{5.4-41}$$

where $\beta = \hat{u}^H U\hat{u}$.

Prediction (PR) coefficients in active W mode $PR_{[3x1]}$ are computed as $PR_{[3x1]} = g\hat{u}$. These coefficients are further scaled as per Eqn (5.4-47).

When $g_{passive} \geq 3$, the downmix matrix with active W prediction coefficients is computed as

$$PRdmx_{[4x4]} = \begin{pmatrix} 1 & (0,0,0) \\ -g\hat{u} & I_3 \end{pmatrix}\begin{pmatrix} 1 & F\hat{u}^H \\ \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} & I_3 \end{pmatrix} = \begin{pmatrix} 1 & F\hat{u}^H \\ -g\hat{u} & I_3 - gF\hat{u}\hat{u}^H \end{pmatrix} \tag{5.4-42}$$

where $F = gf$ and the computation of $f \; and \; g$ is as follows.

First, the post prediction matrix is computed and then the covariance between primary downmix channel and predicted channels is minimized which results in a quadratic equation in f as given in eq. (5.4-42).

$$C^{postpred}_{[4x4]} = PRdmx_{[4x4]} * C^{md}_{[4x4]}(b) * {PRdmx_{[4x4]}}^{H} \quad = \begin{pmatrix} \cdot & \cdot \\ \hat{r} & \cdot \end{pmatrix} \tag{5.4-43}$$

Then $\hat{r}$ is minimized by setting $\hat{u}^H * \hat{r} = 0$ as follows:

$$\hat{u}^H * \hat{r} = f^2\beta g^3 + 2\alpha g^2 f + wg - \beta gf - \alpha = 0 \tag{5.4-44}$$

In eq. (5.4-44), g is set to 3 and $f$is computed by solving the quadratic equation as follows:

$$f = \frac{\left((\beta - 2\alpha g) + \sqrt{4\alpha^2 g^2 + \beta^2 - 4\beta g^2 w}\right)}{\max(2\beta g^2, \varepsilon)} \tag{5.4-45}$$

Prediction (PR) coefficients in active W mode $PR_{[3x1]}$ are computed as $PR_{[3x1]} = g\hat{u}$. These coefficients are further scaled as per eq (5.4-47).

Computation of downmix scaling factor

In active-W prediction mode, an up-mixing strategy is applied at the decoder that is different from the downmix strategy at the encoder, where the primary downmix channel W', obtained by using the prediction matrix in eq (5.4-36) and (5.4-42), is further scaled with an additional gain W_{scale}, computed as per the upmixing gain described in subclause 0. This method attempts to minimize the energy difference between the upmixed signal at the decoder and input signal at the encoder. The gain W_{scale} is computed by estimating the variance of W channel at the upmixer output (as described in subclause 6.3.6.4**) and equating it with variance of W channel input to SPAR. This results in W_{scale} value as given below.

$$W_{scale} = \frac{g_w + \sqrt{{g_w}^2 + 4f_s g^2}}{2} \quad , \tag{5.4-46}$$

where $f_s = \begin{cases} 0.25 & if \; ivas \; bitrate \leq 16.4 \; kbps \; OR \; W_{vad} == 0 \\ 0.5 & otherwise \end{cases}$, $w = C^{md}_{[1,1]}(b)$, $dmx_{[1x4]} = (1 \quad F\hat{u}^H)$

, $w' = dmx_{[1x4]} C^{md}_{[4x4]}(b) {dmx_{[1x4]}}^{H}$, and $g_w = \sqrt{\frac{w}{w'}}$.

The unquantized PR coefficients in active W mode are then computed as follows:

$$PR_{[3x1]}(b) = \frac{g\hat{u}}{W_{scale}} \tag{5.4-47}$$

The downmixing gains $s_{w,}\, s_y, s_z,\, s_x$ mentioned in eq. 5.4-33) are computed as $s_{[1x4]}$ as follows.

$$s_{[1x4]} = dmx_{[1x4]}\, W_{scale} \quad (5.4\text{-}48)$$

5.4.3.6.6.2 Active W prediction coefficients in 2 and 3 channel downmix mode

In 2- and 3-channel downmix mode, the downmix matrix with active W prediction coefficients is computed as per eq. (5.4-42). In this mode f and g are set to 1 and the unit vector is computed as $\hat{u}_i = \begin{cases} 0 & if\ i\ != Res_{ind} - 1 \\ 1 & if\ i\ == Res_{ind} - 1 \end{cases}$, where Res_{ind} is computed in eq. 5.4-27). W_{scale} is set to 1 and Prediction coefficients and downmix matrix is computed as

$$PR_{[3x1]}(b) = \hat{u} \quad (5.4\text{-}49)$$

5.4.3.6.7 Cross-Prediction coefficients computation

To compute cross-prediction coefficients, first, downmix matrix is computed from the prediction coefficients, as

$$PRdmx_{[N_{spar\_ana} x N_{spar\_ana}]} = \begin{pmatrix} 1 & 0 \\ -PR_{[N_{PR}\,x\,1]}(b) & I_{N_{PR}} \end{pmatrix} \begin{pmatrix} W_{scale} & W_{scale} F \hat{u}_{[1xN_{PR}]}{}^{H} \\ 0 & I_{N_{PR}} \end{pmatrix} \quad (5.4\text{-}50)$$

where $PR_{[N_{PR}\,x\,1]}(b)$ is the column vector of prediction coefficients with $N_{PR} = N_{spar\_ana} - 1$, $I_{N_{PR}}$ is an identity matrix and $\hat{u}_{[1xN_{PR}]}$ is the unit vector as given in equations (5.4-36) and (5.4-42), $F = \begin{cases} F & if\ Flag_{activeW} = 1 \\ 0 & if\ Flag_{activeW} = 0 \end{cases}$ and $W_{scale} = \begin{cases} W_{scale} & if\ Flag_{activeW} = 1 \\ 1 & if\ Flag_{activeW} = 0 \end{cases}$.

Then the downmix matrix is remixed as per a remix matrix. The remix matrix rearranges downmix channels as per the priority order which is decided based on perceptual importance of ambisonics channels. Priority order for FOA signal is given below.

$$W > Y > X > Z \quad (5.4\text{-}51)$$

Based on this priority order a remix matrix $remix_{[N_{spar\_ana} x N_{spar\_ana}]}$ is created as given below.

$$remix_{[i,j]} = \begin{cases} I_{[i,j]} & , i > 4, j > 4 \\ remixFOA_{[i,j]}, & \text{otherwise} \end{cases} \quad (5.4\text{-}52)$$

where $1 \le i \le N_{spar\_ana}\ and\ 1 \le j \le N_{spar\_ana}$ and $remixFOA_{[4,x4]} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$. (5.4-53)

The remixed downmix matrix is computed as follows:

$$PRdmx_{[N_{spar\_ana} x N_{spar\_ana}]} = remix_{[N_{spar\_ana} x N_{spar\_ana}]} PRdmx_{[N_{spar\_ana} x N_{spar\_ana}]} \quad (5.4\text{-}54)$$

With the remixed downmix matrix, post-prediction matrix is computed as follows.

$$C^{postpred}_{[N_{spar\_ana} x N_{spar\_ana}]} = PRdmx_{[N_{spar\_ana} x N_{spar\_ana}]}\, C^{md}_{[N_{spar\_ana}\,x\,N_{spar\_ana}]}(b)\, PRdmx_{[N_{spar\_ana} x N_{spar\_ana}]}{}^{H} \quad (5.4\text{-}55)$$

CP coefficients cross-predict any remaining portion of the channels that are not part of the downmix signal (also referred to as fully parametric channels) from the downmix signal. From the post-predicted matrix, CP coefficients are computed as follows.

$$CP_{[(N_{spar\_ana} - N_{dmx})\,x\,(N_{dmx}-1)]}(\mathrm{b}) = \begin{cases} 0, & if\ N_{dmx} = 1 \\ 0, & if\ N_{dmx} = N_{spar\_ana} \\ 0, & if\ W_{vad} = 0 \\ 0, & if\ tr(R_{dd}) < \varepsilon \\ R_{ud}(R_{dd} + I * max(\varepsilon, tr(R_{dd})0.005))^{-1}, & \text{otherwise} \end{cases} \quad (5.4\text{-}56)$$

where $R_{du} = R_{ud}{}^{H} = C^{postpred}_{[2:N_{dmx}, N_{dmx}+1:N_{spar\_ana}]}$, $R_{dd} = C^{postpred}_{[2:N_{dmx}, 2:N_{dmx}]}$ and I is the identity matrix with same dimensions as R_{dd}.

5.4.3.6.8 Decorrelation coefficients computation

The decorrelation coefficients ($D_{[(\boldsymbol{N_{spar\_ana}} - \boldsymbol{N_{dmx}})\ x\ 1]}$) are computed by normalizing the residual energy in fully parametric channels as given below.

$$D_{[(\boldsymbol{N_{spar\_ana}} - \boldsymbol{N_{dmx}})\ x\ 1]}(\mathrm{b}) = \mathrm{diag}\left(\sqrt{\max\left(0, real\left(diag(NRes_{uu})\right)\right)}\right) \qquad (5.4\text{-}57)$$

where

$$NRes_{uu} = \frac{Res_{uu}}{\max(\varepsilon, R_{WW}, \mathrm{scale} * tr(|Res_{uu}|))},$$

$$Res_{uu} = \begin{cases} R_{uu} - (CP(b) * R_{dd} * CP(b)^{H}) & if\ 1 < N_{dmx} < N_{spar\_ana} \\ R_{uu} & \text{otherwise} \end{cases},$$

$$\mathrm{scale} = \begin{cases} 0.75, & if\ W_{vad} == 0\ AND\ N_{dmx} == 1 \\ 1, & otherwise \end{cases},$$

$$R_{WW} = C^{postpred}_{[1,1]},$$

$$R_{uu} = C^{postpred}_{[\boldsymbol{N_{dmx}}+1:\boldsymbol{N_{spar\_ana}},\ \boldsymbol{N_{dmx}}+1:\boldsymbol{N_{spar\_ana}}]},$$

$$R_{dd} = C^{postpred}_{[2:\boldsymbol{N_{dmx}}, 2:\boldsymbol{N_{dmx}}]},$$

and $CP(b)$ are the cross predicted coefficients computed in Eqn (5.4-56)

5.4.3.6.9 Overview of quantization and coding of SPAR parameters

PR, CP and D coefficients are quantized and computed in the sequence that is given in subclause 5.4.3.6.10. Quantization and coding of SPAR parameters includes reading table entries, corresponding to given IVAS bitrate and SBA order, from the SPAR bitrate table (as described in subclause 5.4.3.6.11). Iterating over all the quantization strategies mentioned in the table entries from fine to coarse quantization. For each quantization strategy, iterate over multiple entropy coding strategies. Exit from the quantization and coding loop process when SPAR metadata bitrate is within the bitrate threshold that is defined by target metadata bits $MDbits_{tar}$ and maximum metadata bits $MDbits_{max}$.

5.4.3.6.10 Bitrate distribution and SPAR bitrate table

Table 5.4-8 is the SPAR bitrate distribution table which contains an entry corresponding to each SBA bitrate. Row entry is selected based on IVAS bitrate, bandwidth and SBA analysis order. In addition to various bitrate dependent settings, each table row contains:

a) Target, minimum and maximum bitrate estimate for each downmix signal.

b) Maximum of 3 quantization strategies (from fine to coarse quantization). Each quantization strategy contains number of quantization points for PR, CP and D coefficients $\{\{qlvl1_{pr}, qlvl1_{cp}, qlvl1_{d}, 1\}, \{qlvl2_{pr}, qlvl2_{cp}, qlvl2_{d}, 1\}, \{qlvl3_{pr}, qlvl3_{cp}, qlvl3_{d}, 1\}\}$.

From the row entry, target metadata bits $MDbits_{tar}$ and maximum metadata bits $MDbits_{max}$ are computed as follows.

$$MDbits_{tar} = \left(\left(ivas_{bits} - ivas_{headerbits} - sba_{headerbits} - \left(\sum_{n=1}^{n=N_{dmx}} core_{brs[n][1]}\right) - spar_{CSbits} - spar_{qbits}\right)\frac{nB_{md}}{12}\right) + spar_{CSbits} + spar_{qbits} \qquad (5.4\text{-}58)$$

$$MDbits_{max} = \left(\left(ivas_{bits} - ivas_{headerBits} - sba_{headerbits} - \left(\sum_{n=1}^{n=N_{dmx}} core_{brs[n][2]}\right) - spar_{CSbits} - spar_{qbits}\right)\frac{nB_{md}}{12}\right) + spar_{CSbits} + spar_{qbits} \quad (5.4\text{-}59)$$

where $ivas_{bits} = \frac{ivas_{bitrate}}{frames_{per_{second}}}$, $ivas_{headerBits} = 3$, $sba_{headerbits} = 3$, $spar_{CSbits} = 3$ and $spar_{qbits} = 2$.

Table 5.4-8: SPAR bitrate distribution table

IVAS bitrate [kbps]	SBA analysis order	bwidth	Number of downmix channels	Init time active_w	Core coder bitrates {target, min, max} for each downmix channel	Quantization levels	TD ducking	AGC bits channel index
13.2	1	3	1	1	{ 10000, 8150, 13150 }	{ { 15, 1, 5, 1 },{ 15, 1, 3, 1 },{ 7, 1, 3, 1 } }	0	0
16.4	1	3	1	1	{ 13200, 11350, 16350 }	{ { 15, 1, 5, 1 },{ 15, 1, 3, 1 },{ 7, 1, 3, 1 } }	0	0
24.4	1	3	1	1	{ 16400, 14850, 24350 }	{ { 15, 1, 5, 1 },{ 15, 1, 3, 1 },{ 7, 1, 3, 1 } }	0	0
32	1	3	1	1	{ 24000, 20450, 31950 }	{ { 21, 1, 5, 1 },{ 15, 1, 5, 1 },{ 15, 1, 3, 1 } }	0	0
48	1	3	2	0	{ { 24000, 21000, 31950 },{ 16000, 15000, 20400 } }	{ { 15, 7, 5, 1 },{ 15, 7, 3, 1 },{ 7, 7, 3, 1 } }	1	0
64	1	3	2	0	{ { 38000, 34050, 56000 },{ 16000, 15600, 20400 } }	{ { 21, 7, 5, 1 },{ 15, 7, 5, 1 },{ 15, 7, 3, 1 } }	1	1
80	1	3	2	0	{ { 46000, 43000, 56000 },{ 24000, 23000, 31950 } }	{ { 21, 7, 5, 1 },{ 15, 7, 5, 1 },{ 15, 7, 3, 1 } }	1	0
96	1	3	3	0	{ { 47000, 42600, 56000 },{ 23000, 22600, 31950 },{ 16000, 15600, 20400 } }	{ { 21, 9, 9, 1 },{ 21, 7, 5, 1 },{ 21, 7, 5, 1 } }	1	0
128	1	3	3	0	{ { 55000, 50000, 56000 },{ 36000, 36000, 56000 },{ 27000, 27000, 31950 } }	{ { 21, 11, 9, 1 },{ 21, 9, 7, 1 },{ 21, 7, 7, 1 } }	1	0
160	1	3	3	0	{ { 74000, 70900, 112000 },{ 41000, 40050, 56000 },{ 35000, 34050, 56000 } }	{ { 21, 11, 11, 1 },{ 21, 9, 9, 1 },{ 21, 7, 7, 1 } }	1	0
192	1	3	3	0	{ { 90000, 87900, 112000 },{ 50000, 48050, 56000 },{ 42000, 41050, 56000 } }	{ { 21, 11, 11, 1 },{ 21, 9, 9, 1 },{ 21, 7, 7, 1 } }	1	0
256	1	3	4	0	{ { 90000, 85000, 112000 },{ 70000, 69000, 112000 },{ 50000, 48950, 56000 },{ 36400, 35600, 56000 } }	{ { 31, 1, 1, 1 },{ 1, 1, 1, 1 },{ 1, 1, 1, 1 } }	1	2
256	2	3	4	0	{ { 84650, 83000, 112000 },{ 65850, 64550, 56000 },{ 47000, 46100, 48000 },{ 28200, 27650, 40000 } }	{ { 31, 11, 11, 1 },{ 1, 1, 1, 1 },{ 1, 1, 1, 1 } }	1	2
256	3	3	4	0	{ { 76300, 73550, 112000 },{ 59350, 57200, 56000 },{ 42400, 40850, 48000 },{ 25450, 24500, 40000 } }	{ { 31, 11, 11, 1 },{ 1, 1, 1, 1 }, { 1, 1, 1, 1 } }	1	2

384	1	3	4	0	{ { 128000, 128000, 128000 },{ 100000, 100000, 128000 },{ 79850, 79850, 104000 },{ 66600, 66600, 104000 } }	{ { 31, 11, 11, 1 },{ 1, 1, 1, 1 },{ 1, 1, 1, 1 } }	1	2
384	2	3	4	0	{ { 128000, 128000, 128000 },{ 105350, 103300, 112000 },{ 75200, 73750, 96000 },{ 45100, 44250, 48000 } }	{ { 31, 11, 11, 1 },{ 1, 1, 1, 1 },{ 1, 1, 1, 1 } }	1	2
384	3	3	4	0	{ { 124300, 121550, 128000 },{ 96700, 94550, 112000 },{ 69050, 67500, 96000 },{ 41450, 40500, 48000 } }	{ { 31, 11, 11, 1 },{ 1, 1, 1, 1 },{ 1, 1, 1, 1 } }	1	2
512	1	3	4	0	{ { 128000, 128000, 128000 },{ 128000, 128000, 128000 },{ 128000, 128000, 128000 }, {118450, 118450, 128000 } }	{ { 31, 1, 1, 1 },{ 1, 1, 1, 1 },{ 1, 1, 1, 1 } }	1	2
512	2	3	4	0	{ { 124000, 124000, 128000 },{ 124000, 124000, 128000 },{ 125200, 118450, 128000 },{ 76300, 73000, 128000 } }	{ { 31, 11, 11, 1 },{ 1, 1, 1, 1 },{ 1, 1, 1, 1 } }	1	2
512	3	3	4	0	{ { 118000, 118000, 128000 },{ 118000, 118000, 128000 },{ 117200, 109250, 128000 },{ 72300, 69000, 128000 } }	{ { 31, 11, 11, 1 },{ 1, 1, 1, 1 },{ 1, 1, 1, 1 } }	1	2

5.4.3.6.11 Quantization and coding process

When DTX is OFF or $W_{vad} == 1$, quantization strategies are read from SPAR bitrate distribution table entry corresponding to the given IVAS bitrate, bandwidth value 3 and SBA analysis order. Then the number of quantization points for PR, CP and D coefficients in each of the quantization strategy is read. All SPAR parameters are uniformly quantized within the quantization range of parameters. The quantization range of PR coefficients (PR_{min}, PR_{max}), CP coefficients (CP_{min}, CP_{max}) and D coefficients (D_{min}, D_{max}) is computed as follows.

$$PR_{min} = \begin{cases} -1.2 & if\ active_w == 1 \\ -1 & if\ active_w == 0 \end{cases}, PR_{max} = \begin{cases} 1.2 & if\ active_w == 1 \\ 1 & if\ active_w == 0 \end{cases} \tag{5.4-60}$$

$$CP_{min} = \begin{cases} -0.8 & if\ active_w == 1 \\ -2 & if\ active_w == 0 \end{cases}, CP_{max} = \begin{cases} 0.8 & if\ active_w == 1 \\ 2 & if\ active_w == 0 \end{cases} \tag{5.4-61}$$

$$D_{min} = \begin{cases} 0 & if\ active_w == 1 \\ 0 & if\ active_w == 0 \end{cases}, D_{max} = \begin{cases} 0.8 & if\ active_w == 1 \\ 1 & if\ active_w == 0 \end{cases} \tag{5.4-62}$$

where $active_w$ is the active W flag that is read from the bitrate distribution table.

The quantized coefficient value and the corresponding index to be coded in metadata bitstream is computed as follows.

$$A^{index} = round\left(\frac{((qlvl_A - 1)\, max(A_{min}, min(A_{max}, A)))}{A_{max} - A_{min}}\right) \tag{5.4-63}$$

$$A^{quantized} = A^{index}\left(\frac{A_{max} - A_{min}}{(qlvl_A - 1)}\right) \tag{5.4-64}$$

where A is a SPAR coefficient (either PR or CP or D) and $qlvl_A$ is the number of quantization points as per the quantization strategy read from SPAR bitrate distribution table.

The quantization and coding of SPAR parameters is then done in the quantization and coding loop and the steps performed in the loop are bitrate dependent as described below.

5.4.3.6.11.1 Quantization and coding loop at all bitrates with FOA input and at bitrates 192 kbps and below with HOA2 and HOA3 input

At these bitrates, the quantization and coding are done in following steps.

Step 1: The number of quantization points, for finest quantization strategy are read as $qlvl_{PR}, qlvl_{CP}, qlvl_{D}$.

Step 2: All SPAR parameters, that is PR, CP and D coefficients, are calculated without quantization as described subclauses 5.4.3.6.5 - 5.4.3.6.8.

Step 3: D coefficients are quantized as per $qlvl_D$ and Eqn (5.4-63), (5.4-64). This is done so that D coefficients are computed as per unquantized PR and CP coefficients and do not make up for the quantization noise in PR and CP coefficients.

Step 4: PR coefficients are quantized as per $qlvl_{PR}$ and Eqn (5.4-63), (5.4-64). At IVAS bitrates 13.2 kbps and 16.4 kbps, the quantized PR coefficients are further modified as per Eqn (5.4-65) and (5.4-66).

Step 5: CP coefficients are recomputed, as described in subclause 5.4.3.6.7, by substituting quantized PR coefficients instead of unquantized PR coefficients in Eqn (5.4-50). This is done so that CP coefficients are computed based on the quantized PR coefficients that are used to generate downmix signals.

Step 6: CP coefficients are quantized as per $qlvl_{CP}$ and Eqn (5.4-63), (5.4-64).

Step 7: All the quantized indices are coded non-differentially with Arithmetic coder as described in subclause 5.4.3.6.15 and the actual SPAR MD bits are computed based on arithmetic coder coded bits, $bits_{arithmetic}$, as

$$MDbits_{act}^{nd\_ent} = spar_{CSbits} + spar_{qbits} + bits_{arithmetic}$$

where $spar_{CSbits} = 3$, $spar_{qbits} = \begin{cases} 2 & if\ ivas\ bitrate < 256\ kbps \\ 1 & otherwise \end{cases}$.

This non-differential arithmetic coding scheme is also referred to as BASE coding scheme.

Step 8: If $MDbits_{act}^{nd\_ent} \le MDbits_{tar}$, then the quantization and coding loop is exited, outputting the MD bitstream. If $MDbits_{act}^{nd\_ent} > MDbits_{tar}$, then loop continues with step 9.

Step 9: All the quantized indices are coded non-differentially with base2 coding, that is implemented using Huffman coder with a near flat probability distribution model, as described in subclause 5.4.3.6.16 and the actual SPAR MD bits are computed based on Huffman coder coded bits, $bits_{base2}$, as

$$MDbits_{act}^{base2} = spar_{CSbits} + spar_{qbits} + bits_{base2}.$$

This non-differential Huffman coding scheme is also referred to as BASE_NOEC coding scheme.

Step 10: If $MDbits_{act}^{base2} \le MDbits_{tar}$, then the quantization and coding loop is exited, outputting the MD bitstream. If $MDbits_{act}^{base2} > MDbits_{tar}$, then loop continues with step 11. Both non-differential coding schemes are first tried before time-differential coding so that the loop can exit if the bitrate from non-differential schemes fit within $MDbits_{tar}$ and time differential coding can be avoided as that improves quality during packet loss concealment at decoder. At bitrates 256 kbps and above, with FOA input, SPAR bitrate distribution table is tuned such that $MDbits_{act}^{base2} \le MDbits_{tar}$ is always true.

Step 11: Processing at this step is bitrate dependent. At IVAS bitrates 24.4 kbps and above, all the quantized indices are coded using a time differential coding scheme (as described in subclause 5.4.3.6.13) with Arithmetic coder as described in subclause 5.4.3.6.15 and the actual SPAR MD bits are computed based on arithmetic coder coded bits, $bits_{arithmetic}$, as

$$MDbits_{act}^{td\_ent} = spar_{CSbits} + spar_{qbits} + bits_{arithmetic}.$$

At IVAS bitrates 13.2 kbps and 16.4 kbps, all the quantized indices are coded using band interleaving and 40 ms MD update rate coding scheme (as described in subclause 5.4.3.6.14) with Arithmetic coder and the actual SPAR MD bits are computed based on arithmetic coder coded bits, $bits_{arithmetic}$, as

$$MDbits_{act}^{td\_ent} = spar_{CSbits} + spar_{qbits} + bits_{arithmetic}.$$

Step 12: If $MDbits_{act}^{td\_ent} \leq MDbits_{tar}$, then the quantization and coding loop is exited, outputting the MD bitstream. If $MDbits_{act}^{td\_ent} > MDbits_{tar}$, then loop continues with step 13.

Step 13: Bitstream with minimum MD bitrate is selected as follows.

$$MDbits_{act} = min\ (MDbits_{act}^{nd\_ent}, MDbits_{act}^{base2}, MDbits_{act}^{td\_ent})$$

If $MDbits_{act} \leq MDbits_{max}$, then the quantization and coding loop is exited, outputting the MD bitstream corresponding to $MDbits_{act}$. If $MDbits_{act} > MDbits_{max}$, then the number of quantization points, for next quantization strategy are read as $qlvl_{PR}, qlvl_{CP}, qlvl_{D}$ and the control goes back to Step 2. At 256 kbps and above, SPAR bitrate distribution table is tuned such that the quantization loop exits at this step.

5.4.3.6.11.2 Quantization and coding loop at all bitrates with FOA input and at bitrates 192 kbps and below with HOA2 and HOA3 input

At IVAS bitrates 256 kbps and 384 kbps with HOA2 and HOA3 input, SBA coding architecture is as described in subclauses 5.4.2.2 and 5.4.2.3. At these bitrates, SPAR MD corresponding to FOA channels is coded for 1 to nB_{foa} frequency bands, where $nB_{foa} = 8$, whereas SPAR MD corresponding to HOA2 and HOA3 channels is coded for 1 to nB_{md} frequency bands. SPAR MD corresponding to FOA channels, for frequency bands $nB_{foa} + 1$ to nB_{md} is computed from DirAC MD as per subclause 5.4.3.7.1. The quantization and coding are done in following steps.

Step 1: The number of quantization points, for finest quantization strategy are read as $qlvl_{PR}, qlvl_{CP}, qlvl_{D}$.

Step 2: All SPAR parameters, including PR (prediction), CP (cross prediction) and D (decorrelation) coefficients, are calculated without quantization as described subclauses 5.4.3.6.5 - 5.4.3.6.8.

Step 3: D coefficients are quantized as per $qlvl_{D}$ and Eqn (5.4-63), (5.4-64). This is done so that D coefficients are computed as per unquantized PR and CP coefficients and do not make up for the quantization noise in PR and CP coefficients.

Step 4: For frequency bands 1 to nB_{foa}, PR coefficients corresponding to FOA channels are quantized as per $qlvl_{PR}$ and Eqn (5.4-63), (5.4-64).

For frequency bands $nB_{foa} + 1$ to nB_{md}, PR coefficients corresponding to FOA channels are neither quantized nor coded as they are generated from DirAC MD.

For frequency bands 1 to nB_{md}, PR coefficients corresponding to HOA2 and HOA3 channels are quantized as per $qlvl_{PR}$ and Eqn (5.4-63), (5.4-64).

Step 5: For frequency bands 1 to nB_{foa}, CP coefficients are recomputed, as described in subclause 5.4.3.6.7, by substituting quantized PR coefficients instead of unquantized PR coefficients in Eqn (5.4-50). This is done so that CP coefficients are computed based on the quantized PR coefficients that are used to generate downmix signals.

For frequency bands $nB_{foa} + 1$ to nB_{md}, CP coefficients are recomputed from a combination of DirAC metadata and input covariance C^{md} given in Eqn (5.4-31). This is done by using DirAC to SPAR converted PR coefficients in Eqn (5.4-50) as described in subclause 5.4.3.6.7.

Step 6: CP coefficients are quantized as per $qlvl_{CP}$ and Eqn (5.4-63), (5.4-64).

Step 7: All the quantized indices are coded non-differentially with Arithmetic and the actual SPAR MD bits are computed based on arithmetic coder coded bits, $bits_{arithmetic}$, as

$$MDbits_{act}^{nd\_ent} = spar_{CSbits} + spar_{qbits} + bits_{arithmetic},$$

where $spar_{CSbits} = 3, spar_{qbits} = 1$.

This non-differential arithmetic coding scheme is also referred to as BASE coding scheme.

Step 8: If $MDbits_{act}^{nd\_ent} \leq MDbits_{tar}$, then the quantization and coding loop is exited, outputting the MD bitstream. If $MDbits_{act}^{nd\_ent} > MDbits_{tar}$, then loop continues with step 9.

Step 9: All the quantized indices are coded non-differentially with base2 coding, that is implemented using Huffman coder with a near flat probability distribution model and the actual SPAR MD bits are computed based on Huffman coder coded bits, $bits_{base2}$, as

$$MDbits_{act}^{base2} = spar_{CSbits} + spar_{qbits} + bits_{base2}.$$

This non-differential Huffman coding scheme is also referred to as BASE_NOEC coding scheme.

Step 10: If $MDbits_{act}^{base2} \leq MDbits_{tar}$, then the quantization and coding loop is exited, outputting the MD bitstream. If $MDbits_{act}^{base2} > MDbits_{tar}$, then loop continues with step 11.

Step 11: All the quantized indices are coded using a time differential coding with Arithmetic and the actual SPAR MD bits are computed based on arithmetic coder coded bits, $bits_{arithmetic}$, as

$$MDbits_{act}^{td\_ent} = spar_{CSbits} + spar_{qbits} + bits_{arithmetic}.$$

Step 12: If $MDbits_{act}^{td\_ent} \leq MDbits_{tar}$, then the quantization and coding loop is exited, outputting the MD bitstream. If $MDbits_{act}^{td\_ent} > MDbits_{tar}$, then loop continues with step 13.

Step 13: From the 3 bitstreams from non-differential arithmetic coding, base2 Huffman coding, time differential arithmetic coding respectively, the one with minimum MD bitrate is selected.

$$MDbits_{act} = \min\left(MDbits_{act}^{nd\_ent}, MDbits_{act}^{base2}, MDbits_{act}^{td\_ent}\right).$$

If $MDbits_{act} \leq MDbits_{max}$, then the quantization and coding loop is exited, outputting the MD bitstream corresponding to $MDbits_{act}$. At 256 kbps and above, SPAR bitrate distribution table is tuned such that the quantization loop exits at this step.

5.4.3.6.11.3 Quantization and coding loop at bitrates 256 kbps and 384 kbps with HOA2 and HOA3 input

At IVAS bitrates 256 kbps and 384 kbps with HOA2 and HOA3 input, SBA coding architecture is as described in subclauses 5.4.2.2 and 5.4.2.3. At these bitrates, SPAR MD corresponding to FOA channels is coded for 1 to nB_{foa} frequency bands, where $nB_{foa} = 8$, whereas SPAR MD corresponding to HOA2 and HOA3 channels is coded for 1 to nB_{md} frequency bands. SPAR MD corresponding to FOA channels, for frequency bands $nB_{foa} + 1$ to nB_{md} is computed from DirAC MD. The quantization and coding are done in following steps.

Step 1: The number of quantization points, for finest quantization strategy are read as $qlvl_{PR}, qlvl_{CP}, qlvl_{D}$.

Step 2: All SPAR parameters, including PR (prediction), CP (cross prediction) and D (decorrelation) coefficients, are calculated without quantization.

Step 3: D coefficients are quantized as per $qlvl_D$ and Eqn (5.4-63), (5.4-64). This is done so that D coefficients are computed as per unquantized PR and CP coefficients and do not make up for the quantization noise in PR and CP coefficients.

Step 4: For frequency bands 1 to nB_{foa}, PR coefficients corresponding to FOA channels are quantized as per $qlvl_{PR}$ and Eqn (5.4-63), (5.4-64).

For frequency bands $nB_{foa} + 1$ to nB_{md}, PR coefficients corresponding to FOA channels are neither quantized nor coded as they are generated from DirAC MD.

For frequency bands 1 to nB_{md}, PR coefficients corresponding to HOA2 and HOA3 channels are quantized as per $qlvl_{PR}$ and Eqn (5.4-63), (5.4-64).

Step 5: For frequency bands 1 to nB_{foa}, CP coefficients are recomputed by substituting quantized PR coefficients instead of unquantized PR coefficients in Eqn (5.4-50). This is done so that CP coefficients are computed based on the quantized PR coefficients that are used to generate downmix signals.

For frequency bands $nB_{foa} + 1$ to nB_{md}, CP coefficients are recomputed from a combination of DirAC metadata and input covariance C^{md} given in Eqn (5.4-31). This is done by using DirAC to SPAR converted PR coefficients in Eqn (5.4-50).

Step 6: CP coefficients are quantized as per $qlvl_{CP}$ and Eqn (5.4-63), (5.4-64).

Step 7: All the quantized indices are coded non-differentially with Arithmetic coder and the actual SPAR MD bits are computed based on arithmetic coder coded bits, $bits_{arithmetic}$, as

$$MDbits_{act}^{nd\_ent} = spar_{CSbits} + spar_{qbits} + bits_{arithmetic}$$

where $spar_{CSbits} = 3, spar_{qbits} = 1$.

This non-differential arithmetic coding scheme is also referred to as BASE coding scheme.

Step 8: If $MDbits_{act}^{nd\_ent} \leq MDbits_{tar}$, then the quantization and coding loop is exited, outputting the MD bitstream. If $MDbits_{act}^{nd\_ent} > MDbits_{tar}$, then loop continues with step 9.

Step 9: All the quantized indices are coded non-differentially with base2 coding, that is implemented using Huffman coder with a near flat probability distribution model and the actual SPAR MD bits are computed based on Huffman coder coded bits, $bits_{base2}$, as

$$MDbits_{act}^{base2} = spar_{CSbits} + spar_{qbits} + bits_{base2} .$$

This non-differential Huffman coding scheme is also referred to as BASE_NOEC coding scheme.

Step 10: If $MDbits_{act}^{base2} \leq MDbits_{tar}$, then the quantization and coding loop is exited, outputting the MD bitstream. If $MDbits_{act}^{base2} > MDbits_{tar}$, then loop continues with step 11.

Step 11: All the quantized indices are coded using a time differential coding scheme (as described in subclause 5.3.3.5.13**) with Arithmetic coder and the actual SPAR MD bits are computed based on arithmetic coder coded bits, $bits_{arithmetic}$, as

$$MDbits_{act}^{td\_ent} = spar_{CSbits} + spar_{qbits} + bits_{arithmetic}.$$

Step 12: If $MDbits_{act}^{td\_ent} \leq MDbits_{tar}$, then the quantization and coding loop is exited, outputting the MD bitstream. If $MDbits_{act}^{td\_ent} > MDbits_{tar}$, then loop continues with step 13.

Step 13: From the 3 bitstreams from non-differential arithmetic coding, base2 Huffman coding, time differential arithmetic coding respectively, the one with minimum MD bitrate is selected.

$$MDbits_{act} = \min\,(MDbits_{act}^{nd\_ent}, MDbits_{act}^{base2}, MDbits_{act}^{td\_ent})$$

If $MDbits_{act} \leq MDbits_{max}$, then the quantization and coding loop is exited, outputting the MD bitstream corresponding to $MDbits_{act}$. At 256 kbps and above, SPAR bitrate distribution table is tuned such that the quantization loop exits at this step.

5.4.3.6.11.4 Quantization and coding loop at bitrates 256 kbps and 384 kbps with HOA2 and HOA3 input

At IVAS bitrates 256 kbps and 384 kbps with HOA2 and HOA3 input, SBA coding architecture is as described in subclauses 5.4.2.2 and 5.4.2.3. At these bitrates, SPAR MD corresponding to FOA channels is coded for 1 to nB_{foa} frequency bands, where $nB_{foa} = 8$, whereas SPAR MD corresponding to HOA2 and HOA3 channels is coded for 1 to nB_{md} frequency bands. SPAR MD corresponding to FOA channels, for frequency bands $nB_{foa} + 1$ to nB_{md} is computed from DirAC MD. The quantization and coding are done in following steps.

Step 1: The number of quantization points, for finest quantization strategy are read as $qlvl_{PR}, qlvl_{CP}, qlvl_{D}$.

Step 2: All SPAR parameters, including PR (prediction), CP (cross prediction) and D (decorrelation) coefficients, are calculated without quantization.

Step 3: D coefficients are quantized as per $qlvl_{D}$ and Eqn (5.4-63), (5.4-64). This is done so that D coefficients are computed as per unquantized PR and CP coefficients and do not make up for the quantization noise in PR and CP coefficients.

Step 4: For frequency bands 1 to nB_{foa}, PR coefficients corresponding to FOA channels are quantized as per $qlvl_{PR}$ and Eqn (5.4-63), (5.4-64).

For frequency bands $nB_{foa}+1$ to nB_{md}, PR coefficients corresponding to FOA channels are neither quantized nor coded as they are generated from DirAC MD.

For frequency bands 1 to nB_{md} , PR coefficients corresponding to HOA2 and HOA3 channels are quantized as per $qlvl_{PR}$ and Eqn (5.4-63), (5.4-64).

Step 5: For frequency bands 1 to nB_{foa} , CP coefficients are recomputed by substituting quantized PR coefficients instead of unquantized PR coefficients in Eqn (5.4-50). This is done so that CP coefficients are computed based on the quantized PR coefficients that are used to generate downmix signals.

For frequency bands $nB_{foa}+1$ to nB_{md} , CP coefficients are recomputed from a combination of DirAC metadata and input covariance C^{md} given in Eqn (5.4-31). This is done by using DirAC to SPAR converted PR coefficients in Eqn (5.4-50).

Step 6: CP coefficients are quantized as per $qlvl_{CP}$ and Eqn (5.4-63), (5.4-64).

Step 7: All the quantized indices are coded non-differentially with Arithmetic coder and the actual SPAR MD bits are computed based on arithmetic coder coded bits, $bits_{arithmetic}$, as

$$MDbits_{act}^{nd\_ent} = spar_{CSbits} + spar_{qbits} + bits_{arithmetic},$$

where $spar_{CSbits} = 3$ and $spar_{qbits} = 1$.

This non-differential arithmetic coding scheme is also referred to as BASE coding scheme.

Step 8: If $MDbits_{act}^{nd\_ent} \leq MDbits_{tar}$, then the quantization and coding loop is exited, outputting the MD bitstream. If $MDbits_{act}^{nd\_ent} > MDbits_{tar}$, then loop continues with step 9.

Step 9: All the quantized indices are coded non-differentially with base2 coding, that is implemented using Huffman coder and the actual SPAR MD bits are computed based on Huffman coder coded bits, $bits_{base2}$, as

$$MDbits_{act}^{base2} = spar_{CSbits} + spar_{qbits} + bits_{base2} .$$

This non-differential Huffman coding scheme is also referred to as BASE_NOEC coding scheme.

Step 10: If $MDbits_{act}^{base2} \leq MDbits_{tar}$, then the quantization and coding loop is exited, outputting the MD bitstream. If $MDbits_{act}^{base2} > MDbits_{tar}$, then loop continues with step 11.

Step 11: All the quantized indices are coded using a time differential coding scheme (as described in subclause 5.3.3.5.13**) with Arithmetic coder and the actual SPAR MD bits are computed based on arithmetic coder coded bits, $bits_{arithmetic}$, as

$$MDbits_{act}^{td\_ent} = spar_{CSbits} + spar_{qbits} + bits_{arithmetic} .$$

Step 12: If $MDbits_{act}^{td\_ent} \leq MDbits_{tar}$, then the quantization and coding loop is exited, outputting the MD bitstream. If $MDbits_{act}^{td\_ent} > MDbits_{tar}$, then loop continues with step 13.

Step 13: From the 3 bitstreams from non-differential arithmetic coding, base2 Huffman coding, time differential arithmetic coding respectively, the one with minimum MD bitrate is selected.

$$MDbits_{act} = \min\,(MDbits_{act}^{nd\_ent}, MDbits_{act}^{base2}, MDbits_{act}^{td\_ent})$$

If $MDbits_{act} \leq MDbits_{max}$, then the quantization and coding loop is exited, outputting the MD bitstream corresponding to $MDbits_{act}$. At 256 kbps and above, SPAR bitrate distribution table is tuned such that the quantization loop exits at this step.

5.4.3.6.11.5 Quantization and coding loop at bitrates 512 kbps with HOA2 and HOA3 input

At IVAS bitrates 512 kbps with HOA2 and HOA3 input, SBA coding architecture is as described in subclause 0. At this bitrate, DirAC MD is not converted to SPAR MD at any frequency band and SPAR MD corresponding to all the SBA analysis channels, for frequency bands 1 to nB_{md} is computed, quantized and coded in metadata bitstream. The quantization and coding loop is same as described in subclause 5.4.3.6.11.1.

5.4.3.6.12 Spectral filling in the side channels at 13.2 kbps and 16.4 kbps

At 13.2 and 16.4 kbps, PR and D coefficients may be quantized to 0 even when unquantized PR coefficients are non-zero and the variance of corresponding side channel input to SPAR is also non-zero. This happens due to fewer number of quantization points and large quantization steps and can lead to spectral holes in the side channels at the output of up-mixer at decoder. To avoid spectral holes due to quantization of PR and D coefficients, quantized PR coefficients are further modified as follows.

For each b in range 1 to nB_{md} and for each j in range 2 to $\boldsymbol{N_{spar\_ana}}$.

$$PR_{j-1}^{quantized}(b) = \begin{cases} q_{step} & if\ PR_{j-1}(b) > 0\ , C_{[1,j]}^{md}(b)\ != \ 0 \\ -q_{step} & if\ PR_{j-1}(b) < 0\ , C_{[1,j]}^{md}(b)\ != \ 0 \\ PR_{j-1}^{quantized}(b), & otherwise \end{cases} \tag{5.4-65}$$

where $q_{step} = \frac{PR_{max} - PR_{min}}{(qlvl_{PR} - 1)}$.

$$PR_{j-1}^{index}(b) = \begin{cases} 1 & if\ PR_{j-1}(b) > 0\ , C_{[1,j]}^{md}(b)\ != \ 0 \\ -1 & if\ PR_{j-1}(b) < 0\ , C_{[1,j]}^{md}(b)\ != \ 0 \\ PR_{j-1}^{index}(b), & otherwise \end{cases} \tag{5.4-66}$$

5.4.3.6.13 Time differential coding

SPAR uses time differential coding scheme to encode quantized indices of SPAR coefficients. Table 5.4-9 shows the 4 modes of time differential coding scheme, that is 4a, 4b, 4c, 4d, here “base” refers to non-differential coding schemes. Each mode codes 3/4th of the total bands differentially while remaining bands are coded non-differentially. More particularly, the band specified by 0 is coded non time differentially and the band specified by 1 is coded time differentially. The time differential index is generated by taking the difference between quantized parameter index in current frame and quantized parameter index of the previous frame, applying modulo $qlvl_A$ operation on the difference value as described in Eqn (5.4-67), (5.4-68), (5.4-69) and (5.4-70), here $qlvl_A$ is the number of quantization points as per the quantization strategy read from SPAR bitrate distribution table.

Set of bands to be coded time differentially changes on frame-by-frame basis, Table 5.4-10 shows the mapping of time differential coding mode between previous frame and current frame. All 4 modes of time differential coding are such that complete recovery is possible within 4 frames on the decoder side in the event of packet loss. Coding scheme is set to “base” for the first frame.

Table 5.4-9: Interleaved time differential coding schemes

Coding Scheme	Time Diff Coding, Bands 1-12	Time Diff Coding, Bands 1-8
base or base_noec	0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0
4a	0 1 1 1 0 1 1 1 0 1 1 1	0 1 1 1 0 1 1 1
4b	1 0 1 1 1 0 1 1 1 0 1 1	1 0 1 1 1 0 1 1
4c	1 1 0 1 1 1 0 1 1 1 0 1	1 1 0 1 1 1 0 1
4d	1 1 1 0 1 1 1 0 1 1 1 0	1 1 1 0 1 1 1 0

Table 5.4-10: Mapping of the time differential coding schemes

Previous frame's coding Scheme	Current frame's time differential coding scheme
base or base_noec	4a
4a	4b
4b	4c
4c	4d
4d	4a

5.4.3.6.13.1 Time differential coding across same quantization strategies

Time differential index is computed in two steps. First the difference between current frame's quantized parameter index and previous frame's quantized parameter index is computed as follows:

$$A_{TD}^{index} = A_{current}^{index} - A_{previous}^{index} \tag{5.4-67}$$

Then modulo $qlvl_A$ is performed to keep the index range as $\{A_{min}^{index}, A_{max}^{index}\}$

$$A_{TD}^{index} = \begin{cases} A_{TD}^{index} + qlvl_A & if\ A_{TD}^{index} < A_{min}^{index} \\ A_{TD}^{index} - qlvl_A & if\ A_{TD}^{index} > A_{max}^{index}\ , \\ A_{TD}^{index} & otherwise \end{cases} \tag{5.4-68}$$

where $A_{min}^{index} = round\left(\frac{((qlvl_A-1)*\mathrm{A}_{min})}{A_{max}-A_{min}}\right), A_{max}^{index} = round\left(\frac{((qlvl_A-1)*\mathrm{A}_{max})}{A_{max}-A_{min}}\right)$, A refers to PR, CP and D coefficients and $qlvl_A$ is the number of quantization points as per the current frame's quantization strategy read from SPAR bitrate distribution table. $A_{max} and\ A_{min}$ are the minimum and maximum quantized values as per the current frame's quantization strategy as per Eqn 5.4-60, 5.4-61) and 5.4-62). A_{TD}^{index} is then coded with arithmetic coder.

5.4.3.6.13.2 Time differential coding across different quantization strategies

If, for the same IVAS bitrate, the quantization strategy in the previous frame is different than the quantization strategy in the current frame then the quantized indices from previous frame are first mapped to the quantization strategy of the current frame. This allows time-differential coding between frames without resorting to having to send a non-time-differential frame each time the quantization scheme is changed. The mapping of the indices is performed is done as follows.

$$A_{previous,mapped}^{index} = round\left(\frac{A_{previous}^{index}(qlvl_{A,current}-1)}{qlvl_{A,previous}-1}\right) \tag{5.4-69}$$

Then the difference between current frame's quantized parameter index and previous frame's mapped quantized parameter index is computed as follows.

$$A_{TD}^{index} = A_{current}^{index} - A_{previous,mapped}^{index} \tag{5.4-70}$$

Then modulo $qlvl_A$ is performed as per Eqn (5.4-68)

where $A_{min}^{index} = round\left(\frac{((qlvl_A-1)*\mathrm{A}_{min})}{A_{max}-A_{min}}\right), A_{max}^{index} = round\left(\frac{((qlvl_A-1)*\mathrm{A}_{max})}{A_{max}-A_{min}}\right)$, A refers to PR, CP and D coefficients and $qlvl_A$ is the number of quantization points as per the current frame's quantization strategy read from SPAR bitrate distribution table. $A_{max} and\ A_{min}$ are the minimum and maximum quantized values as per the current frame's quantization strategy as per Eqn 5.4-60, 5.4-61) and 5.4-62). A_{TD}^{index} is then coded with arithmetic coder.

5.4.3.6.13.3 Time differential coding across different quantization strategies

If, for the same IVAS bitrate, the quantization strategy in the previous frame is different than the quantization strategy in the current frame then the quantized indices from previous frame are first mapped to the quantization strategy of the current frame. This allows time-differential coding between frames without resorting to having to send a non-time-differential frame each time the quantization scheme is changed. The mapping of the indices is performed is done as follows.

$$A_{previous,mapped}^{index} = round\left(\frac{A_{previous}^{index}(qlvl_{A,current}-1)}{qlvl_{A,previous}-1}\right) \quad (5.4\text{-}69)$$

Then the difference between current frame's quantized parameter index and previous frame's mapped quantized parameter index is computed as follows.

$A_{TD}^{index} = A_{current}^{index} - A_{previous,mapped}^{index}$. After the modulo operation, A_{TD}^{index} is coded with arithmetic coder.

5.4.3.6.14 Band interleaving and 40ms MD update rate at 13.2 kbps and 16.4 kbps

At 13.2 and 16.4 kbps, SPAR operates at a 40ms metadata update rate, with occasional 20ms metadata updates where permissible by bitrate limitations. For the first frame, or frames where the metadata can be coded within the target metadata bitrate budget, all bands are coded with one of the two non-differential coding schemes as described in subclause 5.4.3.6.11.1. For frames where this limitation is not met, metadata corresponding to half of the bands is sent is current frame, followed by the other half in the next, and so on. These interleaved bands are coded with non-differential arithmetic coding scheme. The choice of which bands to send or omit in each frame is interleaved as shown in Table 5.4-11, here with 40 ms update rate, bands labelled as B are sent in one frame whereas bands labelled as A are sent in second frame. Coding of band interleaved modes in done by reusing time-differential coding modes as shown by the mapping between time-differential coding modes and band interleaved modes in Table 5.4-12.

Table 5.4-11: Frequency band interleaved 40ms metadata update rate coding scheme

	SPAR, Bands 1 to nB_{md}
20 ms update rate frames	Y Y Y Y
40 ms update rate frame	B A B A

Table 5.4-12: Mapping of time differential coding modes to band interleaved coding modes

Coding Scheme	**SPAR, Bands 1 to nB_{md}**
4a	Arithmetic coding of only the A bands.
4b	Arithmetic coding of only the B bands.
4c	Arithmetic coding of only the A bands.
4d	Arithmetic coding of only the B bands.

5.4.3.6.15 Entropy coding with Arithmetic coders

SPAR uses arithmetic coders to entropy-code the quantized indices. Non-differential indices are coded with different probability distribution models as compared to time differential indices. Table 5.4-13, Table 5.4-14 and Table 5.4-15 contains the probability distribution models for PR, CP and D coefficients in time differential and non-differential mode. There are four probability distribution models corresponding to each $qlvl$ that is read from SPAR bitrate distribution table. One out of the four probability distribution models are chosen dynamically on frame-by-frame basis based on the actual symbol distribution in the quantized indices. First, an estimate of coded bits is generated with each probability distribution model as shown below.

For each i in range $1 \leq i \leq 4$

$$bit_i = -1 * \sum_{k=1}^{k=length_{indices}}\left(dist_{curr,k}\log_2\left(\frac{max(\varepsilon, dist_{model,i,k+1})}{\sum_{j=2}^{j=length_{model,i}} dist_{model,i,j}}\right)\right) \quad (5.4\text{-}71)$$

where $length_{indices}$ is the total number of indices to code as per Table 5.4-13, Table 5.4-14 and Table 5.4-15, $dist_{curr,k}$ is the number of occurrences of the kth index in the symbol distribution of quantized SPAR parameter indices in current frame, $dist_{model,i,j}$ is the value of jth index of ith probability distribution model.

Then, the probability distribution model with minimum value of bit_i is selected to code the quantized indices and i is coded with 2 bits in the bitstream as part of Arithmetic coder bits.

Table 5.4-13: Probability distribution models for PR coefficients indices

qlvls from SPAR bitrate distribution table	Active w from SPAR bitrate distribution table	Models for non-differential entropy coding	Models for time differential entropy coding	Indices to code
7	0	{model 1}, {model 2}, {model 3}, {model 4}	{model 1}, {model 2}, {model 3}, {model 4}	-3,-2,-1,0,1,2,3
15	0	{model 1}, {model 2}, {model 3}, {model 4}	{model 1}, {model 2}, {model 3}, {model 4}	-7,-6,-5,-4,-3,-2,-1,0,1,2,3,4,5,6,7
21	0	{model 1}, {model 2}, {model 3}, {model 4}	{model 1}, {model 2}, {model 3}, {model 4}	-10,-9,-8,-7,-6,-5,-4,-3,-2,-1,0,1,2,3,4,5,6,7,8,9,10
31	0	{model 1}, {model 2}, {model 3}, {model 4}	{model 1}, {model 2}, {model 3}, {model 4}	-15,-14,-13,-12,-11,-10,-9,-8,-7,-6,-5,-4,-3,-2,-1,0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15
7	1	{model 1}, {model 2}, {model 3}, {model 4}	{model 1}, {model 2}, {model 3}, {model 4}	-3,-2,-1,0,1,2,3
15	1	{model 1}, {model 2}, {model 3}, {model 4}	{model 1}, {model 2}, {model 3}, {model 4}	-7,-6,-5,-4,-3,-2,-1,0,1,2,3,4,5,6,7
21	1	{model 1}, {model 2}, {model 3}, {model 4}	{model 1}, {model 2}, {model 3}, {model 4}	-10,-9,-8,-7,-6,-5,-4,-3,-2,-1,0,1,2,3,4,5,6,7,8,9,10

Table 5.4-14: Probability distribution models for CP coefficients indices

qlvls from SPAR bitrate distribution table	Models for non-differential entropy coding	Models for time differential entropy coding	Indices to code
7	{model 1}, {model 2}, {model 3}, {model 4}	{model 1}, {model 2}, {model 3}, {model 4}	-3,-2,-1,0,1,2,3
9	{model 1}, {model 2}, {model 3}, {model 4}	{model 1}, {model 2}, {model 3}, {model 4}	-4,-3,-2,-1,0,1,2,3,4
11	{model 1}, {model 2}, {model 3}, {model 4}	{model 1}, {model 2}, {model 3}, {model 4}	-5,-4,-3,-2,-1,0,1,2,3,4,5

Table 5.4-15: Probability distribution models for D coefficients indices

qlvls from SPAR bitrate distribution table	Models for non-differential entropy coding	Models for time differential entropy coding	Indices to code in non-differential mode	Indices to code in differential mode
3	{model 1}, {model 2}, {model 3}, {model 4}	{model 1}, {model 2}, {model 3}, {model 4}	0,1,2	-1,0,1
5	{model 1}, {model 2}, {model 3}, {model 4}	{model 1}, {model 2}, {model 3}, {model 4}	0,1,2,3,4	-2,-1,0,1,2
7	{model 1}, {model 2}, {model 3}, {model 4}	{model 1}, {model 2}, {model 3}, {model 4}	0,1,2,3,4,5,6	-3,-2,-1,0,1,2,3
9	{model 1}, {model 2}, {model 3}, {model 4}	{model 1}, {model 2}, {model 3}, {model 4}	0,1,2,3,4,5,6,7,8	-4,-3,-2,-1,0,1,2,3,4
11	{model 1}, {model 2}, {model 3}, {model 4}	{model 1}, {model 2}, {model 3}, {model 4}	0,1,2,3,4,5,6,7,8,9,10	-5,-4,-3,-2,-1,0,1,2,3,4,5

5.4.3.6.16 Base2 coding with Huffman coder

Huffman coding is done with a near flat probability distribution model such that bits coded with Huffman coder are always less than OR equal to bits coded with base2 coding. Table 5.4-16, Table 5.4-17 and Table 5.4-18 contains the Huffman codebook for every qlvl value that can be read from SPAR bitrate distribution table. Every entry in codebook has three fields {quantized index value, bits for the index, code for the index}.

Table 5.4-16: Huffman codebook for PR coefficients indices

qlvls from SPAR bitrate distribution table	codebook
7	{…}
15	{…}
21	{…}
31	{…}

Table 5.4-17: Huffman codebook for CP coefficients indices

qlvls from SPAR bitrate distribution table	codebook
7	{…}
9	{…}
11	{…}

Table 5.4-18: Huffman codebook for D coefficients indices

qlvls from SPAR bitrate distribution table	codebook
3	{…}
5	{…}
7	{…}
9	{…}
11	{…}

5.4.3.7 DirAC and SPAR parameter merge

SBA coding uses a combination of DirAC and SPAR coding as described in Fig 5.4-2, 5.4-3 and 5.3-4**. For FOA channels, SPAR and DiRAC parameters are integrated based on a frequency split based approach wherein SPAR metadata is generated for frequency band index 1 to 8 whereas DirAC metadata is generated for frequency band index 9 to 12. At 13.2 kbps and 16.4 kbps SPAR bands are merged to 4 bands and DirAC bands are merged to 2 bands to reduce the overall metadata bitrate as described in subclauses 5.4.3.4 and 0. When DTX is ON and $W_{vad} = 0$, SPAR bands are merged to 2 frequency bands and DirAC bands are merged to 2 frequency bands to reduce the overall metadata bitrate as described in subclause 5.4.9. In order to generate SPAR downmix and upmix matrix in DirAC bands, DirAC metadata is converted to SPAR metadata as described in subclause 5.4.3.7.1.

For HOA2 and HOA3 channels, SPAR and DiRAC parameters are integrated based on a channel split based approach wherein at 256kbps and 384 kbps, SPAR metadata is generated for planar HOA2 and planar HOA3 channels whereas DirAC metadata is generated for remaining channels. At 512 kbps, SPAR metadata is generated for HOA2 and planar HOA3 channels whereas DirAC metadata is generated for remaining channels. Input channel indices for metadata analysis in SPAR and DirAC are given in Table 5.4-2.

5.4.3.7.1 DirAC to SPAR parameter conversion

DirAC to SPAR metadata conversion is done by first estimating input covariance matrix from quantized DirAC parameters and quantized SPAR decorrelation (D) coefficients. Then SPAR MD (PR, CP and D coefficients) is computed from covariance matrix as per subclause 5.4.3.6.4.

5.4.3.7.2 Covariance estimation from DirAC parameters (1)

An estimate of covariance matrix of the FOA component of the ambisonics input is generated using quantized DirAC parameters. The covariance estimation depends on number of downmix channels N_{dmx} and VAD flag W_{vad}.

Covariance estimation when $N_{dmx} \leq 2$ and $W_{vad} = 1$

For each DirAC band, covariance is estimated as

$$C_{i,j}^{DiRAC} = \begin{cases} (1-\psi)^2\gamma_i\,\gamma_j + (1-(1-\psi)^2)D_{i-1}^{norm} & if\ i = j\,, i > N_{dmx} \\ (1-\psi)^2\gamma_i\,\gamma_j + \frac{(1-(1-\psi)^2)}{3} & if\ i = j\,, 1 < i \leq N_{dmx} \\ 1 & if\ i = j = 1 \\ (1-\psi)\gamma_i\gamma_j & if\ i \neq j \end{cases} \quad (5.4\text{-}72)$$

Covariance estimation when $N_{dmx} \leq 2$ and $W_{vad} = 0$

For each DiRAC band, covariance is estimated as

$$C_{i,j}^{DiRAC} = \begin{cases} (1-\psi)^2\gamma_i\gamma_j + \frac{(1-(1-\psi)^2)}{3} & if\ i = j\,, i > 1 \\ 1 & if\ i = j = 1 \\ (1-\psi)\gamma_i\,\gamma_j & if\ i \neq j \end{cases} \quad (5.4\text{-}73)$$

Covariance estimation when $N_{dmx} > 2$

For each DiRAC band, covariance is estimated as

$$C_{i,j}^{DiRAC} = \begin{cases} (1-\psi)\gamma_i\gamma_j + \left(\frac{\psi}{3}\right) & if\ i = j\,, i > 1 \\ 1 & if\ i = j = 1 \\ (1-\psi)\gamma_i\gamma_j & if\ i \neq j \end{cases} \quad (5.4\text{-}74)$$

where i and j are FOA channel indices ranging from 1 to 4, Ψ is the quantized DirAC diffuseness parameter , D_{i-1}^{norm} is the normalized D coefficient as given in Eqn (5.4-76). Computation of (5.4-84) γ_i is as follows.

$$\gamma_i = \frac{\sum_{m=1}^{m=N_{sf}} \gamma_{m,i}}{N_{sf}} \tag{5.4-75}$$

where $\gamma_{m,i}$ is the spherical harmonic response for first order ambisonics with ACN ordering and SN3D normalization such that $\gamma_{m,1} = 1$, $\gamma_{m,2} = \sin\theta_m \cos\emptyset_m$, $\gamma_{m,3} = \sin\emptyset_m$,$\gamma_{m,4} = \cos\theta_m \cos\emptyset_m$, m is the subframe index with in a frame, N_{sf} is the number of subframes corresponding to which DirAC metadata is quantized and coded in IVAS bitstream, θ_m is quantized DirAC azimuth angle in subframe m, and $\emptyset_m$ is quantized DirAC elevation angle in subframe m.

Normalized D coefficient D_k^{norm} is computed as follows.

$$D_k^{norm} = \frac{D_k^2(nB_{md})}{(\sum_{k=1}^{k=(4-N_{dmx})} D_k^2(nB_{md}))*\frac{3}{\min(3,\max(0,4-N_{dmx}))}}, 1 \le k \le (4 - N_{dmx}) \tag{5.4-76}$$

where $D_k(nB_{md})$ are the SPAR D coefficients in the last SPAR band nB_{md} as computed in Eqn (5.4-57).

5.4.3.7.2.1 Covariance estimation from DirAC parameters (2)

The inter-channel covariance matrix is required to calculate the SPAR parameters and the downmix matrix (cf. clause 5.4.3.6.4). It is calculated using two different approaches: For the 8 lower parameter bands of the MDFT filterbank (cf. clause xxx), they are calculated directly from the audio channels of the input signal (cf. clause 5.4.3.6.1).

For the 4 higher parameter bands, a model-based approach is employed. This enables to re-use the acoustic model parameters from the DirAC encoder according to clause 5.4.3.4. They are transmitted in the bitstream for these bands. The need to transmit additional metadata for the upmix in these bands can be avoided.

The calculation of the prediction and decorrelation coefficients from the covariance matrix according to clause 5.4.3.6.4 is performed independently on the encoder and decoder side.

Specifically, the calculation of the covariance matrix is based on the real spherical harmonics. Its matrix elements are given by

$$C_{x;y;z,w} = E^{\mathrm{dir}} Y_{0,0}(\theta_D) Y_{1,-1;0;1}(\theta_D) \tag{5.4-77}$$

and

$$C_{w,w} = E^{\mathrm{dir}} Y_{0,0}(\theta_D) Y_{0,0}(\theta_D) + E_w^{\mathrm{diff}}. \tag{5.4-78}$$

This result can be understood from the model of a single point source panned in the ambisonics domain to the DoA given by the angle θ_D. The inter-channel covariance is then given by

$$C_{x;y;z,w} = \int dt\, s^2(t) Y_{0,0}(\theta_D) Y_{1,-1;0;1}(\theta_D) \tag{5.4-79}$$

where s(t) is the time-dependent scalar-valued sound signal emitted by the point source. The signal energy appears due to the integrals over the audio signal s(t). Using the diffuseness parameter Ψ, these energies can be eliminated and expressed by the total energy, yielding:

$$C_{w,w} = (1-\Psi)\, E\, Y_{0,0}(\theta_D) Y_{0,0}(\theta_D) + \Psi\, E \tag{5.4-80}$$

and for the channels for the degree l = 1

$$C_{x;y;z,x;y;z} = (1-\Psi)\, E\, Y_{1,-1;0;1}(\theta_D) Y_{1,-1;0;1}(\theta_D) + \frac{\Psi}{3} E, \tag{5.4-81}$$

and for the channels for the degree l = 2

$$C_{j,j} = (1-\Psi)\, E\, Y_j(\theta_D) Y_j(\theta_D) + \frac{\Psi}{5} E, \tag{5.4-82}$$

and for those of the degree l = 3

$$C_{j,j} = (1-\Psi)\, E\, Y_j(\theta_D) Y_j(\theta_D) + \frac{\Psi}{7} E, \tag{5.4-83}$$

The ratios of the directional and diffuse energies E^{dir} and E^{diff} to the total energy E follows from the diffuseness parameter of the psychoacoustic model (directional audio coding, see clause 5.4.3.4). The index j is a combined index of the real spherical harmonic with and degree l and index m.

The covariance matrix, hence, depends on these model parameters and the downmix matrix, which is employed to transform the audio channels. It is calculated in a signal adaptive way based on model parameters derived from the input audio channels.

Note that the metadata of the acoustic model are quantized prior to the application of Equations (5.4-77) to (5.4-83). This is necessary to ensure exact agreement of the covariance matrices calculated on the encoder and decoder side. They are also encoded in the metadata encoder according to clause 5.2.4.

The calculation of the covariance and downmix matrices is performed individually on each parameter band of the filterbank-domain representation obtained from the analysis filterbank according to clause 5.4.3.3. The above processing is applied to the high frequency bands. For the low frequency bands the mixing matrix is derived differently: the prediction coefficients and decorrelation coefficients are quantized and written to the bitstream directly.

For the low-order operation mode, only a single DoA angle θ_D is estimated by a single parameter estimator. Then this is used in Equations (5.4-77) to (5.4-83). In the high-order operation mode, two sector DoA angles are estimated. In this case, the simple model of a single point source is still employed in the covariance estimation. For a single point source, the two sector DoAs are equal. The DoA of the zeroth sector is used in Eqs. (5.4-77) to (5.4-83).

For configurations with 2 or less channels in the first set of transport channels, the diffuse term in $C_{j,j}$ is renormalized by a factor $P_j^{fact} = \frac{P_j * P_j}{\sum_i P_i P_i} N_{diff}$, where P_i are the SPAR coefficients for the decorrelated signals (cf. clause 180) and N_{diff} the norm of the diffuse energy divided by the number of decorrelator channels:

$$N_{diff} = 3 / \min(3, \max(0, \mathrm{N_{FOA}} - \mathrm{N_{DM}})) \quad \text{for } l=1,$$

$$N_{diff} = 5 / \min(5, \max(0, \min(\mathrm{N}_{TC}, \mathrm{N}_{HOA2}) - \mathrm{N_{DM}})) \quad \text{for } l=2,$$

$$N_{diff} = 7 / \min\big(7, \max(0, \mathrm{N}_{TC} - \mathrm{N_{DM}})\big) \quad \text{for } l=3,$$

Where N_{TC} is the number of channels in the second set of transport channels, $\mathrm{N_{FOA}}$ is the number of FOA channels and $\mathrm{N_{DM}}$ is the number of channels in the first set of transport channels.

Then the covariance matrices from the direct evaluation of the signal (low frequency bands) and the model-based approach (high frequency bands) are combined to obtain the SPAR coefficients and downmix matrices.

5.4.3.7.3 Directional diffuseness

The diagonal elements in the covariance matrix corresponding to the side channels of FOA signal, as given in Eqn (5.4-72), are the sum of directional component, that is computed as a product of energy ratio and spherical harmonics response, and diffuse component in DirAC MD, which is the normalized DirAC diffuseness parameter.

At low bitrates where $N_{dmx} \leq 2$, the diffuse component is computed by normalizing the diffused energy from DirAC with normalized SPAR D coefficients D_k^{norm} and the covariance matrix is estimated based on quantized DirAC parameters and quantized SPAR D coefficients as given in Eqn (5.4-72).

5.4.3.7.4 SPAR parameter estimation from DirAC parameters

PR, CP and D coefficients corresponding to FOA channels and DirAC frequency bands, also referred to as PR^{DirAC}, CP^{DirAC} and D^{DirAC} , are computed by replacing C^{md} matrix with C^{DirAC} matrix in subclauses 5.4.3.6.5 - 5.4.3.6.8. DirAC to SPAR converted parameters are not quantized as they are computed based on quantized DirAC parameters that are transmitted to the decoder.

5.4.3.7.5 Active W prediction from DirAC parameters

In DirAC frequency bands, when active W prediction mode is enabled, primary downmix channel W' is generated by scaling the FOA signal and adding them together as per Eqn 5.4-33). The active W gains ($s_{w,}$ $s_y, s_z,$ s_x) are computed from quantized DirAC metadata parameters by computing the prediction downmix matrix given in Eqn (5.4-36) from the covariance matrix, that is computed in Eqn (5.4-72). The parameters of prediction downmix matrix given in Eqn (5.4-36) are estimated from DirAC parameters as given follows.

$$\hat{u} = \begin{pmatrix} \gamma_2 \\ \gamma_3 \\ \gamma_4 \end{pmatrix}, \alpha = (1 - \psi), U = C^{DiRAC}_{[2:4,2:4]}$$

5.4.4 Downmix matrix calculation

5.4.4.1 SPAR parameters to downmix matrix conversion

In SPAR bands, that is band index 1 to nB_{md}, the downmix matrix, $dmx_{[N_{dmx}\,x\,4]}$, is generated as follows. First, the prediction downmix matrix is generated as per Eqn (5.4-50) but with quantized PR coefficients, $PR^{quantized}_{[\boldsymbol{N_{PR}}\,x\,1]}(b)$, as follows.

$$PRdmx_{[\boldsymbol{N_{spar\_ana}}x\boldsymbol{N_{spar\_ana}}]}(b) = \begin{pmatrix} 1 & Zero_{[1x\,\boldsymbol{N_{PR}}]} \\ -\,PR^{quantized}_{[\boldsymbol{N_{PR}}\,x\,1]}(b) & I_{[\boldsymbol{N_{PR}}x\,\boldsymbol{N_{PR}}]} \end{pmatrix} \begin{pmatrix} W_{scale} & W_{scale}F\hat{u}_{[1x\boldsymbol{N_{PR}}]}{}^{H} \\ Zero_{[\boldsymbol{N_{PR}}\,x\,1]} & I_{[\boldsymbol{N_{PR}}x\,\boldsymbol{N_{PR}}]} \end{pmatrix} \quad (5.4\text{-}84)$$

At 13.2 kbps and 16.4 kbps, in band interleaving mode, half of the frequency bands of $PRdmx$ are generated from previous frame's matrix as described in subclause 5.4.4.3.

Then $PRdmx$ is remixed as per Eqn (5.4-54). After remixing, $PRdmx$ is limited to actual number of downmix channels N_{dmx} as shown below:

$$dmx''_{[i,j]} = PRdmx_{[i,j]} \quad (5.4\text{-}85)$$

where $1 \le i \le N_{dmx}$ and $1 \le j \le 4$.

Then band unmixing is performed on dmx'' as follows:

$$dmx(i + (b1 - 1)bw_{bands}) = dmx''(b1) \quad (5.4\text{-}86)$$

where $1 \le b1 < nB_{md}$ and $1 \le i \le bw_{bands}$.

In DirAC bands, the downmix matrix, $dmx_{[N_{dmx}\,x\,4]}$, is generated as follows. First, the prediction downmix matrix is generated as per Eqn (5.4-50) but with PR coefficients, $PR^{DirAC}_{[\boldsymbol{N_{PR}}\,x\,1]}(b)$, as follows.

$$PRdmx_{[4x4]}(b) = \begin{pmatrix} 1 & Zero_{[1x\,3]} \\ -\,PR^{DirAC}_{[3\,x\,1]}(b) & I_{[3x\,3]} \end{pmatrix} \begin{pmatrix} W_{scale} & W_{scale}F\hat{u}_{[1x3]}{}^{H} \\ Zero_{[3\,x\,1]} & I_{[3x\,3]} \end{pmatrix} \quad (5.4\text{-}87)$$

where $PR^{DirAC}_{[\boldsymbol{N_{PR}}\,x\,1]}(b)$, W_{scale}, F and $\hat{u}_{[1x3]}$ are computed by replacing C^{md} matrix with C^{DirAC} matrix in subclauses 5.4.3.6.5 - 5.4.3.6.8.

Then $PRdmx$ is remixed as per Eqn (5.4-54). After remixing, $PRdmx$ is limited to actual number of downmix channels $\boldsymbol{N_{dmx}}$ as shown in (5.4-85) to generate $, dmx(b)$ in DirAC bands.

In one channel downmix mode, that is $\boldsymbol{N_{dmx}} = 1$, the $PRdmx$ that is computed in Eqn (5.4-87) is further scaled to compensate for the energy difference between C^{md} and C^{DirAC} as described in subclause 5.4.4.2.

5.4.4.2 Energy compensation in DirAC bands

In DirAC bands, when number of downmix channels is one ($\boldsymbol{N_{dmx}} = 1$), primary downmix channel W' is further scaled with an energy compensation factor to prevent spatial collapse by scaling the downmix signal such that the upmixed signal is better energy matched with respect to the input. The scaling is computed in each DirAC band index b as shown below.

$$scale(b) = \max\left(1, min\left(2, \sqrt{tr({C_{norm}}^{in}_{[4x4]}(b))/max\,(\varepsilon, tr(C^{DirAC}_{[4x4]}\,(b))))}\right)\right) \quad (5.4\text{-}88)$$

where ${C_{norm}}^{in}_{[4x4]} = C^{in}_{[4x4]}/\max\,(\varepsilon, C^{in}_{[1,1]})$ and $C^{in}_{[4x4]}$is the input covariance matrix as computed in Eqn (5.4-17).

The prediction downmix matrix, that is computed in Eqn (5.4-87), is then scaled as follows.

$$PRdmx_{[1,i]}(b) = PRdmx_{[1,i]}(b)\, scale(b) \qquad (5.4\text{-}89)$$

where $1 \leq i \leq 4$.

5.4.4.3 Band Interleaving of downmix matrix at 13.2 and 16.4 kbps

When band interleaving mode or 40 ms metadata update rate mode is triggered at 13.2 and 16.4 kbps, the downmix matrix is also generated in band interleaving fashion. The downmix matrix corresponding to the bands that are coded in the metadata bitstream in the current frame, is generated using the quantized prediction coefficients as described in Eqn (5.4-84). The downmix matrix corresponding to the bands that are not coded in the metadata bitstream in the current frame is generated using previous frame's downmix matrix as follows.

$$PRdmx_{[\boldsymbol{N_{spar\_ana}} x \boldsymbol{N_{spar\_ana}}]}(b) = PRdmx^{prev}_{[\boldsymbol{N_{spar\_ana}} x \boldsymbol{N_{spar\_ana}}]}(b) \;, \qquad (5.4\text{-}90)$$

where b is the band index corresponding to which SPAR parameters are not coded in the bitstream in current frame and $PRdmx^{prev}_{[\boldsymbol{N_{spar\_ana}} x \boldsymbol{N_{spar\_ana}}]}$ is the previous frame's downmix matrix.

5.4.5 Downmix generation

5.4.5.1 Time domain to MDFT conversion

Input signals which need to be modified in the parametric downmix process are run through the SPAR filterbank which employs the MDFT for fast convolution (overlap-save) with the SPAR filters. In case only one downmix channel is transmitted, all FOA signals are transformed into the MDFT domain for active downmix processing. Otherwise only the X, Z, Y signals are MDFT transformed for residual computation. Note that the path-through components in the downmix matrix for the W signal are implemented as pure delays according to the SPAR filterbank delay of 1ms and the residual computation happens in the time domain for computational efficiency. The actual filterbank processing happens in the MDFT domain by multiplication with the weighted complex MDFT transforms of the SPAR filters. The filterbank synthesis is realized by the summation of all weighted SPAR filter contributions in the MDFT domain. To ensure smooth transition between frames the downmix signal for the current frame is created twice; one time with previous frame parameters and another time with current frame parameters. The final downmix is computed by crossfading the 2 downmix signals in the time domain.

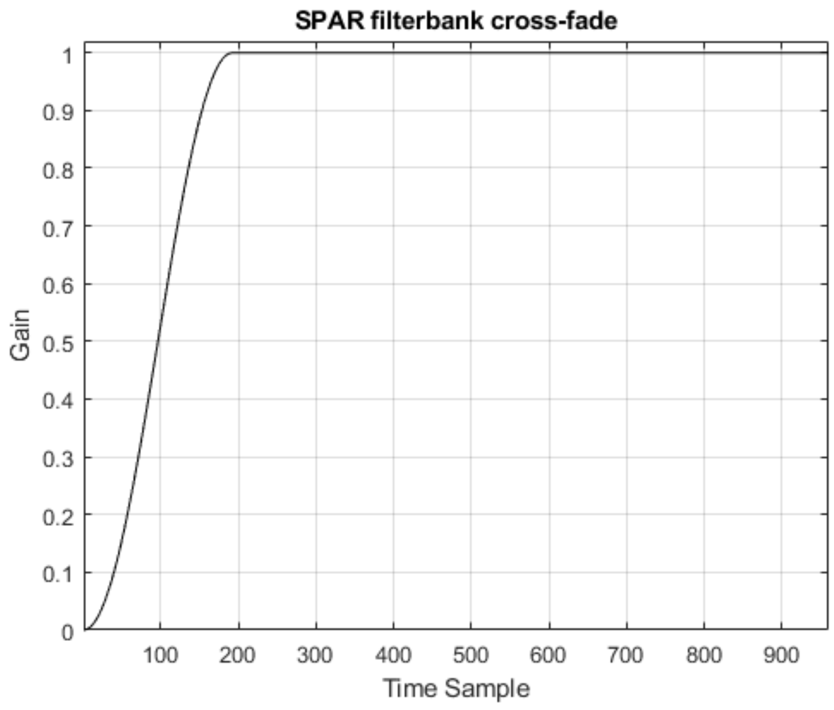
5.4.5.2 Filterbank mixing

The downmixing is performed using the downmix matrix calculated according to clause XX. For each TF tile of the MDFT filterbank, the downmix matrix is pre-multiplied to the vector of input audio channels. Via this downmixing, the second set of transport channels is transformed into a first smaller one:

$$\boldsymbol{x}_{DM}(k,n) = \boldsymbol{D}(k,n)\, \boldsymbol{x}_{HOA}(k,n) \qquad (5.4\text{-}91)$$

Where $\boldsymbol{x}_{DM}(k,n)$ is the downmix signal, i.e., the vector of the channels in the second set of transport channels, and $\boldsymbol{x}_{HOA}(k,n)$ is the HOA input signal. A subset of the ambisonics input channels in $\boldsymbol{x}_{HOA}(k,n)$ is selected for the downmixing depending on the bitrate. For bitrates < 256 kbps, only the first-order ambisonics channels are processed. For bitrates $\boldsymbol{D}(k,n)$ is the downmix matrix, which is computed in a signal-adaptive way for each TF tile indexed by k and n.

5.4.5.3 Windowing and crossfading



5.4.5.4 MDFT Filter Bank Synthesis

5.4.6 Principle Component Analysis (PCA)

5.4.7 Automatic Gain Control (AGC)

AGC aims to reduce audio artifacts caused by an overload condition of an audio signal to be encoded by a target audio codec. An overload condition exists when the audio signals exceed the range expected by the target audio encoder. As SPAR downmixing of Ambisonics input channels may lead to some or more of the downmix channels exceeding the range of the input channels, the IVAS SBA coding mode provides an AGC mechanism for handling of potential overload cases. AGC specified here is a perceptually motivated overload handling without the need for an additional processing delay, i.e., zero delay AGC. AGC for IVAS SBA operates on the generated downmixed signals, $s_i^{DMX\_enc}(n), 1 \leq i \leq N_{dmx}$, described in subclause 5.4.5, which are to be encoded by the core encoder described in subclause 5.4.8, acting as a target audio encoder. Note that AGC is only applied to a single downmixed channel case, $N_{dmx} = 1$. The audio signals expected by the core encoder range within the interval $[-32768, 32767]$.

If an overload condition exists, AGC determines a gain parameter $e(j)$, of the current frame j to be encoded, and a gain transition function to be applied to the current frame samples/signals. The gain transition function is constructed comprising two portions, i.e., a transitory portion and a steady-state portion. The transitory portion is constructed by scaling a prototype smoothing function $p(\cdot)$ by a gain scaling factor obtained from the current and previous frame gain parameters, $e(j) - e(j-1)$. This portion corresponds to a transition from the gain parameter of the previous frame to the current frame. When gain control is applied, it either takes the form of gain increases or decreases over a portion of samples of the current frame. The steady-state portion is merely a constant gain value specified by the last index (right-most) of the transitory portion.

To achieve zero additional delay processing, AGC defines the length of transitory portion based on the overall core codec delay. Given the overall core codec delay $D = 12ms$, corresponding to the total core encoder and decoder delays, $8.75ms$ and $3.25ms$, respectively, and the frame length of $L = 20ms$, the transitory portion length L_p is set to $L - D = 8ms$. This is equal to setting $L = 960$, $D = 576$, and $L_p = 384$ samples at $48kHz$ sampling frequency.

The gain transition function $t(l, j)$ at a sample index l in frame j is thus defined as

$$t(l,j) = \begin{cases} p(l,j)^{(e(j)-e(j-1))}, & l = 0 \ldots L_p - 1 \\ p(L_p - 1, j)^{(e(j)-e(j-1))}, & l = L_p \ldots L - 1. \end{cases}$$

The prototype smoothing function $p(\cdot)$ is defined below depending on a gain transition step size $DBSTEP$ in dB:

$$p(DBSTEP, l, j) = 1 + a(DBSTEP) \cdot \left(\cos\left(\pi \cdot \frac{l}{L_p - 1}\right) - 1\right),$$

where

$$a(DBSTEP) = 0.5 \cdot \left(1 - 10^{\frac{DBSTEP}{20}}\right).$$

$DBSTEP$ has been determined based on subjective and objective quality criteria; thus, it is directly related to the SCE/CPE and MCT core codecs of the IVAS codec. It is set to a predefined value of $DBSTEP = -6\ dB$. Figure 5.4.7-1 depicts the gain transition function at $48kHz$ sampling frequency having positive gain scaling factors.

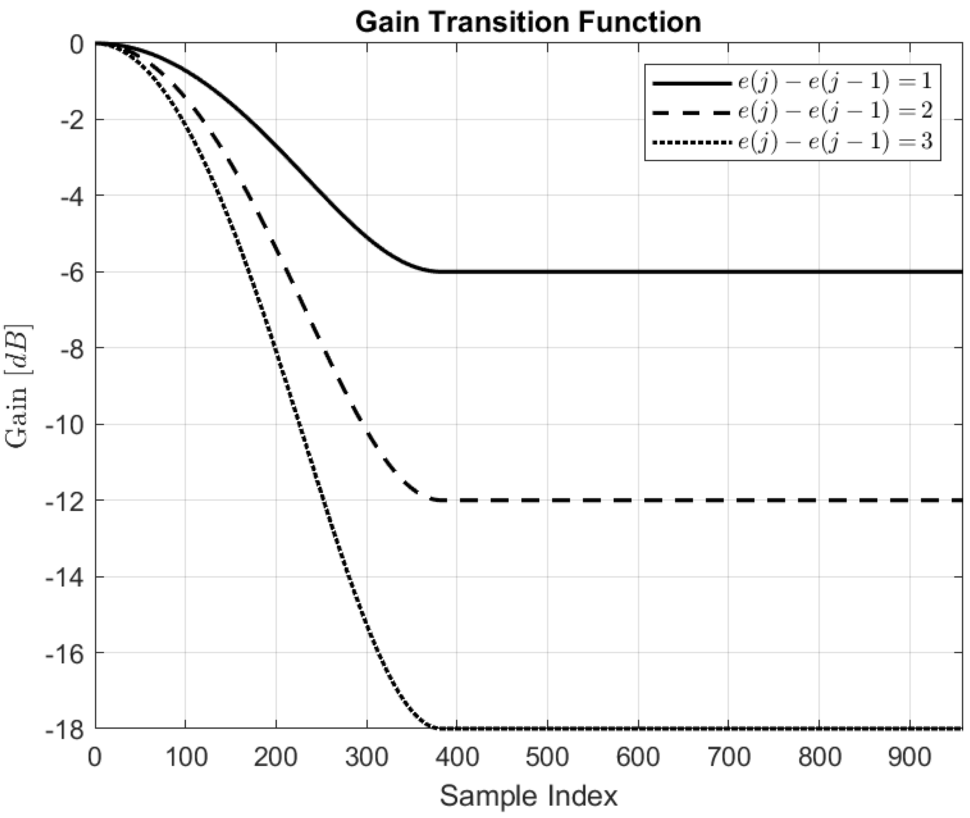


Figure 5.4.7-1 Gain transition function having the scaling factor $e(j) - e(j-1)$ set to 1, 2 and 3.

The following equation defines the relevant relation for constructing the current gain transition function based on the gain transition function applied to the preceding frame $t(DBSTEP, L-1, j-1)$, and where the gain transition step size plays the determining role in adjusting the transition.

$$t(DBSTEP, l, j) = t(l,j) \cdot t(DBSTEP, L-1, j-1), \quad l = 0 \cdots L - 1,$$

where $t(DBSTEP, L-1, j-1)$ is initialized to 1.

Consequently, applying the gain transition function results in the gain adjusted frame $s_i^{AGC\_enc}(n)$, of the downmixed signal $s_i^{DMX\_enc}(n)$. Gain attenuation or amplification results in an attenuated or amplified frame of input signals, respectively. Finally, setting the sample index l to n, and introducing the downmix channel index $i = 1$, indicating a single downmixed channel, gives the following formulation:

$$s_{i=1}^{AGC\_enc}(n) = t(DBSTEP, n, j) \cdot s_{i=1}^{DMX_{enc}}(n),$$

where the gain adjusted frame $s_{i=1}^{AGC\_enc}(n)$ is then encoded by the core encoder.

AGC encodes gain control information in the dedicated AGC metadata bitstream consisting of several bit fields. Firstly, AGC always encodes one bit field indicating the presence or absence of further AGC metadata in the bitstream. If gain control is applied in the current frame, this bit is set to 1 and further AGC metadata is written to the bitstream as described below. If the gain control is not applied, it is set to 0, corresponding to applying a unity gain to the downmix channel, and no further bits are written.

Secondly, based on the information that a gain control is applied in the current frame, AGC encodes the gain parameter $e(j)$ using a field of $\beta_E = ceil(\log_2(E_{MIN} + E_{MAX} + 1))$ bits, where $E_{MAX} = 0$ and $E_{MIN} = \max\big(3, ceil(\log_2(1.5 \cdot N_{HOA}))\big)$, representing the maximum and minimum gain parameter attenuation range, respectively. The term N_{HOA} ideally signifies the amount of HOA components, from which the downmixed signal is calculated. It is set to an FOA case having $N_{HOA} = 4$ components, resulting in β_E set to 2 bits. This gain parameter is encoded using a binary encoding scheme. For a given $\beta_E = 2$ bits, $e(j)$ takes a value from the set $\{0,1,2,3\}$ corresponding to the binary code $\{(00),(01),(10),(11)\}$.

[It is suggested to described AGC specific adaptations in this section]

5.4.8 Core Encoding

Editor's note: SBA specific Bit rate distribution in MCT at high bitrates

5.4.9 DTX operation

In SBA mode, the metadata frame is flagged as inactive frame if W_{vad} is set to 0. The VAD of core coder and front-end VAD, described in subclause 5.3.3.1, are tuned such that if W_{vad} is set to 1, then core coder VAD will be set to 1 as well. However, if W_{vad} is set to 0, then core coder VAD depends on downmix signal and is independent of W_{vad}. As a result, in some frames, SPAR MD may be coded in inactive mode while the frame (MD + core coder) may be coded in active mode. W_{vad} is coded with 1 bit, in SBA metadata bitstream, to indicate if MD is coded in active mode or inactive mode.

A frame is coded as SID or NO DATA frame if W_{vad} is set to 0 and core coder is operating in SID or NO_DATA frame mode.

In inactive mode, SBA MD is coded in four frequency bands with SPAR MD in two low frequency bands and DirAC MD in two high frequency bands.

5.4.9.1 DirAC parameter estimation

5.4.9.2 SPAR parameter estimation

In SPAR inactive coding, the current MD frame is set as active frame or inactive frame based on output of VAD detector W_{vad}, described in subclause "5.3.3.1". The metadata frame is flagged as inactive frame if W_{vad} is set to 0, then a downmix signal and upmix metadata is computed from the FOA input signal to SPAR, where the downmix signal has either one channel or two channels. The upmix metadata is computed in inactive mode such that the downmix signal can be upmixed into FOA signal at the output of the upmixer at decoder. In inactive mode, SPAR parameters PR and D coefficients are computed, quantized and coded in IVAS bitstream. The inactive coding in SPAR is different from active coding in terms of band grouping, covariance smoothing, and quantization and coding of SPAR parameters as described below.

5.4.9.2.1 Banded covariance smoothing

The SPAR metadata is computed based on inter-channel covariance matrix which is obtained by performing temporal smoothing on banded covariance, computed in Eqn (5.4-16**), over multiple frames. The covariance matrix is tracked separately for inactive frames and the temporal smoothing is performed as per Eqn (5.4-17**), where the forgetting factor $(1 - \alpha(b))$ is shorter than that of active frame and the temporal smoothing in inactive frame is higher than that of active frame. The forgetting factor is frequency dependent as given in Eqn (5.4-92). If a frame is identified as a transient, then the covariance of that frame is removed from the temporal smoothing calculation as per Eqn (5.4-92).

$$\alpha(b) = \begin{cases} 1, & if\ frameIdx_{prev} = -1 \\ 0, & if\ T_0 = 1\ OR\ T_1 = 1 \\ \alpha_{sm}(b), & otherwise \end{cases} \quad (5.4\text{-}92)$$

Where,

$$\alpha_{sm}(b) = \min\left(0.4, \frac{\left(\sum_{k=\boldsymbol{bin_{start}}(b)}^{k=\boldsymbol{bin_{end}}(b)} H_{abs,5ms}^{mdft}(b,k)\right) b\ R_{sm_{fact}}}{40}\right) \quad (5.4\text{-}93)$$

$$R_{sm_{fact}} = \begin{cases} 0.5, & if\ ivas\ bit\ rate < 24.4kbps \\ 0.75, & if\ ivas\ bit\ rate \geq 24.4kbps \end{cases} \quad (5.4\text{-}94)$$

Smoothed covariance is then band remixed as per subclause 5.3.3.5.3.

5.4.9.2.2 Quantization and coding in VAD inactive frames

SPAR coefficients are computed as follows. First PR coefficients are computed as per subclause 5.3.3.5.5 and 5.3.3.5.6. CP coefficients are set to 0 as mentioned in subclause 5.3.3.5.7. Then D coefficients are computed as per subclause 5.3.3.5.8. When number of downmix channels N_{dmx} is set to1 then D coefficients are further boosted as follows.

$$D_i(b) = \begin{cases} 0.4, & if\ D_i(\mathrm{b}) > 0.1, D_i(\mathrm{b}) < 0.4, N_{dmx} = 1 \\ D_i(\mathrm{b}), & otherwise \end{cases}$$

Then D coefficients are quantized using uniform quantization given in Eqn (5.4-63**) and (5.4-64**) using the parameters given below.

$$D_{min} = 0, D_{max} = 1.6, qlvl_D = 7$$

Then PR coefficients are quantized as per PR_{min} and PR_{max} given in Eqn (5.4-60**), and the $qlvl_{PR}$ is set based on number of downmix channels and channel priority as follows.

$$qlvl_{PR}^{i} = \begin{cases} 7, & if\ N_{dmx} = 2, i = 3 \\ 9, & otherwise \end{cases}$$

Where, i is the channel index and $i = 3$ corresponds to Z channel as per ACN ordering.

PR coefficients are then quantized using uniform quantization given in Eqn (5.4-63**) and (5.4-64**).

SPAR coefficients are coded in pairs where each PR coefficients that is quantized with 9 quantization points is paired with D coefficient that is quantized with 7 quantization points. Together they form 63 quantization points that are coded with base2 coding using 6 bits. When $N_{dmx} = 2,$ PR coefficients corresponding to Z channel are coded separately with 3 bits.

5.4.9.3 SPAR and DirAC parameter merge

In inactive MD frames, SBA MD is coded for 4 frequency bands out of which SPAR MD is computed for band index b with $1 \leq b \leq 2$ and DirAC MD is computed for band index b with $3 \leq b \leq 4$.The DirAC MD in DirAC bands is converted to SPAR MD as given in subclause 5.4.3.7.

5.4.10 SBA bitrate switching

Bitrate switching is supported from any IVAS bitrate to any other IVAS bitrate. As the configurations of all IVAS components depend strongly on the bitrate, these configurations must be checked and modified accordingly when a bitrate switch is requested. At every frame, it is checked whether the current bitrate R equals that of the previous frame R_{old}. If this is not the case, a re-configuration is triggered.

Then the new SBA analysis order (cf. clause 5.4.3.4) is computed first. If it differs from the previous one, the number of channels analyzed changes and HP20 filter memories are allocated or deallocated to match the new number of channels.

Then the SPAR encoder is re-configured. Specifically, the number of transport channels in the first set (cf. clauses 5.4.4 and 5.4.5) is re-computed according to table I. From this number, the number of CPEs or SCEs follows and is set accordingly. For the case of a single transport channel and one SCE, the core nominal bitrate is set depending on the total IVAS bitrate. When the number of transport channels in the first set changes on the bitrate switch, more changes are made to the SPAR encoder configuration: The number of MDFT filterbank memories in both the parameter-estimation and signal path and the dimensions of the downmix matrix are adjusted (cf. clauses 5.4.3.3 and 5.4.4). The size of the buffers for the covariance matrix and the SPAR coefficients is also adapted to the new number of channels. The covariance smoothing factor is changed depending on the bitrate. If PCA is enabled with the new but not the old bitrate, the necessary initialization is performed.

The DirAC encoder is reconfigured too. First, the decision between the high-order and low-order operation mode (cf. clause 5.4.3.4) is made based on the new bitrate. The encoding unit can switch between a low-order and a high-order operation mode. The low-order mode is active for bitrates less than 384 kbps. The high-order mode is switched on for the bitrates 384 and 512 kbps.

In the low-bitrate mode, the encoding path for the second sector is deactivated and no information for the second sector is written to the bitstream as side information. The first sector then has an omni-directional spatial filter. In the high-bitrate mode, the encoder paths for both sectors are switched on. The parameters for both sectors are contained in the metadata. This is realized by setting a variable to indicate whether the encoder runs in the high- or low-order operation mode.

In addition, the number of parameter bands and DoAs is re-computed. The buffer sizes for the DirAC metadata are adjusted to these new values. If necessary, the buffers are freed and re-allocated with the correct dimensions. It is determined whether the parameter estimation will be performed on the encoder side for all bands (high-order operation mode) or only for the high bands (low-order operation mode). The grouping of the parameter bands and the time resolution of the parameters is set up according to the bitrate. If the number of parameters bands or DoAs changes, the buffers for the quantized energy ratio indices are initialized with zeros.

Finally, the core-coder is reconfigured to reflect the changes in the number of transport channels in the first set and the bitrate.

5.5 Metadata-assisted spatial audio (MASA) operation

5.5.1 MASA format overview

The metadata-assisted spatial audio (MASA) operation encodes the IVAS encoder inputs that use the MASA format. This is a parametric spatial audio format that can be used with any multi-microphone array with suitable capture analysis. The MASA format is optimised for immersive audio capture by smartphones and other form factors that may utilize irregular microphone arrays.

The MASA format is based on audio channels and a set of metadata parameters. The audio signals can be one or two, i.e., mono or stereo. These can be denoted as mono-MASA (MASA1) and stereo-MASA (MASA2), respectively. The metadata parameters include spatial metadata parameters providing information about the audio scene for spatial audio reproduction, and descriptive metadata parameters providing further description about the capture configuration and source format of the spatial audio content.

Each MASA metadata frame, corresponding to 20 ms of audio, includes the descriptive metadata (consisting of a format descriptor and a channel audio format field that further defines the number of directions described by the spatial metadata, number of audio channels, the source format configuration, and a variable description depending on the previous information) and the spatial metadata parameters that are: direction index, direct-to-total energy ratio, diffuse-to-total energy ratio, remainder-to-total energy ratio, spread coherence, and surround coherence.

The direction index (decodable with an elevation and an azimuth component) provides an efficient representation of the multitude of possible spatial directions with about 1-degree accuracy in any arbitrary direction. The direction indices define a spherical grid that covers a sphere with several smaller spheres with centres of the spheres giving the points corresponding with the directions.

Each spatial metadata parameter is given for each of 96 time-frequency (TF) tiles corresponding to 4 temporal subframes and 24 frequency bands. The direct-to-total energy ratio and spread coherence parameters are associated with the direction (parameter). The direction index, direct-to-total energy ratio, and spread coherence parameters are therefore given for each direction described per TF tile (as given by the number of directions descriptive metadata parameter).

For each TF tile, the sum of the different energy ratio parameters is 1.0.

The metadata for MASA format inputs shall be provided as defined in detail in Annex A of [11].

The uncompressed MASA metadata size according to definitions in [11] is between 272.8 and 426.4 kbps (268.8 and 422.4 kbps for spatial metadata only), depending on the number of directions in spatial metadata. As IVAS supports encoding of both mono-MASA and stereo-MASA at IVAS bitrates between 13.2 and 512 kbps, as described in Table 4.2-1, significant compression of the MASA metadata is performed during the encoding process. The compression is based on several overall strategies, including a bitrate dependent configuration for MASA metadata, simplification of the spatial metadata based on combining of spatial metadata parameters in time or frequency, and selecting the method of compression based on the configuration parameter indicating the source format.

5.5.2 MASA format metadata input

5.5.2.1 Obtaining the MASA format metadata

5.5.2.2 Direction index deindexing

5.5.2.3 TF tile based energy calculation

The MASA format encoding algorithms take time-frequency domain transport audio signal energy as an input. This the computation of the energy values is practical to do once as a pre-processing step. This computation is done in time-frequency domain (namely CLDFB) and provides an energy parameter value for each of the 96 time-frequency tiles corresponding to the time-frequency resolution of the associated MASA spatial metadata parameters. The computation of the energy is carried out with the equation

$$E(b,m) = \sum_{k=k_{low}(b)}^{k_{high}(b)} \sum_{i} \left| S_i^{CLDFB}(k,m) \right|^2$$

where $S_i^{CLDFB}(k,m)$ is the CLDFB-domain transformed (see clause (TODO: ref) for description of the CLDFB transform) received transport audio signal $s(m)$, b is the MASA frequency band, m is the time subframe, i is the transport audio signal channel index, k is the frequency bin index of the CLDFB-domain, and $k_{low}(b)$ and $k_{high}(b)$ set the range of CLDFB-domain frequency bins that belong to each MASA frequency band.

The computed energy parameter values are stored into an internal encoder structure to be available for retrieval by further encoding processes.

5.5.3 Coding of MASA format input data

5.5.3.1 Overview

The coding of MASA input format data starts with the encoding of the metadata. The number of bits of the metadata is variable, even within one operation point, but it is limited to a highest allowed value, set for each operational mode. In order to respect the maximum number of bits allowed, the metadata parameters are reduced before encoding. Some of the parameter reductions are dependent on the operation mode, others are explicitly signalled. For each frame the encoded metadata, including the signalling bits, is written at the end of the bitstream. The remaining bits are used for the transport channels encoded as a CPE (see xxx) for stereo-MASA or as a SCE (see xxx) for mono-MASA.

5.5.3.2 MASA metadata pre-encoding processing and signalling

5.5.3.2.1 Overview

5.5.3.2.2 Bitrate dependent parameter settings

Table 5.5-1: MASA parameters coding configuration

Bitrate	Metadata max bits		Nr. Subbands B		Nr. subands with 2 Directions	
		LR mode		Joined sf mode		Joined sf mode
13.2	50	50	5	5	0	0
16.4	60	50	5	5	0	0
24.4	70	60	5	5	0	0
32	85	70	5	8	0	0
48	140		5	12	0	0
64	180		5	12	1	2
80	220		5	12	1	2
96	256		5	18	1	3
128	350		8	18	3	4
160	432		12	18	4	6
192	528		18	18	6	8
256	832		24	24	6	9
384	1024		24	24	9	12
512	1300		24	24	24	24

5.5.3.2.3 Energy ratio compensation

5.5.3.2.4 Merging of MASA spatial parameter metadata across MASA frequency bands and subframes

This clause describes the merging of MASA spatial audio parameter sets across frequency bands and temporal subframes.

When merging is performed across MASA frequency bands, a metric is determined as the number of coding frequency bands (numCodingBands TODO: ADD REF TO CONFIG). This metric is then used to determine whether to merge the spatial audio parameter sets for two or more MASA frequency bands into a merged spatial parameter set. The merged spatial parameter set (over MASA frequency bands) corresponds to a single coding frequency band.

The decision to merge is based is based on whether the metric is smaller than the threshold value of the number of frequency bands in the MASA format (i.e., 24).

The merging is performed for the spatial audio parameter sets of one or more MASA frequency bands, resulting in fewer spatial audio parameter sets than the number of MASA frequency bands. The number of spatial audio parameter sets remaining after the merging process is aligned to the number of coding frequency bands.

The spatial audio parameters sets of consecutive MASA frequency bands are merged by initially converting the azimuth $\theta(b_M,m,i)$ and elevation $\phi(b_M,m,i)$ angles to Cartesian coordinate vectors. The length of the vectors is determined based on the direct-to-total energy ratio $r_{dir}(b_M,m,i)$ and the energy $E(b_M,m)$. This is performed for MASA spatial audio parameter sets which are to be merged by

$$x(b_M,m,i) = \cos\theta(b_M,m,i)\cos\phi(b_M,m,i)\, l(b_M,m,i)$$

$$y(b_M,m,i) = \sin\theta(b_M,m,i)\cos\phi(b_M,m,i)\, l(b_M,m,i)$$

$$z(b_M,m,i) = \sin\phi(b_M,m,i)\, l(b_M,m,i)$$

where

$$l(b_M,m,i) = r_{dir}(b_M,m,i)E(b_M,m)$$

where b_M is the MASA frequency band index, m is the subframe index, and i is the direction index.

The merging process involves mapping a number of MASA frequency bands to a coding frequency band, and then merging the MASA spatial audio parameter sets associated with each of the mapped MASA frequency bands into a single spatial audio parameter set corresponding to the coding frequency band. This is performed, for each mapping of MASA frequency bands to coding frequency band, by summing the vectors and energies corresponding to the MASA frequency bands of the mapping

$$x_{sum}(b,m,i) = \sum_{b_M=b_{M,1}(b)}^{b_{M,2}(b)} x(b_M,m,i)$$

$$y_{sum}(b,m,i) = \sum_{b_M=b_{M,1}(b)}^{b_{M,2}(b)} y(b_M,m,i)$$

$$z_{sum}(b,m,i) = \sum_{b_M=b_{M,1}(b)}^{b_{M,2}(b)} z(b_M,m,i)$$

$$E_{sum}(b,m) = \sum_{b_M=b_{M,1}(b)}^{b_{M,2}(b)} E(b_M,m)$$

where $b_{M,1}(b)$ is the first mapped MASA frequency band of the coding frequency band b and $b_{M,2}(b)$ the last mapped MASA frequency band.

The merged direction for the coding frequency band b is then determined by

$$\theta(b,m,i) = \arctan(y_{sum}(b,m,i), x_{sum}(b,m,i))$$

$$\phi(b,m,i) = \arctan(z_{sum}(b,m,i), \sqrt{x_{sum}(b,m,i)^2 + y_{sum}(b,m,i)^2})$$

where $\arctan(\,,)$ is a variant of arcus tangent that can determine the right quadrant.

Then, the initial merged direct-to-total energy ratio corresponding to the coding frequency band b is determined by

$$r'_{dir}(b,m,i) = \frac{\sqrt{x_{sum}(b,m,i)^2 + y_{sum}(b,m,i)^2 + z_{sum}(b,m,i)^2}}{E_{sum}(b,m)}$$

The merged spread coherence corresponding to the coding frequency band b is determined by

$$\zeta(b,m,i) = \frac{\sum_{b_M=b_{M,1}(b)}^{b_{M,2}(b)} \zeta(b_M,m,i)E(b_M,m)}{E_{sum}(b,m)}$$

The initial merged surround coherence corresponding to the coding frequency band b is determined by

$$\gamma'(b,m) = \frac{\sum_{b_M=b_{M,1}(b)}^{b_{M,2}(b)} \gamma(b_M,m)E(b_M,m)}{E_{sum}(b,m)}$$

In the case the number of coding bands (i.e., numCodingBands) is equal to the number of frequency bands in the MASA format (i.e., 24), the frequency combination need not be performed. The input values are used in the subsequent processing directly, i.e.,

$$\theta(b,m,i) = \theta(b_M,m,i)$$

$$\phi(b,m,i) = \phi(b_M,m,i)$$

$$r'_{dir}(b,m,i) = r_{dir}(b_M,m,i)$$

$$\zeta(b,m,i) = \zeta(b_M,m,i)$$

$$\gamma'(b,m) = \gamma(b_M,m)$$

$$E_{sum}(b,m) = E(b_M,m)$$

It is next checked whether the mergeRatiosOverSubframes flag is enabled TODO: ADD REF TO CONFIG. In the case mergeRatiosOverSubframes is enabled, the direct-to-total energy ratio and surround coherence values for different subframes (there are four subframes m in the MASA metadata) are merged over subframes, resulting in merged direct-to-total energy ratio and surround coherence values for the frame. I.e., in this case, the frame contains only one coding subframe. The merging of subframes is performed separately for all coding frequency bands b. The other parameters are not merged over subframes.

The merging of spatial parameter values over subframes is performed by first summing the energies over subframes

$$E_{sum}(b) = \sum_{m=0}^{3} E_{sum}(b,m)$$

Then, the surround coherence is merged over subframes by

$$\gamma(b) = \frac{\sum_{m=0}^{3} \gamma'(b,m) E_{sum}(b,m)}{E_{sum}(b)}$$

and this value is set to all subframes m yielding the merged surround coherence $\gamma(b,m)$.

The direct-to-total energy ratio is merged over subframes by

$$r_{dir}(b) = \frac{\sum_{m=0}^{3} r'_{dir}(b,m) E_{sum}(b,m)}{E_{sum}(b)}$$

and this value is set to all subframes m yielding the merged direct-to-total energy ratio $r_{dir}(b,m)$.

In the case when the subframes are not merged (i.e., when mergeRatiosOverSubframes flag is not enabled), the initial values are used as the output

$$\gamma(b,m) = \gamma'(b,m)$$

$$r_{dir}(b,m) = r'_{dir}(b,m)$$

5.5.3.2.5 Combining of MASA spatial audio metadata across multiple directions

This clause describes the combining of multiple MASA spatial audio parameter metadata of the same type for a frequency band, where each of the MASA spatial audio parameters which are combined correspond to a different direction.

The input to the combining process is the MASA spatial audio parameters corresponding to two directions for each frequency band b. In effect, the combining process receives two-direction MASA spatial metadata, for each frequency band b, consisting of two azimuth $\theta(b,m,i)$, two elevation $\phi(b,m,i)$, and two spread coherence $\zeta(b,m,i)$ spatial audio parameters, where i is the direction index and holds the values of 0 and 1 corresponding to the two different directions. For a frequency band b, the two azimuth values, the two elevation values and the two spread coherence values are then each combined into a single value, giving a combined azimuth value, a combined elevation value, and a combined spread coherence value.

Initially, the azimuth $\theta(b,m,i)$ and elevation $\phi(b,m,i)$ angles are converted to Cartesian coordinate vectors. The length of the vectors is determined based on the direct-to-total energy ratio $r_{dir}(b,m,i)$. This is performed for each direction i (where m and b correspond to the specific subframe and frequency band in which the combining takes place)

$$x(b,m,i) = \cos\theta(b,m,i)\cos\phi(b,m,i)\, r_{dir}(b,m,i)$$

$$y(b,m,i) = \sin\theta(b,m,i)\cos\phi(b,m,i)\, r_{dir}(b,m,i)$$

$$z(b,m,i) = \sin\phi(b,m,i)\, r_{dir}(b,m,i)$$

Then, the vectors are summed over the two directions

$$x_{sum}(b,m) = \sum_{i=0}^{1} x(b,m,i)$$

$$y_{sum}(b,m) = \sum_{i=0}^{1} y(b,m,i)$$

$$z_{sum}(b,m) = \sum_{i=0}^{1} z(b,m,i)$$

and the length of the sum vector is computed by

$$l_{sum}(b,m) = \sqrt{x_{sum}(b,m)^2 + y_{sum}(b,m)^2 + z_{sum}(b,m)^2}$$

Then, an importance metric is determined, specific to the frequency band b, which represents the importance of having two directions instead of one for the frequency band b. It is determined by

$$\lambda(b) = \frac{\sum_{m=0}^{3}\left(r_{dir}(b,m,0) + r_{dir}(b,m,1) - l_{sum}(b,m)\right)}{M}$$

where M is the number of subframes (i.e., 4).

Then, based on the importance metric $\lambda(b)$, it is determined whether the encoded values for the frequency band b comprises the original azimuth, elevation, and spread coherence for each separate direction, or whether the encoded values comprise the combined azimuth, elevation, and spread coherence. The importance metric $\lambda(b)$ is determined for all frequency bands.

Determining whether to encode two separate spatial audio parameter values or combined spatial audio parameter value for a particular frequency band b is performed with the use of a "two-direction frequency band" $\beta(b)$ variable, which has the value of 1 for the frequency bands that use the two directions and the value of 0 for the frequency bands that use a combined (single direction) value. The value of the $\beta(b)$ variable, for each frequency band b, is determined by sorting the values of $\lambda(b)$ for all frequency bands in order of magnitude to determine the B_{2dir} frequency bands that have the largest values of $\lambda(b)$.

B_{2dir} corresponds to the factor twoDirBands determined in clause X.X.X TODO: ADD REF TO CONFIG. For the B_{2dir} frequency bands b which have the largest values of $\lambda(b)$, $\beta(b)$ is set to 1. For the other frequency bands b, $\beta(b)$ is set to 0.

Then, for the frequency bands b where $\beta(b) = 0$, the directions are combined as follows.

The combined azimuth and elevation angles are determined by

$$\theta(b,m) = \arctan(y_{sum}(b,m), x_{sum}(b,m))$$

$$\phi(b,m) = \arctan(z_{sum}(b,m), \sqrt{x_{sum}(b,m)^2 + y_{sum}(b,m)^2}\,)$$

Then, a ratio sum variable is determined as

$$r_{dir,sum}(b,m) = \sum_{i=0}^{1} r_{dir}(b,m,i)$$

Then, an ambient energy value is determined using the direct-to-total energy ratios by

$$r_{amb,2dir}(b,m) = 1 - \sum_{i=0}^{1} r_{dir}(b,m,i)$$

Then, the combined direct-to-total energy ratio is determined using the ratio sum variable, the ambient energy value, and the length of the sum vector by

$$r_{dir}(b,m)=\frac{l_{sum}(b,m)}{r_{dir,sum}(b,m)+0.5r_{amb,2dir}(b,m)}$$

Then, the combined spread coherence is determined by

$$\zeta(b,m)=\frac{\sum_{i=0}^{1}\zeta(b,m,i)r_{dir}(b,m,i)}{r_{dir,sum}(b,m)}$$

Then, a further ambient energy value (corresponding to the combined direct-to-total energy ratio) is determined by

$$r_{amb,comb}(b,m)=1-r_{dir}(b,m)$$

Then, original surround coherence energy and new surround coherence energy variables are determined as follows

$$\gamma_{origene}(b,m)=r_{amb}(b,m)\gamma(b,m)$$

$$\gamma_{newene}(b,m)=\max\left(r_{amb,comb}(b,m)-r_{amb,2dir}(b,m),0\right)\zeta(b,m)$$

The surround coherence is then adjusted to a new value representing the combined surround coherence by using the original surround coherence energy and new surround coherence energy

$$\gamma(b,m)=\min\left(1,\frac{\gamma_{origene}(b,m)+\gamma_{newene}(b,m)}{r_{amb,comb}(b,m)}\right)$$

5.5.3.2.6 Determining the significance of surround coherence values for encoding

In this section, the encoding of the surround coherence values associated with the frequency bands of a subframe or a frame are dependent on a significance measure calculated for the subframe or the frame, where the significance measure relates to the significance of the surround coherence values in the subframe or the frame. In other words, encoding of the surround coherence values of the subframe or the frame is performed when the significance of the said values is deemed significant enough to warrant encoding.

When there are four subframes m (this is the case when the input format is MASA), the coherence significance measure is determined separately for each subframe m, and using these values it is determined whether to encode the surround coherence values.

When there is only one subframe m (this is the case when the input format is multi-channel, i.e., when operating in the McMASA mode), the coherence significance measure is determined for the whole frame, and using this value it is determined whether to encode the surround coherence values.

The coherence significance measure is calculated by determining a coherent non-directional energy proportion for each frequency band b given by

$$r_{cohnde}(b,m)=r_{diff}(b,m)\gamma(b,m)$$

where $\gamma(b,m)$ is the surround coherence and $r_{diff}(b,m)$ is the diffuse-to-total energy ratio which represents the non-direction energy ratio for the frequency band and is determined using the direct-to-total energy ratio r_{dir} for the band b.

In the case of one direction per sub band the diffuse-to-total energy ratio $r_{diff}(b,m)$ is given by

$$r_{diff}(b,m)=1-r_{dir}(b,m)$$

In the case of two directions per sub band the diffuse-to-total energy ratio $r_{diff}(b,m)$ is given by

$$r_{diff}(b,m)=1-\sum_{i=0}^{1}r_{dir}(b,m,i)$$

The frequency band b can refer to a MASA frequency band or to a coding band, depending on where coherence significance measure is used.

A first initial coherence significance measure is determined by

$$\xi_1(m) = \frac{\sum_{b=0}^{B-1} r_{cohnde}(b,m)}{B}$$

where B is the number of frequency bands.

A second initial coherence significance measure is determined by

$$\xi_2(m) = 0.4 \frac{\sum_{b=0}^{B-1} r_{cohnde}(b,m) r_{diff}(b,m)}{\sum_{b=0}^{B-1} r_{diff}(b,m)}$$

Using them, the coherence significance measure is determined by

$$\xi(m) = \max(\xi_1(m), \xi_2(m))$$

If the number of subframes m is one, the coherence significance measure $\xi(m)$ is then checked against a fixed threshold of 0.1. If the result of the checking indicates that the coherence significance measure $\xi(m)$ is larger than the threshold, then it is determined that the surround coherence values for this frame are significant, and the surround coherence values $\gamma(b,m)$ for the frequency bands b of this frame are encoded. If the coherence significance measure $\xi(m)$ value is not larger than the threshold of 0.1, then it is determined that the surround coherence values are not significant, and the surround coherence values $\gamma(b,m)$ are not encoded for this frame.

If the number of subframes m is four, the coherence significance measure $\xi(m)$ is then checked against a fixed threshold of 0.1 for all subframes m. If the result of the checking indicates that the coherence significance measure $\xi(m)$ is larger than the threshold for any of the subframes m, then it is determined that the surround coherence values for this frame are significant, and the surround coherence values $\gamma(b,m)$ for the frequency bands b of this frame are encoded. If none of the coherence significance measure $\xi(m)$ values for the different subframes m is not larger than the threshold of 0.1, then it is determined that the surround coherence values are not significant, and the surround coherence values $\gamma(b,m)$ are not encoded for this frame.

5.5.3.2.7 Aligning MASA metadata directions

MASA spatial metadata can contain two ordered directions $i \in [0,1]$ associated with two audio sources within an audio scene. Each direction consists of own directional metadata parameters arranged as a grid of time-frequency tiles for identifying a direction of arrival, i.e., azimuth $\theta(b,m,i)$ and elevation $\phi(b,m,i)$, direct-to-total energy ratio $r_{dir}(b,m,i)$, and spread coherence $\zeta(b,m,i)$ for each time-frequency tile, where m and b are the time axis (sub-frame) and frequency axis (parameter band) grid indices. The original ordering of the spatial metadata, i.e., assignment of a specific source direction into a specific metadata direction i, is not always optimal considering the temporal continuity of the metadata within the direction i. To make the metadata better suitable for the further processing steps, the original spatial metadata tiles (b,m,i) are re-ordered between the two directions $i \in [0,1]$ such that an error measure is minimized as follows.

First, the directional metadata parameters for two directions $i \in [0,1]$ associated with two audio sources within an audio scene are obtained, specifically the parameters identifying a direction of arrival, i.e., azimuth $\theta(b,m,i)$ and elevation $\phi(b,m,i)$. An ordering error is determined for each time-frequency tile. More specifically, the ordering considers temporally consecutive (current m and previous $m-1$ subframe which may also belong to a different metadata parameter frame) time-frequency tiles in the same parameter band b. The ordering error is configured to identify the case that the directional metadata parameters describing the source direction in a neighboring time-frequency tile are more similar and/or less different in another direction order index than in the original order index. In other words, that the sound source directions described by the azimuth and elevation metadata parameters are more similar with the previous time-frequency tile in the same parameter band when the metadata direction field assignment $i \in [0,1]$ is modified by reordering it.

The error measure for not reordering the directions is computed as

$$
\begin{aligned}
D_{no\_swap}(b,m) = {} & \arccos\big(\cos(\phi(b,m,0))\cos(\phi(b,m-1,0))\cos(\theta(b,m,0)-\theta(b,m-1,0)) \\
& + \sin(\phi(b,m,0))\sin(\phi(b,m-1,0))\big) \\
& + \arccos\big(\cos(\phi(b,m,1))\cos(\phi(b,m-1,1))\cos(\theta(b,m,1)-\theta(b,m-1,1)) \\
& + \sin(\phi(b,m,1))\sin(\phi(b,m-1,1))\big)
\end{aligned}
$$

The error measure for reordering the directions is computed as

$$
\begin{aligned}
D_{swap}(b,m) = {} & \arccos\big(\cos(\phi(b,m,0))\cos(\phi(b,m-1,1))\cos(\theta(b,m,0)-\theta(b,m-1,1)) \\
& + \sin(\phi(b,m,0))\sin(\phi(b,m-1,1))\big) \\
& + \arccos\big(\cos(\phi(b,m,1))\cos(\phi(b,m-1,0))\cos(\theta(b,m,1)-\theta(b,m-1,0)) \\
& + \sin(\phi(b,m,1))\sin(\phi(b,m-1,0))\big)
\end{aligned}
$$

The values two partial error measures are compared and the decision to reorder the directions is made based on the comparison. If the error measure value for the direction parameters when not reordering the direction is larger than the error measure value for the direction parameter when reordering the associated order indices, i.e., $D_{no\_swap}(b,m) > D_{swap}(b,m)$, the directions are reordered to make the metadata parameters of direction of arrival more similar between temporally consecutive neighboring metadata subframes. In other words, the direction ordering indices are reordered by reversing the association of the spatial metadata parameter to them with

$$\theta'(b,m,0) = \theta(b,m,1)$$

$$\theta'(b,m,1) = \theta(b,m,0)$$

$$\phi'(b,m,0) = \phi(b,m,1)$$

$$\phi'(b,m,1) = \phi(b,m,0)$$

$$r'_{dir}(b,m,0) = r_{dir}(b,m,1)$$

$$r'_{dir}(b,m,1) = r_{dir}(b,m,0)$$

$$\zeta'(b,m,0) = \zeta(b,m,1)$$

$$\zeta'(b,m,1) = \zeta(b,m,0)$$

This re-ordered spatial metadata is used in the further processing instead of the original spatial metadata.

If the input MASA spatial metadata contains only one active direction, no re-ordering is done.

5.5.3.2.8 MASA metadata framing asynchrony correction

5.5.3.2.8.1 Detecting MASA metadata framing asynchrony

The MASA spatial metadata parameters are analyzed for determining if the metadata framing (grouping of four subframes into metadata frames) is in asynchrony compared to the system framing. The analysis obtains the frame of four subframes of spatial metadata parameters associated with the audio signal. Further, the analysis is based on a state structure containing a copy of the MASA spatial metadata parameters of the last subframe of the previous frame, the offset detected (asynchrony information) in the previous frame, and the number of similar subframes from the end of the previous subframe. The analysis returns the asynchrony information consisting of the determined MASA metadata frame mode ("1sf" or "4sf") and framing offset N_{offset} describing the number of subframes containing similar values. The spatial metadata frame processing described in clause xxx (Combination of MASA metadata…) obtains the asynchrony information for each frame of four subframes, and based on the asynchrony information processes the spatial metadata parameter values of the frame of four subframes before the encoder processes the frame further. The purpose of the processing in clause xxx (Combination of MASA metadata…) is to enable more efficient further processing of the spatial metadata parameter frames in the encoding.

In this context, a MASA spatial metadata subframe $I(m)$ consists of the parameters $I(m) = \{N_{dir}, \theta(b,m,i), \phi(b,m,i), r_{dir}(b,m,i), \zeta(b,m,i), \gamma(b,m,i)\}$, where N_{dir} is the number of directions in the frame. When the relevant spatial metadata parameters in two subframes are similar, this is denoted with $I(m) \approx I(n)$. If the relevant spatial metadata parameters in two subframes are not similar, this is denoted with $I(m) \not\approx I(n)$.

The number N_{start} of spatial metadata subframes similar to the first subframe in the current frame is determined using the method described in clause 5.5.3.2.8.2 (Detecting mutually similar spatial metadata in MASA subframes). The value of N_{start} is the largest in the range $1 \le N_{start} \le 4$ such that $I(0) \approx I(m), \forall m \in [0, N_{start} - 1]$. Similarly, the number N_{stop} of spatial metadata subframes similar to the last subframe in the current frame is determined. The value is the largest in the range $1 \le N_{stop} \le 4$ such that $I(3) \approx I(3 - m), \forall m \in [0, N_{stop} - 1]$.

The metadata framing mode is initialized to "4sf", and the offset with

$$N_{offset} = \begin{cases} 0, & \text{if } N_{offset}^{[-1]} > 2 \\ N_{offset}^{[-1]}, & \text{otherwise} \end{cases}$$

The framing mode and offset are determined based on N_{start}, N_{stop}, $N_{offset}^{[-1]}$, and $N_{stop}^{[-1]}$:

If $N_{stop} == 4$ and $N_{start} == 4$, set framing mode to "1sf" and determine the offset with

$$N_{offset} = \begin{cases} N_{stop}^{[-1]}, & \text{if } 0 < N_{stop}^{[-1]} < 3 \text{ and } I(0) \approx I^{[-1]}(3) \\ N_{offset}^{[-1]}, & \text{if } N_{stop}^{[-1]} == 3 \text{ and } I(0) \approx I^{[-1]}(3) \\ 0, & \text{otherwise} \end{cases}$$

and exit the analysis.

If $N_{stop} == 3$, set framing mode to "1sf" and $N_{offset} = 3$, and exit the analysis.

If $N_{stop}^{[-1]} > 0$ and $I(0) \approx I^{[-1]}(3)$ and $N_{start} > 1$ and $N_{start} + N_{stop}^{[-1]} \ge 4$, set framing mode to "1sf" and determine offset with

$$N_{offset} = \begin{cases} min\left(2, N_{stop}^{[-1]}\right), & \text{if } N_{offset}^{[-1]} == 0 \\ N_{offset}^{[-1]}, & \text{otherwise} \end{cases}$$

and exit the analysis. The asynchrony information consists of the determined MASA metadata frame mode ("1sf" or "4sf") and framing offset N_{offset} are provided to further processing as described in clause xxx (Combination of MASA metadata).

5.5.3.2.8.2 Detecting mutually similar spatial metadata in MASA subframes

Two values of the MASA spatial metadata parameters in the MASA metadata subframes $A = I(m)$ and $B = I(n)$ are compared and determined if they are similar. A number of checks are performed and if none of them indicates that the subframes are not similar, the subframes are similar. On the subframe level, if the two subframes have a different number of directions, they are not similar. The other comparisons are performed on each parameter band b for each spatial metadata parameter direction i. The absolute value of the difference between the spatial parameter values are compared to a threshold specific for the parameter to determine if the parameter in the two subframes is similar or not similar. When any spatial metadata parameter in any parameter band b in any spatial metadata direction i is determined to be not similar, the whole subframes can be determined to be not similar.

Obtain the azimuth angle spatial metadata parameter values $\theta_A(b, i)$ and (b, i). Determine the difference in the azimuth angles with

$$d_{azi}\left(\theta_A(b,i), \theta_B(b,i)\right) = |\theta_A(b,i) - \theta_B(b,i)|$$

The angle difference is wrapped if $d_{azi}\left(\theta_A(b,i), \theta_B(b,i)\right) > 180$ by

$$d_{azi}\left(\theta_A(b,i), \theta_B(b,i)\right) = 360 - d_{azi}\left(\theta_A(b,i), \theta_B(b,i)\right)$$

Compare the difference in the azimuth angle spatial metadata parameter values with a threshold, and if $d_{azi}\left(\theta_A(b,i), \theta_B(b,i)\right) > 0.5$, the subframes are not similar.

Obtain the elevation angle spatial metadata parameter values $\phi_A(b, i)$ and $\phi_B(b, i)$. Determine the difference in elevation angles with

$$d_{ele}\big(\phi_A(b,i),\phi_B(b,i)\big) = |\phi_A(b,i) - \phi_B(b,i)|$$

Compare the difference in the elevation angle spatial metadata parameter values with a threshold, and if $d_{ele}\big(\phi_A(b,i),\phi_B(b,i)\big) > 0.5$, the subframes are not similar.

Obtain the direct-to-total energy ratio spatial metadata parameter values $r_{dir;A}(b,i)$ and $r_{dir;B}(b,i)$. Determine the difference in direct-to-total energy ratios with

$$d_{dir}\left(r_{dir;A}(b,i),r_{dir;B}(b,i)\right) = \left|r_{dir;A}(b,i) - r_{dir;B}(b,i)\right|$$

Compare the difference in the direct-to-total energy ratio spatial metadata parameter values with a threshold, and if $d_{dir}\left(r_{dir;A}(b,i),r_{dir;B}(b,i)\right) > 0.1$, the subframes are not similar.

Obtain the spread coherence spatial metadata parameter values $\zeta_A(b,i)$ and $\zeta_B(b,i)$. Determine the difference in spread coherence parameters with

$$d_{spr}\big(\zeta_A(b,i),\zeta_B(b,i)\big) = |\zeta_A(b,i) - \zeta_B(b,i)|$$

Compare the difference in the spread coherence spatial metadata parameter values with a threshold, and if $d_{spr}\big(\zeta_A(b,i),\zeta_B(b,i)\big) > 0.1$, the subframes are not similar.

Obtain the surround coherence spatial metadata parameter values $\gamma_A(b)$ and $\gamma_B(b)$. Determine the difference in surround coherence parameters with

$$d_{sur}\big(\gamma_A(b),\gamma_B(b)\big) = |\gamma_A(b) - \gamma_B(b)|$$

Compare the difference in the surround coherence spatial metadata parameter values with a threshold, and if $d_{sur}\big(\gamma_A(b),\gamma_B(b)\big) > 0.1$, the subframes are not similar.

5.5.3.2.8.3 Combination of MASA metadata over subframes

Obtain the spatial metadata framing asynchrony information from clause x.x.x.x (Detecting MASA metadata framing asynchrony). If the metadata asynchrony information determined by the asynchrony analysis indicates that a non-zero offset $N_{offset} \neq 0$ has been found, indicating that subframes of consecutive subframes possibly in different system frames contain similar spatial metadata parameter values, and a metadata framing mode is "1sf", the spatial metadata parameter values are processed by combining them across the subframes. This processing enables the further processing of the spatial metadata frame of four subframes such that the spatial metadata parameter values can be more efficiently encoded.

The azimuth $\theta(b,m,i)$ and elevation $\phi(b,m,i)$ angles obtained from the spatial metadata are first converted to Cartesian coordinate vectors. The length of the vectors is determined based on the direct-to-total energy ratio $r_{dir}(b,m,i)$ and the energy $E(b,m)$. This is performed by

$$x(b,m,i) = \cos\big(\theta(b,m,i)\big)\cos\big(\phi(b,m,i)\big)\, l(b,m,i)$$

$$y(b,m,i) = \sin\big(\theta(b,m,i)\big)\cos\big(\phi(b,m,i)\big)\, l(b,m,i)$$

$$z(b,m,i) = \sin\big(\phi(b,m,i)\big)\, l(b,m,i)$$

where

$$l(b,m,i) = r_{dir}(b,m,i)E(b,m)$$

where b is the MASA frequency band index, m is the subframe index, and i is the direction index. Then, the vectors are summed over the subframes

$$x_{sum}(b,i) = \sum_{m=0}^{3} x(b,m,i)$$

$$y_{sum}(b,i) = \sum_{m=0}^{3} y(b,m,i)$$

$$z_{sum}(b,i) = \sum_{m=0}^{3} z(b,m,i)$$

The sum of the parameter tile energies is determined with

$$E_{sum}(b) = \sum_{m=0}^{3} E(b,m)$$

The combined direction is determined with

$$\theta'(b,i) = \arctan(y_{sum}(b,i), x_{sum}(b,i))$$

$$\phi'(b,i) = \arctan(z_{sum}(b,i), \sqrt{x_{sum}(b,i)^2 + y_{sum}(b,i)^2}\,)$$

and the combined direct-to-total energy ratio is determined by

$$r'_{dir}(b,i) = \frac{\sqrt{x_{sum}(b,i)^2 + y_{sum}(b,i)^2 + z_{sum}(b,i)^2}}{E_{sum}(b)}$$

The combined direct-to-total energy ratio is used for determining combined diffuse-to-total energy ratio with

$$r'_{diff}(b) = 1 - r'_{dir}(b)$$

for one-direction metadata and by

$$r'_{diff}(b,m) = 1 - \sum_{i=0}^{1} r'_{dir}(b,i)$$

for two-direction metadata.

The spread coherence is combined over subframes by

$$\zeta'(b,i) = \frac{\sum_{m=0}^{3} \zeta(b,m)E(b,m)}{E_{sum}(b)}$$

The surround coherence is combined over subframes by

$$\gamma'(b) = \frac{\sum_{m=0}^{3} \gamma'(b,m)E(b,m)}{E_{sum}(b)}$$

The combined spatial parameter values are assigned to all four subframes in the current system frame

$$\theta'(b,m,i) = \theta'(b,i)\ \forall m \in [0,3]$$

$$\phi'(b,m,i) = \phi'(b,i)\ \forall m \in [0,3]$$

$$r'_{dir}(b,m,i) = r'_{dir}{}'(b,i)\ \forall m \in [0,3]$$

$$\zeta'(b,m,i) = \zeta'(b,i)\ \forall m \in [0,3]$$

$$r'_{diff}(b,m,i) = r'_{diff}(b,i)\ \forall m \in [0,3]$$

$$\xi'(b,m,i) = \xi'(b,i)\ \forall m \in [0,3]$$

5.5.3.2.9 Metadata reductions for low rate

5.5.3.2.10 Reordering of directional data with two concurrent directions

In case of MASA format metadata where at least some frequency bands are configured to encode two sets of directional spatial metadata parameters, an analysis and reordering step is performed before quantization and encoding of the metadata parameters. This is done to optimize the encoding of the directional metadata parameters by fulfilling assumption that for any time frequency tile or frequency band, $r_{dir1} \geq r_{dir2}$, that is, the direct-to-total energy ratio parameter of the first direction is larger than or equal to the direct-to-total energy ratio of the second direction..

In practice, for each frequency coding band containing two sets of directional parameters, a comparison is made to determine if $r_{dir2} > r_{dir1}$, that is, the direct-to-total energy ratio parameter of the second direction is larger than the direct-to-total energy ratio of the first direction. If this is true, then the directional spatial metadata parameters of the first direction are swapped with the directional spatial metadata parameters of the second direction for each time-frequency tile belonging to this frequency coding band. This is swapping done for direct-to-total energy ratio, direction, and spread coherence parameters and is achieved with the following pseudocode algorithm:

```
For subframes in 2dir band
    temp_value = dir1_parameter
    dir1_parameter = dir2_parameter
    dir2_parameter = temp_value
```

5.5.3.2.11 Signalling bits

5.5.3.3 MASA metadata quantization and encoding

5.5.3.3.1 Overview

5.5.3.3.2 Energy ratio encoding

5.5.3.3.2.1 Energy ratio encoding for bitrates up to 256 kbps

5.5.3.3.2.2 Energy ratio encoding for 384 kbps and 512 kbps

5.5.3.3.3 Direction encoding

5.5.3.3.3.1 Direction encoding for bitrates up to 256 kbps

5.5.3.3.3.2 Direction encoding for 384 kbps and 512 kbps

5.5.3.3.4 Spread coherence encoding

5.5.3.3.4.1 Spread coherence encoding for bitrates up to 256 kbps

5.5.3.3.4.2 Spread coherence encoding for 384 kbps and 512 kbps

5.5.3.3.5 Surround coherence encoding

5.5.3.3.5.1 Surround coherence encoding for bitrates up to 256 kbps

5.5.3.3.5.2 Surround coherence encoding for 384 kbps and 512 kbps

5.5.3.3.6 Coding of the second direction parameters

5.5.3.3.6.1 Joint coding of second direction parameters

5.5.3.3.6.2 Independent coding of second direction parameters

5.5.3.4 Encoding of MASA audio transport channels

5.5.3.5 DTX for MASA

5.5.3.6 Bitstream structure

5.6 Object-based audio (ISM) operation

5.6.1 ISM overview

The object-based audio represents a complex audio auditory scene as a collection of individual elements, also known as audio objects, such as speech, music, or a general audio sound. In IVAS, an audio object is referred to as an independent audio stream with metadata (ISM) where the audio stream represents, in a bitstream, an audio waveform and consists of a mono channel coded by a SCE. Then metadata represents a set of information describing an audio stream and an artistic intension used to translate the original or coded audio objects to a reproduction system. The metadata describes spatial properties of each individual audio object, such as position, or an orientation. In the following text, two sets of metadata are considered:

- input metadata: unquantized metadata representation used as an input to the IVAS codec; and
- coded metadata: quantized and coded metadata forming part of a bit-stream transmitted from the encoder to the decoder.

IVAS supports a simultaneous coding of up to four (4) ISMs. The related bitrates and maximum audio bandwidths at which the ISM coding is supported is summarized in Table 5.4-1 while two ISM coding modes are employed: discrete ISM (further referred to as DiscISM) and parametric ISM (further referred to as ParamISM).

Table 5.6-1: Overview of bitrates, bandwidths, and coding modes in ISM format.

IVAS bitrate [kbps]	number of ISMs			
	1	2	3	4
13.2	DiscISM – SWB	n/a	n/a	n/a
16.4	DiscISM – FB	DiscISM – WB	n/a	n/a
24.4	DiscISM – FB	DiscISM – SWB	ParamISM – SWB	ParamISM – SWB
32	DiscISM – FB	DiscISM – FB	ParamISM – SWB	ParamISM – SWB
48	DiscISM – FB	DiscISM – FB	DiscISM – FB	DiscISM – SWB
64	DiscISM – FB	DiscISM – FB	DiscISM – FB	DiscISM – FB
80	DiscISM – FB	DiscISM – FB	DiscISM – FB	DiscISM – FB
96	DiscISM – FB	DiscISM – FB	DiscISM – FB	DiscISM – FB
128	DiscISM – FB	DiscISM – FB	DiscISM – FB	DiscISM – FB
160	n/a	DiscISM – FB	DiscISM – FB	DiscISM – FB
192	n/a	DiscISM – FB	DiscISM – FB	DiscISM – FB
256	n/a	DiscISM – FB	DiscISM – FB	DiscISM – FB
384	n/a	n/a	DiscISM – FB	DiscISM – FB
512	n/a	n/a	n/a	DiscISM – FB

5.6.2 Discrete ISM coding mode

Figure 5.6-1 is a schematic block diagram of the DiscISM coding mode encoder.

At the coded input, there is buffered for each frame N_{ISM} ISMs, i.e. N_{ISM} audio streams with the associated respective NISM metadata. In Figure 5.6-1, the audio streams are denoted by a solid line while metadata and parameters in general by a dotted line.

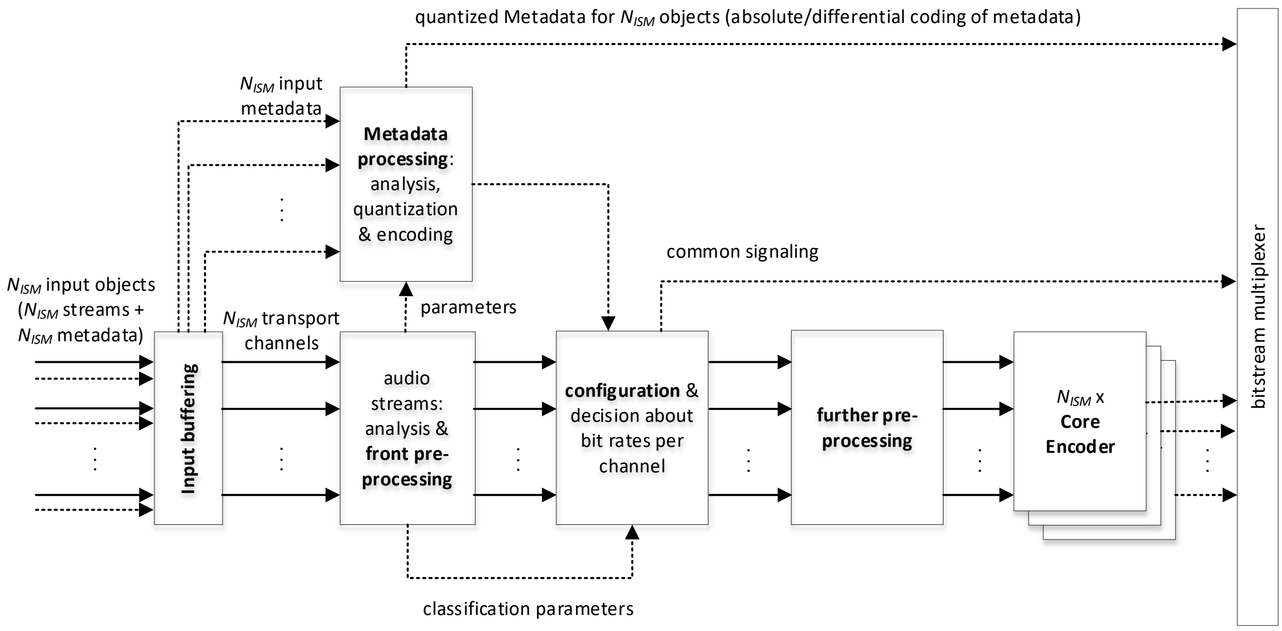

quantized Metadata for N_{ISM} objects (absolute/differential coding of metadata)
N_{ISM} input metadata
Metadata processing: analysis, quantization & encoding
N_{ISM} input objects (N_{ISM} streams + N_{ISM} metadata)
Input buffering
N_{ISM} transport channels
parameters
common signaling
audio streams: analysis & front pre-processing
configuration & decision about bit rates per channel
further pre-processing
N_{ISM} x Core Encoder
bitstream multiplexer
classification parameters

Figure 5.6-1: Block diagram of the DiscISM encoder.

5.6.2.1 DiscISM encoding system

The DiscISM encoding system is composed from several functional blocks as shown in Figure 5.6-1.

In the DiscISM mode, the N_{ISM} audio streams represent N_{ISM} transport channels. These transport channels are first analysed and pre-processed in parallel in the front pre-processing (see Clause 5.2.2.1) block which further provides information about the audio streams to the metadata processing block.

In parallel, the input metadata are analysed, quantized, and encoded in a metadata processing block (see further in Clause 5.6.4). The bit-budget used to quantize the metadata then forms an input to the configuration block.

Further, at the configuration block, a decision about bitrates per transport channel is made in a bit-budget allocator. The bit-budget allocator employs a bitrate adaptation algorithm to distribute the available bit-budget for core-encoding the N_{ISM} audio streams in the N_{ISM} transport channels (see further in Clause 5.6.2.2). Consequently, a better, more efficient distribution of the available bit-budget between the audio streams is achieved.

Once the configuration and bitrate distribution between the N_{ISM} audio streams is completed by the bit-budget allocator, the pre-processor performs a sequential further pre-processing (see Clause 5.2.2.2.1) on each of the N_{ISM} transport channels.

Finally, the N_{ISM} transport channels are sequentially encoded using N_{ISM} fluctuating bitrate core-encoders, specifically by SCE tools (see 5.2.3.1), one per transport channel. The bitrate used by each of the N_{ISM} core-encoders is the bitrate selected by the bit-budget allocator for the corresponding audio stream.

5.6.2.2 Bitrates distribution between audio streams

The bitrate distribution algorithm comprises the following steps 1-6 performed by the bit-budget allocator.

Step 1: The ISM total bit-budget, $bits_{ISM}$, per frame is calculated from the ISM total bitrate ism_total_brate (in case of the ISM format it is equal to the IVAS total bitrate $ivas\_total\_brate$) using the following relation:

$$bits_{ISM} = \frac{ism\_total\_brate}{50} \qquad (5.6\text{-}1)$$

The denominator, 50, corresponds to the number of frames per second, assuming 20 ms long frames in IVAS.

Step 2: The element bitrate $element\_brate$ (resulting from a sum of the metadata bit-budget and core-encoder bit-budget related to one ISM) defined for N_{ISM} streams is constant at a given ism_total_brate, and about the same for the N_{ISM} streams. The corresponding element bit budget, $bits_{element}$, is computed in the bit-budget allocator for the streams $n = 0, \dots, N_{ISM} - 1$ using the following relation:

$$bits_{element}[n] = \left\lfloor \frac{bits_{ISM}}{N_{ISM}} \right\rfloor \qquad (5.6\text{-}2)$$

where $\lfloor x \rfloor$ indicates the largest integer smaller than or equal to x. In order to spend all available ISM total bit-budget $bits_{ISM}$ the element bit-budget $bits_{element}$, of the last audio object is eventually adjusted using the following relation:

$$bits_{element}[N_{ISM} - 1] = \left\lfloor \frac{bits_{ISM}}{N_{ISM}} \right\rfloor + bits_{ISM} \mathrm{mod} N_{ISM} \qquad (5.6\text{-}3)$$

where "mod" indicates a remainder modulo operation. Finally, the element bit-budget $bits_{element}$ of the N_{ISM} audio objects is used to set the value $element\_brate$ for the ISMs $n = 0, \dots, N_{ISM} - 1$ using the following relation:

$$element\_brate[n] = bits_{element}[n] \cdot 50 \qquad (5.6\text{-}4)$$

where the number 50 corresponds to the number of frames per second, assuming 20 ms long frames.

Step 3: The metadata bit-budget $bits_{meta}$ per frame, of the N_{ISM} streams is summed, using the following relation:

$$bits_{meta\_all} = \sum_{n=0}^{N_{ISM}-1} bits_{meta}[n] \qquad (5.6\text{-}5)$$

and the resulting value $bits_{meta\_all}$ is added to an ISM common signalling bit-budget, $bits_{ISM\_signalling}$, resulting in the side bit-budget:

$$bits_{side} = bits_{meta\_all} + bits_{ISM\_signalling} \qquad (5.6\text{-}6)$$

The metadata bit-budget $bits_{meta}$ is obtained in the metadata processing module as described in Clause 5.6.4 while the ISM signalling is described in Clause 5.6.5.1.

Step 4: The side bit-budget, $bits_{side}$, per frame, is split in the bit-budget allocator equally between the N_{ISM} streams and used to compute the core-encoder bit-budget, $bits_{CoreCoder}$, for each of the N_{ISM} streams using the following relation:

$$bits_{CoreCoder}[n] = bits_{element}[n] - \left\lfloor \frac{bits_{side}}{N_{ISM}} \right\rfloor \qquad (5.6\text{-}7)$$

while the core-encoder bit-budget of the last audio stream may eventually be adjusted to spend all the available core-encoding bit-budget using the following relation:

$$bits_{CoreCoder}[N_{ISM} - 1] = bits_{element}[N_{ISM} - 1] - \left\lfloor \frac{bits_{side}}{N_{ISM}} \right\rfloor + bits_{side} \mathrm{mod} N_{ISM} \qquad (5.6\text{-}8)$$

The corresponding total bitrate, $total\_brate$, i.e. the bitrate to code one audio stream in a core-encoder, is then obtained for $n = 0, \dots, N_{ISM} - 1$ using the following relation:

$$total\_brate[n] = bits_{CoreCoder}[n] \cdot 50 \qquad (5.6\text{-}9)$$

where the number 50 corresponds to the number of frames per second, giving 20 ms long frames.

Step 5: The total bitrate, $total\_brate$, in inactive frames may be lowered and set to a constant value in the related audio streams. The so saved bit-budget is then redistributed equally between the audio streams with active content in the frame. Such redistribution of bit-budget is described in the following Clause 5.6.2.2.1, Step 5.

Step 6: The total bitrate, $total\_brate$, in audio streams in active frames is further adjusted between these audio streams based on an ISM importance classification. Such adjustment of bitrate is further described in the following Clause 5.6.2.2.1.

When the audio streams are all classified as inactive, the above last two steps 5 and 6 are skipped. Accordingly, the bitrate adaptation algorithm described in the following Clause 5.6.2.2.1 is employed when at least one audio stream has an active content.

5.6.2.2.1 Bitrate adaptation based on ISM importance

In the ISM format, there is used a classification of ISM importance based on several core-coder parameters, namely local VAD (VAD_{local}, see Clause 5.1.12 in [3][3]), core-encoder type ($coder\_type$, see Clause 5.1.13 in [3]), and core-coder mode ($flag_{tcxonly}$). The ISM importance classification is done as part of the bit-budget allocator for rating the importance of a particular ISM stream based on a metric how critical coding of an audio stream to obtain a given quality of a decoded synthesis is. As a result, four (4) distinct ISM importance classes, $class_{ISM}$, are defined:

- No metadata class, ISM_NO_META: frames without metadata coding, i.e. inactive frames with $VAD_{local} = 0$;
- Low importance class, ISM_LOW_IMP: frames where $coder\_type$ is UNVOICED or INACTIVE;
- Medium importance class, ISM_MEDIUM_IMP: frames where $coder\_type$ is VOICED;
- High importance class ISM_HIGH_IMP: frames where $coder\_type$ is GENERIC.

In case of core-coder employing only the TCX core, i.e. $flag_{tcxonly} = 1$, inactive frames with $VAD_{local} = 0$ are classified as ISM_LOW_IMP and frames with $coder\_type$ being UNVOICED are classified as ISM_HIGH_IMP.

The ISM importance class is then used by the bit-budget allocator, in the bitrate adaptation algorithm (see Clause 5.6.2.2, step 6) to assign a higher bit-budget to streams with a higher ISM importance and a lower bit-budget to streams with a lower ISM importance. Thus, for every audio stream $n,\ n = 0, \dots, N_{ISM} - 1$, the following bitrate adaptation algorithm is used by the bit-budget allocator:

Step 1: In frames classified as $class_{ISM}$ = ISM_NO_META, a constant low bitrate $B_{VAD0} = 2450$ bps is assigned.

Step 2: In frames classified as $class_{ISM}$ = ISM_LOW_IMP, the core-encoder total bitrate, $total\_brate$, is lowered as:

$$total\_brate_{new}[n] = \max(\alpha_{low} \cdot total\_brate[n], B_{low}) \quad (5.6\text{-}10)$$

where the constant $\alpha_{low} = 0.6$. Then the constant B_{low} represents a minimum bitrate threshold supported by the core-coder for a particular configuration, which is dependent on the internal sampling rate of the codec and the coded audio bandwidth.

Step 3: In frames classified as $class_{ISM}$ = ISM_MEDIUM_IMP, the core-encoder total bitrate, $total\_brate$, is lowered as

$$total\_brate_{new}[n] = \max(\alpha_{med} \cdot total\_brate[n], B_{low}) \quad (5.6\text{-}11)$$

where the constant $\alpha_{med} = 0.8$.

Step 4: In frames classified as $class_{ISM}$ = ISM_HIGH_IMP, no bitrate adaptation is used, i.e.

$$total\_brate_{new}[n] = total\_brate[n]. \quad (5.6\text{-}12)$$

Step 5: Finally, the saved bit-budget (a sum of differences between the initial ($total\_brate$) and new ($total\_brate_{new}$) total bitrates) is redistributed equally between the streams with active content in the frame. The following bit budget redistribution logic is used:

First, the saved bit-budget is computed using the following relation:

$$bits_{diff} = \sum_{n=0}^{N_{ISM}-1} \left(bits_{CoreCoder,new}[n] - bits_{CoreCoder}[n] \right) \quad (5.6\text{-}13)$$

where

$$bits_{CoreCoder,new}[n] = \frac{total\_brate_{new}[n]}{50} \quad . \quad (5.6\text{-}14)$$

Next, the saved bit-budget is redistributed equally between the core-encoder bit-budgets of the streams with active content in a given frame using the following relation:

$$bits_{CoreCoder,final}[n] = bits_{CoreCoder,new}[n] + \left\lfloor \frac{bits_{diff}}{N_{VAD1}} \right\rfloor \quad (5.6\text{-}15)$$

for all n with $VAD_{local} = 1$ where N_{VAD1} is the number of streams with active content. The core-encoder bit-budget of the first audio stream with active content is then eventually increased using the following relation:

$$bits_{CoreCoder,final}[n] = bits_{CoreCoder,new}[n] + \left\lfloor \frac{bits_{diff}}{N_{VAD1}} \right\rfloor + bits_{diff} \mathrm{mod} N_{VAD1} \qquad (5.6\text{-}16)$$

where n in (5.6-16) corresponds to the first stream with $VAD_{local} = 1$.

The corresponding final core-encoder total bitrate, $total\_brate_{final}$, is finally obtained for each audio stream $n,\ n = 0, \dots, N_{ISM} - 1$, as follows:

$$total\_brate_{final}[n] = bits_{CoreCoder,final}[n] \cdot 50. \qquad (5.6\text{-}17)$$

5.6.3 Parametric ISM coding mode

5.6.3.1 General

In ISM mode, if the total bitrate configured is 24.4 kbit/s or 32 kbit/s and if there are three or four input objects, the Parametric ISM (ParamISM) mode is employed. ParamISM allows for encoding audio objects and their related metadata (if present), where the metadata indicates, among other characteristics, the direction information of each audio object. To reduce data and allow for efficient transmission, ParamISM employs a downmixer for downmixing the audio objects into two transport channels for further encoding. The metadata is used to generate new parameter data that describes two relevant audio objects and is transmitted along with the transport channels. Figure 5.2-11 shows an overview of the ParamISM encoder (band-wise processing not reflected).

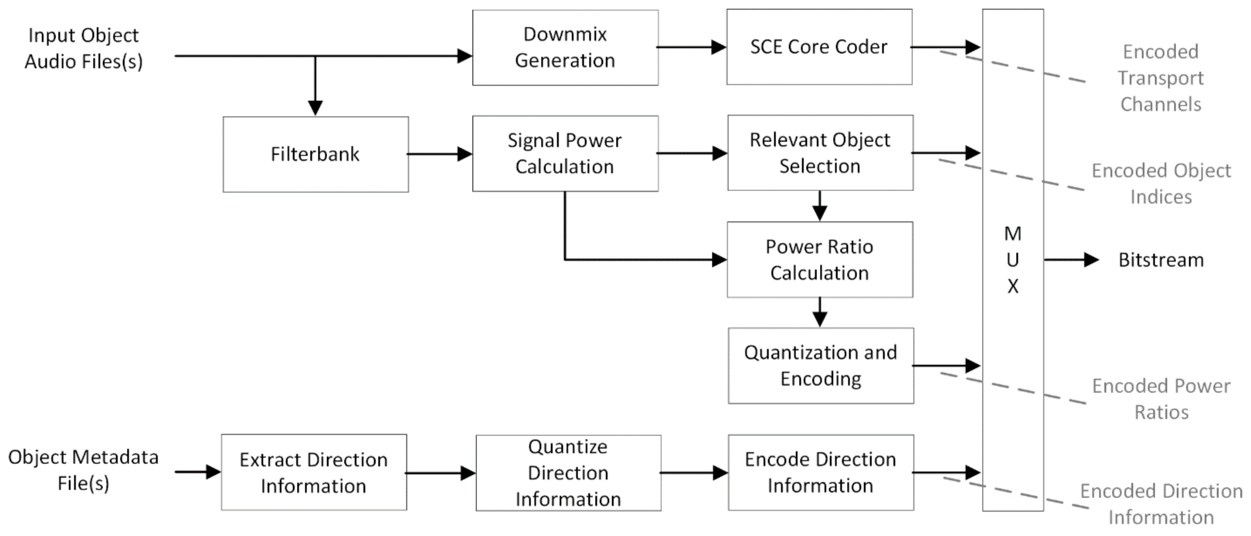

Input Object Audio Files(s)
Downmix Generation
SCE Core Coder
Encoded Transport Channels
Filterbank
Signal Power Calculation
Relevant Object Selection
Encoded Object Indices
Power Ratio Calculation
M U X
Bitstream
Quantization and Encoding
Encoded Power Ratios
Object Metadata File(s)
Extract Direction Information
Quantize Direction Information
Encode Direction Information
Encoded Direction Information

Figure 5.6-2: Parametric ISM encoder overview.

5.6.3.2 Parameters

ParamISM identifies the two relevant audio objects in each parameter band, i.e., the two most dominant objects in terms of signal power. To determine the two relevant objects, the audio objects are transformed into a spectral representation by means of an MDFT filterbank as described in 5.2.5. Each time frame of 960 samples (in the case of 48 kHz input) is divided into 4 subframes, each of which the filterbank is applied to. In the time-frequency representation, each original frame then consists of 4-by-240 time-frequency (T/F) tiles, where in the temporal direction, 4 time slots, and in the frequency direction, 240 frequency bins are used.

The signal power of each object is calculated and stored for each T/F tile of the current frame:

$$P_i\,(k, n) = |X_i(k, n)|^2, \qquad (5.6\text{-}18)$$

where X_i denotes the *i*-th input object in the spectral domain, k denotes the frequency bin index, and n denotes the time slot index.

To reduce the data load and thus the number of parameters to be transmitted, the T/F files are further grouped into parameter bands. To that end, all 4 time slots are grouped into 1 slot and the 240 frequency bins are grouped into 11 bands. Consequently, the signal power of each audio object within each parameter band is calculated by summation over all T/F tiles contained in the parameter band and used as the selection criterion:

$$P_i(l) = \sum_{n=0}^{3} \sum_{k=B(l)}^{B(l+1)} P_i(k,n), \tag{5.6-19}$$

where l is the parameter band index with $0 < l < 10$ and $B(l)$ defines the parameter band borders in the frequency direction:

$$B(l) = [0, 4, 8, 12, 16, 24, 32, 44, 60, 84, 124, 240] \tag{5.6-20}$$

The two objects that comprise the greatest signal power values are then selected as the relevant objects of the parameter band and identified by their object index. The object index of each input object is derived from the input channels fed into the IVAS encoder, i.e., the object occupying the first input channel is identified as object 1, the object occupying the second input channel is identified as object 2, and so on, up to the maximum of 4 possible input objects.

The two relevant objects are further characterized by a power ratio that signals the contribution of each of the two objects to the sum of the signal powers of both objects.

$$r_1(l,m) = \frac{P_1(l,m)}{(P_1(l,m) + P_2(l,m))} \tag{5.6-21}$$

$$r_2(l,m) = \frac{P_2(l,m)}{(P_1(l,m) + P_2(l,m))} = 1 - r_1(l,m) \tag{5.6-22}$$

If both signal power values are 0, the power ratios are both set to 0.5. Since the relevant objects are always sorted from high-to-lower signal energy, the first object index corresponds to the most dominant object and the corresponding energy ratio describes how its signal energy compares to that of the second relevant object. As both ratios sum up to 1, only the first power ratio, which always has a value between 0.5 and 1, must be quantized and encoded and transmitted via the bitstream. Specifically, the power ratio associated with the most dominant object is quantized with 3 bits and transmitted as a power ratio index:

$$\lfloor idx = 14(r_1 - 0.5) \rfloor \tag{5.6-23}$$

The direction information provided in the input metadata is quantized to a lower resolution for further processing. The azimuth values are quantized using 7 bits, while the elevation values are quantized using 6 bits as described in 5.6.4.1.

5.6.3.3 Downmix

Apart from the set of parameters described above, ParamISM requires a transmission of the actual audio objects. This is done in the form of a downmix, consisting of two transport channels. More concretely, the downmixer is configured to downmix the audio objects in response to the direction information provided in the metadata.

To that end, two virtual microphone signals arranged at the same horizontal position (center) and different orientations are generated. More specifically, two virtual cardioid microphone signals oriented in opposing directions, +90° and -90° with respect to the center line, are employed, forming a left-right pair of virtual microphones.

The positions and orientations of the virtual cardioids are static over all time frames and only the direction information of each object may change from frame to frame according to the associated metadata which gives the direction information for each frame of the corresponding audio object.

For each audio object, the downmixer derives a weighting information for each transport channel using the direction information for the corresponding audio object, thus acquiring the contribution of each object to each transport channel. Prior to obtaining the weights from the virtual cardioids, the direction information provided in the metadata of each object is quantized as described above in the section on the parameter calculation. The left and right cardioids, and thus the weighting information, are defined as:

$$w_{L,i} = 0.5 + 0.5 * \cos(\theta_i - \pi/2) \tag{5.6-24}$$

$$w_{R,i} = 0.5 + 0.5 * \cos(\theta_i + \pi/2) \tag{5.6-25}$$

As elevation is not considered for downmixing, only the azimuth θ_i is used as direction information.

The weights obtained from the virtual cardioids for each time frame and audio object are applied in the time domain in a sample-by-sample manner before combining the weighted samples into the two, left and right, transport channels:

$$DMX_L = \sum_{i\in N} x_i * w_{L,i} \quad (5.6\text{-}26)$$

$$DMX_R = \sum_{i\in N} x_i * w_{R,i} \quad (5.6\text{-}27)$$

Here, N denotes the number of input objects and x_i describes the current time frame of the i-th object, consisting of, e.g., 960 samples at a sampling rate of 48 kHz. For simplicity, a sample index is omitted.

To avoid sudden directional changes between frames, smoothing is applied to the cardioid signals. [Editor's note: to be elaborated on in next version]

The transport channels and the signal energies contained in each transport channel highly depend on the object positions. As such, the transport channels may contain less energy than the input objects, a fact which may be reflected in the decoded output signal. To mitigate, an energy compensation is employed in the form of a downmix gain that is multiplied to the transport channels.

$$gain_{DMX} = \sqrt{\frac{E_{in}}{E_{DMX}+\epsilon}}, \quad (5.6\text{-}28)$$

where $E_{in}$is the combined signal energy of all input objects of the current frame and $E_{DMX} = DMX_L{}^2 + DMX_R{}^2$, with DMX_L the left cardioid transport channel and DMX_R the right cardioid transport channel.

To also avoid any sudden changes between frames, the downmix gain is applied to the transport channels with the same smoothing approach as employed for the cardioid signals. [Editor's note: to be elaborated on in next version]

5.6.3.4 Encoding

In summary, the encoded audio signal of each input audio frame comprises two encoded transport channels that are fed into the core coder as two SCEs, and the parameter data, further consisting of:

- two object identifications for the relevant audio objects of each time-frequency tile in the form of object indices 0, 1, 2, or 3
- quantized and encoded direction data for each audio object in the time frame, with the direction data being constant for all time-frequency tiles of the time frame
- one quantized power ratio describing the ratio between the two relevant objects of each time-frequency tile
- one flag indicating whether the frame was classified as noisy speech [Editor's note: to be elaborated on in next version]

5.6.4 ISM metadata coding

In the ISM format, several groups of metadata parameters are analysed, coded, quantized, and transmitted in the bitstream.

The first group of metadata parameters comprises an azimuth and an elevation. If available as the input metadata, they are coded at all bitrates and encoded and quantized as described in Clause 5.6.4.1.

The second group of metadata parameters comprises the first group of metadata parameters and extended metadata parameters. The extended metadata parameters comprise a yaw, a pitch, and a radius. The coding of the extended metadata is supported for $ism\_total\_brate \geq 64$ kbps.

Finally, when a non-diegetic audio stream is encoded, a non-diegetic gain is coded and quantized while metadata parameters from first and second group are not coded and transmitted. The coding of non-diegetic gain is supported for $ism\_total\_brate \geq 64$ kbps.

The metadata are processed in the metadata processing module from Figure 5.6-1 where the metadata and core-coder parameters of each of the N_{ISM} audio stream are analyzed in order to determine whether the current frame is inactive ($VAD_{local} = 0$) or active ($VAD_{local} = 1$) with respect to this particular audio stream. In inactive frames, no metadata is coded relative of that ISM. In active frames, the metadata are quantized and coded for this audio stream using a variable bitrate. More details about metadata quantization and coding are provided in the following Clauses.

Once the metadata of the N_{ISM} audio streams are analyzed, quantized and encoded, information from the metadata processing module about the bit-budget $bits_{meta}$ for the coding of the metadata per audio stream is supplied to a configuration and decision processor (and more specifically to the bit-budget allocator) described in Clause 5.6.2.2. When the configuration and bitrate distribution between the audio streams is completed, the coding continues with further pre-processing. Finally, the N_{ISM} audio streams are encoded using an encoder comprising N_{ISM} fluctuating bitrate SCE core-encoders.

5.6.4.1 Direction metadata encoding and quantization

The metadata processing module of Figure 5.6-1 quantizes and encodes the metadata of the N_{ISM} audio streams sequentially in a loop while a certain dependency is employed between quantization of audio streams and the metadata parameters of these audio streams.

In case of the first group of metadata parameters, two metadata parameters, azimuth and elevation (as included in the N_{ISM} input metadata), are considered and further referred as a directional metadata. The metadata processing module comprises a quantizer of the metadata parameter indexes using the following resolution of metadata parameters in order to reduce the number of bits being used:

a) Azimuth parameter: An azimuth parameter index from the input metadata is quantized to B_{az}-bit index where $B_{az} = 7$. Giving the minimum and maximum azimuth limits (-180 and +180^o), a quantization step for a 7-bit uniform scalar quantizer is 2.5^o between -140^o and 135^o and 5^o otherwise.

b) Elevation parameter: An elevation parameter index from the input metadata is quantized to B_{el}-bit index where $B_{el} = 6$. Giving the minimum and maximum elevation limits (-90^o and +90^o), a quantization step for 6-bit uniform scalar quantizer is 2.5^o between -70^o and 65^o and 5^o otherwise.

Both azimuth and elevation indexes, once quantized, are coded using either absolute or differential coding. In the case of absolute coding, the current signed value of a parameter is coded. In case of differential coding, a signed difference between a current value and a previous value of a parameter is coded. As the indexes of the azimuth and elevation parameters usually evolve smoothly (i.e. a change in azimuth or elevation position can be considered as continuous and smooth), the differential coding is used by default. However, absolute coding is used in the following instances:

a) There is too large a difference between current and previous values of the parameter index which would result in a higher or equal number of bits for using differential coding compared to using absolute coding (this may happen exceptionally);

b) No metadata were coded and sent in the previous frame;

c) There were too many consecutive frames with differential coding. In order to control decoding in a noisy channel when a frame is lost, the metadata encoder codes the metadata parameter indexes using absolute coding if a number of consecutive frames which are coded using differential coding is higher than $\beta = 10$ frames.

The metadata encoder produces a 1-bit absolute coding flag, $flag_{abs}$, to distinguish between absolute and differential coding.

In the case of absolute coding, the coding flag, $flag_{abs}$, is set to 1, and is followed by the B_{az}-bit (or B_{el}-bit) index coded using absolute coding, where B_{az} and B_{el} refer to the indexes of the azimuth and elevation parameters to be coded, respectively.

In the case of differential coding, the 1-bit coding flag, $flag_{abs}$, is set to 0 and is followed by a 1-bit zero coding flag, $flag_{zero}$, signaling a difference Δ between the B_{az}-bit index (respectively the B_{el}-bit index) in the current and previous frames equal to 0. If the difference Δ is not equal to 0, the metadata encoder continues coding by producing a 1-bit sign flag, $flag_{sign}$, followed by a difference index, of which the number of bits is adaptive, in a form of a unary code indicative of the value of the difference Δ. Figure 5.6-3 is a diagram showing different scenarios of bit-stream coding of one metadata parameter.

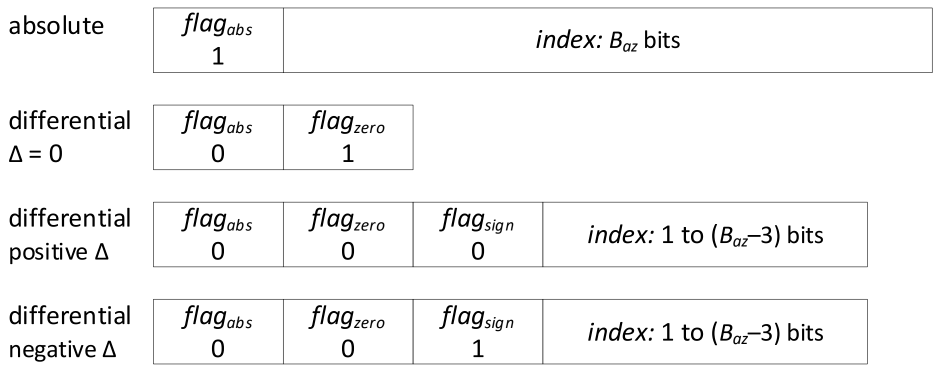


Figure 5.6-3: Different scenarios of bit-stream coding of one metadata parameter in ISM format.

There are thus in total four (4) different scenarios how to encode a metadata parameter:

Scenario 1: In the case of absolute coding (first line of Figure 5.6-3), the absolute coding flag, $flag_{abs}$, and the B_{az}-bit index (respectively the B_{el}-bit index) are transmitted.

Scenario 2: In the case of differential coding with the difference Δ between the B_{az}-bit index (respectively the B_{el}-bit index) in the current and previous frames equal to 0 (second line of Figure 5.6-3), the absolute coding flag $flag_{abs} = 0$, and the zero coding flag $flag_{zero} = 1$ are transmitted;

Scenario 3: in the case of differential coding with a positive difference Δ between the B_{az}-bit index (respectively the B_{el}-bit index) in the current and previous frames (third line of Figure 5.6-3), the absolute coding flag $flag_{abs} = 0$, the zero coding flag $flag_{zero} = 0$, the sign flag $flag_{sign} = 0$, and the difference index (1 to ($B_{az} - 3$)-bits index (respectively 1 to ($B_{el} - 3$)-bits index)) are transmitted; and

Scenario 4: in the case of differential coding with a negative difference Δ between the B_{az}-bit index (respectively the B_{el}-bit index) in the current and previous frames (last line of Figure 5.6-3), the absolute coding flag $flag_{abs} = 0$, the zero coding flag $flag_{zero} = 0$, the sign flag $flag_{sign} = 1$, and the difference index (1 to ($B_{az} - 3$)-bits index (respectively 1 to ($B_{el} - 3$)-bits index)) are transmitted.

5.6.4.1.1 Intra-object metadata coding logic

The logic used to set absolute or differential coding is further extended by an intra-object metadata coding logic. Specifically, in order to limit a range of metadata coding bit-budget fluctuation between frames and thus to avoid a too low bit-budget left for the core-encoders, the metadata encoder limits absolute coding in a given frame to one, or generally to a number as low as possible of, metadata parameters.

In the case of azimuth and elevation metadata parameter coding, the metadata encoder uses a logic that avoids absolute coding of the elevation index in a given frame if the azimuth index was already coded using absolute coding in the same frame. In other words, the azimuth and elevation parameters of one audio stream are never both coded using absolute coding in the same frame. As a consequence, the absolute coding flag, $flag_{abs,el}$, for the elevation parameter is not transmitted in the audio stream bit-stream if the absolute coding flag, $flag_{abs,az}$, for the azimuth parameter is equal to 1, and vice versa.

5.6.4.1.2 Inter-object metadata coding logic

The metadata encoder applies a similar logic from Clause 5.6.4.1.1 to metadata coding of different audio streams. The implemented inter-object metadata coding logic minimizes the number of metadata parameters of different audio streams coded using absolute coding in a current frame. This is achieved by the metadata encoder by controlling frame counters of metadata parameters coded using absolute coding chosen from robustness purposes and represented by the parameter $\beta = 10$ frames. In general, a scenario where the metadata parameters of the audio streams evolve slowly and smoothly is considered. In order to control decoding in a noisy channel where indexes are coded using absolute coding every β frames, the azimuth B_{az}-bit index of audio stream #1 is coded using absolute coding in frame M, the elevation B_{el}-bit index of audio stream #1 is coded using absolute coding in frame $M + 1$, the azimuth B_{az}-bit index of audio stream #2 is encoded using absolute coding in frame $M + 2$, the elevation B_{el}-bit index of stream #2 is coded using absolute coding in frame $M + 3$, etc.

An advantage of the inter-object metadata coding logic and the intra-object metadata coding logic is to limit a range of fluctuation of the metadata coding bit-budget between frames. Moreover, they increase robustness of the codec in a noisy channel; when a frame is lost, then only a limited number of metadata parameters from the audio objects coded using absolute coding is lost. Consequently, any error propagated from a lost frame affects only a small number of metadata parameters across the audio objects and thus does not affect whole audio scene.

5.6.4.2 Extended metadata encoding and quantization

In addition to the direction metadata (azimuth and elevation) described in clause 5.6.4.1, the metadata is extended for higher bitrate operations ($ism\_total\_brate \geq 64$ kbps) of DiscISM mode to optionally include yaw φ, pitch ψ, and radius r. The default values and the range for these parameters are:

$$\varphi = 0; \quad \in [-180, 180] \tag{5.6-29}$$

$$\psi = 0; \quad \in [-90, 90] \tag{5.6-30}$$

$$r = 1; \quad \in [0, 15.75] \tag{5.6-31}$$

Yaw φ and pitch ψ are used to define the directivity of the object. Yaw represents the horizontal orientation of the source around the z-axis which is perpendicular to the ground plane. Meanwhile pitch represents the vertical orientation around the y-axis. The quantization of the extended metadata (yaw and pitch) follows the same procedure with the direction metadata (azimuth and elevation) which is explained in detail in clause 5.6.4.1.

The radius defines the distance from the origin (0,0) when the listener is also placed at the origin by default. The radius is set to $r = 1$ by default placing the source(s) at the one-unit distance on the unit circle around the listener. The quantization of the radius is uniform with a unit step value of 0.25.

The gain metadata coding operation is the same with the direction metadata coding where absolute or differential coding is used according to the principles described in Clause 5.6.4.1.

5.6.4.3 Panning gain in non-diegetic rendering

5.6.5 Bitstream structure

Figure 5.6-4 is a schematic diagram illustrating the structure of the bit-stream produced by the bitstream multiplexer from Figure 5.6-1 at the encoder. Regardless whether metadata are present and transmitted or not, the structure of the bit-stream is structured as illustrated in Figure 5.6-4.

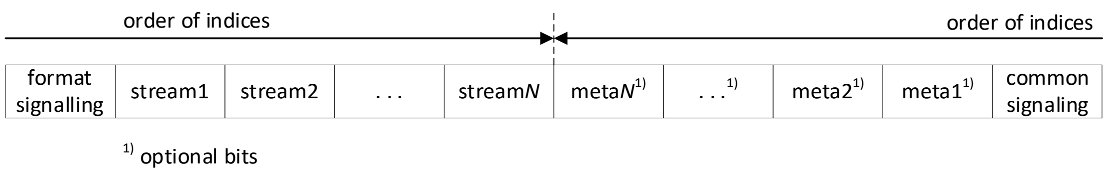


Figure 5.6-4: Structure of ISM format bitstream.

5.6.5.1 ISM signalling

The bitstream multiplexer from Figure 5.6-1 first writes the ISM common signalling from the end of the bit-stream. The ISM common signalling is produced by the configuration and decision processor from Figure from Figure 5.6-1 and comprises a variable number of bits representing:

a) a number of ISMs, N_{ISM}: the signalling of the number NISM of coded ISMs present in the bitstream is in the form of a unary code with a stop bit (e.g. for $N_{ISM} = 3$ ISMs, the first 3 bits of the ISM common signalling are "110").

b) extended metadata presence flag: written at $ism\_total\_brate \geq 64$ kbps, one one-bit flag per frame;

c) non-diegetic object flag: written only when extended metadata presence flag is equal to 1, one one-bit flag per frame;

d) the ISM importance class: comprises two bits per audio stream to indicate the ISM importance class, $class_{ISM}$, (ISM_NO_META, ISM_LOW_IMP, ISM_MEDIUM_IMP, and ISM_HIGH_IMP), as defined in Clause 5.6.2.2.1;

(e) a metadata presence flag, $flag_{meta}$: written if metadata are not present or the $class_{ISM}$= ISM_NO_META; the flag comprises one bit per ISM to indicate whether metadata for that particular ISM are present ($flag_{meta} = 1$) or not ($flag_{meta} = 0$) in the bitstream;

f) an ISM VAD flag, $flag_{VAD}$: the ISM VAD one-bit flag is transmitted per ISM when $flag_{meta} = 0$, respectively $class_{ISM}$ = ISM_NO_META, and distinguishes between the following two cases:

1) input metadata are not present so that the audio stream needs to be coded by an active coding mode ($flag_{VAD} = 1$); and

2) input metadata are present and transmitted so that the audio stream can be coded by an inactive coding mode ($flag_{VAD} = 0$).

5.6.5.2 Coded metadata payload

The bitstream multiplexer is next supplied with the coded metadata and writes the metadata payload sequentially from the end of the bitstream for the ISMs for which the metadata are coded in the current frame. The metadata bit-budget for each audio object is not constant but rather inter object and inter frame adaptive. Different metadata format scenarios are shown in Figure 5.6-3.

In the case that metadata are not present or are not transmitted for at least some of the ISMs, the metadata flag is set to 0 for these ISMs. Then, no metadata indices are sent in relation to those audio objects, i.e. $bits_{meta}[n] = 0$.

5.6.5.3 Audio streams payload

Finally, the multiplexer receives the NISM audio streams coded by the N_{ISM} core encoders through the N_{ISM} transport channels, and writes the audio streams payload sequentially for the N_{ISM} audio streams in chronological order from the beginning of the bitstream (See Figure 5.6-4). The respective bit-budgets of the N_{ISM} audio streams are fluctuating as a result of the bitrate adaptation algorithm described in Clause 5.6.2.2.1.

5.6.6 DTX operation

This Clause describes a method for discontinuous transmission (DTX) of audio objects (ISMs) in the ISM format within IVAS. It consists in analysing the audio streams for producing signal activity information on the audio streams, detecting, in response to the activity information on the audio streams, a DTX signal segment of the audio objects, producing a SID frame within the DTX signal segment, wherein the segment and frame detection comprises (a) updating a global SID counter of inactive frames, and (b) signalling the SID frame within the DTX signal segment depending on a value of the global SID counter, and finally encoding the SID frame using SID frame coding of CNG parameters.

5.6.6.1 ISM DTX operation overview

Similar to the EVS, the ISM DTX coding employs a more efficient DTX coding (“regular”) strategy at lower bitrates while a conservative coding strategy is used at higher bitrates. The conservative coding means that the DTX segments correspond to signals with a very low energy. The DTX strategy overview per number of ISMs and per IVAS total bitrate is summarized in Table 5.6-2.

Table 5.6-2: Overview of DTX strategies in ISM format.

bitrate [kbps]	number of ISMs			
	1	2	3	4
13.2	regular	n/a	n/a	n/a
16.4	regular	regular	n/a	n/a
24.4	regular	regular	regular	regular
32	conservative	regular	regular	regular
48	conservative	regular	regular	regular
64	conservative	conservative	regular	regular
80	conservative	conservative	regular	regular
96	conservative	conservative	conservative	regular
128	conservative	conservative	conservative	conservative
160	n/a	conservative	conservative	conservative
192	n/a	conservative	conservative	conservative
256	n/a	conservative	conservative	conservative
384	n/a	n/a	conservative	conservative
512	n/a	n/a	n/a	conservative

In order to correctly set the number of transport channels and to enable smooth transitions between active and inactive segments, the CNG spatial parameters depend on the ISM mode (DiscISM, ParamISM). Otherwise, the DTX/SID classification and CNG core-coding are the same regardless the ISM mode.

In the DiscISM mode, the number of core-encoders (SCEs) is usually equal to the number N_{ISM} of streams. In case of ParamISM, the number of core-encoders (SCE) is two (2) which less than the number of audio streams (3 or 4). It is obvious that coding several ISMs in an SID frame would result in N_{ISM} times (2.4 kbps + metadata) bitrate which would further result in a too high IVAS SID bitrate. In order to keep it reasonably low, the IVAS ISM SID bitrate is set to 5.2 kbps similarly as in other IVAS formats and an efficient coding of SID frames is employed.

5.6.6.2 Classification of Inactive Frames in ISM Encoder

As described in Clause 5.6.2.1, the audio stream analysis and front pre-processing block classifies the input audio streams and produces the voice/signal activity detection (VAD/SAD) flag f_{SAD}, one per audio stream, while the SAD flag with DTX hangover addition as described in Paragraph 5.1.12.8 of Reference [3] is employed. A change from $f_{SAD} = 1$ to $f_{SAD} = 0$ indicates a start of inactive signal segment for a particular audio signal. It is obvious that the start of an inactive signal segment usually happens at different time instances for different audio streams.

By default, a DTX segment is declared when an inactive signal segment is declared for all audio streams. Then, additional classification stages are used to classify the inactive sound signal segments. Also, the values of metadata have an impact whether a frame is declared as an inactive frame. The classification logic of inactive (SID or NO_DATA) frames or active frames is shown in Figure 5.6-5.

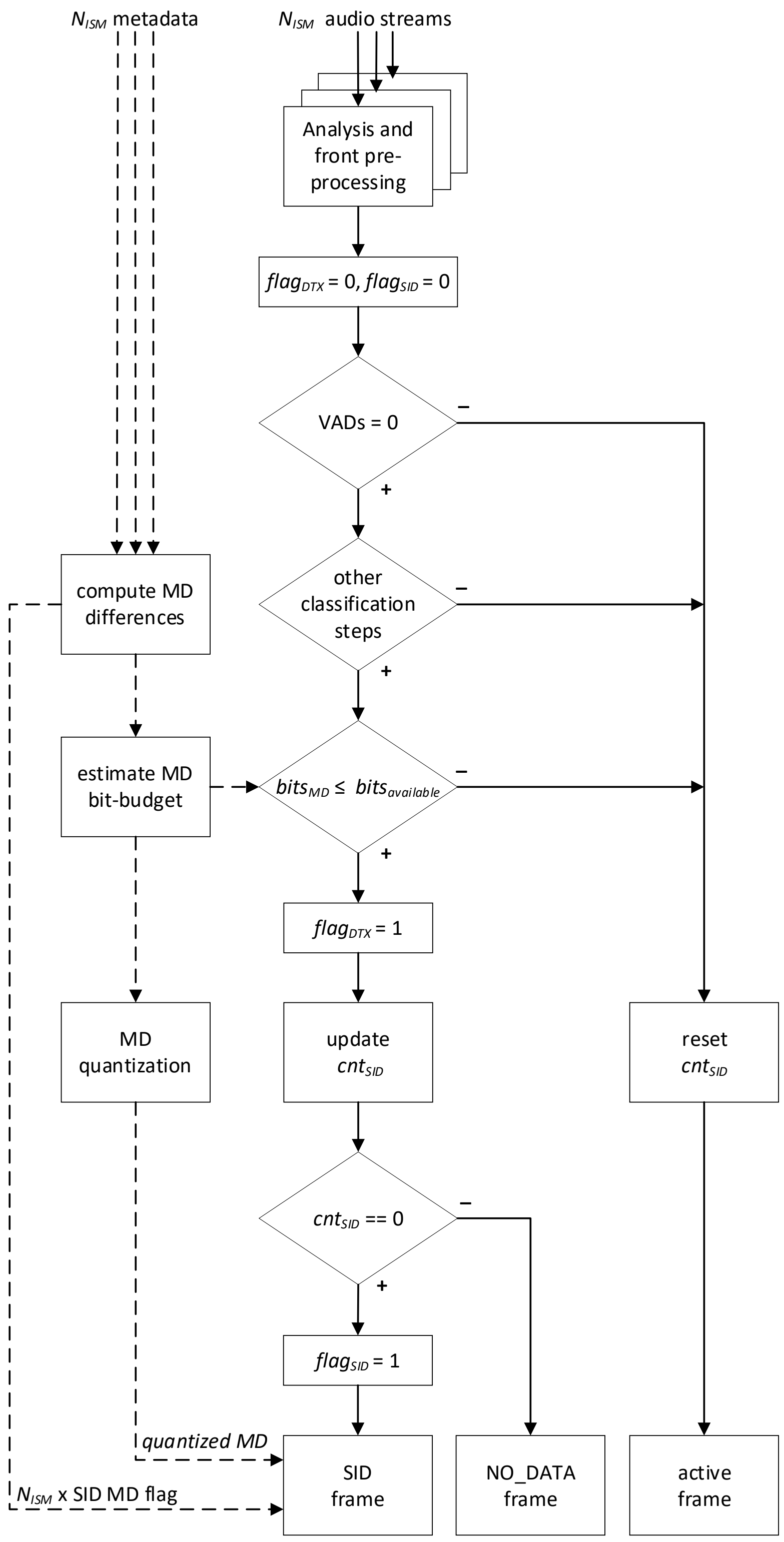


Figure 5.6-5: Flow chart of a SID/DTX controller logic in the DTX ISM format operation.

5.6.6.2.1 Global SID Counter

Figure 5.6-5 is a flow chart illustrating a DTX controller used in the DTX operation implemented in the ISM encoder from Figure 5.6-1. Referring to Figure 5.6-5, in order to control the SID/NO_DATA frame decision and the start of a DTX segment, the DTX controller uses a global SID counter, cnt_{SID}. This global SID counter cnt_{SID} allows to control the SID update rate and synchronize the SID/NO_DATA across all audio streams. Consequently, the global SID counter cnt_{SID} enables efficient tuning of a hang-over (hysteresis) DTX logic, or permits other than a default SID update rate of 8 frames or SID adaptive update rate. The global SID counter cnt_{SID} is thus superior to per audio stream SID counters and it effectively ensures that the individual per audio stream SID counters are synchronized.

Referring to Figure 5.6-5, the DTX controller receives the information from the analysis and front pre-processing block, including the SAD information. The DTX controller then initializes to “0” the DTX flag, $flag_{DTX}$, and the SID flag $flag_{SID}$.

By default, the DTX controller detects a DTX signal segment (SID or NO_DATA frame) when the VAD flags, $flag_{VAD}[n],\ n = 0, \dots, N_{ISM} - 1$, of all audio streams are equal to 0 and signals it by setting the DTX flag to 1, i.e. $flag_{DTX}$. This can be expressed using the following relation:

$$flag_{DTX} = \begin{cases} 1 & \text{if } flag_{VAD} = 0 \quad \text{for all streams} \\ 0 & \text{otherwise} \end{cases} \tag{5.6-32}$$

where $flag_{DTX}$ is the DTX flag. When the VAD flags, $flag_{VAD}[n],\ n = 0, \dots, N_{ISM} - 1$, of all audio streams are not equal to 0, the DTX controller signals an active frame and selects active frame coding.

Further, the DTX controller signals SID frames within the DTX signal segment using a SID flag, $flag_{SID}$. The SID flag $flag_{SID}$ is set (a) to 1 when the global SID counter cnt_{SID} equals to 0 in which case the DTX controller signals a SID frame and selects SID frame coding and (b) to 0 when the global SID counter cnt_{SID} does not equal to 0 in which case the DTX controller signals a NO_DATA frame and selects NO_DATA frame coding. This can be expressed using the following relation:

$$flag_{SID} = \begin{cases} 1 & \text{if } cnt_{SID} = 0 \\ 0 & \text{otherwise} \end{cases} . \tag{5.6-33}$$

Next, the DTX controller resets the SID counter cnt_{SID} to -1 in every active frame and it is incremented by 1 at every inactive frame up to the value corresponding to the SID update rate (by default 8 frames). If the value of counter cnt_{SID} reaches the SID update rate, it is set to 0 and the DTX controller signals a SID frame and selects SID frame coding.

The DTX controller can alter the DTX flag $flag_{DTX}$ by using other classification stages based on core-encoder preprocessing values for all audio objects. Specifically, the DTX controller comprises a logic that forces active frame coding ($flag_{DTX} = 0$) and selection of active frame coding in cases when 1) mean value of the LT (Long-Term) background noises over all audio streams, $noise_{mean}$, is higher than a first threshold β_1, or 2) mean value of LT background noises over all audio streams, $noise_{mean}$,, is higher than a second threshold β_2 and LT background noise variations over all audio streams, $noise\_var_{mean}$,, is higher than a third threshold β_3. This can be expressed using the relation:

$$flag_{DTX} = 0 \quad \text{if} \quad noise_{mean} > \beta_1 \text{ or } (noise_{mean} > \beta_2 \text{ and } noise\_var_{mean} > \beta_3) \tag{5.6-34}$$

where the thresholds are set to $\beta_1 = 50$, $\beta_2 = 10$, and $\beta_3 = 2$. The assessment of the LT background noise variations is introduced in order to check for background noise similarities among all streams. The parameter $noise_{mean}$, represents the mean value of the long-term background noise energy values of all audio streams (see Paragraph 5.1.11 in Reference [3] for details about the background noise energy). The parameter $noise\_var_{mean}$, then represents the variation of the long-term background noise energy values of all audio streams.

5.6.6.3 SID Metadata Analysis, Quantization and Coding

The analysis, quantization and coding of metadata in SID frames follows several principles from active frame coding as described in Clause 5.6.4.1 though there are a few differences. First, in the SID frame, always only two metadata parameters, azimuth and elevation (as included in the NISM input metadata), are coded. Then, both parameters are coded absolutely meaning that the differential coding is not used in order to avoid potential degradation in long segments of inactive frames due to a lost SID frame in case of a noisy channel if relative coding would be used.

The IVAS SID bitrate is 5.2 kbps from which roughly one half is reserved for a metadata (MD) payload. In order to transmit as much as possible MD values, some compromises are made.

First, one possibility to match the bit-budget constraints is to lower the resolution of MD values. Specifically, when the number of audio objects is low, the available bit-budget for coding the metadata (MD) is relatively generous and the MD resolution is thus kept relatively high. On the other hand, when the number of coded audio objects is high, the resolution of MD values is relatively low and less bits are needed to encode them. There are thus two scenarios used:

a) azimuth and elevation indexes are encoded by means of $B_{az} = 8$ bits and $B_{el} = 7$ bits when one or two audio streams are coded,

b) azimuth and elevation indexes are encoded by means of $B_{az} = 6$ bits and $B_{el} = 5$ bits when three or four audio streams are coded,

while the actual number of coded bits and thus the resolution is explicitly known from the ISM common signalling (see Clause 5.6.6.5.1 below).

Second, in order to keep the SID bit-budget as low as possible, a saving in MD bit-budget is achieved by computing a flag, one flag per audio stream, indicating that the MD parameters have not changed (or has not changed significantly) since the last frame and consequently that the metadata (MD) parameters for a specific audio stream are not coded and transmitted. Similarly, this flag serves as an indication that the input MD parameters are not present for that specific audio object.

The DTX controller computes a MD on/off flag, $flag_{MD}[n]$, $n = 0, \ldots, N_{ISM} - 1$, for all MD parameter values. Specifically, a flag $flag_{MD,\theta}$, is calculated for the azimuth MD parameter using the following relation:

$$flag_{MD,\theta}[n] = \begin{cases} 1 & \text{if } |\theta[n] - \theta_{last}[n]| > \delta_\theta \\ 0 & \text{otherwise} \end{cases} \quad (5.6\text{-}35)$$

where $\theta[n]$ is the current frame azimuth for stream n, $\theta_{last}[n]$ is the last frame azimuth for stream n and δ_θ is azimuth maximum difference value set to $\delta_\theta = 10$. In the same manner (see Equation (5.6-35)), the DTX controller computes a flag $flag_{MD,\varphi}[n]$ for the elevation MD parameter of stream n while the elevation maximum difference value is set to $\delta_\varphi = 10$. The final MD on/off flag $flag_{MD}[n]$ for one audio stream n is obtained as a logical OR (symbol V in the Equation (5.6-36) below) operation between all particular MD flags:

$$flag_{MD}[n] = flag_{MD,\theta}[n] \ \vee \ flag_{MD,\varphi}[n]\,. \quad (5.6\text{-}36)$$

Then, the flag $flag_{MD}[n]$ represented as a 1-bit information per audio stream, is inserted into the bit-stream in the SID frame right after the ISM signalization of the number N_{ISM} of coded audio streams (see Clause 5.6.6.5.1 below).

Third, the DTX controller comprises a mechanism that estimates the bit-budget $bits_{MD}$ for metadata quantization. It should be noted that it is only the bit-budget for the quantization of metadata values which is estimated (computed in advance) at this stage while the quantization itself is possibly performed only later. When the estimated bit-budget $bits_{MD}$ is higher than a maximum available bit-budget for the MD coding, $bits_{available}$, the flag $flag_{DTX}$ is reset to 0 (see Figure 5.6-5) and active frame coding is performed. On the other hand, when the estimated MD bit-budget, $bits_{MD}$, is lower or equal to the maximum available bit-budget for the MD coding, $bits_{available}$, the flag $flag_{DTX}$ is not changed, i.e. $flag_{DTX} = 1$, and the DTX coding segment continues. While the maximum available bit-budget for the MD coding, $bits_{available}$, is computed as the difference between the codec SID bit-budget minus the bit-budget needed for other than MD coding (e.g. SID frame signalling, core-coder SID bit-budget, spatial information bit-budget, ISM signalling) in the SID frame.

5.6.6.3.1 CNG parameters encoding

5.6.6.4 ISM DTX core encoding

In case of SID frame, the core-encoder SID bitrate of 2.4 kbps is assigned to one SCE core-encoder in which a FD-CNG from EVS (see Clause 5.6.3 in [3]) is used. The processing in other core-coders is then skipped.

5.6.6.5 SID bitstream structure

Figure 5.6-6 is a schematic diagram illustrating, for a frame, the structure of the SID bitstream produced by the bitstream multiplexer from Figure 5.6-1 and transmitted from the ISM format encoder. Regardless whether metadata are present and transmitted or not, the structure of the SID bit-stream is structured as illustrated in Figure 5.6-6.

First the bitstream multiplexer writes the indices of SID format signalling followed by the indices of one core-encoder SID from the beginning of the bit-stream while the indices of ISM common signalling, spatial information, and metadata are written from the end of the bitstream.

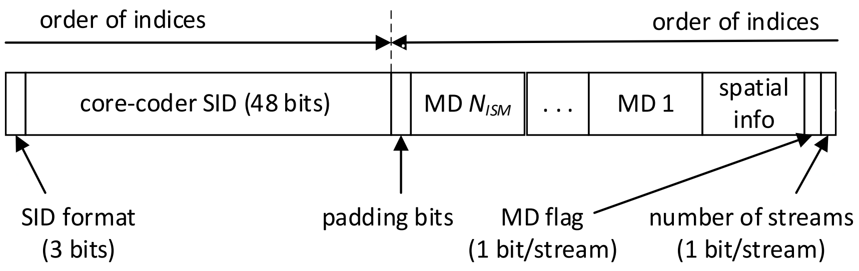


Figure 5.6-6: Structure of SID ISM format bitstream.

Further the Table 5.6-3 summarizes the bit-budget of different indexes present in the SID frame depending on the number of transported ISMs.

Table 5.6-3: SID payload in ISM format

	DiscISM mode				**ParamISM mode**	
parameter	**1 ISM**	**2 ISMs**	**3 ISMs**	**4 ISMs**	**3 ISMs**	**4 ISMs**
SID format	3	3	3	3	3	3
core-coder SID	48	48	48	48	48	48
number of streams	1	2	3	4	3	4
SID metadata flag	1	2	3	4	3	4
ISM mode flag	n/a	n/a	1	1	1	1
noisy speech flag	n/a	n/a	n/a	n/a	1	1
SCE ID	n/a	1	2	2	1	1
coherence	n/a	4	8	12	4	4
azimuth [max]	8	16	18	24	18	24
elevation [max]	7	14	15	20	15	20
padding bits	36	10	3	-14$^{1)}$	7	-6$^{1)}$
total	104	104	104	104	104	104

Note 1): The negative values mean that an active frame coding can be used as a result of the MD quantization logic following the third principle from Clause 5.6.6.3.

5.6.6.5.1 ISM Common Signalling in SID Frame

The bitstream multiplexer writes the ISM common signalling from the end of the bitstream. The ISM common signalling in SID frame is produced by the configuration and decision processor (bit-budget allocator) and comprises a variable number of bits representing:

(a) a number N_{ISM} of audio objects: the signaling for the number N_{ISM} of coded audio streams present in the bitstream is in the form of a unary code with a stop bit similarly as in the active frame (for example, for $N_{ISM} = 3$ audio objects, the first 3 bits of the ISM common signaling would be "110" written in a backward order).

(b) a metadata on/off flag $flag_{MD}[n]$, $n = 0, \ldots, N_{ISM} - 1$, one per audio stream, as defined in Equation (5.6-36).

5.6.6.5.2 SID Data Payload

In a SID frame, right after the ISM common signalling, in the backward order, there are written into the bitstream a) spatial information indices, and finally b) metadata values as quantized in Clause 5.6.6.3 above.

5.6.6.5.3 SID Audio Streams Payload

In active frames, the multiplexer receives N_{ISM} audio streams coded by N_{ISM} core-encoders through N_{ISM} transport channels, and writes them from the beginning of the bitstream right after the IVAS format bits as described in Clause 5.6.5.3.

However, in SID frames, the bitstream multiplexer receives one audio stream coded by one of the core-encoders through one transport channel, and writes it from the beginning of the bitstream (see Figure 5.6.5) right after the IVAS SID format bits (signalization of the SID mode related to the IVAS format).

5.6.7 Bitrate switching

5.7 Multi-channel audio (MC) operation

5.7.1 MC format overview

Coding of multi-channel (MC) inputs is available for the channel layouts 5.1, 7.1, 5.1+2, 5.1+4, and 7.1+4. The coding technique is selected from a set of MC coding modes based on the the available bitrate and specified channel layout. The general principle in technique selection is to aim for best possible quality given the allowed bitrate. The MC coding modes thus include multi-channel MASA (McMASA, Clause 5.7.3) mode, Parametric multi-channel coding (ParamMC, Clause 5.7.4) mode, Parametric multi-channel upmix (ParamUpmix, Clause 5.7.5) mode, and discrete multi-channel coding (DiscMC, Clause 5.7.5.1) mode. The use of individual modes is summarized in Table 5.7-1. For all techniques, LFE channel coding is also offered either separately or within the technique. The multi-channel operation supports output to mono, stereo, multi-channel (at the same or any other layout with up to 16 speakers), Ambisonics (at up to order 3), and binaural. Full bitrate switching is supported.

Table 5.7-1: Overview of bitrates and coding modes in MC format

Bitrate [kbps]	MC layout				
	5.1	7.1	5.1.2	5.1.4	7.1.4
13.2, 16.4, 24.4, 32	McMASA	McMASA	McMASA	McMASA	McMASA
48, 64, 80	ParamMC	ParamMC	ParamMC	McMASA	McMASA
96	DiscMC	ParamMC	ParamMC	ParamMC	McMASA
128	DiscMC	DiscMC	DiscMC	ParamMC	ParamMC
160	DiscMC	DiscMC	DiscMC	DiscMC	ParamUpmix
192, 256, 384, 512	DiscMC	DiscMC	DiscMC	DiscMC	DiscMC

5.7.2 Common multi-channel processing

5.7.2.1 LFE channel encoding

5.7.2.1.1 Common LFE pre-processing

5.7.2.1.1.1 Low-pass filtering

The LFE signal $lfe_{inp}(n)$ of the input audio sampled at 16, 32 or 48 kHz, n being the sample index, is low-pass filtered to suppress undesired high frequency components for subwoofer. The LFE encoder low-pass filter LP_{lfe_enc} is a 4th order Butterworth filter with cut-off frequency of 130 Hz (–3 dB). The transfer function of LP_{lfe_enc} is given by

$$H_{130Hz}(z) = \left(\frac{b_{00}+b_{01}z^{-1}+b_{02}z^{-2}}{1+a_{01}z^{-1}+a_{02}z^{-2}}\right)\left(\frac{b_{10}+b_{11}z^{-1}+b_{12}z^{-2}}{1+a_{11}z^{-1}+a_{12}z^{-2}}\right) \quad (5.7\text{-}1)$$

The coefficients of the LP_{lfe_enc} filter for a given input sampling frequency are given in the table below.

Table 5.7-2: Coefficients of the 130Hz LP_{lfe_enc} filter

	16 kHz	32 kHz	48 kHz
$\boldsymbol{b_{00}}$	3.97464794223146e-07f	2.56674586654460e-08f	5.12617881476274e-09f
$\boldsymbol{b_{01}}$	7.94929601777927e-07f	5.13349181918215e-08f	1.02523584294987e-08f
$\boldsymbol{b_{02}}$	3.97464788468481e-07f	2.56674582938215e-08f	5.12617879059970e-09f
$\boldsymbol{b_{10}}$	1.0f	1.0f	1.0f
$\boldsymbol{b_{11}}$	1.99999996645833f	1.99999996645833f	1.99999984394358f
$\boldsymbol{b_{12}}$	1.00000001447843f	1.00000001447843f	1.00000000471366f
$\boldsymbol{a_{01}}$	-1.90746797905194f	-1.95329015717623f	-1.96875982668433f
$\boldsymbol{a_{02}}$	0.909956295365785f	0.953926661219383f	0.969044914826862f
$\boldsymbol{a_{11}}$	-1.95913666348268f	-1.98000953138860f	-1.98677297369091f
$\boldsymbol{a_{12}}$	0.961692382252710f	0.980654742275836f	0.987060670205863f

The input signal, filtered by the $\boldsymbol{LP_{lfe\_enc}}$ filter, is denoted as $LFEin_{LP}(n)$.

5.7.2.1.2 MDCT based LFE encoding

In Parametric upmix encoding mode (see subclause 5.6.4) and MCT (Clause 5.6.5), the LFE channel is coded with a MDCT based encoding approach.

5.7.2.1.2.1 Windowing and MDCT

MDCT based coding of the LFE channel is based on applying a symmetric Kaiser-Bessel derived (KBD) window of 20 ms length followed MCDT transform. This operation is applied in two subframes of the window size, i.e., twice per frame with a stride of 10 ms.

As illustrated in the Figure below, the left half of the window applied before MDCT comprises a 1 ms zero-pad portion, followed by a 8 ms cross-over portion and a 1 ms portion equalling 1. The right half of the window is obtained through mirroring at the 10 ms point. The size of the cross-over portion of 8 ms causes a corresponding lookahead delay.

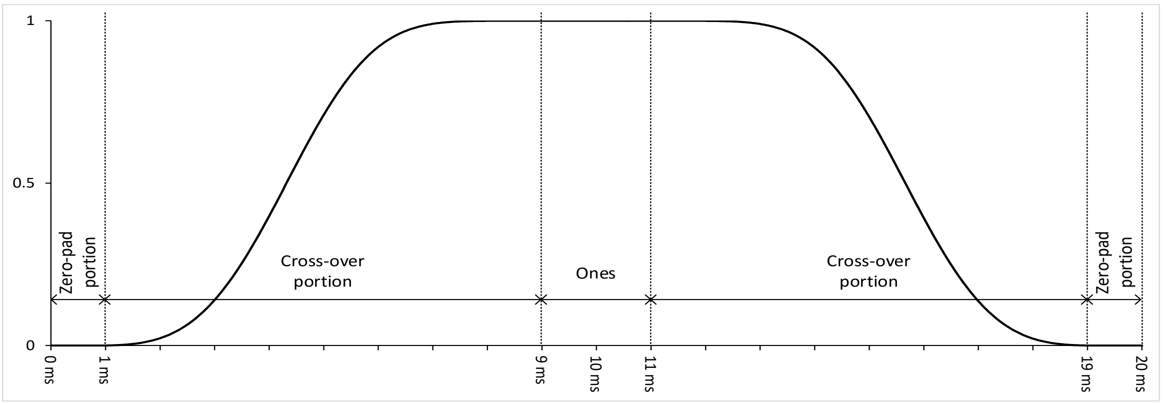


Figure 1: KBD window used in MDCT based LFE channel codec

In each run of the windowing and MDCT operation the windowed samples $x_w(n)$ are subsequently transformed using an L-point MDCT of the following form:

$$X(k) = \sum_{n=0}^{L-1} x_w(n) \cos\left[\frac{2\pi}{L}\left(n+\frac{1}{2}+\frac{L}{4}\right)\left(k+\frac{1}{2}\right)\right], \; k = 0 \ldots \frac{L}{2}-1 \quad , \qquad (5.7\text{-}2)$$

where L equals 960, 64 or 320 for an input audio sampling frequency of, respectively, 48000 Hz, 32000 Hz or 16000 Hz.

The MDCT is efficiently computed using by means of FFT. This requires the following steps.

Firstly, the windowed samples are folded by subdividing the buffer into 4 portions of length $L/4$ and then adding the sign-inverted and time reversed first portion to the second portion and adding the time-reversed forth portion to the third portion:

$$x_{wf}(n) = x_w\left(n+\frac{L}{4}\right) - x_w\left(-n+\frac{L}{4}-1\right),\ n = 0 \ldots \frac{L}{4}-1 \quad \text{, and} \tag{5.7-3}$$

$$x_{wf}(n) = x_w\left(n+\frac{L}{4}\right) + x_w(-n+L-1),\ n = \frac{L}{4} \ldots \frac{L}{2}-1 \quad . \tag{5.7-4}$$

Next, the folded samples are twiddled and combined to a sequence of $L/4$ complex samples, as follows:

$$\tilde{x}_{wf}(n) = -x_{wf}(2n)e^{j\frac{2\pi}{L}(n+1/8)} + jx_{wf}\left(\frac{L}{2}-2n-1\right)e^{j\frac{2\pi}{L}(n+1/8)} \quad . \tag{5.7-5}$$

After these preparations, the buffer $\tilde{x}_{wf}(n)$ of $L/4$ samples is transformed into frequency domain using a complex valued FFT, yielding$\tilde{X}_{wf}(k)$.

Finally, to generate the MDCT output samples, the buffer $\tilde{X}_{wf}(k)$ is firstly scaled with an MDCT gain scaling factor $g = g(L)$ that depends on the applied window length L. This removes gain dependencies on the input audio sampling frequency. After that, the following post-rotation operation is carried out with the same twiddling factors as used prior to the FFT:

$$\begin{aligned} X\left(\frac{L}{2}-2k-1\right) &= \Re\mathrm{e}\left(g\tilde{X}_{wf}(k)e^{j\frac{2\pi}{L}(k+1/8)}\right) \\ X(2k) &= \Im\mathrm{m}\left(g\tilde{X}_{wf}(k)e^{j\frac{2\pi}{L}(k+1/8)}\right) \end{aligned} \ ,\ k = 0 \ldots \frac{L}{4}-1 \quad . \tag{5.7-6}$$

Specifically, the MDCT coefficients for the first and second subframes are respectively denoted as $X_1(k)$ and $X_2(k)$.

5.7.2.1.2.2 Quantization and entropy coding

Quantization of MDCT coefficients (following the LPF curve)

Arithmetic coding

5.7.2.1.2.2.1 Silence mode LFE coding

5.7.3 Multi-channel MASA (McMASA) coding mode

5.7.3.1 McMASA coding mode overview

Multi-channel MASA (McMASA) encodes multi-channel playback audio signals that it receives using transport audio signals, MASA spatial audio parameters, and LFE parameters. The MASA spatial audio parameters comprise “directional” (or “spatial”) parameters (azimuth, elevation, and direct-to-total energy ratio) and “coherence” parameters (spread coherence and surround coherence). The LFE parameters contain a lower frequency effect (LFE) information parameter indicating the proportion of the LFE energy in the multi-channel signals, called the LFE-to-total energy ratio. The MASA spatial audio parameters are used together with the transport audio signals for providing spatial audio reproduction in the decoder. The LFE parameters are used together with the transport audio signals for determining the LFE signal in the decoder.

There are two operation modes in McMASA, “normal” and “separate-channel” mode. The mode is selected based on the bitrate. The separate-channel mode is used with bitrates equal to or larger than 64 kbps. The normal mode is used with bitrates smaller than 64 kbps. In case the separate-channel mode was selected, the separate channel index is determined, which equals to the centre channel index, i.e., $i_{sep} = i_c$.

TODO: REFER TO MCMASA/MASA CONFIG HERE, SETS THE NUMBER OF BANDS, ETC.

The input to the processing is time-domain multi-channel audio signals $s(n,i)$ (where n is the sample index and i the channel index). There are N input channels TODO: ADD REFERENCE TO EARLIER TEXT WHERE THE INPUT MULTI-CHANNEL FORMATS ARE DISCUSSED, E.G., 5.1, 7.1+4, ETC.

LFE energies are determined using the input multi-channel playback audio signals $s(n,i)$. This is explained in detail in clause 5.7.3.2. The outcome is LFE energy $E_{LFE}(m)$ for the first frequency band (where m is the subframe index). In addition, in the separate-channel mode, the total low-frequency energy $E_{LF,tot}(m)$ is obtained (which depicts the energy of all channels of the multi-channel signals in the same frequency band where the LFE energy was determined).

Then, the LFE signal is combined with the centre channel by

$$s'(n, i_c) = s(n, i_c) + s(n, i_{LFE})$$

where i_c is the channel index of the centre channel and i_{LFE} of the LFE channel. Other channels of $s'(n, i)$ correspond to $s(n, i)$.

When operating in the separate-channel mode, the channel to be separated is first identified based on the i_{sep}, and then the corresponding channel is separated to its own signal

$$s_{sep}(n) = s(n, i_{sep})$$

The remaining channels of the multi-channel signals are separated to the multi-channel signal $s''(n, i)$ (i.e., $s''(n, i)$ otherwise corresponds to $s'(n, i)$, but the channel i_{sep} is not present). In the normal mode, $s''(n, i)$ corresponds to $s'(n, i)$.

Using the multi-channel playback audio signals $s''(n, i)$, MASA spatial audio parameters are determined by analysing the multi-channel playback audio signals $s''(n, i)$. This is explained in detail in clause 5.7.3.3. As the outcome, azimuth $\theta(b, m)$, elevation $\phi(b, m)$, direct-to-total energy ratio $r_{dir}(b, m)$, spread coherence $\zeta(b, m)$, and surround coherence $\gamma(b, m)$ are obtained. They are determined for each frequency band b and subframe m of the multi-channel playback audio signals (there are 5 frequency bands and 4 subframes). In addition, in the normal mode, the total low-frequency energy $E_{LF,tot}(m)$ is obtained for the first frequency band.

TODO: SETTING ZEROS FOR HIGHER BANDS OF THE SPATIAL METADATA WITH LOW SAMPLING RATES, IF IT IS NOT MENTIONED ELSEWHERE.

The LFE-to-total energy ratio $\Xi(m)$ is determined using the LFE energy $E_{LFE}(m)$ and the total low-frequency energy $E_{LF,tot}(m)$

$$\Xi(m) = \frac{E_{LFE}(m)}{E_{LF,tot}(m)}$$

The LFE-to-total energy ratio $\Xi(m)$ is determined for the first frequency band (as were the LFE energy $E_{LFE}(m)$ and the total low-frequency energy $E_{LF,tot}(m)$). In the decoder, rendering based on the LFE-to-total energy ratio (and the transport audio signal(s)) enables the determination of the low frequency effect (LFE) channel.

McMASA transport audio signals are determined by downmixing the multi-channel playback audio signals $s''(n, i)$ (based on the analysis of the multi-channel playback audio signals $s''(n, i)$). The number of transport audio signals is therefore smaller than the number of multi-channel playback audio signals. This is explained in detail in clause 5.7.3.5. The outcome is the transport audio signal(s) $s_{trans}(n, i)$, having 1 or 2 channels, depending on the bitrate.

Using the determined the transport audio signals $s_{trans}(n, i)$ and MASA spatial audio parameters (azimuth $\theta(b, m)$, elevation $\phi(b, m)$, direct-to-total energy ratio $r_{dir}(b, m)$, spread coherence $\zeta(b, m)$, and surround coherence $\gamma(b, m)$), the sound scene captured (or in other words, represented) by the original multi-channel playback audio signals can then be reproduced in the decoder.

Next, it is checked if the coherence parameters are significant enough so that they are to be encoded and transmitted. First, all_coherence_zero parameter is set to one, meaning that no coherence value is significant enough to be transmitted. Then, all spread coherence values $\zeta(b, m)$ are compared against the MASA coherence threshold (having the value of 0.1). If the value for any time-frequency tile (b, m) is larger than the threshold, all_coherence_zero is set to zero. Then, if all_coherence_zero is still having the value of one, the significance of the surround coherence is inspected. This is done using the method described in clause X.X.X.X TODO: ADD REF. If surround coherence is determined to be significant, the value of all_coherence_zero is set to zero. Otherwise, it is kept as one.

The determined spatial audio metadata (azimuth $\theta(b, m)$, elevation $\phi(b, m)$, direct-to-total energy ratio $r_{dir}(b, m)$, spread coherence $\zeta(b, m)$, and surround coherence $\gamma(b, m)$) and LFE metadata (LFE-to-total energy ratio $\Xi(m)$) are provided for encoding. This is explained in detail in clause 5.7.3.6.

TODO: EXPLAIN SOMEWHERE THAT META ENCODING IS DONE FIRST, AND THE REMAINING BITS ARE GIVEN TO CORE CODEC.

The determined transport audio signal(s) $s_{trans}(n, i)$ are provided for encoding. In case of separate-channel mode, also the separated channel signal $s_{sep}(n)$ is provided for encoding. This is explained in detail in clause 5.7.3.7.

5.7.3.2 LFE energy computation

5.7.3.2.1 LFE energy computation in normal mode

First, the LFE signal $s(n, i_{LFE})$ is converted to the time-frequency domain using the filter bank described in clause X.X.X TODO: ADD REF. The outcome is the time-frequency domain LFE signal $S(k, m, i_{LFE})$ (where k is the frequency bin index, m the subframe index corresponding also to the slot index, and i_{LFE} is the channel index of the LFE signal). The LFE energy $E_{LFE}(m)$ is then computed for first frequency band (which corresponds to the four first frequency bins)

$$E_{LFE}(m) = \sum_{k=0}^{3} |S(k, m, i_{LFE})|^2$$

which corresponds to a frequency range from 0 to 400 Hz.

5.7.3.2.2 LFE energy computation in separate-channel mode

First, the LFE signal $s(n, i_{LFE})$ and the separate-channel signal $s(n, i_{sep})$ are delayed by $L_{lfe,ana,del}$ samples to provide the correct alignment. $L_{lfe,ana,del}$ corresponds to 9.5 milliseconds, and the value depends on the input sampling rate (e.g., 456 samples at 48 kHz). The result is the delayed versions $s_{del}(n, i_{LFE})$ and $s_{del}(n, i_{sep})$, which are next low-pass filtered.

The low-pass filtering is performed by

$$s_{lp}(n, i) = s_{lp}(n-1, i) + c_{lp} s_{del}(n, i) - c_{lp} s_{del}(n - L_{lp}, i)$$

where L_{lp} corresponds to 5 milliseconds (e.g., 240 samples at 48 kHz), and $c_{lp} = 1/L_{lp}$. The cross-over frequency of this filter is at 120 Hz; thus this signal is associated with the frequency band from 0 to 120 Hz.

The LFE energy $E_{LFE}(m)$ is then computed by

$$E_{LFE}(m) = \sum_{n=n_1(m)}^{n_2(m)} s_{lp}(n, i_{LFE})^2$$

and the total low-frequency energy $E_{LF,tot}(m)$ is computed by

$$E_{LF,tot}(m) = E_{LFE}(m) + \sum_{n=n_1(m)}^{n_2(m)} s_{lp}(n, i_{sep})^2$$

5.7.3.3 McMASA spatial audio parameter estimation

First, the multi-channel playback audio signals $s''(n, i)$ are converted to the time-frequency domain using the filter bank described in clause X.X.X TODO: ADD REF. The outcome is the time-frequency domain multi-channel signals $S(k, m, i)$ (where k is the frequency bin index, m the subframe index corresponding also to the slot index, and i is the channel index). It can be presented in a vector form as $\mathbf{s}(k, m)$ (which is a column vector with channels as rows).

There are N_{ana} analysis channels. $N_{ana} = N - 1$ for the normal mode, and $N_{ana} = N - 2$ for the separate-channel mode.

Next, the covariance matrix is computed by

$$\mathbf{C}_s(b, m) = \sum_{k=k_1(b)}^{k_2(b)} \mathbf{s}(k, m)\, \mathbf{s}(k, m)^H$$

where $()^H$ denotes the conjugate transpose, and k_1 and k_2 are the first and the last bin of the frequency band b. The size of the covariance matrix is N_{ana} x N_{ana}, and individual entry is referred to by $c_s(b, m, i, j)$, where i and j refer to loudspeaker channels.

The energy of the multi-channel signals is determined by

$$E(b,m) = \sum_{i=0}^{N_{ana}-1} c_s(b,m,i,i)$$

The diagonal entries are real, so the resulting energy values are real as well.

In the normal mode, the total low-frequency energy $E_{LF,tot}(m)$ is computed by

$$E_{LF,tot}(m) = \sum_{i=0}^{N_{ana}-1} \sum_{k=0}^{3} |S(k,m,i)|^2$$

which corresponds to the frequency band from 0 to 400 Hz.

The loudspeaker directions are described by azimuth $\theta_{LS}(i)$ and elevation $\phi_{LS}(i)$ angles. The multi-channel signals are converted to FOA signals (where the channels 0, 1, 2, and 3 correspond to spherical harmonics W, Y, Z, and X) by

$$S_{FOA}(k,m,0) = \sum_{i=0}^{N_{ana}-1} S(k,m,i)$$

$$S_{FOA}(k,m,1) = \sum_{i=0}^{N_{ana}-1} \sin\theta_{LS}(i)\cos\phi_{LS}(i)\, S(k,m,i)$$

$$S_{FOA}(k,m,2) = \sum_{i=0}^{N_{ana}-1} \sin\phi_{LS}(i)\, S(k,m,i)$$

$$S_{FOA}(k,m,3) = \sum_{i=0}^{N_{ana}-1} \cos\theta_{LS}(i)\cos\phi_{LS}(i)\, S(k,m,i)$$

The loudspeaker setup is converted also to an even distribution version of the original setup, see clause 5.7.3.4 for details. The FOA signals are computed also based on the even setup (the even setup is handled as a horizontal setup)

$$S_{FOAeven}(k,m,0) = \sum_{i=0}^{N_{ana}-1} S(k,m,i)$$

$$S_{FOAeven}(k,m,1) = \sum_{i=0}^{N_{ana}-1} \sin\theta_{LSeven}(i)\, S(k,m,i)$$

$$S_{FOAeven}(k,m,2) = 0$$

$$S_{FOAeven}(k,m,3) = \sum_{i=0}^{N_{ana}-1} \cos\theta_{LSeven}(i)\, S(k,m,i)$$

TODO: ADD DIRECTION AND DIFFUSENESS COMPUTATION ONCE CORRESPONDING DIRAC PARTS HAVE BEEN WRITTEN. REFER TO THE SAME SUBSECTIONS AND USE THE SAME VARIABLES, ETC. WHERE POSSIBLE. OUTCOME FROM THIS SHOULD BE azimuth $\theta(b,m)$, elevation $\phi(b,m)$, diffuseness $r_{diff}(b,m)$. IF SOMETHING ELSE, EDIT THE TEXT BELOW. DIFFUSENESS IS MERGED OVER TIME, EITHER HERE, OR LATER WHEN OTHER VARIABLES ARE MERGED.

Next, the coherence parameters are determined between the multi-channel playback audio signals. First, absolute values of the complex coherences in the covariance matrix are computed

$$\mathbf{C}_{\mathrm{s,abs}}(b,m) = |\mathbf{C}_{\mathrm{s}}(b,m)|$$

The energies of the loudspeaker channels can be obtained by

$$E(b,m,i) = c_{s,abs}(b,m,i,i)$$

Next, the channel with the largest energy is selected

$$i_{loudest}(b,m) = \underset{\mathrm{i}}{\mathrm{maxindex}}(E(b,m,i))$$

where $\underset{\mathrm{i}}{\mathrm{maxindex}}()$ returns the index of the entry that has the largest value (i.e., the index of the channel that has the largest energy).

Normalized coherences between the channels i and j are computed by

$$c_{s,\mathrm{norm}}(b,m,i,j) = \frac{c_{s,\mathrm{abs}}(b,m,i,j)}{\sqrt{E(b,m,i)E(b,m,j)}}$$

The initial surround coherence is computed by taking the minimum value of the normalized coherences between the loudest channel $i_{loudest}(b,m)$ and all other channels and squaring that value

$$\gamma'(b,m) = (\underset{\mathrm{i}}{\min}(c_{s,\mathrm{norm}}(b,m,i,i_{loudest}(b,m))))^2$$

where $\underset{\mathrm{i}}{\min}()$ return the minimum value. The resulting surround coherences $\gamma'(b,m)$ are limited to values between 0 and 1 to account for numerical inaccuracies.

Spread coherence computation is performed differently depending on the elevation angle. If elevation $\phi(b,m) \geq 17.5$ degrees, the spread coherence is set to zero $\zeta(b,m) = 0$. Otherwise, i.e., when elevation $\phi(b,m) < 17.5$ the spread coherence is determined as follows.

First, the loudspeaker channel having the azimuth $\theta_{LS}(i)$ closest to the azimuth $\theta(b,m)$ is determined (denoted $i_{closest}(b,m)$).The determination is performed among the loudspeaker channels being on the horizontal plane, i.e., $\phi_{LS}(i) = 0$. In addition, the loudspeaker closest on the left side of $i_{closest}(b,m)$ is determined (denoted $i_{left}(b,m)$), and the loudspeaker closest on the right side of $i_{closest}(b,m)$ is determined (denoted $i_{right}(b,m)$). This determination is also performed among the loudspeaker channels being on the horizontal plane, i.e., $\phi_{LS}(i) = 0$.

Then, stereo coherence is determined by

$$c_{st}(b,m) = c_{s,\mathrm{norm}}\left(b,m,i_{left}(b,m),i_{right}(b,m)\right)$$

and a stereo energy relation parameter $\xi_{lr/lrc}(b,m)$ is determined by

$$\xi_{lr/lrc}(b,m) = \frac{E\left(b,m,i_{left}(b,m)\right) + E\left(b,m,i_{right}(b,m)\right)}{E\left(b,m,i_{left}(b,m)\right) + E\left(b,m,i_{right}(b,m)\right) + E\left(b,m,i_{closest}(b,m)\right)}$$

Using them, a stereoness parameter is determined by

$$\mu(b,m) = c_{st}(b,m)\xi_{lr/lrc}(b,m)$$

Then, front coherence is determined by

$$c_{clr}(b,m) = \min(c_{s,\mathrm{norm}}\left(b,m,i_{closest}(b,m),i_{left}(b,m)\right), c_{s,\mathrm{norm}}\left(b,m,i_{closest}(b,m),i_{right}(b,m)\right))$$

and a front energy relation parameter $\xi_{clr}(b,m)$ is determined by

$$\xi_{clr}(b,m) = \min\left(\frac{E\left(b,m,i_{left}(b,m)\right)}{E\left(b,m,i_{closest}(b,m)\right)}, \frac{E\left(b,m,i_{closest}(b,m)\right)}{E\left(b,m,i_{left}(b,m)\right)}, \frac{E\left(b,m,i_{right}(b,m)\right)}{E\left(b,m,i_{closest}(b,m)\right)}, \frac{E\left(b,m,i_{closest}(b,m)\right)}{E\left(b,m,i_{right}(b,m)\right)}\right)$$

Using them, a coherent panning parameter is determined by

$$\kappa(b,m) = c_{clr}(b,m)\xi_{clr}(b,m)$$

Using these variables, the spread coherence is determined by

$$\zeta(b,m) = \begin{cases} \max(0.5, \mu(b,m) - \kappa(b,m) + 0.5), & if \max\big(\mu(b,m), \kappa(b,m)\big) > 0.5 \;\&\; \kappa(b,m) > \mu(b,m) \\ \max\big(\mu(b,m), \kappa(b,m)\big), & else \end{cases}$$

The resulting spread coherences $\zeta(b,m)$ are limited to values between 0 and 1 to account for numerical inaccuracies.

Then, stereo energy ratio tuning parameter is determined by

$$r_{st}(b,m) = c_{st}(b,m)\frac{E\left(b,m,i_{left}(b,m)\right) + E\left(b,m,i_{right}(b,m)\right)}{E(b,m)} - \gamma'(b,m)$$

and coherent panning ratio tuning parameter is determined by

$$r_{clr}(b,m) = c_{clr}(b,m)\frac{E\big(b,m,i_{closest}(b,m)\big) + E\left(b,m,i_{left}(b,m)\right) + E\left(b,m,i_{right}(b,m)\right)}{E(b,m)} - \gamma'(b,m)$$

Using them, an initial coherence-based energy ratio tuning parameter is determined by

$$r'_{coh}(b,m) = \max\big(r_{st}(b,m), r_{clr}(b,m)\big)$$

The resulting $r'_{coh}(b,m)$ values are limited to values between 0 and 1 to account for numerical inaccuracies.

The surround coherence values $\gamma(b,m)$ computed for different subframes are combined so that all the subframes within a frame have the same value. This is performed determining the value for the first subframe by

$$\gamma(b,0) = \frac{\sum_{m=0}^{3} E(b,m)\gamma'(b,m)}{\sum_{m=0}^{3} E(b,m)}$$

and setting the same value to all subframes m of the frame, yielding $\gamma(b,m)$.

The coherence-based energy ratio tuning parameter values $r'_{coh}(b,m)$ computed for different subframes are combined so that all the subframes within a frame have the same value. This is performed determining the value for the first subframe by

$$r_{coh}(b,0) = \frac{\sum_{m=0}^{3} E(b,m)r'_{coh}(b,m)}{\sum_{m=0}^{3} E(b,m)}$$

and setting the same value to all subframes m of the frame, yielding $r_{coh}(b,m)$.

The direct-to-total energy ratios are determined by

$$r_{dir}(b,m) = \max\left(1 - r_{diff}(b,n), r_{coh}(b,m)\right)$$

5.7.3.4 Even loudspeaker setup determination

The even setup is determined separately for the loudspeaker channels on the horizontal plane, i.e., $\phi_{LS}(i) = 0$ and the loudspeakers not on the horizontal plane, i.e., $\phi_{LS}(i) \neq 0$. The same method is applied for both; hence it is explained below only once. The number of channels fed to the method below is N_{even}.

The spacing between the loudspeakers is first computed by

$$\alpha_{spacing} = \frac{360}{N_{even}}$$

Then, the direction of the first loudspeaker is selected by

$$\alpha_{start} = -\alpha_{spacing}\frac{N_{even}}{2}$$

when N_{even} is odd, and by

$$\alpha_{start} = -\alpha_{spacing}\left(\frac{N_{even}}{2} - 0.5\right)$$

when N_{even} is even.

The even directions as ordered from the smallest azimuth to largest are then determined by

$$\theta_{order}(i') = i'\alpha_{spacing} + \alpha_{start}$$

where i' is the loudspeaker channel index from 0 to $N_{even} - 1$. These angles are ordered to the same order as the input loudspeaker channels were, yielding the even loudspeaker angles $\theta_{LSeven}(i)$. As said, this procedure is repeated separately for the horizontal and the non-horizontal loudspeakers (in case there are non-horizontal loudspeakers).

5.7.3.5 McMASA transport audio signal generation

McMASA transport audio signals are created by downmixing the multi-channel signals $s''(n,i)$. There are N_{ana} input channels to be downmixed. $N_{ana} = N - 1$ for the normal mode, and $N_{ana} = N - 2$ for the separate-channel mode.

Multi-channel energy is computed by

$$E_{mc} = \sum_{i=0}^{N_{ana}-1} \sum_{n=0}^{L_{frame}-1} s''(n,i)^2$$

where L_{frame} is the length of the frame in samples.

Then, prototype downmix signals are created. In case of one transport audio signal ($N_{trans} = 1$), the input channels are summed

$$s_{proto}(n,i') = \sum_{i=0}^{N_{ana}-1} s''(n,i)$$

In case of two transport audio signals ($N_{trans} = 2$), the input channels on the left (i.e., $\theta_{LS}(i) > 0$) are summed to the first prototype downmix signal $s_{proto}(n,0)$, and the input channels on the right (i.e., $\theta_{LS}(i) < 0$) are summed to the second prototype downmix signal $s_{proto}(n,1)$. In the normal mode, the centre channel (i.e., $\theta_{LS}(i) = 0$) is summed to both prototype downmix signals after being multiplied by $1/\sqrt{2}$.

The energy of the prototype downmix signals is computed by

$$E_{proto} = \sum_{i'=0}^{N_{trans}-1} \sum_{n=0}^{L_{frame}-1} s''(n,i')^2$$

The energies are smoothed over time by

$$E_{mc,sm} = 0.1E_{mc} + 0.9E_{mc,sm}^{[-1]}$$

$$E_{proto,sm} = 0.1E_{proto} + 0.9E_{proto,sm}^{[-1]}$$

where $E_{mc,sm}^{[-1]}$ and $E_{proto,sm}^{[-1]}$ are the smoothed values computed for the previous frame.

Equalization gain is next determined for keeping the time-smoothed overall energy the same for the transport audio signals as the input multi-channel audio signals. It is determined by

$$g_{EQ} = \frac{E_{mc,sm}}{E_{proto,sm}}$$

Then, an interpolator is determined

$$g_{interp}(n) = \frac{n}{L_{frame}}$$

Using these gains, the transport audio signals are created by equalizing the prototype downmix signals

$$s_{trans}(n,i) = (g_{interp}(n)g_{EQ} + (1 - (g_{interp}(n))g_{EQ}^{[-1]})\, s_{proto}(n,i)$$

where $g_{EQ}^{[-1]}$ is the equalization gain determined for the previous frame.

5.7.3.6 McMASA metadata encoding

5.7.3.6.1 Spatial metadata encoding

5.7.3.6.2 LFE-to-total energy ratio encoding

For the operating bitrates of 13.2 and 16.4 kbps, the LFE-to-total energy ratios $\Xi(m)$ for each subframe of the current frame are quantized as a single ratio for the whole frame. For higher bitrates, an additional ratio is quantized for each subframe using residual VQ when the LFE frames have a higher energy. Inactive LFE frames for all bitrates are indicated with one bit (0) in the bitstream, and no further LFE-to-total ratio energy coding is performed. The bit allocation, dependent on the McMASA bitrate, is provided in Table 4.4-1.

Table 5.7-3: Bit allocation for LFE-to-total energy ratio

Bitrate (kbps)	Bits used (inactive frames)	Bit allocation (active frames)	Bits used (active frames)
13.2	1	1 (activity / energy ratio modulation)	1
16.4	1	1 (activity) + 3 (scalar quantization)	4
≥ 24.4	1	1 + 3 (scalar quant) + 0…4 (subframe VQ for high energy ratio frames)	4…8

The activity of LFE is detected from the subframe $\Xi(m)$ LFE-to-total energy ratios. LFE-to-total energy ratio $\Xi(m)$ is only sent when any of the subframes in the frame have a $\Xi(m)$ which is above a threshold of 0.005. Otherwise, if the maximum $\Xi(m)$ of all the subframes in the frame is less than the threshold, one bit is sent with a zero (0) index for all bitrates.

In the case when a $\Xi(m)$ is above the threshold for any of the subframes in the current frame, the encoding process comprises determining an averaged $\log_2$ LFE-to-total energy ratio for a frame by

$$\Xi_{log_2} = \sum_{m=0}^{3} 0.25 * \Xi_{log_2}(m)$$

where the $\log_2$ LFE-to-total energy ratio for each subframe $\Xi_{log_2}(m)$ is clamped between [-9,1] by

$$\Xi_{log_2}(m) = \begin{cases} -9, if\ \log_2(\Xi(m)) < -9 \\ \log_2(\Xi(m))) \\ 1, if\ \log_2(\Xi(m)) > 1 \end{cases}$$

At the lowest bitrate of 13.2 kbps only one bit is allocated for the LFE-to-total energy ratios for the frame. In this case the LFE-to-total energy ratio bit is set to (1) if Ξ_{log_2} is higher than both a threshold value (MCMASA_LFE_1BIT_THRES=0.03) and a value depending on the previous frame's quantized value $\hat{\Xi}_{prev}$. The comparisons are made in the linear domain by converting the averaged $\log_2$ LFE-to-total energy ratio according to $\Xi = 2^{\Xi_{log_2}}$. Therefore, the condition for setting the LFE-to-total energy ratio bit to 1, is expressed as when $\Xi > 0.03$ and $\Xi > [0.5 * (0.09 + \hat{\Xi}_{prev}) + 0.5 * (0.67 + \hat{\Xi}_{prev}]$ are true the LFE-to-total energy ratio bit is set to (1). If either of conditions are not met LFE-to-total energy ratio bit is set to (0). In practice, one bit is thus used to bump up or dampen the LFE-to-total energy ratio on a frame-by-frame basis.

At the second lowest bitrate of 16.4 kbps 1 (activity) + 3 = 4 bits are used to encode the LFE-to-total energy ratios for the frame when the frame is classified as an active frame, with the first bit being used to signal that the frame is an active frame. When the first bit is 1 (active) next three bits are used to scalar quantize $\hat{\Xi}_{log_2}$ for the frame using a linear scalar quantizer using a codebook of $[-6.5, \ldots, 0.5]$ with step of 1.0. This codebook is given in Table 5.7-4 as the second column.

Table 5.7-4: Scalar quantizer codebook used by LFE-to-total energy ratio for bitrates ≥ 16.4 bpss and bit allocation for residual subframe energy ratio VQ encoding for bitrates ≥ 24.4 kbps.

Scalar codebook indice	Scalar $\hat{\Xi}_{log_2}$ quantized value	Bits used for residual subframe VQ for bitrates ≥ 24.4 kbps
0	-6.5	0
1	-5.5	0
2	-4.5	1
3	-3.5	2
4	-2.5	3
5	-1.5	4
6	-0.5	4
7	0.5	4

For bitrates 24.4 kbps and above, an additional subframe residual $\tilde{\Xi}_{log_2}(m)$ is calculated for each subframe m. This can be obtained by

$$\tilde{\Xi}_{log_2}(m) = \hat{\Xi}_{log_2} - \Xi_{log_2}(m)$$

The subframe residuals $\tilde{\Xi}_{log_2}(m)$ for the subframes of the frame are then encoded as a 4-dimensional vector using a vector quantizer with a 16-entry trained codebook. The entries of the trained codebook are given by the array `McMASA_LFEGain_vectors[4x16]`.

The residual is adaptively encoded depending on the scalar codebook index used to quantize the $\hat{\Xi}_{log_2}$. For low energy ratio frames (i.e., when $\hat{\Xi}_{log_2} \leq -5.5$), which are associated with the scalar codebook indices 0 and 1, only the $\hat{\Xi}_{log_2}$ value is quantized with the scalar quantizer and transmitted. For the next higher energy ratios, corresponding to scalar codebook indices (2-4), between 1 to 3 bits are used to quantise and encode the residual subframes. Finally, for the highest energy ratios, corresponding to scalar codebook indices (5-7), the maximum of 4 bits are used to quantise and encode the subframe residuals, which corresponds to the case when $\hat{\Xi}_{log_2} \geq -1.5$. Table 5.7-4 (third column) provides LFE-to-total energy ratio subframe VQ bit allocation.

5.7.3.7 McMASA transport audio signal encoding

TODO. TRANSPORT AUDIO SIGNAL ENCODING AND THE SEPARATE CENTRE CHANNEL ENCODING.

5.7.3.8 McMASA bitstream structure

5.7.4 Parametric MC coding mode

5.7.4.1 ParamMC coding mode overview

The Parametric Multi-channel coding mode encoder generates a transport audio signal with at least two channels from the original multi-channel input signal. It estimates channel level and inter channel correlation parameters from the input audio signal. The transport audio signal channels are coded using either MDCT Stereo (for 2 transport audio channels) or MCT (for 3 transport audio channels) and are written to the bitstream. The estimated parameters are encoded as side information. The channel levels are encoded as inter-channel level differences ICLD for all channels and as the inter-channel correlations ICCs as a selected subset of all possible combinations of the original channels.

The ICLDs and ICCs are estimated and coding for a number of parameter bands in the frequency domain. To lower the side information load the parameter bands are divided into two sets that are sent alternately for each frame resulting in one set containing the channel level and correlation information for two consecutive frames when the signal has non-transient behaviour. In case of a transient signal parameter bands are combined and the reduces set of parameters is sent for the full band width in a frame with a transient, in this case the parameters are containing the channel level and correlation information specific to this frame.

5.7.4.2 ParamMC transport audio signal generation

Depending on the bit rate and the input channel configuration the number of transport audio signals n_t and a downmix matrix $\mathbf{M}_{l,k}$ is chosen. (add tables for ts and dmx matrix selection) The transport audio signal is generated by a time-domain sample-wise down mix of the input channels using the static downmix matrix:

$$ts_k[n] = \sum_{n_i}^{l=0} is_l[n]\mathbf{M}_{l,k} \tag{5.7-7}$$

where ts_k is the kth transport audio signal channel, with $k = 0,1 \dots n_t$, is_l is the lth input audio channel, with $l = 0,1, \dots n_i$, and n is the sample index, with n=0,1,…,framelength.

5.7.4.3 ParamMC time domain transient detection

On the generated time domain transport audio signals of the current frame, a transient analysis is done to determine the occurrence of a transient within the frame.

The Time-domain transient detection (Clause 5.2.2.1.6) is run for each transport channel. First for the two last subblocks of the previous frame it is checked if the energy of the segment A transient is detected if the energy of a segment $E_{TD}(i)$, i=-2..7 exceeds the accumulated energy by a constant factor of 8.5 and the transient position index is set to max(i,*0)* and the transient indicator for this channel is set to 1. The overall transient flag is set to one if at least one transport channel has the channel transient indicator set to 1. The overall transient position index is set to the smallest one of all channels having the transient indicator set to 1.

5.7.4.4 ParamMC covariance estimation

The input signal is transformed into the frequency domain using the MDFT (ref to MDFT) using the parameters from table xxx, resulting in the input audio signal in frequency domain Y, the frame now being subdivided into an integer number of consecutive slots with one MDFT spectrum per slot and channel

A reference down mix representation is generated by applying the downmix matrix from 5.6.4.2 also on the input audio MDFT samples.

$$\mathrm{X}_{\mathrm{l},s}[\mathrm{b}] = \sum_{n_i}^{l=0} Y_{k,s}[b]M_{l_k} \tag{5.7-8}$$

For both downmix and input MDFT representations covariance matrix estimates are calculated for each bin and timeslot in a frame

$$\mathbf{C}_{y,s,b} = Y_s[b]Y_s[b]^* \tag{5.7-9}$$

$$\mathbf{C}_{x,s,b} = X_s[b]Y_s[b]^* \tag{5.7-10}$$

The covariance matrix estimates are accumulated per parameter band b_p for either all MDFT timeslots when no transient was detected or for the MDFT timeslots after the transient in case a transient was detected, and all MDFT bins belonging to a specific parameter band b_p to create the sum of the covariances as a basis for the transmitted channel level and correlation information.

$$\mathbf{C}_{y,b_p} = \sum_{b_{end}}^{b=b_{start}} \sum_{ts_{end}}^{s=ts_{start}} \mathbf{C}_{y,s,b} \tag{5.7-11}$$

$$\mathbf{C}_{x,b_p} = \sum_{b_{end}}^{b=b_{start}} \sum_{ts_{end}}^{s=ts_{start}} \mathbf{C}_{x,s,b} \tag{5.7-12}$$

5.7.4.5 ParamMC default settings for the LFE channel

The parameters for the LFE channel are only sent for the first parameter band. To ensure correct estimates in the following steps, the covariance estimate entries related to the LFE for all other bands are set to zero.

$$c_{y,b_{p_{LFE\_INDEX,*}}} = \mathbf{0} \tag{5.7-13}$$

$$c_{y,b_{p_{*,\mathrm{LFE\_INDEX}}}} = \mathbf{0} \tag{5.7-14}$$

where b_p is the parameter band, with b_p=1…$n_B - 1$.

5.7.4.6 ParamMC parameter based transient detection

To make sure that in frames where no time domain transient is detected the ICLDs in non-transmitted parameter bands change too much, the ICLDs for those bands are calculated according to 5.6.4.6.2, but with no limiting strategy applied and if the number of ICLDs where the difference to the ICLDs of the exceeding a parameter band specific threshold $ICLD_{thr,diff}$ is greater than 3 the transient flag is set to one.

5.7.4.7 ParamMC band combining in case of transients

In case of a set transient flag, the estimates co-variances are of two bands each are combined together to form a new set of estimates for a new set of parameter bands with a reduced number of parameter and with less frequency resolution and the step size s_p is set to 2.

$$C_{y,b_p} = \begin{cases} C_{y,b_p} + C_{y,b_p+1} if b_p + 1 < n_B \\ C_{y,b_p} else \end{cases} \tag{5.7-15}$$

$$C_{x,b_p} = \begin{cases} C_{x,b_p} + C_{x,b_p+1} if b_p + 1 < n_B \\ C_{x,b_p} else \end{cases} \tag{5.7-16}$$

where b_p is the parameter band, with b_p=0,s_p, $2s_p$,…$n_B - 1$.

If the transient flag is 0, the step size s_p, is set to 1.

5.7.4.8 ParamMC complex valued to real valued covariance conversion

A parameter band index b_{p,max_abs_cov} is determined, all entries in the (complex valued) covariance estimates for parameter bands below this index are converted to real values using the absolute value of the estimates, all entries for the parameter bands covariance values being equal or above this index are converted to real values using the real part of the complex estimates:

$$C_{y,b_p} = \begin{cases} |C_{y,b_p}| if b_p < b_{p,max\_abs\_cov} \\ C_{y,b_p} else \end{cases} \tag{5.7-17}$$

$$C_{x,b_p} = \begin{cases} C_{x,b_p} + C_{x,b_p+1} if b_p + 1 < n_B \\ C_{x,b_p} else \end{cases} \tag{5.7-18}$$

where b_p is the parameter band, with b_p=0,s_p, $2s_p$,…$n_B - 1$.

5.7.4.9 ParamMC LFE activity detection

If the energy of the LFE channel, i.e. the entry in the first parameter band in the main diagonal element of C_y corresponding to the LFE channel is lower than the threshold $PARAM\_MC\_LFE\_ON\_THRESH = 8000$, then the LFE activity flag f_{LFE} is set to zero, otherwise to one.

5.7.4.10 ParamMC parameter quantization

5.7.4.10.1 Parameter quantizers

The parameter quantizers for ParamMC are realised with a non-linear mid-step quantization characteristic, i.e. the quantized value and its index are those of the quantizer level and its index closest to the unquantized value. Quantizers have a length l_q.

5.7.4.10.2 Interchannel Level differences (ICLDs)

The ICLD values are represented as normalized and logarithmic values per parameter band and input channel using the energy $\mathrm{P_i}$ of the original channel in this parameter band. The normalization is based on the energy $\mathrm{P_{dmx,i}}$ of the reference (transport) channels belonging to the to be quantized, the reference energy being a linear combination of the values of the covariance information of the downmix signal.

$$\mathrm{\chi_i} = 10\log_{10}\left(\frac{\mathrm{P_i}}{\mathrm{P_{dmx,i}}}\right) \tag{5.7-19}$$

$$P_i = C_{y,b_p,i,i} \tag{5.7-20}$$

$$P_{dmx,i} = f_{ICLD}[i]\sum_{j<n_r}^{j=0} C_{x,k,k} \tag{5.7-21}$$

where: $f_{ICLD}[i]$ is a factor dependent on the bit rate and the input multichannel format (see Table xxx)

$n_r\ is\ the\ number\ of\ downmix\ channels\ associated\ with\ the\ ICLD\ of\ the\ inputchannel\ i$

K is the index of a downmix channel determined by a map: $k = m_{ild}[j]$

Two limiting strategies are applied to the original channel energies to avoid artefacts. The first one checks if the sum of the energies of the downmix channels is smaller than the sum of the energies of the original channels.

$$E_{tot} = \sum_{i<n_i}^{i=0} C_{y,b_p,i,i},\ if i \neq LFE\_INDEX \vee f_{LFE} = 1 \tag{5.7-22}$$

$$E_{dmx} = \sum_{i<n_{dmx}}^{i=0} C_{x,b_p,i,i} \tag{5.7-23}$$

$$fac_{ener} = 10log_{10}\left(\frac{E_{tot}+\epsilon}{E_{dmx}+\epsilon}\right) \tag{5.7-24}$$

If fac_{ener} is larger than the intraframe limiter threshold $PARAM\_MC\_ENER\_LIMIT\_INTRAFRAME = 1.5$, then the first limiter is applied:

$$P_i = 10^{\frac{0.3log(fac_{ener}-PARAM\_MC\_ENER\_LIMIT\_INTRAFRAME+1)-(fac_{ener}-PARAM\_MC\_ENER\_LIMIT\_INTRAFRAME)}{10}} P_i$$

The second limiting step checks that ILDs do not change too much

$$\Delta fac_{ener} = fac_{ener} - fac_{ener,prev} \tag{5.7-25}$$

If the frame is not marked as a transient frame and Δfac_{ener} is larger than $PARAM\_MC\_ENER\_LIMIT\_INTERFRAME = 2$ and Δfac_{ener} is smaller than $PARAM\_MC\_ENER\_LIMIT\_MAX\_DELTA\_FAC = 15$ the second limiter is applied and fac_{ener} is corrected:

$$fac_{limit} = 10^{\frac{0.3log(\Delta fac_{ener}-PARAM\_MC\_ENER\_LIMIT\_INTERFRAME+1)-(\Delta fac_{ener}-PARAM\_MC\_ENER\_LIMIT\_INTERFRAME)}{10}}$$

$$P_i = fac_{limit}P_i \tag{5.7-26}$$

$$fac_{ener} = fac_{ener} + 10log_{10}(fac_{limit}) \tag{5.7-27}$$

fac_{ener} is saved as $fac_{ener,prev}$, if the current frame is a transient frame, it is also saved as the $fac_{ener,prev}$ for the parameter band $b_p + 1$ as long as this is smaller than n_B.

The unquantized ILDs $\mathrm{\chi_i}$ are quantized using a quantizer as described in xxx with a length $l_{Q,ICLD} = 16$ (see Table xxx), and the quantizer indices stored parameter band wise for encoding. The unquantized values are also stored per parameter band for the parameter based transient detection in the next frame.

5.7.4.10.3 Inter-channel coherences (ICCs)

The normalized inter channel coherences ICCs for a specific parameter band are generated from the full covariance matrix of the input channels of the parameter band using the relation

$$\xi_{i,j} = \frac{C_{y_{i,j}}}{\sqrt{C_{y_{i,i}}C_{y_{j,j}}}} \tag{5.7-28}$$

Only for a sub-set of all possible channel combinations for the channels i, j the ICCs are estimated, the sub-set depending on the bit rate, the input format and if it is the first parameter band and $f_{LFE} = 1$. (add Tables!)

The unquantized ICCs are quantized according to XXX using the quantizer from table xxx with $l_{Q,ICC} = 8$ and the quantization indices stored in the order determined by the order of the subset in the chosen table for each parameter band.

5.7.4.11 ParamMC parameter encoding

5.7.4.11.1 Common ParamMC parameter encoding

* f_{LFE} with 1 bit

* the encoded band width with 2 bits

* the parameter frame indicator with 1 bit

* the transient flag with 1 bit, signalling the occurrence of a transient in the frame

** if the transient flag is 1, the transient position with 3 bits, signalling in which slot the transient has occurred

5.7.4.11.2 ICC and ICLD Parameter quantization indices encoding

5.7.4.11.2.1 General parameter indices encoding

The quantized parameter indices are rearranged for all parameter bands belonging to the parameters bands to be encoded determined either by the coding band mapping and the current parameter frame indicator or all bands if is a transient frame. The order is first all parameters not belonging to the coding of the LFE and the parameters belonging to the coding of LFE at the very end of the sequence if $f_{LFE} = 1$ and either the transient flag is one or the parameter band with index 0 is part of the set of parameter bands to be coded. Both a sequence of absolute indices and delta indices are generated. Both sequences are encoded using the range coder (reference missing) with a codebook depending on the bit rate and input format with distinct code books for absolute and delta indices. The bit demand of the two encoding runs is compared and the minimum bit demand is chosen and a flag f_r is set to 0 if the range coding with absolute indices yields less bits and to 1 if range coding with delta indices yields less bits. To ensure a deterministic upper bound of the bit demand for coding parameters the bit demand per index $n_{uniform}$ determined by the quantizer size is multiplied by the number of indices to code in the sequence. If this uniform coding takes less bits than the better of range coding absolute and delta indices a zero bit is written to the bit stream and each index is written to the bit stream with the number of bits needed for uniform coding. Otherwise a one is written to the bit stream, followed by writing the flag f_r with one bit followed by the encoded range coding values with the fewer number of bits.

5.7.4.11.2.2 ICLD parameter indices encoding

The quantization indices of the quantized ICLD values are encoded and written to the bit stream using the code books specified in table xxx and a uniform bit demand $n_{uniform} = 4$

5.7.4.11.2.3 ICC parameter indices encoding

The quantization indices of the quantized ICC values are encoded and written to the bit stream using the code books specified in table xxx and a uniform bit demand $n_{uniform} = 3$

5.7.4.12 ParamMC transport audio signal encoding

The time domain downmix channels are either encoded with MDCT stereo in case of bit rates using 2 transport channels or with MCT in case of bit rates with 3 transport channels. Note that since we ran the time domain transient detector already in the ParamMC processing it is in the case of ParamMC coding not necessary to run it again in the pre-processing of the core coding, but the results can be directly re-used.

5.7.5 Parametric upmix encoding mode

5.7.5.1 General

For 7.1.4 operation at 160 kbps, a parametric upmix coding approach is used offering highly efficient coding for this particular configuration.

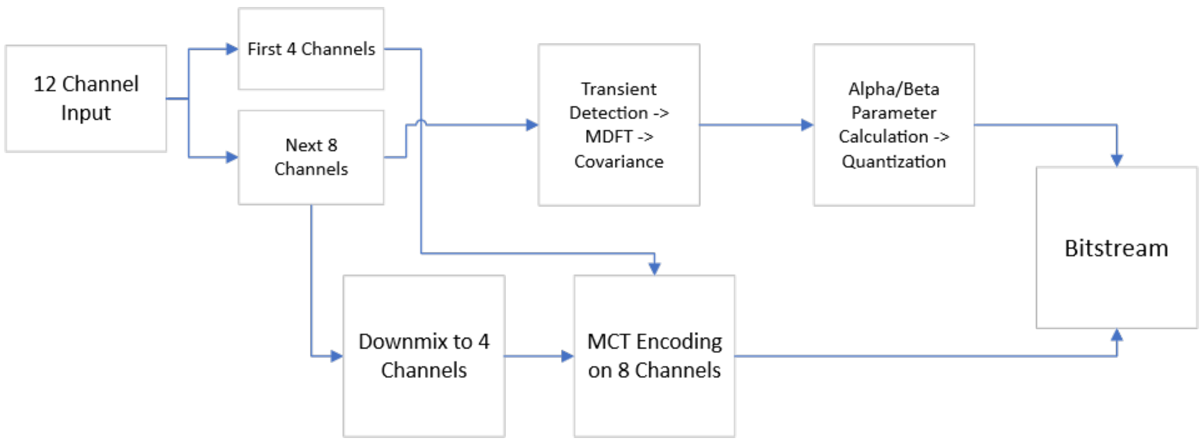

12 Channel Input
First 4 Channels
Next 8 Channels
Transient Detection -> MDFT -> Covariance
Alpha/Beta Parameter Calculation -> Quantization
Bitstream
Downmix to 4 Channels
MCT Encoding on 8 Channels

Figure 7

Figure 5.7.5-8: Schematic of parametric upmix encoder

5.7.5.2 Sectioning and downmixing

Input 12 channel 7.1.4 signal is subdivided into two sections P1 and P2: Left/Right/Center/LFE (P1) and the other 8 channels (P2). The P2 section is processed by the Metadata Parameter Calculation and Quantization block to generate ParamUpmix parameters. At the same time P2 is downmixed from 8 channels to 4 by combining the following channels using an index of 0 for the first channel of the input 12 channels: [4+6], [5+7], [8+10], [9+11].

Denoting the channels of the P2 section as $P2_k$, $k = 0 \dots 7$, the downmix operation calculates the Mid transformation of the 8 channels as follows:

$$M2_k = \frac{1}{2}(P2_k + P2_{k+2}) \quad k = 0,1 \ , \tag{5.7-29}$$

$$M2_k = \frac{1}{2}(P2_k + P2_{k+2}) \quad k = 2,3 \ . \tag{5.7-30}$$

The 4 Mid channels $M2_k$, k=0…3 are associated with the P1 4 channels to make 8 channels which go on to be coded by the MCT (Reference) and respectively the LFE codec [Ref: LFE encoder].

5.7.5.3 Metadata parameter calculation

The Metadata Parameter Calculation block takes the P2 channels above and transient detection is performed on them (Reference). The transient detector output data (TD) is passed to the Covariance Calculator (Reference). The P2 channels are also transformed to frequency domain by the MDFT block (Reference).

The 8 MDFT channels are arranged into 4 pairs $q_k, k = 0 \dots 3$, of complex data, analogous to the pairs in downmixing above.

$$\begin{aligned} q_k &= \{P2_k, P2_{k+2}\}, \ k = 0,1 \\ q_k &= \{P2_{k+2}, P2_{k+4}\}, \ k = 2,3 \end{aligned} \tag{5.7-31}$$

These pairs are passed into the Covariance Calculator (Reference) along with the transient detector data (TD) above.

The output will for each frequency band (12 bands) be a matrix of covariance coefficients, i.e., for each pair of channels and for each frequency band there will be a 2x2 matrix of which the real part will be retained:

$$M_{m,k} \quad = \Re\mathrm{e}\begin{Bmatrix} R_{xx} & R_{yx} \\ R_{xy} & R_{yy} \end{Bmatrix}, \ k = 0 \dots 3, m = 0 \dots 11 \ , \qquad (5.7\text{-}32)$$

where x and y respectively relate to the first and second element of the pairs q_k.

The following pseudocode will demonstrate the calculation of alpha and beta unquantized parameters. Herein alpha parameters are normalized cross-correlation parameters whereas beta parameters control the decorrelator in the decoding process.

```
e = 0.000000000000001
for ( b = 0; b < number_of_pairs; b++ )
{
    for ( bnd = 0; bnd < number_of_frequency_bands; bnd++ )
    {
        C = Rxy[b][bnd] / ( Ryy[b][bnd] + e )
        alpha[b][bnd] = 2.0f * C - 1.0
        Rxx_est = C * C * Ryy[b][bnd]
        dRxx = Rxx[b][bnd] - Rxx_est
        dRxx = max( dRxx, 0.0f )
        beta[b][bnd] = 2.0 * sqrtf( dRxx / ( Ryy[b][bnd] + e ) )
    }
}
```

5.7.5.4 Quantization

Alpha parameters are quantized to the nearest value given by the following table:

Table 5.7-5: Quantizer for Alpha parameter

Index	Value	Index	Value
0	-2.0	17	0.190625
1	-1.809375	18	0.3625
2	-1.6375	19	0.515625
3	-1.484375	20	0.65
4	-1.35	21	0.765625
5	-1.234375	22	0.8625
6	-1.1375	23	0.940625
7	-1.059375	24	1.0
8	-1.0	25	1.059375
9	-0.940625	26	1.1375
10	-0.8625	27	1.234375
11	-0.765625	28	1.35
12	-0.65	29	1.484375
13	-0.515625	30	1.6375
14	-0.3625	31	1.809375
15	-0.190625	32	2.0
16	0.0		

Beta parameters are quantized using a two stage process where alpha parameters are first quantized using the above method, and these quantized values are used to index into a set of tables which beta parameters will be quantized by, to the nearest value in the table.

The below table gives the beta quantization table index to give the beta table to be used.

Table 5.7-6: Quantizer selector Beta parameter quantizer

Quantized Alpha Index	Beta Table Index	Quantized Alpha Index	Beta Table Index
0	0	17	1
1	1	18	2
2	2	19	3
3	3	20	4
4	4	21	5
5	5	22	6
6	6	23	7
7	7	24	8
8	8	25	7
9	7	26	6
10	6	27	5
11	5	28	4
12	4	29	3
13	3	30	2
14	2	31	1
15	1	32	0
16	0		

Table 5.7-7: Quantizers for Beta parameter

Beta Table Index			Beta Table Index		
0	**Quantized Beta Index**	**Value**	**5**	**Quantized Beta Index**	**Value**
	0	0.0		0	0.0
	1	0.2375		1	0.101123
	2	0.55		2	0.2341797
	3	0.9375		3	0.3991699
	4	1.4		4	0.5960938
	5	1.9375		5	0.8249512
	6	2.55		6	1.085742
	7	3.2375		7	1.378467
	8	4.0		8	1.703125
1	**Quantized Beta Index**	**Value**	**6**	**Quantized Beta Index**	**Value**
	0	0.0		0	0.0
	1	0.2035449		1	0.08386719
	2	0.4713672		2	0.1942188
	3	0.8034668		3	0.3310547
	4	1.199844		4	0.494375
	5	1.660498		5	0.6841797
	6	2.18543		6	0.9004688
	7	2.774639		7	1.143242
	8	3.428125		8	1.4125
2	**Quantized Beta Index**	**Value**	**7**	**Quantized Beta Index**	**Value**
	0	0.0		0	0.0
	1	0.1729297		1	0.06995117
	2	0.4004688		2	0.1619922
	3	0.6826172		3	0.276123
	4	1.019375		4	0.4123438
	5	1.410742		5	0.5706543
	6	1.856719		6	0.7510547
	7	2.357305		7	0.9535449
	8	2.9125		8	1.178125
3	**Quantized Beta Index**	**Value**	**8**	**Quantized Beta Index**	**Value**
	0	0.0		0	0.0
	1	0.1456543		1	0.059375
	2	0.3373047		2	0.1375
	3	0.5749512		3	0.234375
	4	0.8585938		4	0.35
	5	1.188232		5	0.484375
	6	1.563867		6	0.6375
	7	1.985498		7	0.809375
	8	2.453125		8	1.0
4	**Quantized Beta Index**	**Value**			
	0	0.0			
	1	0.1217188			
	2	0.281875			
	3	0.4804688			
	4	0.7175			
	5	0.9929688			
	6	1.306875			
	7	1.659219			
	8	2.05			

5.7.5.5 Entropy Coding

Quantized alpha and beta indices are Huffman coded differentially in time or frequency. If no other frame has yet been encoded or the previous frame was not using ParamUpmix then frequency differential coding is used since there is no previous set of quantized indices to use for a time differential. In other cases, the length of the codes required across all bands is determined for both differential in time and in frequency and the shorter overall differential type is used. The determined codes are then written to the bitstream.

5.7.6 Discrete Multi-channel coding mode

5.7.7 MC bitrate switching

5.8 Combined Object-based audio and SBA (OSBA) operation

5.8.1 OSBA format overview

The encoder supports combined input with 1 – 4 ISMs and an SBA signal of order 1 – 3. Depending on the bitrate, different methods are employed to combine these input signals. For bitrates less than 256 kbps, a simple pre-rendering of the objects into the SBA signal is performed. For bitrates at and higher than 256 kbps, the individual objects are transmitted and rendered on the decoder side.

5.8.2 Low-bitrate pre-rendering coding mode

In the low-bitrate OSBA mode, the encoder generates a description of a combined audio scene based on a first-order ambisonics signal.

The input interface receives via the IVAS encoder API calls, one signal describing an audio scene in ISM format (first format) and another signal describing an audio scene in ISM format (second format) at the same time.

The first scene description in the first format (ISM format) is then converted into first-order ambisonics as the common format. The SBA input signal (second format) is truncated to first order. The ISM input signal is converted to first-order ambisonics by panning the objects in the ambisonics domain to the positions described by the metadata:

$$\boldsymbol{x}_{FOA,O}(t) = \boldsymbol{Y}(\theta_O)x_O(t) = \left[x_O(t)Y_{00},\ x_O(t)Y_{1-1}, x_{O(t)}Y_{1,0}, x_O(t)Y_{11}\right]. \tag{5.8-20}$$

$\boldsymbol{x}_{FOA,O}(t)$ is the converted audio signal corresponding to the object with the index O. $\boldsymbol{Y}(\theta_O)$ is the vector of the real spherical harmonics evaluated at the DoA angle θ_O that represents the position of the object on the sphere. $x_O(t)$ is the object audio signal in time-domain.

From the two scenes described in the common format (first-order ambisonics) the description of the combined audio scene is acquired. Specifically, the two first-order ambisonics signals describing the two individual scenes are added up and multiplied by a factor of 0.5, thereby obtaining the description of the combined audio scene in the common format.

From this common format (first-order ambisonics), transport channels are generated in the SPAR downmixing (cf. clause xxx). This downmix is performed in the on each time-frequency tile.

On the low frequency bands, the covariance is estimated from the first-order ambisonics channels of the combined scene in the common format. On the high-frequency bands, a DirAC parameter analysis is performed, generating direction-of-arrival and diffuseness values. These parameters represent the combined scene. They are, in turn, used in the estimation of the covariance for the calculation of the downmix matrix (cf. clauses xxx and xxx) and encoded by the DirAC metadata encoder according to clause xxx).

The transport channels in the first set resulting from the downmix are then encoded using the MCT according to clause 6.1.4.3), which, in turn, is based on EVS as a core coder. The bitstream containing the encoded metadata and audio channels is then written to the encoder output interface (IVAS encoder API).

In summary, the processing of the first-order ambisonics signal (common format and common scene) is equivalent as a regular ambisonics input at the respective bitrate. The only difference to the bitstream of a regular coded SBA signal is that the number of ISMs in the input is signaled to the decoder such that the correct number of channels can be provided for external output.

5.8.3 High-bitrate discrete coding mode

In the high-quality high-bitrate OSBA mode, the objects are coded with the regular ISM processing as in clause xxx. The SBA scene is coded as in the regular SBA coding at the respective bitrate. In principle, the regular SBA and ISM encoders run alongside each other in an unmodified way. The configuration of the SBA encoder is the same as with regular SBA input at the same total IVAS bitrate.

The ISM metadata are coded in the bitstream together with the SBA metadata and the required bits are subtracted from the budget of the core coder. The downmix audio channels of the SBA encoder and the object audio channels are encoded jointly using the MCT (see clause xxx). Thereby a dynamic bit allocation between the audio downmix channels of the SBA encoder and the object audio channels is achieved. An object that is silent, therefore, uses only very few bits.

5.8.4 OSBA bitrate switching

When the bitrate is switched in OSBA mode, both encoders (SBA and ISM) are re-configured. The configuration is the same as is there were running as separate instances of IVAS. One special case for OSBA is the switching between bitrates lower or higher than 256 kbps. Then the encoder switches between the pre-rendering and the discrete coding mode.

When the high-bitrate mode is switched on, additional re-configurations are required as compared to bitrate switching for SBA in clause xxx. Specifically, the number of MCT channels is set according to the SBA configuration and the number of objects. The ISM mode flag is set to signal discrete object coding.

When the high-bitrate mode is switched off, the ISM mode flag is set to signal pre-rendering mode.

5.9 Combined Object-based audio and MASA (OMASA) operation

5.9.1 OMASA format overview

5.9.2 OMASA format configurations

5.9.3 OMASA pre-coding processing tools

5.9.3.1 OMASA spatial audio parameter analysis

First, the object audio signals $s(n,i)$ are converted to the time-frequency domain using the filter bank described in clause X.X.X. TODO: ADD REF. The outcome is the time-frequency domain object signals $S(k,m,i)$ (where k is the frequency bin index, m the subframe index corresponding also to the slot index, and i is the object index).

Then, the time-frequency domain object signals are converted to FOA signals (where the channels 0, 1, 2, and 3 correspond to spherical harmonics W, Y, Z, and X) by

$$S_{FOA}(k,m,0) = \sum_{i=0}^{N_{obj}-1} S(k,m,i)$$

$$S_{FOA}(k,m,1) = \sum_{i=0}^{N_{obj}-1} \sin\theta_{obj}(i)\cos\phi_{obj}(i)\,S(k,m,i)$$

$$S_{FOA}(k,m,2) = \sum_{i=0}^{N_{obj}-1} \sin\phi_{obj}(i)\, S(k,m,i)$$

$$S_{FOA}(k,m,3) = \sum_{i=0}^{N_{obj}-1} \cos\theta_{obj}(i)\cos\phi_{obj}(i)\, S(k,m,i)$$

where N_{obj} is the number of objects, $\theta_{obj}(i)$ the azimuth angle of the object i for this frame, and $\phi_{obj}(i)$ the elevation angle.

The energy of the object signals is determined by

$$E(b,m) = \sum_{k=k_1(b)}^{k_2(b)} \sum_{i=0}^{N_{obj}-1} |S(k,m,i)|^2$$

and k_1 and k_2 are the first and the last bin of the frequency band b. The frequency bands follow the MASA frequency bands TODO: ADD REF TO MASA FREQUENCY BANDS.

TODO: ADD DIRECTION AND DIFFUSENESS COMPUTATION ONCE CORRESPONDING DIRAC PARTS ARE READY. REFER TO THE SAME SUBSECTIONS AND USE THE SAME VARIABLES, ETC. WHERE POSSIBLE. OUTCOME FROM THIS SHOULD BE azimuth $\theta(b,m)$, elevation $\phi(b,m)$, diffuseness $r_{diff}(b,m)$. IF SOMETHING ELSE, EDIT THE TEXT BELOW. CHECK THAT DIFFUSENESS IS MERGED OVER TIME, EITHER HERE, OR LATER WHEN OTHER VARIABLES ARE MERGED.

The direct-to-total energy ratio is determined from the diffuse-to-total energy ratio by

$$r_{dir}(b,m) = 1 - r_{diff}(b,n)$$

The spread and surround coherences are set to zero

$$\zeta(b,m) = 0$$

$$\gamma(b,m) = 0$$

5.9.3.2 OMASA MASA metadata combining

The MASA spatial audio parameters determined from the second audio stream (i.e., from the objects) are merged with the spatial audio parameters of the first, the original, MASA audio stream. The merge is done for each parameter coding band b and parameter subframe m independently.

For the first, the original, MASA audio stream, obtain the signal energy parameter E_{MASA} describing the energy of the transport signal, and the direct-to-total energy parameter $r_{dir;MASA}$. The signal energy parameter can be determined from the transport audio signals or received as an input to the metadata merge. Based on the signal energy parameter E_{MASA} and direct-to-total energy parameter $r_{dir;MASA}$ of the original MASA stream, a weighting value w_{MASA} for the first stream is generated with

$$w_{MASA}(b,m) = E_{MASA}(b,m) r_{dir;MASA}(b,m)$$

Similarly for the second audio stream, obtain the signal energy parameter E_{ISM} and the direct-to-total energy ratio parameter $r_{dir;ISM}$. These are determined based on the MASA spatial metadata and transport signal for the second stream determined from the original ISM object signals and spatial metadata as described in clause x.x.x.x (OMASA spatial audio parameter analysis).

Generate a diffuse-energy compensated direct-to-total energy ratio parameter $r_{dir;comp}$ based on the first stream diffuse-to-total energy ratio $r_{diff;MASA}$, signal energy parameter E_{MASA}, and the energy parameter E_{ISM} of the second stream from the converted objects with

$$r_{dir;comp}(b,m) = 1 - \frac{r_{diff;MASA}(b,m)E_{MASA}(b,m)}{E_{MASA}(b,m) + E_{ISM}(b,m)}$$

Based on this diffuse-energy compensated direct-to-total energy ratio parameter, the direct-to-total energy ratio parameter $r_{dir;ISM}$ of the second stream, and the signal energy parameter E_{ISM} of the second stream, generate a weighting value w_{ISM} for the second audio stream with

$$w_{ISM}(b,m) = E_{ISM}(b,m)\frac{r_{dir;comp}(b,m) + r_{dir;ISM}(b,m)}{2}$$

Select the spatial metadata parameters to use in the merge result based on comparing the two weight values. If $w_{ISM}(b,m) > w_{MASA}(b,m)$, the direction parameters of the MASA stream from converted object signals are used and the diffuse-energy compensated direct-to-total energy ratio parameter is used.

$$\theta'(b,m) = \theta_{ISM}(b,m)$$

$$\phi'(b,m) = \phi_{ISM}(b,m)$$

$$r'_{dir}(b,m) = min\left(r_{dir;ISM}(b,m), r_{dir;comp}(b,m)\right)$$

$$\zeta'(b,m) = \zeta_{ISM}(b,m)$$

$$r'_{diff}(b,m) = 1 - r'_{dir}(b,m)$$

Otherwise, the spatial parameters of the original MASA stream, including the original direct-to-total energy ratio parameter, are used

$$\theta'(b,m) = \theta_{MASA}(b,m)$$

$$\phi'(b,m) = \phi_{MASA}(b,m)$$

$$r'_{dir}(b,m) = r_{dir;MASA}(b,m)$$

$$\zeta'(b,m) = \zeta_{MASA}(b,m)$$

$$r'_{diff}(b,m) = 1 - r'_{dir}(b,m)$$

The value of the energy parameter of the signal is updated with

$$E'(b,m) = E_{MASA}(b,m) + E_{ISM}(b,m)$$

5.9.3.3 OMASA audio signals downmix

First, the stereo amplitude panning gains $g(i,j)$ are determined for each object i and downmix channel j, based on the object directions (azimuth $\theta_{obj}(i)$ and elevation $\phi_{obj}(i)$), as described in clause 6.4.6.6 (Determining direct part gains X.X.X TODO: ADD REF) (for the "stereo" mode operation).

Then, an interpolator is determined

$$g_{interp}(n) = \frac{n}{L_{frame}}$$

Using these gains, the downmix signals are created by amplitude panning the object signals to the downmix stereo signals

$$s_{obj,dm}(n,j) = \sum_{i=0}^{N_{obj}-1} (g_{interp}(n)g(i,j) + \left(1 - (g_{interp}(n)\right) g^{[-1]}(i,j))\, s(n,i)$$

where $g^{[-1]}(i,j)$ is the amplitude panning gains determined for the previous frame.

5.9.3.4 Determination of an object to be separated

The input to the processing is the object and MASA audio signals. $s(n,i)$ is used to refer to both signals, $s_{obj}(n,i)$ is used to refer to object audio signals, and $s_{MASA}(n,i)$ is used to refer to the MASA audio signals. There are $N = N_{obj} + N_{MASA}$ input channels, where N_{obj} is the number of object channels and N_{MASA} the MASA channels. As an outcome, the separated object channel index i_{sep} is determined for this frame.

First, the energy of each input channel is determined by

$$E(i) = \sum_{n=0}^{L_{frame}-1} s(n,i)^2$$

where L_{frame} is the length of the input frame. These energies are smoothed over time by

$$E_{sm}(i) = 0.2E(i) + 0.8E_{sm}^{[-1]}(i)$$

Then, the loudest object $i_{loudest}$ is determined by selecting the channel which has the largest value of $E_{sm}(i)$ (this selection is performed among the object channels).

The determination of separated object depends on the "change object mode" η of the previous frame. If the "change object mode" was enabled in the previous frame, then the loudest object is set as the separated object, i.e.,

$$i_{sep} = i_{loudest}, \quad if\ \eta^{[-1]} = 1$$

If the "change object mode" was not enabled in the previous frame (i.e., $\eta^{[-1]} = 0$), the following is performed.

First, it is checked if the loudest object for this frame is the same as the separated object of the previous frame. If it is, the loudest object is separated, i.e.,

$$i_{sep} = i_{loudest}, \quad if\ i_{sep}^{[-1]} = i_{loudest}$$

If this is not the case (i.e., $i_{sep}^{[-1]} \neq i_{loudest}$), then the following is performed.

First, the selected channels energy is determined by

$$E_{sel} = E(i_{loudest}) + E\left(i_{sep}^{[-1]}\right) + E^{[-1]}(i_{loudest}) + E^{[-1]}\left(i_{sep}^{[-1]}\right)$$

which is a sum of the energies of the current and the previous frames for the loudest object of the current frame and the separated object of the previous frame.

Then, the sum of the energies for all the input channels of the current and the previous frames is determined

$$E_{tot} = \sum_{i=0}^{N-1}\left(E(i) + E^{[-1]}(i)\right)$$

Using them, the selected channels ratio is computed by

$$r_{sel} = \frac{E_{sel}}{E_{tot}}$$

Then, an adaptive threshold (in decibels) is determined by

$$\tau_{sel} = 9r_{sel} + 1$$

Then, the ratio between the loudest object and the previous separated object is computed in decibels by

$$r_{objects} = 10\log_{10}\left(\frac{E_{sm}(i_{loudest})}{E_{sm}\left(i_{sep}^{[-1]}\right)}\right)$$

The ratio between the objects is compared to the adaptive threshold to determine if the separated object is to be changed. If the ratio is not larger than the threshold, the separated object is not changed, i.e.,

$$i_{sep} = i_{sep}^{[-1]}, \quad if\ r_{objects} \le \tau_{sel}$$

If the ratio is larger than the threshold (i.e., $r_{objects} > \tau_{sel}$), the following is performed. First, it is checked if the selected channels ratio r_{sel} is smaller than the threshold of 0.25 (i.e., $r_{sel} < 0.25$). In that case, the levels of $i_{sep}^{[-1]}$ and $i_{loudest}$ are relatively low in comparison to the total energy (e.g., if the MASA part is louder), and the separate object is changed with a "hard switch" as the other channels are masking possible artefacts from the abrupt change in the signal waveform, i.e.,

$$i_{sep} = i_{loudest}, \quad if\ r_{sel} < 0.25$$

If the selected channels ratio is not smaller than the threshold of 0.25 (i.e., $r_{sel} \ge 0.25$), the separate object channel is not changed abruptly. Instead, fade out fade in procedure is performed (explained in detail in clause 5.9.3.5). For the current frame, the separated channel of the previous frame is used, i.e.,

$$i_{sep} = i_{sep}^{[-1]}, \quad if\ r_{sel} \ge 0.25$$

Moreover, the "change object mode" η is enabled for this frame (i.e., $\eta = 1$). As a result, a new separate object channel will be determined in the next frame. In all other cases, the "change object mode" η is disabled for this frame (i.e., $\eta = 0$).

5.9.3.5 Separation of an object from other objects

The separation of the object from the other objects depends on the value of the "change object mode" η determined in clause xxx. The audio signal $s(n,i)$ refers to the object channels in this clause. The separation can be performed via fade out fade in when the separated object changes, or the separated object can change instantly. The fade out gains are determined by

$$g_{fadeout}(n) = \frac{1 + \cos\left(\pi \frac{n}{L_{frame}}\right)}{2}$$

The fade in gains are determined by

$$g_{fadein}(n) = 1 - g_{fadeout}(n)$$

If the "change object mode" is enabled for this frame (i.e., $\eta = 1$), the separate object signal is determined by

$$s_{sep}(n) = s(n, i_{sep}) g_{fadeout}(n)$$

The remaining objects signal is determined by

$$\begin{cases} s_{rem}(n,i) = s(n,i) g_{fadein}(n), & if\ i = i_{sep} \\ s_{rem}(n,i) = s(n,i), & if\ i \ne i_{sep} \end{cases}$$

If the "change object mode" is not enabled for this frame, but it was enabled for the previous frame (i.e., $\eta^{[-1]} = 1$), the separate object signal is determined by

$$s_{sep}(n) = s(n, i_{sep}) g_{fadein}(n)$$

The remaining objects signal is determined by

$$\begin{cases} s_{rem}(n,i) = s(n,i) g_{fadeout}(n), & if\ i = i_{sep} \\ s_{rem}(n,i) = s(n,i), & if\ i \ne i_{sep} \end{cases}$$

If the "change object mode" is not enabled for this frame nor the previous frame (i.e., $\eta = 0$ and $\eta^{[-1]} = 0$), the separate object signal is determined by

$$s_{sep}(n) = s(n, i_{sep})$$

The remaining objects signal is determined by

$$\begin{cases} s_{rem}(n,i) = 0, & if\ i = i_{sep} \\ s_{rem}(n,i) = s(n,i), & if\ i \neq i_{sep} \end{cases}$$

5.9.4 Low-bitrate pre-rendering coding mode

5.9.4.1 Overview

5.9.4.2 Low-bitrate pre-rendering coding method

5.9.5 One object with MASA representation coding mode

5.9.5.1 Overview

5.9.5.2 One object with MASA representation coding method

5.9.6 Parametric one object coding mode

5.9.6.1 Overview

5.9.6.2 OMASA parametric information determination

5.9.6.3 One object with parametric representation coding method

5.9.6.3.1 Coding method overview

5.9.6.3.2 Encoding of MASA to total ratios

5.9.6.3.3 Encoding of ISM ratios

5.9.7 Discrete coding mode

5.9.8 Inter-format bitrate adaptation

5.9.9 OMASA bitrate switching

5.9.10 OMASA bitstream structure

5.10 EVS-compatible mono audio operation

Editor's note: More than referencing EVS? Anything from SCE description?

6 Functional description of the decoder

Editor's note: Referencing rendering (26.254) in each subclause, e.g., 6.2.x Binaural rendering of SBA.

6.1 Decoder overview

The IVAS codec's decoder processes the received bitstream and outputs the audio channels in one of the following formats: mono, stereo, objects, multichannel, FOA, HOA2, HOA3, loudspeaker rendering, binaural rendering, binaural rendering with a room effect.

From the received bitstream, first the IVAS format followed by number of transport channels is decoded. Then the transport channels synthesis is decoded by the decoding tools, namely Single Channel Elements decoder (SCE decoder comprising one core-coder, see Clause 6.2.3.1), Channel Pair Elements decoder (CPE decoder comprising two core-coders, see Clause 6.2.3.2), and Multichannel Coding Tool decoder (MCT decoder comprising a joint coding of multiple core-coders, see Clause 6.2.3.3) while the core-coder is inherited from the EVS codec with additional flexibility and variable bitrate (Clause 6.2.2). Finally, the transport channels are processed and upmixed (and rendered in case of internal renderer) in a scene decoder.

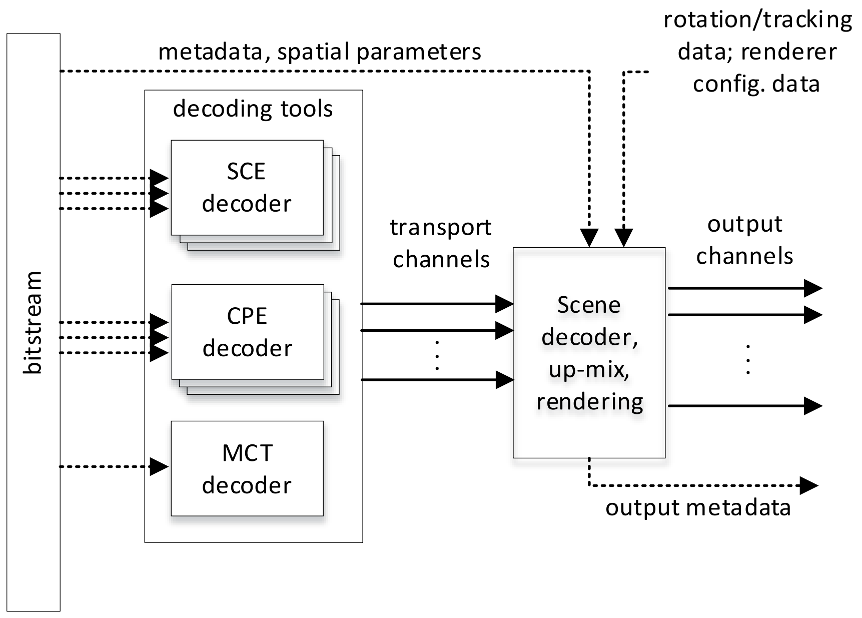

bitstream
metadata, spatial parameters
rotation/tracking data; renderer config. data
decoding tools
SCE decoder
CPE decoder
MCT decoder
transport channels
Scene decoder, up-mix, rendering
output channels
output metadata

Figure 6.1-1: Decoder Data Flow from IVAS bitstream to output data

The decoder receives all quantized parameters and generates a synthesized signal. Thus, for the majority of encoder operations it represents the inverse of the quantized value to index operations.

As EVS, for the AMR-WB interoperable operation the index lookup is performed using the AMR-WB codebooks and the decoder is configured to generate an improved synthesized signal from the AMR-WB bitstream. For EVS decoding the bitstream decoding functions are equivalent to EVS. IVAS is considered a new bitstream that shall be signalled to the decoder, i.e., whether to operate in EVS (mono) mode or IVAS mode.

The IVAS decoder includes an error concealment scheme for erroneous or lost frames, an adaptive jitter buffer algorithm run on transport channels before the rendering, output signal resampling and internal or external rendering to multiple configurations and options.

6.2 Common processing and decoding tools

6.2.1 Common decoder processing overview

6.2.1.1 High-pass filtering

The high-pass (HP) filtering operation is done on all transport channels (i.e. before rendering) except of the LFE channel. The procedure from Clause 5.2.1.1 is used.

6.2.1.2 Limiter

6.2.1.2.1 Overview

In IVAS, a two-stage distortion limiter placed at the output stage of the decoder provides protection against distortions in the output signal. The limiter is implemented as shown in Figure 5.2-6. It consists of a peak level detection (first stage detection), a strong saturation detection (second stage detection), a calculation of parameters, and an attenuation (IIR-based low-pass) filtering. It is noted that for the multichannel output signal, the distortion may occur in one, several, or all output channels. The attenuation is then to be applied to all output channels regardless if any of them originally contains saturations or not in order to preserve the spatial characteristics of the output signal.

To avoid any processing delay, no look-ahead is used. The limiter is active in the current frame if either the threshold value is exceeded in this frame, or the threshold has been exceeded in frames processed earlier and the system has not yet returned to its default state. Time taken by the system to return to within 1% of its default state (release time) is governed by a heuristic value ranging from 0 to 1. The release heuristic increases on each frame that contains a sample with a value above the threshold and decreases on each frame with all sample values below the threshold. Values of 0 and 1 map to the shortest (0.005 seconds) and longest (1 second) release time, respectively. The goal of this heuristic is to avoid the "pumping" effect when only sharp transients exceed the threshold, but also keep the gain curve smoother if the threshold is exceeded in many frames in a short span of time.

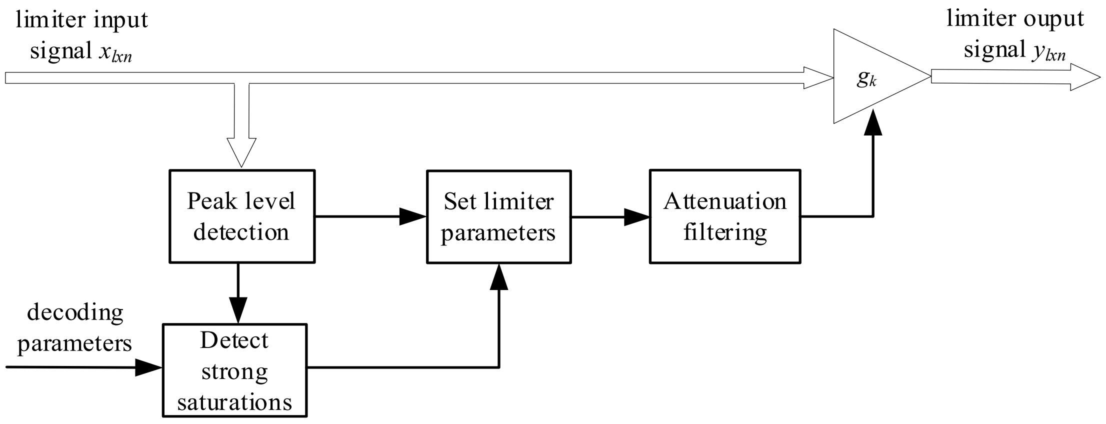


Figure 6.2-1: Schematic block diagram of the IVAS limiter

6.2.1.2.2 Detailed algorithmic description

The input to the limiter at frame with index k is a buffer containing a time-domain signal $x_{l \times n}$ consisting of l samples per each of n channels. The maximum absolute sample (peak) value $x_{k\,max}$ is found in the limiter multichannel input signal $x_{l \times n}$:

$$x_{k\,max} = \max_{\substack{0 \le i < l \\ 0 \le c < n}} (|x_{ic}|)_{(i,c)} \tag{6.2-1}$$

Based on $x_{k\,max}$ the values of release heuristic r_k and provisional target (control) gain γ_k are calculated for the frame:

If $x_{k\,max} > x_{threshold}$, then:

$$\gamma_k = \frac{x_{threshold}}{x_{k\,max}} \tag{6.2-2}$$

$$r_k = \min\left\{1, r_{k-1} + 4\frac{l}{f_s}\right\} \tag{6.2-3}$$

else, if $x_{k\,max} \leq x_{threshold}$, then:

$$\gamma_k = 1 \tag{6.2-4}$$

$$r_k = \max\left\{1, r_{k-1} - \frac{l}{f_s}\right\} \tag{6.2-5}$$

where the limiter threshold $x_{threshold}$ is a constant equal to 32729 (~-0.01 dBFS).

At this stage an additional heuristic is used to detect strong saturations in the limiter input signal. These are likely to occur when decoding frames containing bit errors, although in exceptional circumstances may also be encountered when processing valid frames. The heuristic is described in 6.2.1.2.3 below. It yields the decision whether strong limiting should be applied in the current frame and updates the provisional target gain γ_k.

The final value of the target gain $g_{k\,target}$ is determined as follows:

$$g_{k\,target} = \begin{cases} \gamma_k, & \text{if strong limiting active} \\ \max\{\gamma_k, 0.1\}, & \text{otherwise} \end{cases} \tag{6.2-6}$$

Time-constants a_{attack} and $a_{k\,release}$ used for smoothing the gain curve in the current frame are calculated as follows:

$$a_{attack} = 0.01^{\frac{1}{f_s t_{attack}}} \tag{6.2-7}$$

$$a_{k\,release} = 0.01^{\frac{1}{f_s t_{k\,release}}} \tag{6.2-8}$$

where t_{attack} and $t_{k\,release}$ describe the attack and release time in seconds that the gain curve takes to reach 99% of its target value. t_{attack} is a constant always equal to 0.005 and $a_{k\,release}$ depends on the release heuristic value in the current frame:

$$t_{k\,release} = 0.005 \cdot 200^{r_k}. \tag{6.2-9}$$

Next, l values for the gain vector G are calculated:

$$g_i = \begin{cases} a_{attack}\left(g_{i-1} - g_{k\,target}\right) + g_{k\,target}, & g_{i-1} > g_{k\,target} \\ a_{release}\left(g_{i-1} - g_{k\,target}\right) + g_{k\,target}, & g_{i-1} \leq g_{k\,target} \end{cases}. \tag{6.2-10}$$

Finally, the limiter output signal $y_{l \times n}$ is created by applying the gain vector to each sample of each limiter input channel via element-wise multiplication:

$$Y_{(i,c)} = (x_{ic} g_i)_{(i,c)}. \tag{6.2-11}$$

Note that due to the lack of look-ahead, the non-zero attack time and the restricted gain reduction range, it is possible that the limiter output contains samples whose absolute value exceeds the limiter threshold. This is accounted for at later stages of the processing chain.

Initial values for recursively defined variables:

- Limiter gain before frame 0: $g = 1$

- Release heuristic before frame 0: $r = 0$

6.2.1.2.3 Handling of strong saturations

The second stage of the limiter from Figure 6.2-1 contains a detection of strong saturations to handle saturations with an unexpectedly high level. A schematic block diagram of its logic is shown in Figure 6.2-2.

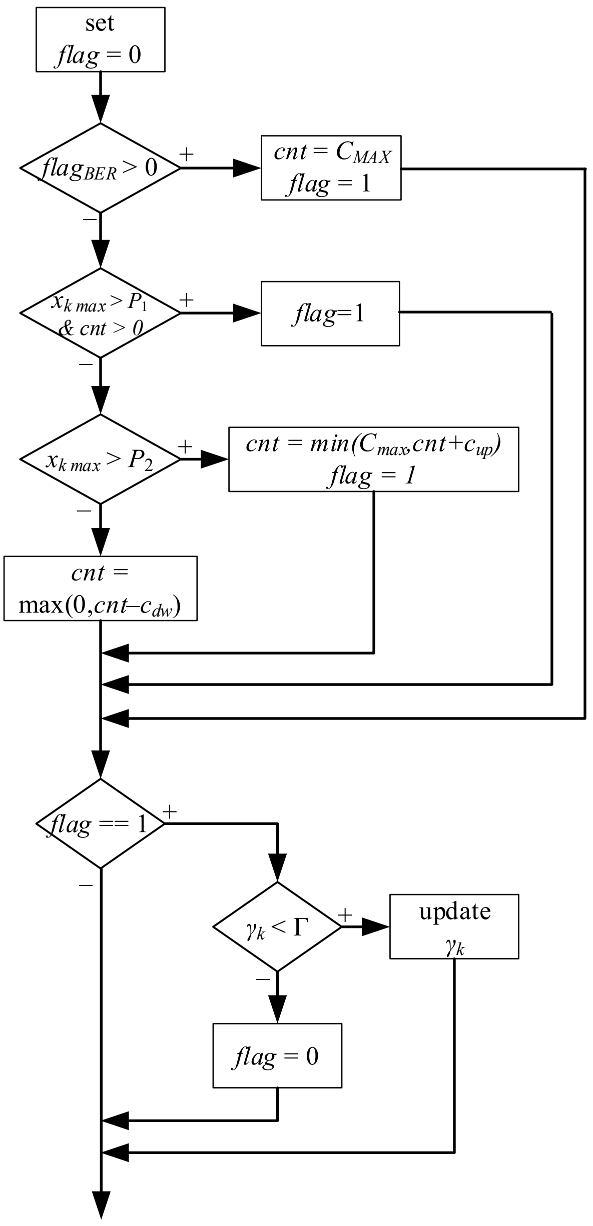


Figure 6.2-2: Schematic block diagram of the strong saturation detection logic

The strong saturation detector is applied to every channel of the limiter input multichannel signal and produces a saturation detection flag $flag$ indicating whether a strong saturation was detected or not, and an updated provisional target (control) gain γ_k. Referring to Figure 6.2-2, the saturation detection starts with initialization, at the beginning of every frame of multichannel signal, of the saturation detection flag: $flag = 0$. The saturation detector then comprises two parts:

- a first calculator which updates a saturation detection counter cnt which stores a metric measuring probability that a strong saturation is present in a current signal synthesis processing frame and produces the saturation detection flag $flag$; and

- a second calculator which (a) decides if additional attenuation is to be applied to the limiter input signal $x_{l \times n}$ and (b) determines an updated provisional target gain γ_k value to control the attenuation to be applied to the limiter input signal $x_{l \times n}$.

The first calculator of the saturation detector starts with a decision whether bit errors, signalled from the decoder by a parameter $flag_{BER}$ were detected in the received bitstream of the current frame or not. If bit errors were detected, indicated by parameter $flag_{BER} == 1$, the saturation detection counter cnt is updated to its maximum value $cnt = C_{MAX}$, $C_{MAX} = 50$, and the saturation detection flag $flag$ is set to its saturation indicating value $flag = 1$. The saturation detection then continues in the second calculator.

If the parameter $flag_{BER} == 0$, indicating that no bit errors are detected in the received bitstream, the saturation detector continues with a second decision using the detected peak value $x_{k\,max}$ from (6.2-1). If the peak value $x_{k\,max}$ is greater than a given threshold P_1 and the saturation detection counter $cnt > 0$, the saturation detection flag is set to its strong saturation indicating value of $flag = 1$. The saturation detection is then continued in the second calculator.

Then, if (a) the parameter $flag_{BER} == 0$, and (b) the peak value $x_{k\,max}$ is equal to or lower than the threshold P_1 and/or the strong saturation detection counter $cnt = 0$, the first calculator continues with a third decision whether the peak value $x_{k\,max}$ is greater than a given threshold P_2. If $x_{k\,max} > P_2$, the strong saturation detection counter cnt is updated to the minimum between C_{MAX} and the sum of $cnt + c_{up}$ and the saturation detection flag is set to $flag = 1$. Here $P_1 = 3 \cdot x_{threshold}$, $P_2 = 10 \cdot x_{threshold}$ and the constant $c_{up} = 20$ represents an incrementation step up. The saturation detection then continues in the second calculator.

Next, if the parameter $flag_{BER} == 0$, (b) the peak value $x_{k\,max}$ is equal to or lower than the threshold P_1 and/or the saturation detection counter $cnt = 0$, and (c) the peak value $x_{k\,max}$ is equal to or lower than another threshold P_2, the first calculator updates the saturation detection counter cnt to the maximum between 0 and the difference $cnt - c_{dw}$ where the constant c_{dw}=1 represents an incrementation step down. The saturation detection then continues in the second calculator.

At this point, the contribution of the first calculator of the strong saturation detector is completed.

In the second calculator of the strong saturation detector, if the saturation detection flag $flag = 0$, the strong saturation detecting method is ended and no additional attenuation is requested at the output of the second stage of the two-stage multichannel limiter.

On the other hand, if the saturation detection flag $flag = 1$, the second calculator of the strong saturation detector continues with a decision using the provisional target gain (aka gain correction factor) γ_k.

If the gain correction factor γ_k is lower than threshold $\Gamma = 0.3$, the second calculator updates its value as follows:

$$\gamma_k := \frac{\gamma_k}{\alpha} \qquad (6.2\text{-}12)$$

where $\alpha = 3.0$ is an additional correction factor to control the strength of the additional limitation of the provisional target gain γ_k. Consequently, the target gain from the first stage of the limiter is updated as well.

Otherwise, if $\gamma_k \geq \Gamma$, the second calculator sets the saturation detection flag to $flag = 0$ and performs no updating of the provisional target gain γ_k.

6.2.1.3 Time-domain decorrelator

The time-domain decorrelator takes an input audio signal and generates up to 8 outputs that are decorrelated from the input signal as well as from each other. These decorrelated outputs are used by SBA decoder (subclause 6.4), where they are combined with the transient-containing input signal to fill in energy that was lost in the downmix process. The decorrelators are also used in MASA (subclause 6.5), Parametric Multichannel (subclause 6.7.3) and Parametric Upmix (subclause xxx) processing modes.

The time-domain decorrelator includes a transient processing stage and a decorrelation stage, as shown in figure 6.2-3. The transient processing stage is used to scale the input and output of the decorrelation stage for the purpose of reducing decorrelation of transient signals, which can be perceptually undesirable.

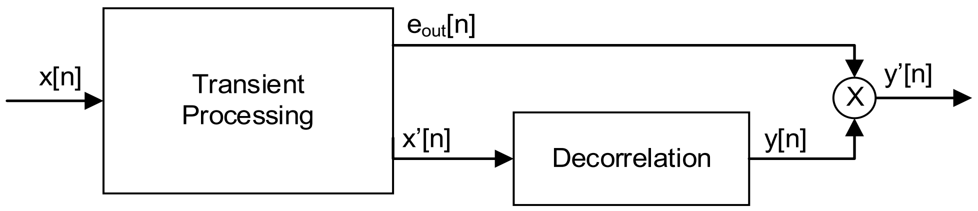


Figure 6.2-3: Time-domain decorrelator block diagram

The transient processing stage begins by filtering the input $x[n]$ with a first-order high-pass filter with a cut-off frequency of 3kHz to obtain $x_{hp}[n]$. The input signal envelope $e[n] = |x_{hp}[n]|$ is then used to obtain a fast envelope $e_1[n]$ and a slow envelope $e_2[n]$. The fast envelope e_1 is obtained by filtering e with an integrator having a 5ms integration time, whilst e_2 is obtained by filtering e with an integrator having a 100ms integration time. The fast and slow envelopes are then used to determine a time-varying scaling function $e_{in}[n]$ as follows:

$$e_{in}[n] = \min\left(c_{in\_duck}(e_{in}[n-1]-1)+1, \alpha_{duck}\frac{e_2[n]}{e_1[n]}\right), \tag{6.2-13}$$

where α_{duck} is a constant specific to the operating mode and $c_{in_duck}$is a constant that depends on the sampling rate.

A second time-varying scaling function $e_{out}[n]$ is computed as

$$e_{out}[n] = \min\left(c_{out\_duck}(e_{out}[n-1]-1)+1, \alpha_{duck}\frac{e_1[n]}{e_2[n]}\right), \tag{6.2-14}$$

where $c_{out_duck}$is a constant that depends on the sampling rate. The input $x'[n]$ to the decorrelation stage is given by $x'[n] = e_{in}[n]x[n-d]$, where d represents a 2ms delay.

The decorrelation stage consists of a cascade of $k = 1..K$ first-order all-pass filters for each output channel i. Each filter $y_{ap(i,k)}[n]$ is of the form $y_{ap(i,k)}[n] = g_{ik}x[n] + x[n-m_k] - g_{ik}y_{ap(i,k)}[n-m_k]$. The number of all-pass filters K depends on the number of decorrelator output channels N_{out}:

$$K = \max\left(2, 2^{\lceil log_2(N_{out})\rceil}\right) \tag{6.2-15}$$

The gain coefficients for each filter in each channel are given by

$$g_{ik} = 0.4\mathrm{h}_{ik}, \tag{6.2-16}$$

where $h_{ik}$are the coefficients of the 8x8 Hadamard matrix Ψ_{H8}.

$$\Psi_{H8} = \begin{pmatrix} h_{11} & \cdots & h_{18} \\ \vdots & \ddots & \vdots \\ h_{81} & \cdots & h_{88} \end{pmatrix} \tag{6.2-17}$$

The delay m_k of each filter is computed as follows:

$$\mathrm{R}_k = 3^{\frac{k-1}{4}} \tag{6.2-18}$$

$$m_k = \frac{f_s R_k \tau_{ap}}{\sum_{j=1}^{K} R_j}, \tag{6.2-19}$$

where f_s is the sampling rate in Hz and $\tau_{ap} = 20\text{ms}$. The outputs $y_i[n]$ of the decorrelation stage for channels $i = 1..N_{out}$ are scaled by e_{out} to produce the decorrelator output $y_i'[n]$:

$$y_i'[n] = y_i[n]e_{out}[n]. \tag{6.2-20}$$

6.2.2 Core-decoder processing

Similar to the core-coder (Clause 5.2.2), the core-decoder module in IVAS is based on the EVS codec from [3] while it is supplemented by more flexibility and adaptation for a multichannel processing. In the remaining part of Clause 6.2.2, only the differences of IVAS core-decoder wrt. the EVS decoder Specification from [3] are then provided.

The core-decoder module bitrate (one per one core-decoder) is not directly signalled in the bitstream but it is derived from the bit-budgets of all decoder modules in the following steps:

First, an IVAS format signalling is read from the bitstream.

Second, stereo or spatial decoding tools processing (if present) is done and their respective bit-budgets computed.

Similarly, a metadata decoding module (if present) is processed and its bit-budget is computed.

The bit-budgets for IVAS format signalling, the stereo / spatial decoding tools and metadata decoding module are then subtracted from the IVAS total bit-budget.

Finally, the remaining bit-budget is distributed among core-decoders based on the type of the core. The details of the LP-based core decoding are provided in Clause 6.2.2.1 while the MDCT-based core decoder details are provided in Clause 6.2.2.2.

6.2.2.1 LP based decoding

6.2.2.1.1 Variable bitrate ACELP decoding

Similar to the encoder, the bit-budget allocated to the ACELP core decoding module can fluctuate between a relatively large minimum and maximum bitrate span with a granularity of 1 bit. The decoder processing is then the same as used at the encoder and described in Clause 5.2.2.2.2, subclause Variable bitrate ACELP coding, including the bit-budget allocator, innovation codebook bit-budget distribution and bit-budget distribution in high bitrate ACELP.

6.2.2.1.2 Fast algebraic codebook decoding

The received algebraic codebook index is used to extract the positions and amplitudes (signs) of the excitation pulses and to find the algebraic codevector as described in Clause 6.1.1.2.1. of [3]. When the number of bits assigned to decode the codebook index belongs to the fast algebraic codebook configurations from Table 5.2-3, a dedicated decoding processing supporting these configurations is run while this decoding processing simply extends the decoding present algorithms from EVS so it supports small codebook sizes and a longer subframe length.

6.2.2.1.3 Comfort Noise Addition

6.2.2.1.4 AVQ Bit Savings Decoder

6.2.2.1.5 BPF

6.2.2.1.6 Bandwidth extension in time domain

6.2.2.1.7 Multimode FD bandwidth extension coding

6.2.2.2 MDCT based decoding

6.2.2.2.1 Variable bitrate MDCT coding

6.2.2.2.2 Improved handling of envelope stability parameter in HQ MDCT

In the linear predictor-based coding mode Section 6.1 in [3][3], stability factor θ is computed based on a distance measure between adjacent LP filters 6.1.1.3.2 in [3]. In HQ MDCT coding mode, the spectral envelope stability measure is described according to 6.2.3.2.1.3.2.3 in [3].

The transition between different modes is improved in IVAS operations by handling the spectral envelope stability parameter, es (in [3] $S^{[m]}$). Since the spectral envelope values $E^{MDCT}[b]$ (in [3]] $E_M^{[n]}(b)$) and the Euclidian distance between them D_{LP}^{MDCT} (in [3]] $\widetilde{D}_M^{[n]}$) are only available in HQ MDCT mode, and the stability factor θ is calculated in the LP based coding mode, the smooth transition between these modes requires reinitialization of these parameters and a memory module.

If both CURRENT_MODE and the mode for the previous frame are coded in HQ MDCT coding mode according to the bitstream, the envelope stability es (in [3]] $S^{[m]}$) is calculated from D_{LP} following the equations given in Equation 1839 in [3] similar to the EVS operation. When the processing of the current frame is completed, the CURRENT_MODE is stored in the variable PREVIOUS_MODE (a coding mode memory introduced in IVAS?) to be used in the following frame. The energy stability $E_{\Delta,LP}^{[-1]}$ and shape stability $\theta_{LP}^{[-1]}$ measures of the previous frames are stored in the memory as well. $\theta_{LP}^{[-1]}$ is the shape stability for the previous frame, implemented as the long-term estimate of the stability factor $\theta^{[-1]}$ from previous frame and $E_{\Delta,LP}^{[-1]}$ is the energy stability, implemented as the long-term estimate of the absolute log energy difference between synthesis frames estimated in the previous frame.

If the current mode is HQ MDCT and a transition occurred from the LP based mode to HQ MDCT, meaning CURRENT_MODE ≠PREVIOUS_MODE, D_{LP} cannot be calculated since $D_{LP}^{MDCT[-1]}$ and the spectral envelope value $E^{MDCT[-1]}[b]$ are outdated or do not exist for the Equations 1987-1838 in [3] For handling transitions between modes in IVAS operations, an estimator module is introduced. Estimation of D_{LP}^{MDCT} (or $D_{M,LP}^{[m]}$) is based on the energy stability $E_{\Delta,LP}^{[-1]}$ and shape stability $\theta_{LP}^{[-1]}$ values from the previous frame:

$$D_{LP}^{MDCT} = D_{LP,est}^{MDCT} = P_1 + P_2\,\theta_{LP}^{[-1]} + P_3 E_{\Delta,LP}^{[-1]} \tag{6.2-21}$$

where P_1, P_2, P_3 are constants, set as $P_1 = 2.93$, $P_2 = -2.20$ and $P_3 = 0.741$ using a linear regression model.

The envelope stability es is then derived from the equation 1839 in [3]] replacing the $\widetilde{D}_M^{[n]}$ with its estimation, $D_{LP,est}^{MDCT}$. The determined energy stability and shape stability parameters are stored in the memory together with the other parameters of the multi-mode decoder. Following the estimation of es, decoding of the current frame is proceeded and the memory is updated with the estimated parameters.

The energy stability of the current frame $E_{\Delta,LP}$ is then determined by using the definition as the long-term estimate of the absolute log energy difference according to:

$$E_{\Delta,LP} = \beta E_\Delta + (1-\beta)E_{\Delta,LP}^{[-1]} \tag{6.2-22}$$

$$E_\Delta = \left|E_{log} - E_{log}^{[-1]}\right| \tag{6.2-23}$$

$$E_{log} = \log_2 \frac{1}{L_{out}} \sum_{k=0}^{L-1} \hat{x}(;k)^2 \tag{6.2-24}$$

where $\hat{x}(;k)$ is the output synthesis of the current frame, L_{out} denotes the output synthesis frame length and β is a low-pass filter coefficient.

Subsequently, the stability factor θ (stab_fac) for the current frame is estimated based on D_{LP}^{MDCT} and $E_{\Delta,LP}$:

$$\theta_{est} = Q_1 + Q_2 D_{LP}^{MDCT} + Q_3 E_{\Delta,LP} \tag{6.2-25}$$

where Q_1, Q_2 and Q_3 are constants derived from a linear regression model and set as $Q_1 = 1.093$, $Q_2 = -5.84 \cdot 10^{-5}$ and $Q_3 = 0.125$. The estimated stability factor θ_{est} then becomes θ and the shape stability θ_{LP} is derived by low-pass filtering the stability factor similar to the Equation 1542 in [3]:

$$\theta_{LP} = \gamma\theta + (1-\gamma)\theta_{LP}^{[-1]} \tag{6.2-26}$$

where γ is a low-pass filter coefficient.

The resulting energy stability and shape stability measures are stored in the memory.

If the received frame is coded in LP based coding mode, in addition to the EVS operations, the energy stability $E_{\Delta,LP}$ is calculated using Equation (6.2-22). Since the stability factor θ is already available in this mode, shape stability is derived from the Equation (6.2-26). Both parameters are stored in the memory for future reference.

6.2.2.2.3 PLC method selection

In case of a packet loss when the last decoded good frame was coded with the HQ MDCT mode, the preliminary signal analysis described in 5.4.3.1 [4] is performed. The PLC method selection 5.4.3.2 [4] is amended in IVAS operation in the following way:

- An output sampling rate $f_{s,out}$ equal to 8000 Hz is only supported in EVS operation, meaning that the PLC method specified in An output sampling rate $f_{s,out}$ equal to 8000 Hz is only supported in EVS operation, meaning that the PLC method specified in [4] 5.4.3.3, 5.4.3.4 is applied.
- Otherwise, let the expression $evs\_mode\_selection$ denote the decision to use the PLC algorithm specified in [4] 5.4.3.6, and let the expression $ivas\_mode\_selection$ be defined according to

$$ivas\_mode\_selection = T_c < 56 \lor c < 0.85$$

where $\lor$ is the 'inclusive or' operator, c is the correlation parameter and T_c is the pitch computed as in [4] 5.4.3.1.2.

- In case of EVS operation, $evs\_mode\_selection$ corresponds to the decision to use the PLC algorithm specified in [4] 5.4.3.6.
- In case of IVAS operation,
    - if the expression
      $$evs\_mode\_selection \land ivas\_mode\_selection$$
      is true, the PLC algorithm specified in [4] 5.4.3.6 is used. Here, $\land$ denotes logical 'and'.
    - Otherwise, the PLC algorithm specified in [4] 5.4.3.5 is used.

6.2.2.2.4 Phase ECU enhancements

If the PLC method selection described in clause 5.2.2.1 results in the usage of [4] 5.4.3.5, the Phase ECU frame loss concealment method is used. The basis for this method is the same as in [4] 5.4.3.5, with the enhancements described in this clause. When the Phase ECU is activated, a frequency domain analysis of the previously decoded synthesis is performed as described in [4] 5.4.3.5.2, rendering a frequency spectrum $X_F(k)$ and a magnitude spectrum $|X_F(k)|$. In IVAS operation, the tonal components of $X_F(k)$ are found by identifying peaks of the spectrum using a refined peak picking method. Which results in a list of possible tones with the centre located in bins, $k_p(j),\ j = 0, \ldots, N_{peaks} - 1$.

The refinement consists of a threshold adjustment of the peak finder used for locating potential tones, for IVAS the significance level is set to 0.93, instead of the 0.97 used for EVS. This reduces the number of found peaks as it avoids classifying sidelobes around tones as false tones, this improves performance in clean tones.

Additionally, the interpolation of the found peaks is made more accurate by using the complex information in the frequency spectral estimate $X_F(k)$ when available for the three bins used for interpolation. That is, the centre peak and the two surrounding bins all have complex valued $X_F(k)$. In the other cases the parabolic interpolation from EVS is used.

The higher interpolation accuracy is achieved using the following equation.

$$k'_p(j) = K_{jacob}\ RE\left\{\frac{X_F(k_p(j)-1) - X_F(k_p(j)+1)}{2\,X_F\big(k_p(j)\big) - X_F(k_p(j)-1) - X_F(k_p(j)-1)}\right\} \quad (6.2\text{-}27)$$

where K_{jacob} is scale factor dependent on the frame length and window type use for the spectral analysis of the prototype frame. In the IVAS case K_{jacob} =1.1453. Resulting in the peaks frequency of

$$f_{k_p(j)} = \big(k_p(j) + k'_p(j)\big)\ C \quad (6.2\text{-}28)$$

Where C is the coarse resolution in Hz/bin for the spectral analysis, in this case $C = 32.25$.

The result of the refined peak picking method is the set of peak locations at fractional frequencies $k'_p(j), j = 0, \ldots, N_{peaks} - 1$, which is represented by an integer peak location $k_p(j)$ and the peak neighbourhood $k_p(j) - \delta_1, \ldots, k_p(j) - \delta_2$, where δ_1, δ_2 is defined as in [4] 5.4.3.5 equation (190). The bins of the spectrum that are not covered by the peaks $k_p(j)$ and their peak neighbourhood is defined as the noise component of the spectrum. A relative energy E_{NSR} in the range $[0,1]$ between the noise component of the spectrum and the peak component of the spectrum is calculated.

$$E_{NSR} = \frac{\sum_{k \notin G_{peak}(j)} |X_F(k)|^2}{\sum_{k=0}^{N-1} |X_F(k)|^2} \quad (6.2\text{-}29)$$

A noise attenuation factor $g_{PHECU,noise}$ is set depending on E_{NSR} according to

$$g_{PHECU,noise} = \begin{cases} 0.5, & E_{NSR} < 0.03 \\ 1, & E_{NSR} \geq 0.03 \end{cases} \quad (6.2\text{-}30)$$

The noise attenuation factor is applied to the noise component of the spectrum. Once a time domain substitution frame $x_{ph}(n)$ has been obtained as described in [4] 5.4.3.5.3, an MDCT frame completion is performed.

With MDCT frame completion the substitution frame, $x_{ph}(n)$, is combined with already decoded signal from previous frame (or previously recreated, in case of burst errors) through copying and overlap adding of the different signal segments depending on if both signals are available or not for those segments. For the initial part of the MDCT frame only the previously decoded (or recreated) signal is available and therefore copied into a processing buffer. The second part where both already decoded signal and the substitution frame are available are combined with overlap add to the processing buffer. Finally, the remaining part of the substitution frame is copied into the MDCT processing buffer. The processing buffer is then processed using the MDCT processing windowed and TDA processed used in the encoder [4] 5.3.2.2 and then completed with the normal decoder side processing [4] 6.2.4.1, consisting of ITDA, windowing and ola processing in the same way as normal decoder thereby generating an approximation of the lost frame.

6.2.2.2.5 Long term prediction processing

LTP post filtering is used in the similar way as described in 5.1.14.1.1.1 and 6.9.2 in [3]. LTP post filtering is used for post-processing TCX20 and TCX10/5 frames at bitrates below 96 kbps. Different to [3] there is no LTP gain reduction at 48 kbps.

The parameters of the LTP filter, being the LTP gain and the pitch gain, may change and are updated in regular update intervals. The framing of 20 ms is chosen as the regular update interval. LTP filter coefficients are derived from the LTP parameters in the same way as in [3]; the LTP filter coefficients change whenever the LTP parameters change.

If the LTP gain is non-zero in the previous and in the current frame, it means that the LTP post filtering is active in the previous and in the current frame. If the LTP is active in the previous and in the current frame and if the LTP parameters have changed, OAO smoothing is used for the transition in the initial subinterval of 5 ms. OAO smoothing is more efficient than the smoothing in [3] that uses zero input response.

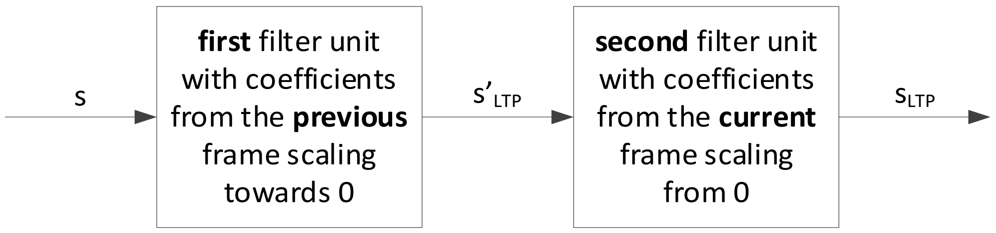

s
first filter unit with coefficients from the **previous** frame scaling towards 0
s'LTP
second filter unit with coefficients from the **current** frame scaling from 0
sLTP

Figure 6.2-4: Block diagram of the LPT post filtering

OAO smoothing is accomplished by two filter units. The block diagram for the LTP post filtering that uses OAO smoothing is shown in Figure 6.2-4. The first filter unit filters the input signal $s(n)$ using the coefficients according to the LTP parameters associated to the previous frame:

$$s'_{LTP}(n) = s(n) + c_{k-1}(n)\left(\sum_{i=0}^{P} b_{k-1,i}s(n-i) - \sum_{j=1}^{Q} a_{k-1,j}s'_{LTP}(n-j)\right), kT \le n < kT + T_l$$

where $c_{k-1}(n)$ changes from the LTP gain in the previous frame towards 0 when n increases. The second filter unit filters the output of the first filter unit $s'_{LTP}(n)$ using the coefficients according to the LTP parameters associated to the current frame:

$$s_{LTP}(n) = s'_{LTP}(n) + c_k(n)\left(\sum_{i=0}^{P} b_{k,i}s'_{LTP}(n-i) - \sum_{j=1}^{Q} a_{k,j}s_{LTP}(n-j)\right), kT \le n < kT + T_l$$

where $c_k(n)$ changes from 0 to the LTP gain in the current frame when n increases.

6.2.2.3 Switching coding modes

6.2.2.4 DTX/CNG operation

6.2.3 Common audio decoding tools

This Clause describes common audio coding tools used for decoding of the transport channels. Similar to the encoder tools, the individual decoder tools can be used simultaneously multiple times similarly as a combination of different tools can be used simultaneously to decode different numbers of transport channels. The exact set-up depends on the actual IVAS format, IVAS total bitrate, and its actual mode and it is described in particular IVAS format decoder related Clauses.

6.2.3.1 Single Channel Element (SCE) decoder

The Single Channel Element (SCE) decoder is a decoding tool that decodes one transport channel. It is based on one core-decoder module. Its block diagram is shown in Figure 6.2-5.

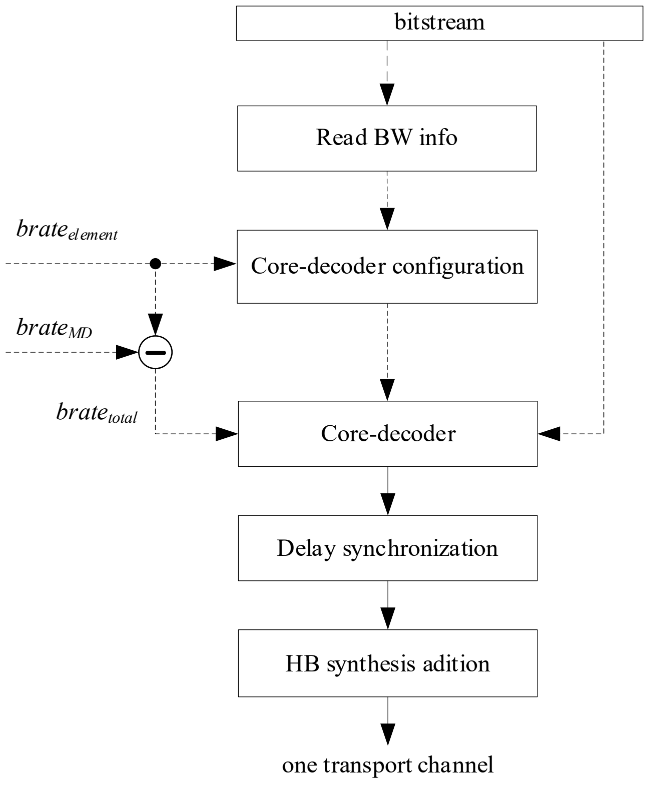


Figure 6.2-5: Block diagram of the Single Channel Element (SCE) decoder

As seen from Figure 6.2-5, the SCE decoder tool consists of the following modules:

(a) The audio bandwidth information is read from the bitstream.

(b) Core-decoder configuration module: high-level parameters like core-decoder internal sampling-rate, core-decoder nominal bitrate are set. Note that it depends on the element bitrate, $brate_{element}$, which is constant at one IVAS total bitrate, $brate_{IVAS}$, and ensures that there is no frequent switching between different core-decoder set-ups.

(c) One variable bitrate core-decoder (Clause 6.2.2).

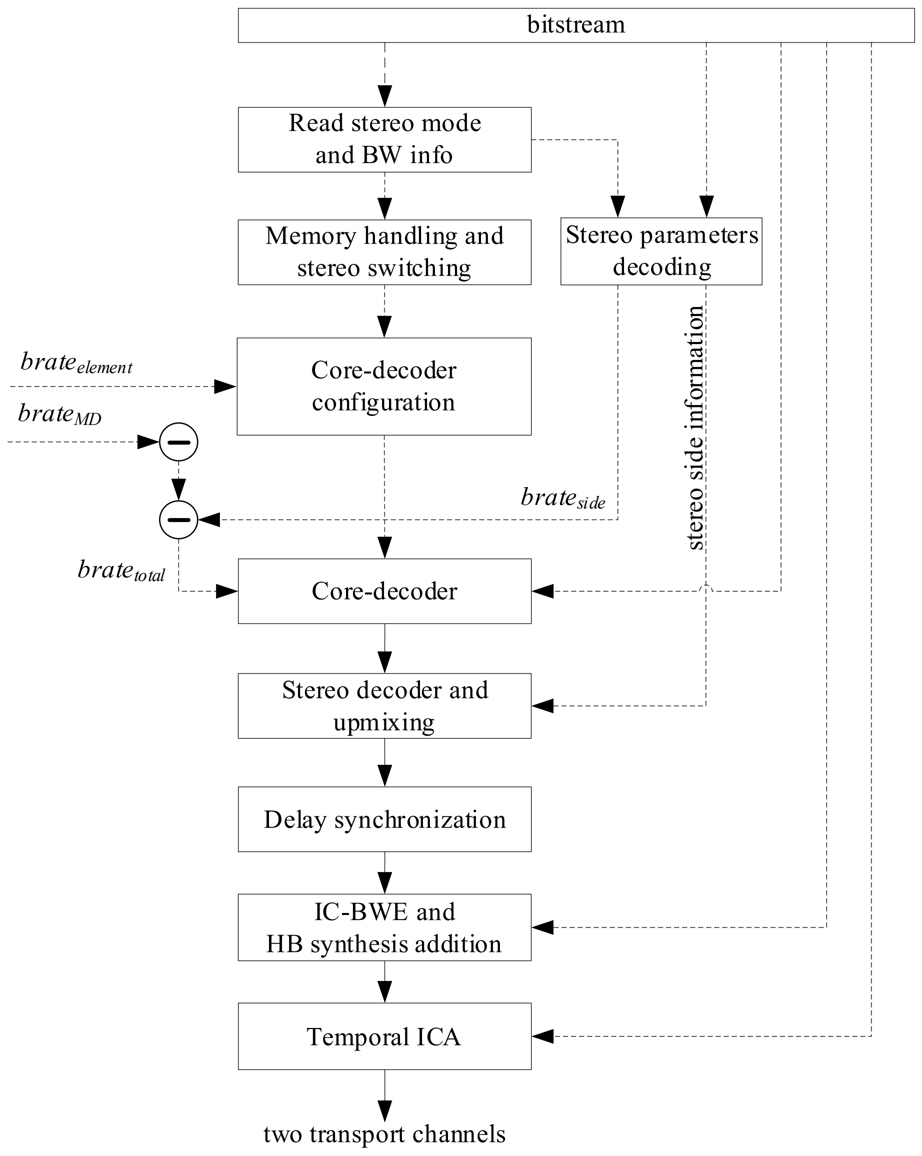
(d) Delay synchronization: the low-band synthesis is synchronized to be in line between different IVAS elements so it is delayed by $3.25 - 1.25 = 2.0$ ms.

(e) The high-band synthesis from BWE modules is synchronized between IVAS elements/formats so it is delayed by $3.25 - 2.3125 = 0.9375$ ms.

Note that the core-decoder module (c) depends on the core-decoder total bitrate $brate_{total}$ which is a difference between the element bitrate $brate_{element}$ and bitrate related to encode the metadata $brate_{MD}$ (if present) as defined in (5.2-92).

6.2.3.2 Channel Pair Element (CPE) decoder

The Channel Pair Element (CPE) decoder is a decoding tool that decodes two transport channels. It is based on one or two core-coder module(s) supplemented by stereo coding tools. Its block diagram is shown in Figure 6.2-6.



bitstream
Read stereo mode and BW info
Memory handling and stereo switching
Stereo parameters decoding
$brate_{element}$
Core-decoder configuration
$brate_{MD}$
stereo side information
$brate_{side}$
$brate_{total}$
Core-decoder
Stereo decoder and upmixing
Delay synchronization
IC-BWE and HB synthesis addition
Temporal ICA
two transport channels

Figure 6.2-6: Block diagram of the Channel Pair Element (CPE) decoder

As seen from Figure 5.2-8, the CPE decoder consists of the following modules:

(a) The stereo mode and audio bandwidth information are read from the bitstream.

(b) Memory handling and stereo switching (Clause 6.3.5) controls the switching between DFT stereo, TD stereo and MDCT stereo coding modes and comprises dynamic allocation/deallocation of static memory of stereo modes data structures depending on the current stereo mode. Also, the TD stereo mode is set in this module.

(c) Core-decoder configuration, one per transport channel: high-level parameters like core-decoder internal sampling-rate, core-decoder nominal bitrate etc. are set. Note that it depends on the element bitrate, $brate_{element}$, which is constant at one IVAS total bitrate, $brate_{IVAS}$, and ensures that there is no frequent switching between different core-decoder set-ups.

(d) One or two variable bitrate core-coder(s) (Clause 6.2.2) performed sequentially on one or two downmixed channels.

(e) Stereo decoder and upmixing module in case of DFT stereo (Clause 6.3.3.2) or TD stereo (Clause 6.3.3.3) while the stereo parameters are decoded in the stereo parameters decoding module.

(f) Delay synchronization: the low-band synthesis is synchronized to be in line between different IVAS elements so it is delayed depending on the stereo mode: the delay is $3.25 - 3.125 = 0.125 \text{ ms}$ in case of DFT stereo, $3.25 - 1.25 = 2.0 \text{ ms}$ in case of TD stereo and MDCT stereo.

(g) Inter-Channel BWE (IC-BWE) decoder module (Clause 6.3.3.1.2) decodes high-frequencies in DFT stereo and TD stereo modes and add the HB synthesis to the low-band synthesis.

(h) Temporal Inter-Channel Alignment (ICA) decoder module to time-align the two transport channels (Clause 6.3.3.1.1).

Note that the core-decoder module (d) depends on the core-decoder total bitrate $brate_{total}[n], n = 1, 2$. The sum of total bitrates of both transport channels is then a difference between the element bitrate $brate_{element}$ and bitrates related to decode the metadata $brate_{MD}$ (if present) and stereo side information $brate_{side}$ (if present) as defined in (5.2-93).

6.2.3.3 Multichannel coding tool (MCT) decoder

6.2.3.3.1 General overview

The decoding process of MCT is very similar to the MDCT-stereo based decoder described in clause 6.3.4. After the decoding of the side bit information and the MCT parameters from the bit stream, the number of bits allocated to each channel can be determined with the decoded bitrate ratios and the exact same procedure as the encoder, described in clause 5.2.3.3.5. Then, inverse quantization of the spectra of the jointly coded channels, followed by the noise filling as described takes place. Afterwards, IGF or stereo IGF is applied. From the bitstream signalling, it is also known if there are channel-pairs that were jointly coded. The inverse processing should start with the last channel-pair block formed in the encoder in order to convert back to the original whitened spectra of each channel. For each channel pair the inverse stereo processing is applied base on the stereo mode and the band-wise M/S decision. For all channels that were involved in channel pair blocks and were jointly coded the spectrum is de-normalized to the original energy level based on the $\widehat{ILD}(k)$ values that were sent from the encoder. Finally, TNS and SNS is applied to reconstruct the original spectra of the individual channels and the inverse MDCT transforms the signals to the time domain.

6.2.3.3.2 Core and SNS parameter decoding

The decoding of the core parameter side bits -including the SNS decoding-is applied for each active channel (excluding the LFE channels if it exists) and the process is the same as for MDCT-stereo which is described in clauses 6.3.4.2.2-6.3.4.2.3. The active channels are assigned to $\frac{N}{2}$ CPEs, which are configured as MDCT-stereo CPEs per default. As the encoding of the core side parameters are written to the bitstream before the MCT channel pairing process, the decoded parameters represent the parameters for each active channel in sequential order $0, \cdots, N-1$.

6.2.3.3.3 MCT parameter decoding

The MCT parameters that are decoded needed to perform the inverse MCT processing, are the following in the order they are being read from the bit stream:

- The number of selected channel pairs $Block\_cnt$
- If $\text{Bloc}k\_c\text{nt} \neq 0$ then for each active channel the $\widehat{ILD}$ is read, meaning the quantized level differences to the mean energy level, otherwise $\widehat{ILD}$ are set to *0*.
- The flag for signalling downscaling with bit *0* and upscaling with bit *1* for the normalization of each channel as described in the encoder subclause 5.2.3.3.3.1.
- Finally, the pairwise processing data is decoded. More precisely, for each channel pair block from $0, \cdots, \text{Bloc}k\_c\text{nt} - 1$:
    - The channel pair index is read, which discloses the selected channels for the particular channel pair block as described in Table 5.2-18
    - The stereo information for the joint processing of the particular channel pair, which include:
        - The stereo mode for the core coding bands
        - The respective MS mask if the stereo mode is band-wise MS
        - If IGF is active the stereo mode for the IGF bands
        - The respective MS mask if the IGF encoding mode is band-wise MS

6.2.3.3.4 Bitrate distribution

The number of bits allocated to each channel must be known before fully decoding the bitstream and the spectral coefficients for each channel.

The bit ratio for each channel $\hat{r}_i$is decoded from the bit stream. After the decoding of the core coding parameters and the MCT side data, the number of bits $bits_{remaining}$ remaining to decode the spectral coefficients of each channel is known. The exact same procedure described for the encoder in subclause 5.2.3.3.5 is applied to retrieve the number of bits allocated to each channel $bits_i$.

6.2.3.3.5 Spectral data decoding and noise filling

Spectral data for each active channel are decoded using the range decoder described in 6.3.4.3.2 with the respective number of assigned bits $bits_i$. Then inverse quantization and noise filling is applied as for MDCT -stereo and as is described in the respective clauses 6.3.4.3.1.

6.2.3.3.6 Application of IGF and stereo IGF

IGF is applied on the whitened, ILD compensated spectral domain.

If from the decoded MCT parameters it is established, that either there are no channel pair blocks, i.e. $Block\_cnt = 0$, or some of the active channels are not part of a channel-pair block, then single channel IGF is applied as is described in [3] subclause XX and IGF subclause of IVAS.

If there are channel-pair blocks, then the same conditions apply as described in MDCT-stereo and clause 6.3.4.3.3. If either the core stereo encoding mode or the IGF stereo encoding more is not DUAL_MONO, stereo IGF as described in 6.3.4.3.3.2 is applied. Otherwise, the IGF is applied to each channel separately.

6.2.3.3.7 MCT decoding

6.2.3.3.7.1 Inverse M/S processing of channel pairs

If there are channel pair blocks present, then for each block the inverse M/S processing is applied depending on the core stereo encoding mode and the IGF stereo encoding mode (if IGF is active) as described in the MDCT-stereo decoder clause in 6.3.4.4.1.

6.2.3.3.7.2 ILD de-normalization

To retrieve the original whitened signals that were input to the MCT at encoder side, the de-normalization towards the initial energies of the signals with use of the decoded quantized ILDs $\widehat{ILD}$ is applied. That is summarized in the following:

If $\widehat{ILD} \neq 0$ and if upscaling is flagged

$$\hat{a}_i = \frac{\widehat{ILD}(i)}{ILD_{RANGE}} \tag{6.2-31}$$

if downscaling is flagged then the normalization factor is derived from

$$\hat{a}_i = \frac{ILD_{RANGE}}{\widehat{ILD}(i)} \tag{6.2-32}$$

where $i = 0,1,\cdots,N_{ACTIVE} - 1$.

Then the spectrum of each channel is de-normalized by scaling with the de-normalization factor $\hat{a}_i$.

6.2.3.3.8 Noise shaping

As in MDCT stereo, after the retrieval of the whitened signals of all active channels, temporal and noise shaping is applied to reconstruct the original spectra. As described in clause 6.3.4.5, the TNS may be applied either prior or after the scaling of the MDCT spectrum with the SNS coefficients.

Detailed description of the SNS decoding and spectral shaping can be found in clause 6.3.4.5.2.

Detailed description of the TNS can be found in subclause 6.2.2.3.10 of [3].

6.2.3.3.9 Inverse MDCT transform to time domain

Finally the signals are transformed back to the time domain using an inverse MDCT transform. In subclause 6..2.4 of [3], the details of the processing including the inverse MDCT, overlap-add and windowing are described.

6.2.4 Common spatial metadata decoding tools

6.2.4.1 Direction metadata decoding tools

6.2.4.1.1 Overview

6.2.4.1.2 Direction dequantization tools

6.2.4.1.2.1 Joint azimuth-elevation dequantization

6.2.4.1.2.2 Azimuth dequantization

6.2.4.1.3 Direction metadata raw decoding

6.2.4.1.4 Direction metadata entropy decoding tools

6.2.4.1.4.1 Quasi uniform decoding

The decoding function, *decode_quasi_uniform(alphabet_size)* is defined in pseudo-code as:

```
bits = floor(log2(alphabet_sz))
thresh = 2 ^ (bits + 1) - alphabet_sz
value = read_bits(bits)
if (value >= thresh)
        value = (value * 2 + read_bits(1)) - thresh
return value
```

6.2.4.1.4.2 Extended Golomb Rice decoding

The extended Golomb-Rice decoding can be performed with the following pseudo-code, given as input the alphabet size and the extended GR parameter, that indicates the limit of the most significant bits.

```
Decode_extended_GR():
    msb_size = ( alph_size + ( 1 << gr_param ) - 1 ) >> gr_param;
    if ( msb_size <= 3 )
        value = decode_quasi_uniform( index, alph_size );
    else
        msb = 0;
        while ( ( msb < msb_size - 1 ) && ( read_bits(1) != 0 ) )
            msb++;

        if ( msb == msb_size - 1 )
            lsb = decode_quasi_uniform( alph_size - ( ( msb_size - 1 ) << gr_param ) );
        else
            lsb = 0;
            for ( i = 0; i < gr_param; i++ )
                lsb = ( lsb << 1 ) + read_bits(1);

        value = ( msb << gr_param ) + lsb;
```

6.2.4.2 Diffuseness and energy ratio decoding methods

This decoding method is the counter part of the encoding method described in clause xxx.

If the *nbands* is equal to one, only one diffuseness value is transmitted is coded in binary code on 3 bits.

Otherwise, the *dif_use_raw_coding* 1-bit flag is read.

If it is equal to 0, entropy coding was used.

For nbands<8, the next 1-bit *dif_have_unique_value* is read. If this flag is 1 only one value is decoded using the *decode_quasi_unform()* function using an alphabet of 8. The unique decoded diffuseness value is then copied to all the nbands. If *dif_have_unique_value* is zero, all diffuseness values consist only of two consecutive values. First the minimal diffuseness value is decoded *decode_quasi_uniform()* function using an alphabet of 8, then for each nbands an offset of 0 or 1 and coded on 1bit binary code is decoded. The final diffuseness is then:

```
Diffuseness[b] = diff_min + offset[b]
```

For nbands>=8, a Huffman coding strategy was used, and first an average diffuseness is decoded from a 3-bits binary code. The average diffuseness is adjusted for each frequency bands by decoding an unary code with leading 1 and terminating 0, and this up to 9 leading value, where the value 9 is amended by a 2-bit binary code. The decoded value is then used to update the average diffuseness value as follows:

```
Diffuseness[b] = diff_avq + (value[b]+1)/2 if value[b] % 2== 1
Diffuseness[b] = diff_avq - (value[b]/2) if value[b] % 2== 0
```

In of case *dif_use_raw_coding* is equal to 1, all nbands diffuseness values are decoded using *decode_quasi_unform()* function using an alphabet of 8.

As in the encoding, the minimum coded diffuseness index is computed. It is used to set the dif_index_max_ec using the subsequent entropy coding 1. If the minum diffuseness is above 6 dif_index_max_ec = 7, otherwise dif_index_max_ec = 5.

6.2.4.2.1 Diffuseness decoding

6.2.4.2.2 Diffuseness and energy ratio decoding with two concurrent directions

6.2.4.3 Direction metadata decoding methods

6.2.4.3.1 Direction metadata decoding overview

6.2.4.3.2 Decoding EC1

6.2.4.3.2.1 Introduction

A first 1-bit flag is read. If its value if 1 the raw decoding will be performed, otherwise, the frame-wise entropy decoding will follow.

6.2.4.3.2.2 Raw decoding

If the raw coding method was signalled, the direction azimuth and elevation are decoded from the lowest to the highest frequency band, and then for each time blocks.

In case of 2D scene, the elevation angle is set to zero and the azimuth codebook is selected depending of the decoded diffuseness index, using *decode_quasi_unifirm()* function. In case of HR MASA, the maximum resolution is used to decode the azimuth angle.

In case of 3D, both elevation and azimuth are decoded jointly reading and decode a spherical index per frequency band and time block.

6.2.4.3.2.3 Frame-wise entropy decoding

The direction parameters are decoded form the lowest transmitted frequency band *start_band* to the highest *nbands-1*. For each frequency band the previously diffuseness index is used to deduce the direction quantization resolution, except for HR MASA, where the maximal resolution is used corresponding to a diffuseness index of 0.

If the diffuseness index is above the threshold *diffuseness_index_max_ec*, the direction parameters are raw coded in the bitstream and the spherical indices are read and decoded for each time block according to the associated resolution in case of 3D. In case of 2D, *decoder_quasi_uniform()* is used to decode the *nblocks* azimuth indices associated to the derived resolution and elevation angles are set to zero.

For the other frequency bands with diffuseness indices below and equal to *diffuseness_index_max_ec,* the direction parameters are entropy coded.

The average elevation angle is first decoded in case of 3D, using *decode_quasi_uniform()* followed by *deorder().* The extended GR parameter between 0 and 4 is then decoded using *decode_quasi_uniform().* In case extended GR parameter is equal to 4, all the elevation quantized distanced from the average elevation angle zero. The average elevation is then projected for each band and block to the respective codebook and angular resolution. Along with the diffuseness index, it will be used to select the codebook and alphabet of the azimuth. If the extended GR parameter is below 4, the distance in index from the projected elevation index is decoded using *decode_extended_gr().* The distance index is reordered before being added to the projected average elevation index and used for the inverse quantization of the elevation angle.

The azimuth angles are decoded in a second step, by decoding the average azimuth index using *decode_quasi_uniform()* and *deorder()*. The extended GR parameter between 0 and 5 is then decoded using *decode_quasi_unform().* In case the GR parameter is 5, all the azimuth distance index are set to zero, and the projected average azimuth indices are computed for each frequency bands and used for each band and block for the inverse quantization. For the GR parameter below 5, the extended_GR_decode() followed by the *reorder()* to adjust the projected average azimuth index, which later on used for the inversion quantization.

6.2.4.3.3 Decoding EC2

6.2.4.3.4 Decoding EC3

6.2.4.4 Coherence decoding

6.2.4.4.1 Spread coherence decoding

6.2.4.4.1.1 Spread coherence decoding for bitrates up to 256 kbps

6.2.4.4.1.2 Spread coherence decoding for 384 kbps and 512 kbps

6.2.4.4.2 Surround coherence decoding

6.2.4.4.2.1 Surround coherence decoding for bitrates up to 256 kbps

6.2.4.4.2.2 Surround coherence decoding for 384 kbps and 512 kbps

6.2.4.4.3 DTX Decoding

DTX decoding, common to SBA-DirAC and MASA, is the dual part of the DTX encoding described in clause xxx. In the case of non-active frames and for the SID frames, 5 bands are transmitted for MASA and only 2 for SBA-DirAC. Only one direction per parameter band is transmitted, and a time resolution of 20ms is used, i.e. only 1 time block is used. The total payload allocated to the SID sound field parameters is a maximum of 2.8kbps, corresponding to the total SID bit rate of 5.2 kbps minus the 2.4 kbps SID of core-coding.

For MASA, one signalling bit is read to signal in case of 2 transport channels to know if DFT stereo or discrete CPE MDCT coding is used for the core-coding. Another bit is read to know whether the transmitted soundfield parameters are for a 2D or 3D scene.

First the diffuseness indices are decoded for each parameter bands using the *decode_quasi_uniform()* function with an alphabet of 4. In case of SBA-DirAC, the diffuseness indices are sufficient to decode the rest of the SID payload, while in MASA the same heuristic as done at the encoder side and described in section in clause xxx must be performed.

The directions, one DoA per parameter band and frame, are decoded in a second step. It is achieved by decoding the spherical indices in 3D coded by raw coding and decoding the azimuth angle by using the function *decode_quasi_uniform()* in case of 2D.

To avoid abrupt transitions between the new directions decoded in the current SID frame and the previous directions, and to increase the temporal resolution from 20ms to the 5ms resolution used in the upmix, a smoothing is performed at the time block resolution, i.e. 5ms, by interpolating the two sets of directions by successively adding the Cartesian coordinate of the new direction to the previous direction at each time block step of the frame, and this for each parameter band. The 4 resulting Cartesian coordinates per parameter band associated to the 4 time blocks are transformed into polar coordinates for further processing.

6.2.5 Common filter bank operation tools

Editor's note: Used in SBA/MASA decoding and decoding of some MC modes.

6.2.5.1 Complex Low-delay Filter Bank (CLDFB) analysis/synthesis common parts

6.2.5.2 Complex Low-delay Filter Bank (CLDFB) analysis

6.2.5.3 Complex Low-delay Filter Bank (CLDFB) synthesis

6.3 Stereo audio decoding

6.3.1 Stereo format decoder overview

6.3.2 Common stereo decoding tools

6.3.2.1 Common Stereo postprocessing

6.3.2.2 ITD compensation

6.3.3 Unified stereo decoding

6.3.3.1 Common unified stereo decoding tools

6.3.3.1.1 Inter-channel Alignment (ICA) decoder

6.3.3.1.2 Inter-channel BWE (IC-BWE) decoder

Decoder side Inter-Channel Bandwidth Extension (IC-BWE) functionality for a stereo audio signal is structured in two stages, see block diagram in Fig 6.3-1. First, the spectral mapping of a reference channel to non-reference channels in the synthesized high-band of the signal i.e., the core decoding operation for IC-BWE, takes place, creating non-reference channels using the reference channel and various parameters. Consequently, the incorporation of the high-band (HB) synthesis into the final output takes place along with the management of the transitions between different core decoders. These two stages are explained next in further detail.

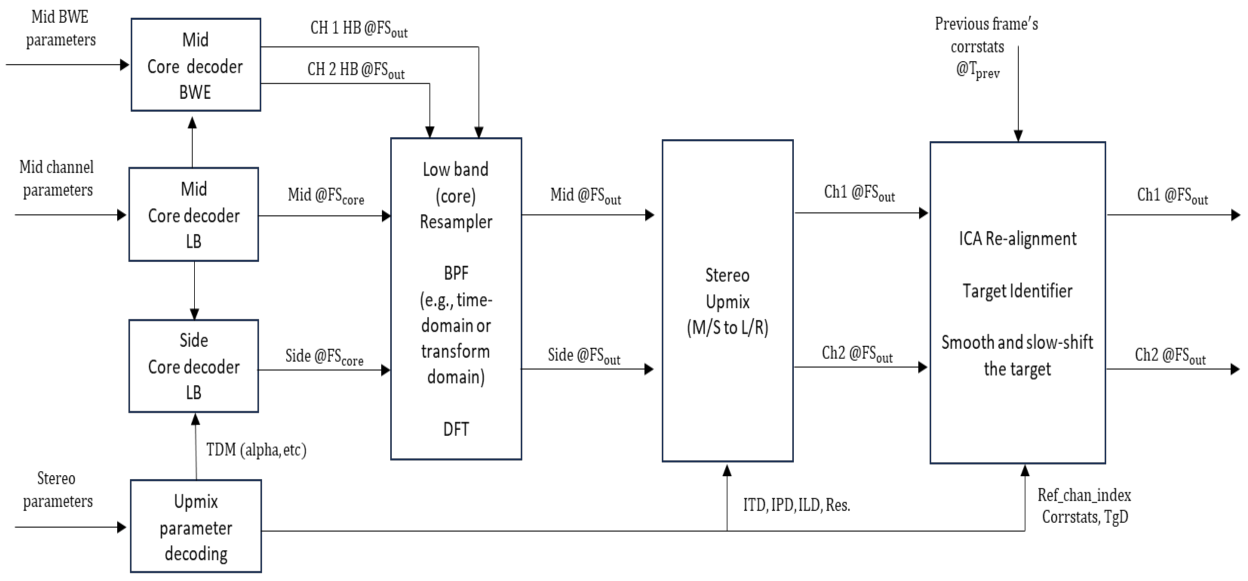


Figure 6.3-1: IC-BWE decoder block diagram

There are certain conditions including (core DFT mode with) mono output, no-data frame (core bitrate below a threshold), core not being ACELP, and some others, which cause IC-BWE process to be skipped. Time domain high-band stereo filling takes place. If core bitrate is below a threshold along with core DFT mode, the reference channel HB synthesis is copied to the non-reference channel HB synthesis.

For the proper IC-BWE processing, spectral and gain shape indices are retrieved from the bitstream for the current frame and stored. In case the current frame is bad, last successfully decoded indices are retrieved. Spectral mapping indices are dequantized, and gain shape indices are mapped (using pre-defined lookup tables) to reverse the encoding process induced changes and restore values closer to the original values of these parameters. The spectral and gain shape mappings are crucial aspects of the signal that determine its frequency content and volume levels, respectively.

An excitation signal is generated for the non-reference channel based on the reference channel. This is done by creating a combination of a noise component and pitch structure derived from the reference channel. A voicing factor is used to blend these two components together.

The output high-band synthesis signal is generated by filtering the excitation signal through a synthesis filter (a linear predictive coding (LPC) filter) and adjusting the spectral and gain shape. to the excitation signal.

The output high-band synthesis signal is resampled to the output sample frequency to change the sample rate of the signal to match the output sample frequency.

The final resampled HB synthesis signal for the non-reference channel is stored, ready to be used by consequent decoding processing.

For the incorporation of the high-band (HB) synthesis signal into the final decoder output, it is checked if the current mode is time domain (TD) with left-right channels (LRTD). If affirmative, the following steps are performed:

· The high-band (HB) synthesis, in all channels, is adjusted by adding a delay.

· If the output is mono, the delayed stereo HB synthesis signal is downmixed to mono and added to the output.

· Otherwise i.e., if the output is not mono, the delayed HB synthesis is added to the delayed ACELP synthesis for each channel.

Decoder IC-Realignment

The BWE synthesized audio data for the current frame for both channels in their respective buffers is preserved for use in the next frame. Dequantization of the inter-channel temporal shift and gain factor parameters takes place. Similar to what occurred at the encoder, there is a channel Identification and adjustment stage. The reference and target channels are identified. If the temporal shifts between the current and previous frames differ, the target signal is adjusted accordingly.

The adjusted target channel is scaled by a dequantized gain factor to balance the audio levels between the two channels.

6.3.3.2 DFT-based stereo decoding

6.3.3.2.1 General

The DFT stereo decoding is the duality of the DFT-based stereo encoding described in 5.3.3.2.2. The mono downmix is coded and decoded by one the core-coders, before performing the upmix in frequency domain. Some stereo processing steps are still performed in time domain like, like the ITD handling, and the decorrelator.

6.3.3.2.2 STFT analysis

A time-frequency transform like at the encoder side is applied to the decoded downmix $\hat{m}$ yielding time-frequency vectors:

$$\widehat{M_i}[\mathrm{k}] = \sum_{n=0}^{\mathrm{N}-1} \hat{m}[n + i.M - L1 - zp] w_{dec}[n\%2, n] e^{-j\frac{2\pi \mathrm{k} n}{N}} \quad (6.3\text{-}1)$$

$w_{dec}[n\%2, n]$ are the analysis window functions at the decoder side defined for odd and even i indexes, i being the index of the M size stride (stride parameter), zp corresponds to the zero padding of the analysis windows, $L1$ the outer overlapping size, k is the frequency index, N the DFT size, and j is the imaginary unit. Unlike the encoder side, the stride is of only 10ms, and two DFTs are performed pr frame of 20ms. The overlapping in the outer bounds of the two DFTs analyses are of 3.125ms, while the inner overlapping is of 9.375ms. Zero padding is added at both ends equally to end up with windows of a total duration of 20ms as depicted in Figure 6.3-1.

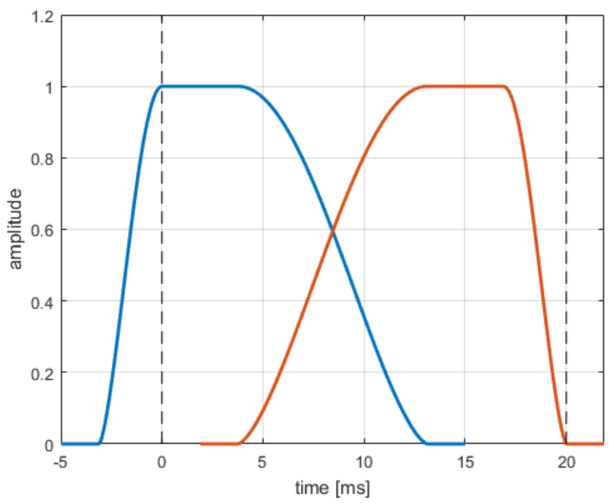


Figure 6.3-1: Analysis and synthesis windows of DFT stereo at decoder side

Same windows are used for the STFT synthesis, since overlapping windows are defined as sine windows.

$$w_{dec}[0,n] = \begin{cases} sin\left(\frac{\pi.(n-zp+0.5)}{2.L1}\right) for\ zp \le n < zp + L1 \\ 1, for\ zp\ +\ L1 \le n < M - L2/2 \\ sin\left(\frac{\pi.(n-zp+0.5)}{2.L2}\right) for\ M - L2/2 \le n < M + L2/2 \end{cases} \tag{6.3-2}$$

and zero otherwise. The second window corresponds to inverse of the first one,

$$w_{dec}[1,n] =\ w_{dec}[0, N-1-n] \tag{6.3-3}$$

where $L1$ is the size in samples of the outer overlapping of 3.125ms, $L2$ is the size in samples of the inner overlapping of 9.375ms.

6.3.3.2.3 Stereo parameter decoding

The global IPD index is decoded on 4 bits for the active frames and on 2 bits for the SID frame. The global IPD is then recovered by:

$$\widehat{gIPD} = gIPD_{index}.deltaIPD - \pi, \tag{6.3-4}$$

where $deltaIPD = 2\pi/gIPD_{index\_max}$, where $gIPD_{index\_max} = 2^{gIPD_{bits}}$.

The side and residual prediction gains are coded jointly. First the absolute ILD level is decoded along with sign of the side gain: The ILD level is coded on 4 bit value between 0 and 15, and is used to select the codebook used for coding jointly the absolute value of the side gain and the residual prediction gain:

$$\begin{cases} |\hat{g}| = gains_{cdbk}[8 * ILD_{level} + gains_{index}][0] \\ \hat{r} = gains_{cdbk}[8 * ILD_{level} + gains_{index}][1] \end{cases} \tag{6.3-5}$$

where $gains_{cdbk}[]$ is the table given in Table 5.3-7, in clause 0.

The side decoded gain is then given by:

$$\hat{g} =\ sign(g).|\hat{g}| \tag{6.3-6}$$

6.3.3.2.4 Residual signal decoding

If the residual coding is employed, i.e. for bit-rate at and higher than 32kbps, the residual signal is decoded for the low frequencies.

First the global gain ggi is read on 7 bits, If the global gain index is 127, all the residual signal vector is set to zero Otherwise, the global gain is first decoded as:

$$gg = dequantize\_gain(gg) = 10^{\frac{90}{20\cdot 127}ggi}, \text{ for } ggi \in \{0,\dots,126\} \quad (6.3\text{-}7)$$

The residual vector is split into 8-diemnsion blocks, and for each of them a coding parameter is read by entropy decoding using a probability model associated to the bit-rate.

Entropy decoding for param[k]=0 of a block starts by decoding *nz_count*, the number of nonzero values of ±1, with raw coding using 2 bits. Then, the nonzero mask is decoded, which contains *nz_count* ones and *blk_length-nz_count* zeros, with raw coding of the sign bits of the nonzero values.

```
decode_low_entropy_block(block, blk_length)
{
  nz_count = rc_uni_dec_read_bits(2)
  left_1 = nz_count
  left_0 = blk_length - nz_count

  for (i = 0; i < blk_length; i++)
  {
     if (left_1 == 0)
     {
        sym = 0
     }
     else if (left_0 == 0)
     {
        sym = 1
     }
     else
     {
        count0 = left_0 * ECSQ_tab_inverse[left_0 + left_1]
        sym = rc_uni_dec_read_bit_prob_fast(count0, 14)
     }

     if (sym != 0)
     {
        sym *= 1 - 2 * rc_uni_dec_read_bit() /* apply the sign */
        left_1--
     }
     else
     {
        left_0--
     }

     block[i] = sym
  }
}
```

Entropy decoding for $param[k] \geq 1$ of a block starts by computing $shift = \max(0, param[k] - 3)$, which represents the number of least significant bits of the absolute values that are coded approximately uniformly or with raw coding. The most significant bits of each absolute value are decoded using a probability model selected by $param[k]$ together with escape coding. For $shift \leq 4$, decoding of LSBs takes into account that for absolute values the probability of zero (0 maps to one value) is half of the probability of nonzeros (1 maps to two values, ± 1). For larger shifts, the length difference is negligible and raw coding is used using $shift$ bits. For nonzero values, the sign bits are decoded using raw coding.

```
decode_normal_block(block, blk_length, param)
{
  shift = max(0, param - 3)

  for (i = 0; i < blk_length; i++)
  {
    sym = arith_decode_prob_escape(ECSQ_tab_vals[param - 1], 16)

    if (shift != 0)
    {
      if ((sym > 0) || (shift > 4))
      {
        lsbs = rc_uni_dec_read_bits(shift)
      }
      else /* (sym == 0) && (shift <= 4) */
      {
        lsbs = rc_uni_dec_read_symbol_fast(ECSQ_tab_abs_lsbs[shift], 1 << shift, 14)
      }
      sym = (sym << shift) | lsbs
    }
    if (sym != 0)
    {
      sym *= 1 - 2 * rc_uni_dec_read_bit() /* apply the sign */
    }

    block[i] = sym
  }
}
```

The decoding of the entire quantized q_input vector can be expressed in terms of the previous two functions, which decode low entropy blocks and normal blocks, after decoding their $param[k]$.

```
decode_raw_vector(q_input, N)
{
  block_cnt = (N + 7) / 8

  for (k = 0; k < block_cnt; k++)
  {
    blk_length[k] = min(8, N - 8 * k)
    param[k] = rc_uni_dec_read_symbol_fast(ECSQ_tab_param, 16, 14)

    if (param[k] == 0)
    {
      decode_low_entropy_block(block[k], blk_length[k])
    }
    else
    {
      decode_normal_block(block[k], blk_length[k], param[k])
    }

    for (i = 0; i < blk_length[k]; i++)
    {
      q_input[8 * k + i] = block[k][i]
    }
  }
}
```

The decoded residual signal q_input is then scaled by the decoded global gain gg. The resulting decoded residual signal is then transformed back to the time domain using an inverse MDCT of size 20ms with an overlapping of 8.75ms using sine windowing. The time domain decoded residual signal is transformed to the DFT domain using the same STFT analysis used for the decoder downmix. The resulting signal is then $\widehat{Res}_i[k]$ and used in the subsequent stereo upmixing. In case the residual coding is disabled, the spectrum $\widehat{Res}_i[k]$ is set to zero.

6.3.3.2.5 Low-cost adaptation of bass post filter

The bass post-filter introduced for EVS in 6.1.4.2 [3] adapts the post-filter strength $\alpha \in [0, 0.5]$ to how well the post-filtered signal correlates with the input signal. A low correlation suggests the filter may have a degrading impact, and as a result the filter output is attenuated. The post-filter strength is also adapted to the LP filter stability, where a low stability leads to an attenuated filter. In EVS operations the post filter strength α and the fundamental pitch period in samples T are both updated each subframe. To prevent the abrupt changes at the subframe boundaries, an adaptation mechanism is introduced for IVAS operations.

The adaptation mechanism includes a post-filter adaptation module to decide whether to apply the post filter or not based on the energy discontinuities in the signal between consecutive subframes. The input to the adaptation module consists of the reconstructed frequency domain signal $\hat{S}$ generated by the decoder and the post-filter difference signal s_{diff} which describes the difference between the filtered and non-filtered signal in time domain.

Adaptation module first measures the energy of the spectrum $E_{\hat{S}_{cb}}$ in the critical frequency band $[k_{start}, k_{end}]$ for each frame:

$$E_{\hat{S}_{cb}} = \frac{1}{k_{end}-k_{start}+1}\sum_{k_{start}}^{k_{end}} \hat{S}(k)^2 \qquad (6.3\text{-}8)$$

The frequency bin limits k_{start} and k_{end} are set to match the frequency range of the critical band. If the MDCT frame length is equal to $N_{MDCT} = 160$, the sampling rate is 8000 Hz and the critical frequency range is 1000 Hz – 1600 Hz, hence $k_{start} = 39$ and $k_{end} = 64$. Otherwise, the limits are selected as such $k_{start} = 28$ and $k_{end} = 40$.

The calculated energy $E_{\hat{S}_{cb}}$ is then subject to low pass filtering to mimic the overlap-add in the synthesis of $\hat{S}$ with:

$$\tilde{E}_{\hat{s}_{cb}} = \gamma E_{\hat{s}_{cb}} + (1-\gamma) E_{\hat{s}_{cb}}^{[-1]} \tag{6.3-9}$$

where the value of low pass filter coefficient γ is equal to $\gamma = 0.61$.

The size of discontinuities is measured by averaging the step at the subframe boundaries of the filter difference signal s_{diff}. The filter difference signal is defined as the impact of the filter expressed as a negative difference signal or a correction signal or an error signal, $s_{diff} = \alpha(\hat{s} - s_p)$ where s_p is the post-filter output. The average energy of the step at the subframe boundaries $\tilde{E}_{step}$ is then:

$$\tilde{E}_{step} = \frac{1}{N_{sf}} \sum_{i=1}^{N_{sf}} \left(s_{diff}(n_i) - s_{diff}(n_i - 1) \right)^2 \tag{6.3-10}$$

where i denotes the subframe number, N_{sf} is the number of subframes and equals to $N_{sf} = 5$. The subframe indices of the subframe boundaries $n_1, n_2, \ldots, n_{N_{sf}}$ mark the start of each new subframe. A frame length of $N = 160$ would mean that $n_1 = 0, n_2 = 32, n_3 = 64, n_4 = 96, n_5 = 128$. The first sample $n_1 = 0$ in this case refers to the last sample of the previous frame $s_{diff}^{[-1]}(N-1)$.

The ratio between $\tilde{E}_{step}$ and $\tilde{E}_{\hat{s}_{cb}}$ is then calculated to provide a decision variable, $\tilde{E}_{ratio}$:

$$\tilde{E}_{ratio} = \frac{\tilde{E}_{step}}{\tilde{E}_{\hat{s}_{cb}}} \tag{6.3-11}$$

The range of the decision variable $\tilde{E}_{ratio}$ is limited with an upper limit of $E_{ratio,lim} = 2$ in the following expression:

$$\tilde{E}_{ratio,1} = \begin{cases} \tilde{E}_{ratio}, & \tilde{E}_{ratio} \le \tilde{E}_{ratio,lim} \\ \tilde{E}_{ratio,lim}, & \tilde{E}_{ratio} > \tilde{E}_{ratio,lim} \end{cases} \tag{6.3-12}$$

Following that $\tilde{E}_{ratio,1}$ is low pass filtered with a filter coefficient of $\beta = 0.68$:

$$\tilde{E}_{ratio,LP} = \beta \tilde{E}_{ratio,1} + (1-\beta) \tilde{E}_{ratio,LP}^{[-1]} \tag{6.3-13}$$

Finally, the decision to activate the post-filter is taken by subjecting the resulting $\tilde{E}_{ratio,LP}$ value to the threshold $E_{thr} = 1$:

$$D_{bpf} = \begin{cases} active, & \tilde{E}_{ratio,LP} < E_{thr} \\ inactive, & \tilde{E}_{ratio,LP} \ge E_{thr} \end{cases} \tag{6.3-14}$$

As a result, if $D_{bpf} = active$, the post-filter is applied to the reconstructed signal. When $D_{bpf} = inactive$, the output is the reconstructed signal.

6.3.3.2.6 Residual predicted signal generation

6.3.3.2.6.1 Stereo filling

Stereo filling technology is used to estimate the component of the side channel being uncorrelated to the mid channel. This component is also called the stereo residual. The stereo filling is controlled by a control parameter, the residual prediction gain, received from the encoder, provided to a decorrelator for adaptive adjustment of the amount of decorrelation. The parameter is obtained as described in clause 5.3.3.2.2.8 and relates to the performance of the parametric description of the input audio. The prediction gain is low when the input audio signal is well represented by the parametric stereo representation, and high when there are uncorrelated components that are not fully captured by the stereo model, i.e., there is a stereo residual component.

Depending on the bitrate, stereo residual coding or various stereo filling modes are utilized, as shown in table 6.3-1.

Table 6.3-1: Stereo residual modes

	13.2 kbps	16.4 kbps	24.4 kbps	32 kbps
Residual coding	-	-	-	0-1 kHz
Enhanced stereo filling	0-6.4 kHz	0-6.4 kHz	0-8 kHz	-
FD stereo filling	> 6.4 kHz (TCX/HQ)	> 6.4 kHz (TCX/HQ)	> 8 kHz (TCX/HQ)	1-8 kHz, > 8 kHz (TCX/HQ)
TD stereo filling	> 6.4 kHz (ACELP)	> 6.4 kHz (ACELP)	> 8 kHz (ACELP)	> 8 kHz (ACELP)

At 32 kbps, the residual is encoded for the lowest frequency bands up to frequency band $b_{res\_cod\_max} - 1$, see 6.3.3.2.4.

Stereo filling is applied up to at most 16 kHz (SWB), which corresponds to at most 13 frequency bands, i.e.

$$b_{respred} = \min(N_b, 12).$$

where N_b is the number of decoding frequency bands.

Depending on the stereo filling mode, two different methods are applied for low frequency bands $\{b_{res\_cod\_max}, \dots, b_0\}$ where b_0 corresponds to the frequency band not having any coefficient $k > k_0$ with

$$k_0 = \left\lfloor \frac{N_{FFT}}{2} + 0.5 \right\rfloor,$$

where N_{FFT} denotes the number of frequency coefficients and $\lfloor\cdot\rfloor$ denotes the floor function. For the higher bitrate, 32 kbps, low-complex FD stereo filling is applied for these lower frequency bands. For the lower bitrates, up to 24.4 kbps, enhanced stereo filling is applied, see clause 6.3.3.2.6.4.

For the higher frequency bands $\{b_0, \dots, b_{respred}\}$ the low-complex FD stereo filling is applied for the TCX/HQ core codec mode (see 6.3.3.2.6.2). For high-band ACELP, a separate time-domain stereo filling is applied (see 6.3.3.2.6.5). The transitions between the modes are handled specially, see clause 6.3.3.2.6.2 and 6.3.3.2.6.5.3.

6.3.3.2.6.2 FD stereo filling

The FD stereo filling consists of a low-complex decorrelation of the downmix signal S_{dmx} by frequency-dependent adaptive mixing of downmix signals of past frames.

A short delay offset index i_{short} is obtained as

$$i_{short} = D_{MAX} - D_{short} + b \bmod 2 \;,$$

where $D_{MAX} = 4$ and $D_{short} = 2$. The modulo operation mod makes the delay index offset i_{short} vary by one per consecutive frequency bands. To get the optimal decorrelation parameter per frequency band, the filter length is adapted per band such that a longer delay is set for the lower frequency bands and a shorter delay for the higher frequency bands. A long delay index offset i_{long} is obtained as

$$i_{long} = \max\left(4, \left\lfloor (\mathrm{D_{MAX}} + 4 - 1)\frac{b}{N_{bands} - 1} + 0.5 \right\rfloor\right) - 4 \qquad (6.3\text{-}18)$$

To make sure i_{short} is indeed corresponds to a shorter or equal delay:

$$i_{short} := \max(i_{short}, i_{long})$$

As each frame consists of two subframes, even indices i_{short} and i_{long} are obtained as

$$i_{short} := 2\left\lfloor \frac{i_{short}}{2} \right\rfloor$$
$$i_{long} := 2\left\lfloor \frac{i_{long}}{2} \right\rfloor$$

Corresponding delays d_{short} and d_{long} are obtained as

$$d_{short} = D_{MAX} - i_{short}$$
$$d_{long} = D_{MAX} - i_{long}$$

Finally, offset indices to use for the stereo filling are obtained as

$$i_{short} := \left(i_{short} + i_{past} + 1\right) \bmod D_{MAX}$$
$$i_{long} := \left(i_{long} + i_{past} + 1\right) \bmod D_{MAX}$$

where i_{past} corresponds to the latest index of an array in which the downmix signals has been stored.

In addition, corresponding gains g_{short} and g_{long} are obtained based on a determined target delay for the stereo filling as described in clause 6.3.3.2.6.3. However, if the delays d_{short} and d_{long} are equal,

$$g_{short} = 1$$
$$g_{long} = 0.$$

Further, transients are avoided in the stereo filling, by setting

$$g_{short} = 0 \quad if \quad g_p(i_{short}, b) = -1$$
$$g_{long} = 0 \quad if \quad g_p\left(i_{long}, b\right) = -1$$

where $g_p(i, b)$ denotes the prediction gain of a previous frame of index i in an array of previous prediction gains. $g_p(i, b)$ is set to -1 for earlier frames having $attackPresent$ or $wasTransient$ flags set, otherwise it reflects the prediction gain for the corresponding frequency band b.

If both gains g_{short} and g_{long} are zero, the stereo filling component of $S_{DMX}(k)$ is set to zero. Otherwise, the decorrelator is adaptively adjusted to generate the stereo filling, estimating the component of the side channel being uncorrelated to the downmix signal, as

$$S_{DMX}(k) = g_2(m, b)\, S_{AVG}(k) \qquad\qquad \forall k \in k_b$$

where k_b denotes the frequency indices of frequency band b, and the weighted average $S_{AVG}(k, n)$ is

$$S_{AVG}(k)= g_{short}S_{DMX\_past}(i_{short}, k) + g_{long}S_{DMX\_past}\left(i_{long}, k\right)$$

The decorrelation signal strength g_2, controlling the amount of decorrelated signal in the stereo synthesis, is determined by calculation of

$$g_2(m, b) = \min\left(G_{amp}\, \overline{g_p}, (1 - G_{rest})\, \overline{g_p} + G_{rest}\, g_p^*(m, b)\ \right)$$

where $G_{amp} = 0.6$ and $\overline{g_p}$ is obtained as a weighted average

$$\overline{g_p} = g_{short}\, g_p(i_{short}, b) + g_{long}\, g_p(i_{long}, b), \text{ and}$$

$$g_p^*(m, b) = f\, g_p(m, b).$$

with

$$f = \sqrt{\frac{E_{DMX}+0.001}{E_{DMX\_past}+0.001}}$$

corresponding to the ratio between the energy of the current downmix spectrum $S_{DMX}(k, n)$ and averaged previous downmix signals $S_{AVG}(k, n)$ as obtained from equation (6.3-27). The energies are obtained as

$$E_{DMX}(m, b) = \sum_{k=k_b} S_{DMX}^2(k, n), \text{ and}$$

$$E_{DMX\_past}(m, b) = \sum_{k=k\_b} S_{AVG}^2(k, n),$$

where k_b is the set of frequency coefficients in band b, excluding any coefficient $k > k_0$.

For the higher frequency bands $\{b_0, \dots, b_{respred}\}$, in TCX/HQ core codec mode, stereo filling components are obtained in accordance with equation (6.3-25). However, for frames where ACELP was used, the gains are set to zero, i.e.

$$g_{short} = 0 \quad if \quad core\_hist\left(\frac{d_{short}}{2}\right) = ACELP$$
$$g_{long} = 0 \quad if \quad core\_hist\left(\frac{d_{short}}{2}\right) = ACELP$$

where $core\_hist$ is an array containing information of what core codec mode has been used.

In transitions from ACELP to TCX/HQ mode, the high band energy $E_{hb}(m)$ is updated as

$$E_{hb}(m) = E_{hb}(m-1) + \frac{2}{N_{FFT}} E_{DMX}(m,b), \tag{6.3-34}$$

where $E_{DMX}(m,b)$ is also covering all the coefficients up to $N_{FFT}/2$.

In the transition from TCX/HQ core codec mode to ACELP, a fixed short delay index i_{short} is determined as

$$i_{short} = D_{MAX} - D_{sf} \tag{6.3-35}$$

with $D_{sf} = 2$. The delay offset index i_{short} is then obtained as given in equation (6.3-22). Stereo filling components are obtained as in equation (6.3-26) with $g_{long} = 0$ and with g_{short} determined as

$$\begin{matrix} g_{short} := \frac{1+g_{short}}{2} & if \quad g_{short} \geq 0 \\ g_{short} := 0 & if \quad g_{short} < 0 \end{matrix} \tag{6.3-36}$$

6.3.3.2.6.3 Determination of stereo filling target delay

A target delay for the stereo filling is determined as basis for determining the gains g_{short} and g_{long}. The sum and maximum of the prediction gains over the frequency bands are determined as

$$max\_pg = \max_b g_p(b) \tag{6.3-37}$$

$$sum\_pg = \sum_b g_p(b) \tag{6.3-38}$$

Then, if $sum\_pg > 0$, the mean and variation of the prediction gain control parameter are estimated by calculation of

$$\overline{g_p} := \alpha_M \, sum\_pg + (1 - \alpha_M)\, \overline{g_p} \tag{6.3-39}$$

$$\widetilde{g_p} := \alpha_V \, var\_pg + (1 - \alpha_V)\, \widetilde{g_p} \tag{6.3-40}$$

where $\alpha_M = 0.1$ and α_V=0.1, and the variation of the prediction gain is computed as

$$var\_pg = |sum\_pg - lt\_pred\_gain|. \tag{6.3-41}$$

If $\overline{g_p} > 0$, a ratio of the variation and mean of the control parameter is determined as

$$\hat{g}_p = \min\left(1.5 * L_{ratio}, \frac{\widetilde{g_p}}{\overline{g_p}}\right), \tag{6.3-42}$$

where L_{ratio}=0.18.

A long-term variation to mean ratio is computed as

$$\begin{matrix} \hat{g}_{p,LT} := \alpha_{up}\, \hat{g}_p + (1 - \alpha_{up})\hat{g}_{p,LT} & if \quad \hat{g}_p > \hat{g}_{p,LT} \\ \hat{g}_{p,LT} := \alpha_{down}\, \hat{g}_p + (1 - \alpha_{down})\hat{g}_{p,LT} & else \end{matrix} \tag{6.3-43}$$

where α_{up} and α_{down} are determined based on the max peak gain of equation (6.3-37) as

$$\begin{matrix} \begin{cases} \alpha_{up} = 0.03 \\ \alpha_{down} = 0.05 \end{cases} & if \quad max\_pg > T_{pg} \\ \begin{cases} \alpha_{up} = 0.1 \\ \alpha_{down} = 0.001 \end{cases} & else \end{matrix} \tag{6.3-44}$$

The peak gain threshold $T_{pg} = 0.6$. Subsequently, a decorrelation parameter representing the targeted decorrelation filter length d_{target} is calculated as

$$d_{target} = D_{short} \qquad if \quad \hat{g}_{p,LT} \geq L_{ratio}$$
$$d_{target} = \min\left(D_{long}, D_{short} + \left(D_{offset} + D_{long} - D_{short}\right)\left(1 - \hat{g}_{p,LT}/L_{ratio}\right)\right) \qquad else \tag{6.3-45}$$

where $D_{short} = 2$ and D_{long}=4 are filter lengths of a short and a long decorrelation filter, $D_{offset} = 2$. The short and long stereo filling gains are obtained based on the determined target gain d_{target} as

$$g_{short} = \frac{D_{long}-d_{target}}{D_{long}-D_{short}} \tag{6.3-46}$$

$$g_{long} = \sqrt{1 - g_{short}^2} \tag{6.3-47}$$

6.3.3.2.6.4 Enhanced Stereo Filling

For replacing residual parts in the stereo upmix of the decoded base downmix channel for bitrates below 32 kbps, a special filling signal is generated for the lowband (i.e. all frequencies up to the core sampling rate, which is either 12.8 or 16 kHz). This so-called Enhanced Stereo Filling (ESF) signal is generated by filtering the core downmix channel (sampled at the core sampling frequency) with an allpass-based decorrelation filter. Since it is applied in time-domain the filtering is a broadband processing of the decoded core downmix signal. The ESF signal is subsequently transformed to DFT domain along with the downmix signal. In the spectral domain the two upmix channels are determined by combining weighted representations of the downmix signal and the ESF signal. The weighting and upmixing is thereby done in a band-wise manner using the energies of the two signals and the transmitted and dequantized residual prediction gain parameter in the respective stereo band.

In order to obtain the desired stereo filling signal a special allpass filter is applied to the downmix channel. This filter is designed to have short, dense impulse responses which is achieved by applying multiple stages of basic allpass filters where each stage consists of cascaded Schroeder allpass filters nested into a third Schroeder filter, i.e.

$$B(z) = H\left(\left(z^{-d_3}S(z)\right)^{-1}\right), \tag{6.3-48}$$

where

$$S(z) = \frac{g_1+z^{-d_1}}{1-g_1z^{-d_1}}\frac{g_2+z^{-d_2}}{1-g_2z^{-d_2}} \tag{6.3-49}$$

and

$$H(z) = \frac{g_3+z^{-1}}{1-g_3z^{-1}}. \tag{6.3-50}$$

The design of such a basic allpass filter unit is illustrated in Figure 6.3-2.

The delays d_i and the gains g_i used in each stage are chosen to be rather small to avoid an overly reverberant filling signal. To further create dense and random-like impulse responses the delays are also selected to be mutually prime for all allpass filters.

With the input signal to the filter always sampled at the core sampling rate, which is either 12.8 kHz or 16 kHz for all DFT Stereo modes, a filter that was found to give suitable results for both possible sampling rates is

$$F(z) = \prod_{i=1}^{3} B_i(z) \tag{6.3-51}$$

where B_i are the cascaded basic allpass filters units with gains and delays as shown in Table 6.3-2.

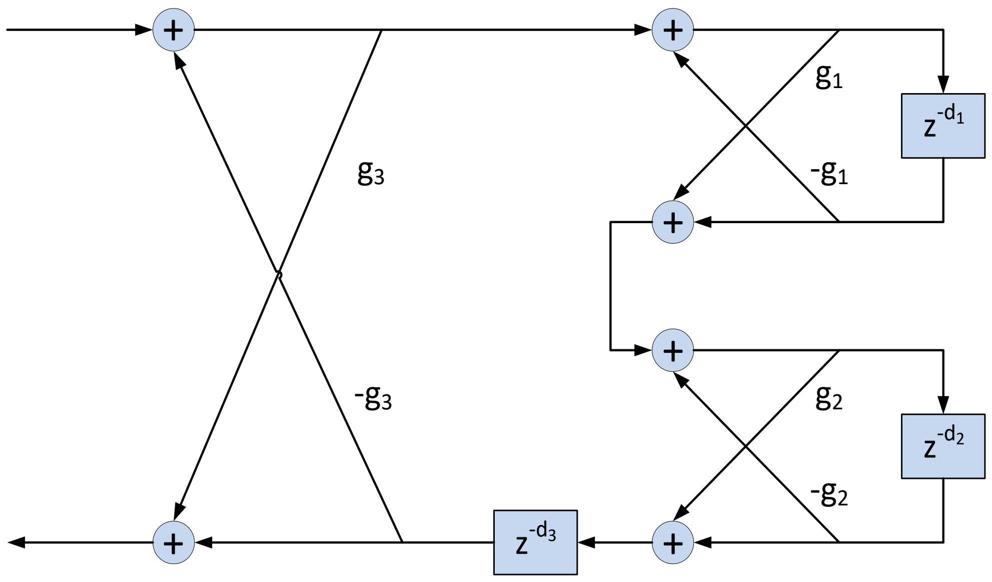


Figure 6.3-2: Basic allpass filter unit

Table 6.3-2: Allpass delays and gains

Filter	g_1	d_1	g_2	d_2	g_3	d_3
$\boldsymbol{B_1(z)}$	0.5	2	-0.2	47	0.5	61
$\boldsymbol{B_2(z)}$	-0.4	29	0.2	41	-0.5	73
$\boldsymbol{B_3(z)}$	0.4	31	-0.3	37	0.5	59

To avoid smeared transients when applying the enhanced stereo filling signal during stereo upmix, zeroes are fed to the allpass filter units in frames where an attack was detected by the transient detector on encoder side [reference?] and this control information is available at the decoder. For smooth transitions between regular input and zeroed transient frames, a small linear fade over 32 (at 12.8 kHz) or 40 samples (at 16 kHz) is done in both directions.

After generating the allpass-filtered filling signal in time-domain, the signal is brought to DFT domain via STFT analysis (see 6.3.3.2.2). As part of the transform the signal is also upsampled from the core sampling rate to the output sampling rate to align it with the transformed downmix signal.

In DFT domain, the ESF signal now substitutes the missing residual signal $\hat{\rho}$ for all bands b that are part of the low-band and is used together with the bandwise residual gains $\hat{r}_b$ and the energy-adjusting factor $g_{norm,b}$ during the stereo upmix (see 6.2.2.2.4).

The energy-adjusting gain $g_{norm,b}$ is calculated from the energies of downmix signal $\widehat{M}_{t,b}$ and residual signal $\hat{\rho}_{t,b}$ as

$$g_{norm,b} = \sqrt{\frac{E_{\widehat{M}_{t,b}}}{E_{\hat{\rho}_{t,b}}}} \qquad (6.3\text{-}52)$$

where the energies are calculated for each band as

$$E_{\widehat{M}_{t,b}} = \sum_{k\in I_b} \left|\widehat{M}_{t,k}\right|^2 \qquad (6.3\text{-}53)$$

and

$$E_{\hat{\rho}_{t,b}} = \sum_{k\in I_b} \left|\hat{\rho}_{t,k}\right|^2. \qquad (6.3\text{-}54)$$

The energies are further smoothed using previous energies $E_{\widehat{M}_{t-1,b}}$ and $E_{\hat{\rho}_{t-1,b}}$ and smoothing factor α as

$$E_{\widehat{M}_{t,b}} \leftarrow \alpha\, E_{\widehat{M}_{t-1,b}} + (1-\alpha)\, E_{\widehat{M}_{t,b}} \qquad (6.3\text{-}55)$$

and

$$E_{\hat{\rho}_{t,b}} \leftarrow \alpha\, E_{\hat{\rho}_{t-1,b}} + (1-\alpha)\, E_{\hat{\rho}_{t,b}} \tag{6.3-56}$$

where α is set to 0 directly after transient frames, i.e. no smoothing, and to 0.2 for all other frames.

To obtain a smoother output and to partially compensate the ambience loss induced by low-rate coding, a compressor is applied to the energy-adjusting gain g_{norm} which compresses the values towards one.

The compression is done via

$$\tilde{g}_{norm,b} = \exp\left(f\left(\log\left(g_{norm,b}\right)\right)\right), \tag{6.3-57}$$

where $f(t)$ is a non-linear function defined as

$$f(t) = t - \int_0^t c(\tau)d\tau \tag{6.3-58}$$

and the function c satisfies

$$0 \le c(t) \le 1. \tag{6.3-59}$$

The value of c around t then specifies how strongly this region is compressed, where the value 0 corresponds to no compression and the value 1 corresponds to total compression. Furthermore, the compression scheme is symmetric if c is even, i.e. $c(t) = c(-t)$.

Concretely, $c(t)$ is defined as

$$c(t) = \begin{cases} 1 & -\alpha < t < \alpha, \\ 0 & else, \end{cases} \tag{6.3-60}$$

which gives rise to

$$f(t) = t - max\{min\{\alpha, t\}, -\alpha\}. \tag{6.3-61}$$

This simplifies the calculation of $\tilde{g}_{norm}$ to

$$\tilde{g}_{norm} = g_{norm,b} min\left\{max\left\{\exp(-\alpha), \frac{1}{g_{norm,b}}\right\}, \exp(\alpha)\right\}. \tag{6.3-62}$$

With α chosen as $\log(1.25)$ the final compression is given as

$$\tilde{g}_{norm,b} = g_{norm,b} min\left\{max\left\{0.8, \frac{1}{g_{norm,b}}\right\}, 1.25\right\}. \tag{6.3-63}$$

In all bands of the low-band where ESF is used, this compressed value $\tilde{g}_{norm,b}$ is then used for energy-normalization of the ESF signal.

6.3.3.2.6.5 TD stereo filling

6.3.3.2.6.5.1 General

For ACELP frames the bandwidth extension (as described in EVS [3] Clause 5.2.6.1 and EVS [3] Clause 5.2.6.2) is not transformed to DFT domain due to delay reasons. Therefore, the upmix to stereo is done in time-domain via Inter-Channel Bandwidth Extension (IC-BWE) [reference IC-BWE chapter here]. IC-BWE mainly aims at restoring correct panning in the bandwidth extension range. For also restoring ambience by adding a residual prediction signal to the ACELP highband, two different methods are used depending on whether the current frame is following another ACELP frame before or is a transition frame directly after a TCX or HQ frame.

For both cases, the high-band energy of the current ACELP frame is required. This can be easily obtained by calculating it on the high-band signal in time-domain before stereo upmix.

6.3.3.2.6.5.2 Regular ACELP frame: calculation of average prediction gain and addition in time-domain

For non-transition ACELP frames, the stereo filling is done entirely in time-domain and can be seen as an extension of IC-BWE. This requires saving the high-band signal of the previous frame which will be used as the stereo filling signal. For energy normalization of the filling signal, a single normalization gain for the complete bandwidth extension range has to be calculated from the energies $E_{\widehat{m},HB}$ and $E_{\widehat{m}_{SF}}$ of the HB of input signal $\widehat{m}$ and the TD stereo filling signal $\widehat{m}_{SF}$, as well as the bandwise residual prediction parameters of the current and previous frame $\hat{r}_{t,b}$ and $\hat{r}_{t-1,b}$ for all subbands b inside the bandwidth extension range.

From the HB prediction gains weighted average gains are computed as

$$\bar{r}_t = \frac{\sum_{b\in HB} \hat{r}_{t,b} w_b}{\sum_{b\in HB} w_b} \tag{6.3-64}$$

and

$$\bar{r}_{t-1} = \frac{\sum_{b\in HB} \hat{r}_{t-1,b} w_b}{\sum_{b\in HB} w_b} \tag{6.3-65}$$

where w_b are bandwise weighting factors depending on the band resolution (see Table 5.3-5 in clause 5.3.3.2.2.6). The weighting factors are given as defined in Table 6.3-3 and Table 6.3-4.

Table 6.3-3: Residual gain weighting ERB 4 scale

b	9	10	11	12
w_b	1	1	1	1

Table 6.3-4: Residual gain weighting ERB 8 scale

b	4	5	6
w_b	1	0.5	0.4

Using $\bar{r}_t$ and $\bar{r}_{t-1}$ a normalized gain $\bar{r}_{norm,HB}$ is calculated as

$$\bar{r}_{norm,HB} = \min\left(\bar{r}_{t-1}, 0.4\,\bar{r}_{t-1} + 0.6\,\bar{r}_t \sqrt{\frac{E_{\widehat{m},HB}}{E_{\widehat{m}_{SF}}}}\right). \tag{6.3-66}$$

Finally, the TD stereo filling signal $\widehat{m}_{SF}$ is added to the output signal after stereo upmix of the HB:

$$l \leftarrow l + \bar{r}_{norm,HB}\widehat{m}_{SF}, \tag{6.3-67}$$

$$r \leftarrow r - \bar{r}_{norm,HB}\widehat{m}_{SF}. \tag{6.3-68}$$

Since $\bar{g}_{norm,HB}$ was derived from residual prediction gains that were calculated in DFT domain and therefore from a windowed signal, it has faded with the gain from the previous frame in the DFT overlap region:

$$\bar{r}_{norm,HB} = win_{DFT}\bar{r}_{norm,t,HB} + (1 - win_{DFT})\bar{r}_{norm,t-1,HB} \tag{6.3-69}$$

6.3.3.2.6.5.3 TCX/HQ→ACELP transition frame: bandwise stereo filling in frequency-domain

For transition frames where the current ACELP frame was preceded by a TCX or HQ frame, the stereo filling is done in a bandwise manner in frequency domain, instead. In this case, an FD stereo filling signal is available and the normalization and addition of the filling signal can, in general, be done the same as for regular FD stereo filling in the high-band (as described in clause 6.3.3.2.6.2).

However, no bandwise energies are available for current HB signal. Therefore, only one normalization factor is calculated as

$$g_{norm} = \sqrt{\frac{E_{\widehat{m},HB}}{E_{\widehat{m}_{SF},HB}}} \qquad (6.3\text{-}70)$$

where $E_{\widehat{m},HB}$ is the energy of current HB signal calculated in time-domain and $E_{\widehat{m}_{SF},HB}$ the energy of the stereo filling signal. This single normalization factor is applied for all bands in the high-band. The normalized residual gain thus becomes

$$\hat{r}_{norm,b} = g_{norm}\hat{r}_b. \qquad (6.3\text{-}71)$$

6.3.3.2.7 Stereo upmix

The upmix is done in frequency domain. using the dequantized stereo parameters are calculated as:

$$\hat{L}_i[k] = \widehat{M}_i[k].(1+\hat{g}_i[b]) + \hat{r}_i[b].g_{norm}.\hat{\rho}_i[k] + \widehat{Res}_i[k] \qquad (6.3\text{-}72)$$

and

$$\hat{R}_i[k] = \widehat{M}_i[k].(1-\hat{g}_i[b]) - \hat{r}_i[b].g_{norm}.\hat{\rho}_i[k] - \widehat{Res}_i[k] \qquad (6.3\text{-}73)$$

for k ∈ I_b, where $\hat{\rho}_i[b]$ is a substitute when the decoded residual signal $\widehat{Res}_i[k]$ is missing, and g_{norm} is the energy adjusting factor:

$$g_{norm} = \sqrt{\frac{E_{\widehat{M}_i[k]}}{E_{\hat{\rho}_i[k]}}} \qquad (6.3\text{-}74)$$

The channels are then rotated for reinjecting the global IPD:

$$\begin{cases} \hat{L}_i[k] = \hat{L}_i[k].e^{j2\pi.\widehat{gipd}} \\ \hat{R}_i[k] = \hat{R}_i[k] \end{cases} \qquad (6.3\text{-}75)$$

The left and right channel waveforms are then generated by applying STFT synthesis to the reconstructed left and right spectra.

6.3.3.2.8 STFT synthesis

From the reconstructed left and right spectra $\hat{L}_i[k]$ and $\hat{R}_i[k]$, stereo time domain signal is synthesized by the inverse DFTs:

$$\hat{l}_i[n] = \sum_{k=0}^{N-1} \hat{L}_i[k] \cdot e^{\frac{2\pi jkn}{N}}, \text{ for } 0 \le n < N, \qquad (6.3\text{-}76)$$

and

$$\hat{r}_i[n] = \sum_{k=0}^{N-1} \hat{R}_i[k] \cdot e^{\frac{2\pi jkn}{N}}, \text{ for } 0 \le n < N. \qquad (6.3\text{-}77)$$

Finally, windowing and overlap-adding allow reconstructing a frame of 2.*M* samples for the two channels, by computing the following equalities at every even DFT index *i*:

$$\hat{l}'[(i/2)\cdot 2.\mathrm{M}+n-zp] = \begin{cases} \hat{l}_{i-2}[2M-zp-L1+n]\cdot w_{dec}[1,2M-zp-L1+n] + \hat{l}_{i-1}[n]\cdot w_{dec}[0,n], \text{for } zp \le n < zp+L1 \\ \hat{l}_{i-1}[n] \quad \text{, for } zp+L1 \le n < zp+(2M-L1-L2)/2 \\ \hat{l}_{i-1}[n]\cdot w_{dec}[0,n] + \hat{l}_i[\alpha+n]\cdot w_{dec}[1,\alpha+n], \text{for } zp+(2M-L1-L2)/2 \le n < zp+(2M-L1+L2)/2 \\ \hat{l}_i[\alpha+n] \quad \text{, for } zp+(2M-L1+L2)/2 \le n < zp+2M-L1 \end{cases}$$

(6.3-78)

and

$$\hat{r}'[(i/2)\cdot 2.\mathrm{M}+n-zp] = \begin{cases} \hat{r}_{i-2}[2M-zp-L1+n]\cdot w_{dec}[1,2M-zp-L1+n]+\hat{r}_{i-1}[n]\cdot w_{dec}[0,n], \text{for } zp \le n < zp+L1 \\ \hat{r}_{i-1}[n] \quad ,\text{for } zp+L1 \le n < zp+(2M-L1-L2)/2 \\ \hat{r}_{i-1}[n]\cdot w_{dec}[0,n]+\hat{r}_i[\alpha+n]\cdot w_{dec}[1,\alpha+n], \text{for } zp+(2M-L1-L2)/2 \le n < zp+(2M-L1+L2)/2 \\ \hat{r}_i[\alpha+n] \quad ,\text{for } zp+(2M-L1+L2)/2 \le n < zp+2M-L1 \end{cases} \quad (6.3\text{-}79)$$

where $\alpha = -zp - L1 - (2M - L1 - L2/2)$ is the offset between the beginning points of the 2 windows within a frame of *2M* samples. The synthesis windows are the same as the analysis windows at the decoder side.

6.3.3.2.9 Parametric stereo decoding

6.3.3.2.10 Side prediction residual decoding

6.3.3.2.10.1 Enhanced stereo filling

At the lower bitrates, where coding of the side channel residual is not applied, there is an enhanced stereo filling technology for the lower frequencies.

[To be completed]

6.3.3.2.11 DFT-based stereo PLC

6.3.3.2.11.1 General

In case of a packet loss, the PLC operation is activated for the DFT stereo. The decoded down-mix signal is retrieved by running the down-mix decoder PLC to retrieve a down-mix PLC frame $\hat{s}_M(n)$. A DFT representation of the down-mix signal $\hat{S}_M(k)$ is obtained in the same way as in error-free decoding [xref previous clause]. The stereo parameters that was decoded in the previous frame is generally reused together with $\hat{s}_M(n)$ in the same was as in 6.3.3.2.9, with two additions: the side gain parameters is reconstructed as described in 6.3.3.2.11.2 and the side gain prediction residual (if present) is reconstructed as described in 6.3.3.2.11.3. The reconstructed down-mix PLC frame $\hat{S}_M(k)$ is used together with the reconstructed parameters and the reconstructed side prediction residual $\hat{S}_R(k)$ to perform a DFT stereo synthesis as described in 6.3.3.2.9.

6.3.3.2.11.2 Side gain recovery after frame loss

During error-free decoding of active frames, the side gain parameter $\hat{g}_S[b]$ for band b is decoded as described in 6.3.3.2.9. To improve the error recovery after a frame loss, a prediction memory corruption flag $g_{S,CORRUPT}$ is maintained. It is initialized to $FALSE$, and upon reception of a bad frame it is set to $TRUE$. Once a non-predictive frame is received [ref to sg decoding?] the flag is restored to $FALSE$. If $g_{S,CORRUPT} = FALSE$, a low-pass filtered mean of $\hat{g}_S[b]$ is obtained according to

$$\begin{cases} \bar{g}_{S,LP} = 0.425\bar{g}_S + 0.575\bar{g}_{S,LP}^{[-1]}, & |\bar{g}_S| \ge 0.6 \\ \bar{g}_{S,LP} = 0, & |\bar{g}_S| < 0.6 \\ \bar{g}_S = \frac{1}{N_{bands}} \sum_{b=0}^{N_{bands}-1} \hat{g}_S[b] \end{cases} \quad (6.3\text{-}80)$$

where N_{bands} is the number of DFT stereo bands currently in use for $\hat{g}_S[b]$. If $g_{S,CORRUPT} = TRUE$, the value from the previous frame is used.

$$\bar{g}_{S,LP} = \bar{g}_{S,LP}^{[-1]} \quad (6.3\text{-}81)$$

When a bad frame is indicated, the side gain parameter from the previous frame is used $\hat{g}_S[b] = \hat{g}_S^{[-1]}[b]$. If a good frame is received, and after decoding of $\hat{g}_S[b]$ and updating of the $\bar{g}_{S,LP}$ it is found that the following is true

$$\begin{cases} g_{S,CORRUPT} = TRUE \\ \bar{g}_{S,LP} > 0.6 \vee \bar{g}_{S,LP} < 0.6' \end{cases} \quad (6.3\text{-}82)$$

where $\vee$ denotes inclusive or, the decoded side gain of the previous frame is used, $\bar{g}_{S,LP} = \bar{g}_{S,LP}^{[-1]}$. The condition $|\bar{g}_{S,LP}| > 0.6$ indicates that the reconstructed sound source has been stable and concentrated to either the left or the right channel in the recently decoded frames. This is a situation which is sensitive to error-propagation and reusing the previously decoded parameters improves the performance in the state of corrupted memory.

6.3.3.2.11.3 Side prediction residual PLC

In case of a packet loss for a DFT stereo frame where a decoded side prediction residual is present, the PLC operation is activated to produce a concealment frame of the side prediction residual. This is achieved by combining the Phase ECU with the predicted stereo residual obtained by the stereo filling algorithm [xref previous clause]. First, the reconstructed down-mix PLC frame $\hat{S}_M(k)$ is run through the frequency domain decorrelator [xref previous clause] to obtain $\hat{S}_D(k)$. The magnitude of the previously decoded side prediction residual $\hat{S}_R^{[-1]}(k)$ is combined with the phase from $\hat{S}_D(k)$ to retain the correlation property with respect to the down-mix PLC frame $\hat{S}_M(k)$. This could be done by matching the magnitude of $\hat{S}_D(k)$ with the magnitude of $\hat{S}_R^{[-1]}(k)$. A low-complex adjustment is made by matching the order of the absolute values of the real and imaginary part and the signs for each bin of $\hat{S}_R^{[-1]}(k)$ with each bin of $\hat{S}_D(k)$. Following this principle, the phase matched $\tilde{S}_R(k)$ is calculated according to

$$\begin{cases} \tilde{S}_R(k) = a + jb \\ a = sign\left(Re\left(\hat{S}_D(k)\right)\right)c \\ b = sign\left(Im\left(\hat{S}_D(k)\right)\right)d \end{cases} \quad (6.3\text{-}83)$$

where c, d is

$$\begin{cases} c = \left|Re\left(\hat{S}_R^{[-1]}(k)\right)\right| \\ d = \left|Im\left(\hat{S}_R^{[-1]}(k)\right)\right| \end{cases} \quad (6.3\text{-}84)$$

in the case where the order of the absolute values for the real and imaginary components are the same, i.e.

$$\left|Re\left(\hat{S}_D(k)\right)\right| \geq \left|Im\left(\hat{S}_D(k)\right)\right| \wedge \left|Re\left(\hat{S}_R^{[-1]}(k)\right)\right| \geq \left|Im\left(\hat{S}_R^{[-1]}(k)\right)\right| \quad (6.3\text{-}85)$$

and otherwise

$$\begin{cases} c = \left|Im\left(\hat{S}_R^{[-1]}(k)\right)\right| \\ d = \left|Re\left(\hat{S}_R^{[-1]}(k)\right)\right| \end{cases} \quad (6.3\text{-}86)$$

An example of the low complex phase matching is illustrated in 5.3-15, where the operation moves the phase within the correct $\pi/4$ section of the unit circle.

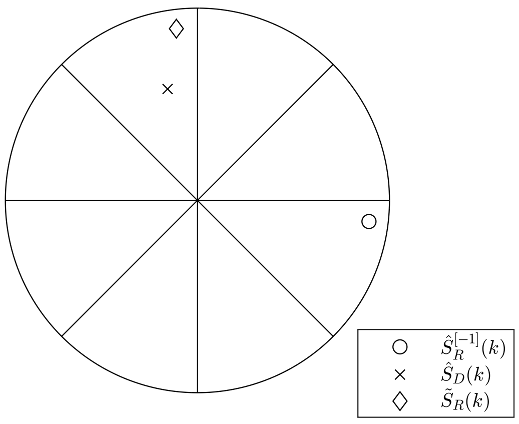


Figure 6.3-3: Low complex phase matching within $\pi/4$ of target

The Phase ECU algorithm as described in [4] 5.4.3.5.2 and 5.4.3.5.3 is applied on $\hat{S}_R^{[-1]}(k)$, where the N_{peaks} peaks $k_p(i), i = 0, \ldots, N_{peaks} - 1$ are identified and then refined to $k'_p(i), i = 0, \ldots, N_{peaks} - 1$ on a fractional frequency scale. As described in [xref previous clause on DFT Stereo], the second analysis frame of the subframe of the DFT stereo is a time-reversed version of the first subframe. Since the last residual subframe is generated from the second subframe, the first ECU subframe is time-reversed to create the matching window shape. The phase adjustment for the time-reversed ECU frame is calculated according to

$$\begin{cases} \Delta\phi_i = -2\phi_i - 2\pi k'_p(i)\left(N_{step21} + N_{lost} \cdot L_{DFT}\right)/L_{DFT} \\ \phi_i = \arctan\left(\hat{S}_R^{[-1]}\left(k'_p(i)\right)\right) - f_{frac}(1/3.0329 - \pi) \\ f_{frac} = k'_p(i) - k_p(i) \end{cases} \tag{6.3-87}$$

where N_{step21} is the number of samples between the end of the second DFT stereo subframe to the start of the first subframe, N_{lost} is the number of consecutively lost frames and L_{DFT} is the length of the DFT analysis frame. For the first lost frame $N_{lost} = 1$. The second subframe is not time reversed and the phase adjustment $\Delta\phi_i$ is computed similar to [4] 5.4.3.5.2, except the frame length and frame alignment gives different constants.

$$\Delta\phi_i = 2\pi k'_p(i) N_{lost} \tag{6.3-88}$$

As in [4] 5.4.3.5.3, the peaks of $\hat{S}_R^{[-1]}(k)$ are adjusted by applying the phase adjustment $\Delta\phi_i$ to the peak bins and their neighbourhood bins according to

$$\begin{cases} \hat{S}_{R,adj}(k) = e^{j\Delta\phi_i}\hat{S}_R^{[-1]}(k), & 1st\ subframe \\ \hat{S}_{R,adj}(k) = \left(e^{j\Delta\phi_i}\hat{S}_R^{[-1]}(k)\right)^*, & 2nd\ subframe \end{cases} \tag{6.3-89}$$

for $k \in K_{peak}$, where $(\cdot)^*$ denotes complex conjugate and results in a time reversal for the 2nd subframe and K_{peak} denotes the set of bins that are part of the peaks and their neighbouring bins in the current frame.

$$K_{peak} = \left\{-1, k_p(i), 1\right\}, i = 0, \ldots, N_{peaks} - 1 \tag{6.3-90}$$

The Phase ECU algorithm in [4] 5.4.3.5.2 and 5.4.3.5.3 is complemented with a separate source for the noise component of the spectrum. The side prediction residual concealment spectrum $\hat{S}_R(k)$ is formed by combining the phase adjusted peaks and their neighbouring bins with the energy adjusted decorrelated down-mix signal $\tilde{S}_R(k)$, i.e.

$$\hat{S}_R(k) = \begin{cases} \hat{S}_{R,adj}(k), k \in K_{peak} \\ \tilde{S}_R(k), k \notin K_{peak} \end{cases} \tag{6.3-91}$$

The non-peak bins may be seen as the noise component of the spectrum. If no peaks are found, $\hat{S}_R(k)$ will comprise only $\tilde{S}_R(k)$ the energy adjusted noise component. The concealment spectrum $\hat{S}_R(k)$ for the decoded residual signal will be transformed to time domain and included in the reconstructed stereo signal as described in 6.3.3.2.10.

6.3.3.3 TD-based stereo decoding

6.3.4 MDCT-based stereo decoding

6.3.4.1 General Overview

The decoding process for the MDCT-based stereo comprises of the decoding of all the side parameters, namely, the parameters of the tools like TNS, SNS, TCX LTP, the IGF parameters and the stereo parameters. Then the bitrate ratio decoded from the bitstream determines the number of bits to be read for each channel and the decoding of the spectral data follows using the range decoder. After noise filling, IGF is applied, the inverse stereo processing takes place and resulting intermediate signal is de-normalized using the global ILD value parsed by the bitstream. Finally, spectral noise shaping is applied to de-whiten the signals, TNS filtering is also applied, if it is active, either before or after the de-whitening process and the inverse MDCT transform is carried out to obtain the original signals of the two channels in time-domain.

6.3.4.2 Decoding of parameters

6.3.4.2.1 TCX block configuration

As in EVS, the coding modes TCX20 or TCX10 are signaled within the bit stream for each channel. Overlap code for the current frame is formed from the short/long transform decision bit and from the binary code that is read from the bit stream for the overlap width as defined in clause 5.3.2.3 of [3].The TCX block is then configured as described in clause 6.2.4.2 of [3].

Additionally, the previous frame overlap coding mode is also read from the bit stream to prevent using a wrong length for the inverse transform if previous frame is lost.

Finally, the kernel type is also read from the bitstream to determine whether kernel switching, described in clause 5.3.3.3.3.4.2 was enabled at the encoder.

6.3.4.2.2 Core parameters decoding

For each channel the following parameters are decoded as defined in EVS [3]:

- Global gain (clause 6.2.2.2.4)
- Noise fill parameter (clause 6.2.2.2.5)
- LTP data (clause 6.2.2.2.6)
- TNS parameters (clause 6.2.2.2.7)
- IGF parameter decoding (clause 6.2.2.2.9)

6.3.4.2.3 SNS parameters decoding

6.3.4.2.3.1 Scale parameter decoding

Depending on the codec bitrate and the core coder sampling rate, the SNS scale parameters were either encoded using a multi-stage stochastic VQ (MSVQ) or a 2-stage split/AVQ quantizer. Refer to clause 5.3.3.3.3.6.3, especially table Table 5.3-11 for the concrete assignment of VQ technologies to operating points. For each operating point, the corresponding dequantizer is selected according to the same table.

Decoding the SNS scale parameters includes obtaining codevectors using the transmitted VQ indices and restoring the SNS scale parameters back to left/right representation if joint encoding was used. The information about whether joint encoding was used or if the SNS scale parameters were encoded separately for each channel is read from the bitstream in form of a single signalling bit. In operating points where the MSVQ is used for quantizing the SNS scale parameters and both channels use short transform lengths, a bit for each subframe is read. The bit to determine between joint or separate encoding mode is only written by the encoder if both channels use the same transform block length as joint coding is not possible if transform lengths differ between the channels. Therefore, the decoder does not read the respective bit from the bitstream in that case and defaults to assume the separate encoding mode has been used. This makes it necessary to read and decode the transform length for both channels from the bitstream prior to decoding the SNS scale parameters.

6.3.4.2.3.1.1 2-Stage Split/AVQ dequantizer

The dequantizer for the split/AVQ case comprises a first stage split vector dequantizer that obtains an intermediate quantized SNS scale parameter vector, a second stage algebraic vector dequantizer that obtains an SNS scale parameter residual vector and a combiner stage that combines the intermediate quantized vector and the residual vector to obtain the reconstructed SNS scale parameter vector.

The dequantization process depends on the employed encoding scheme (joint or separate encoding, low-bitrate mode and inter-subframe coding for short blocks). Operation of the separate encoding mode without low-bitrate mode for long blocks in both channels will be described first followed by a description of the differences in the other encoding modes.

In separate encoding mode for long blocks in both channels, three indices are read from the bitstream per channel:

A 10 bit index for the 1st stage split VQ dequantizer $ind_{SNS,1}$

A base codebook index for the AVQ dequantizer

A Voronoi extension index for the AVQ dequantizer

The two indices for each split of the 1st stage are calculated as

$$ind_0 = ind_{SNS,1} \; mod \; 2^5 \tag{6.3-92}$$

$$ind_1 = \left\lfloor \frac{ind_{SNS,1}}{2^5} \right\rfloor. \tag{6.3-93}$$

With these, the first stage vector dequantizer obtains the dequantized vectors for each split and puts them together into the 1st stage intermediate quantized vector $\widehat{SNS}_{VQ,1}$ as given in Equation (5.3-198) The second stage vector dequantizer obtains the 2nd stage quantized residual vector $\widehat{SNS}_{VQ,2}$ as described in EVS [3], clause 6.1.1.2.1.6.1. The final quantized SNS scale parameter vector is then calculated by the combiner stage by adding the two dequantized vectors after weighting the residual vector by the factor of 0.4 according to Equation (5.3-200). This is done separately for both channels.

In joint encoding mode, the SNS scale parameters were quantized in a mid/side representation so that the first set of the jointly encoded scale parameters comprises mid scale parameters and the second set of the jointly encoded scale parameters comprises side scale parameters. The mid representation is first dequantized as described above. For decoding the side representation, first the *zero_side* flag is read from the bitstream as a single bit. If the value of this flag is 1, the side SNS scale parameter vector is set to zero and SNS decoding stops. Otherwise, a base codebook index and a Voronoi extension index for the AVQ are read from the bitstream and AVQ decoding is done as described in EVS [3], clause 6.1.1.2.1.6.1, to obtain the side SNS scale parameter vector, i.e.

$$\widehat{SNS}_{side}(i) = \begin{cases} \widehat{SNS}_{VQ,2}(i), if \; zero\_side = 0 \\ 0, if \; zero\_side = 1 \end{cases}. \tag{6.3-94}$$

If at least one of the channels uses short blocks, no joint encoding is possible. Instead, low-bitrate mode encoding can be indicated in the bitstream if the bitrate is 48kbps. This is signaled using a single bit which must be read from the bitstream before decoding the SNS scale parameters. If low-bitrate mode is signaled, the second stage decoding is skipped and only the first stage split VQ decoding is run to calculate the quantized output vector. At other bitrates and if the low-bitrate signaling bit is set to zero, the SNS scale parameters for channels with long blocks are decoded as described above for the separate encoding mode with long blocks in both channels. For channels that use short blocks, two sets of SNS scale parameters need to be decoded. The first set for the first subframe is always decoded as described above for separate encoding mode with long blocks in both channels. If no low-bitrate mode is used, inter-subframe coding of the SNS scale parameters is signaled in the bitstream with one bit. This bit is read from the bitstream and if its value is one, no first stage is decoded for the second subframe's set of SNS scale parameters. Instead, only the AVQ indices are read from the bitstream and decoded as described in EVS [3], clause 6.1.1.2.1.6.1. The output quantized SNS scale parameter vector is then calculated as the sum of the AVQ decoded output vector and the first subframe's decoded SNS parameter vector.

6.3.4.2.3.1.2 MSVQ

In operating points that use the MSVQ to quantize the SNS scale parameter, decoding is simply done by reading the respective codebook index from the bitstream and choosing the respective code vector from the respective codebook. The codebook to use for the lookup is chosen depending on the kind of representation in which the SNS scale parameters were quantized and the used transform block length according to Table 5.3-13. Decoding is done subframe-wise. If the transform lengths are the same in both channels, a bit is read from the bitstream to determine between left/right or mid/side encoding mode. If the bit's value is one, joint encoding mode is assumed, otherwise separate encoding mode is assumed. In case the transform lengths differ between the channels, no bit is read and separate encoding mode is assumed per default.

The MSVQ decoder output is given by

$$\widehat{SNS}(i) = \sum_{k=0}^{N} V_k\left(I_{MSVQ}(k), i\right) \tag{6.3-95}$$

where $V_k(j,i)$ is the *i*-th coefficient of the *j*-th vector in the selected codebook of stage *k*. V_k is selected according Table 5.3-13. If the joint encoding mode is signaled, the SNS scale parameters were transmitted in a mid/side representation together with a *zero_side* flag represented by a single bit. This flag is read from the bitstream prior to decoding the side SNS scale parameters. If the flag's value is 1, no MSVQ decoding is performed for the side SNS scale parameter vector and all of its parameters are set to zero instead.

6.3.4.2.3.1.3 Restoring left/right representation of jointly coded SNS scale parameters

If the joint encoding mode was signaled, the SNS scale parameter vectors for the respective (sub)frame were transmitted in mid/side representation and need to be converted back to a left/right representation. To achieve this, the scale parameter encoder combines the two vectors of jointly encoded scale parameters, comprising mid or side scale parameters, respectively, to obtain the left and right scale parameters by using two different combination rules. The first combination rule includes adding the mid and side scale parameters while the second combination rule includes subtracting the side scale parameters from the mid scale parameters. Specifically, the combination is done as described in clause 5.3.3.3.3.6.3.4.

6.3.4.2.4 Stereo parameters decoding

6.3.4.2.4.1 ITD decoding

For the mid-high bitrates namely 48 and 64 kbps, where the time-domain audio signals are time-aligned depending on the inter-channel time difference (ITD) as described in XXX, the ITD value is read from the bit stream and de-quantized. Then the ITD is re-introduced between the two stereo channels as described in xxx.

6.3.4.2.4.2 Stereo mode of the core bands

The stereo encoding mode is decoded from the bitstream and it is determined whether the encoded audio signal is encoded using full mid-side (MS_FULL), full dual-mono (DUAL_MONO) or band-wise encoding mode (BW_MS).

If the stereo mode is:

- DUAL_MONO then the MS mask is set to 0 for all core spectral bands $i = 0 \dots n_{Bands,core} - 1$;
- MS_FULL then the MS mask is set to 1 for all core spectral bands $i = 0 \dots n_{Bands,core} - 1$;
- BW_MS then the MS mask value of the m_i is read from the bit stream for each spectral band $i = 0 \dots n_{Bands,core} - 1$;

6.3.4.2.4.3 Global ILD decoding

The global ILD is read from the bitstream depending on the core coding mode and on whether the core coding mode is in sync in both channels. More specifically, if the core coding mode is the same for the channels and equal to TCX20 then the one global ILD value per frame is read by decoding b_{ILD} bits. If the core coding mode is TCX10 for both channels, then the global ILD value is read for each subframe. If the core coding mode is different between channel, then only one global ILD value is read, and for the channel with TCX10 the same global ILD is used for both subframes.

6.3.4.2.4.4 Stereo mode of the IGF bands

Subsequently, for the bitrates that intelligent gap filling is used, the stereo mode for the IGF bands is read from the bit stream. Similarly, as in described in clause 6.3.4.2.4.2 the MS mask is set to 1 if the stereo mode is MS_FULL or is set to 0 if stereo mode is DUAL_MONO or is parsed from the bit stream for each IGF spectral band $i = n_{Bands,core} \cdots n_{Bands,IGF} - 1$.

6.3.4.2.4.5 Split ratio and bitrate distribution

Finally, the split ratio $\hat{r}_{split}$ for the distribution of the remaining bits between the two channels is parsed from the bit stream. The procedure as described in clause 5.3.3.3.7 is applied to determine the number of bits to be read for each channel to decode the spectral coefficients using the range coder.

6.3.4.3 Decoding process

6.3.4.3.1 Spectral data decoding and noise filling

The arithmetic decoder is replaced in IVAS by the range decoder and is described in the following clause 6.2.3.3.2 of the present document. The remaining processing for each channel is the same as in EVS and clause 6.2.2.3 of [3]. More specifically:

- Global gain decoding as described in clause 6.2.2.3.3
- Residual bits decoding and global gain adjustment as described in clause 6.2.2.3.4
- Noise filling as described in clause 6.2.2.3.6

6.3.4.3.2 Range decoder

6.3.4.3.3 Application of stereo IGF

6.3.4.3.3.1 General

After the decoding and de-quantizing of the spectral coefficients of the first and second channel is performed, stereo IGF is applied to obtain the full-bandwidth spectra of the two channels.

As described at the respective encoding clause 5.3.3.3.6.1 stereo IGF is applied only for the case where stereo encoding is allowed in general i.e. if the coding mode is the same for both channels and therefore, the spectral resolution is the same and if either the core stereo encoding mode or the IGF stereo encoding more is not DUAL_MONO. Otherwise, the IGF is applied to each channel separately as is done for EVS and described in [3] clause 6.2.2.3.8.

6.3.4.3.3.2 Stereo IGF apply

6.3.4.4 Stereo decoding process

6.3.4.4.1 Inverse mid-side transform

As described in 6.3.4.2.4.2 and 6.3.4.2.4.4 the stereo encoding mode is retrieved from the bitstream for both the so called-core bands, i.e. up to the cross-over band to IGF and the IGF bands if IGF is active. Depending on the stereo encoding mode:

- full mid-side: the output first channel of the intermediate audio output is retrieved from the inverse mid operation of the encoded first and second channel and the second channel from the inverse side operation of the encoded first and second channel.

$$\bar{X}_{1,q}[k] = \frac{\sqrt{(2)}}{2}\left(\tilde{X}_{1,q}[k] + \tilde{X}_{2,q}[k]\right), \quad for\ k = 0, \dots, L_{frame} - 1 \tag{6.3-96}$$

$$\bar{X}_{2,q}[k] = \frac{\sqrt{(2)}}{2}\left(\tilde{X}_{1,q}[k] - \tilde{X}_{2,q}[k]\right), \quad for\ k = 0, \dots, L_{frame} - 1 \tag{6.3-97}$$

where: $\tilde{X}_{1,q}, \tilde{X}_{2,q}$ are the M/S encoded signals of the first and second channels ;

$\bar{X}_{1,q}, \bar{X}_{2,q}$ represent the intermediate outputs, meaning they are still normalized with respect to their original energy values;

L_{frame} is the length of the frame up to the core bandwidth

- full dual-mono: the first and second channel of the encoded signals remain unaltered for the intermediate output.
- band-wise mid-side, then the stereo encoding mode for each spectral band of the total spectral bands is determined. If the stereo mode for the given spectral band:

- is dual-mono than the spectral band of the first channel of the intermediate output is the same as the input first channel and the second channel of the intermediate output is the same as the second channels of the encoded signal.
- is mid-side than the respective spectral band of the first intermediate channel is obtained from the inverse mid operation of the respective spectral bands of the first encoded channel and the second audio channel and the respective spectral band of the intermediate second channel is obtained from the inverse side operation of the first encoded channel and the second encoded channel.

The inverse operation to obtain the intermediate output stereo signal is depicted in Equations (6.3-98) and (6.3-99) below.

$$\overline{X}_{1,q}\left[b_{offset}(i)+k\right]=\begin{cases}\frac{\sqrt{2}}{2}\left(\tilde{X}_{1,q}\left[b_{offset}(i)+k\right]+\tilde{X}_{2,q}\left[b_{offset}(i)+k\right]\right), & \text{if } m_i=1\\ \tilde{X}_{1,q}\left[b_{offset}(i)+k\right], & \text{otherwise}\end{cases} \quad (6.3\text{-}98)$$

$$\overline{X}_{2,q}\left[b_{offset}(i)+k\right]=\begin{cases}\frac{\sqrt{2}}{2}\left(\tilde{X}_{1,q}\left[b_{offset}(i)+k\right]-\tilde{X}_{2,q}\left[b_{offset}(i)+k\right]\right), & \text{if } m_i=1\\ \tilde{X}_{2,q}\left[b_{offset}(i)+k\right], & otherwise\end{cases} \quad (6.3\text{-}99)$$

where: $i=0,1,\cdots,n_{bands,core}-1$;

$k=0...N_{bins}(i)-1$;

$\tilde{X}_{1,q}^{i}\left[b_{offset}(i)+k\right],\tilde{X}_{2,q}^{i}\left[b_{offset}(i)+k\right]$ is the k-th bin of the i-th spectral band of the retrieved encoded signals of the first and second channels;

$\overline{X}_{1,q}^{i}\left[b_{offset}(i)+k\right],\overline{X}_{2,q}^{i}\left[b_{offset}(i)+k\right]$ is the k-th bin of the i-th spectral band of the intermediate first and second channel output;

$b_{offset}(i)$ is the spectral offset of the i -th band defined in Table 5.3-14

m_i is the decoded M/S mask decision of the the *i*-th spectral band;

$N_{bins}(i)$is the width of the *i*-th spectral band as in number of spectral bins defined in Table 5.3-14

;

For the bitrates that IGF is active, the same inverse mid-side processing is applied for the IGF bands depending on the respective stereo mode.

6.3.4.4.2 De-normalization to global ILD

Depending on the de-normalization value, either the first or second channel is de-normalized to obtain the first channel and the second channel of the decoded audio.

The de-normalization value g_{ILD} is calculated as in Equation (5.3-212), where in this case $\widehat{ILD}$ is the quantized ILD value decoded from the bit stream. Then the de-normalization processing applies as:

$$X_{1,q}^{MDCT}[k]=1/g_{ILD}\cdot\overline{X}_{1,q}^{MDCT}[k], \quad \text{for } k=0\,...\,L_{frameTCX}-1, \quad \text{if } g_{ILD}<1 \quad (6.3\text{-}100)$$

or

$$X_{2,q}^{MDCT}[k]=g_{ILD}\cdot\overline{X}_{2,q}^{MDCT}[k], \quad \text{for } k=0\,...\,L_{frameTCX}-1, \quad \text{if } g_{ILD}>1 \quad (6.3\text{-}101)$$

6.3.4.5 Post-stereo decoding process

6.3.4.5.1 General

After the decoded stereo signal is obtained, the first and second channel are further processed to conduct a decoder side temporal noise shaping if it was signaled as active in the decoded parameters and a spectral noise shaping to obtain the perceptually un-whitened original audio spectra of the first and second channel. Finally, the inverse MDCT transform is performed to obtain the final time-domain first and second channel decoded audio signal.

6.3.4.5.2 Spectral noise shaping

6.3.4.5.2.1 General overview

The spectral shaping applied on the signal in the encoder using SNS needs to be reverted during decoding. Applying the inverse shaping restores the spectral curve of the original signal and shapes the quantization noise introduced into the signal to be minimally perceived. The SNS decoder comprises a scale parameter decoder that decodes the encoded SNS scale parameters which were either encoded jointly or separately and a spectral signal processor that processes the decoded MDCT spectrum by scaling the MDCT coefficients using scale factors obtained from the decoded SNS scale parameters. The encoded SNS scale parameters are decoded using one of two vector dequantizers that provide the decoded SNS scale parameters from the quantization indices read from the bitstream. After decoding the scale parameters, the scale parameter decoder interpolates them to obtain the scale factors to use for scaling the MDCT spectrum by applying each scale factor on all spectral samples in each frequency band.

6.3.4.5.2.2 Spectral Shaping

After the SNS scale parameters have been reconstructed, scale factors are interpolated from them and the inverse shaping with respect to the decoder is applied on the MDCT spectrum.

6.3.4.5.2.2.1 Interpolation of SNS scale factors

The scale parameter decoder performs the interpolation of the SNS parameters in the same way as done in the encoder, i.e. according to Equation (5.3-205) While being interpolated, the scale parameters are still in a log domain of base 2. After the interpolation is performed, the parameters are converted back to linear domain to obtain the set of scale factors used for shaping the MDCT spectrum, according to

$$g_{SNS}(b) = 2^{scf(b)}, b = 0, \dots, 63. \qquad (6.3\text{-}102)$$

6.3.4.5.2.2.2 Scaling the MDCT spectrum

The interpolated scale factors are applied to the MDCT spectrum by the spectral signal processor in the same bands as used in the SNS encoder (see clause 5.3.3.3.3.6.1). This results in a scaled decoded MDCT spectrum with restored spectral shape as in the original encoded signal and perceptually shaped quantization noise. Scaling of the spectrum is done as described in clause 5.3.3.3.3.6.4.2.

6.3.4.5.3 Temporal noise shaping

If from the decoded TNS data TNS is active, and it is signalled from the bitstream that TNS is applied on the whitened spectrum than the TNS filtering is applied prior to the SNS.

Furthermore, if it is signalled from the bit stream, TNS filtering is applied on the intermediate signal after the SNS.

The TNS filtering is the same as for EVS and is described in detail in 6.2.2.3.10 of [3].

If the configuration, determined as described in clause 6.2.4.2 of [3], indicates that some sub-frames are coded using TCX5 then sub-frame containing MDCT bins of 2 TCX5 sub-frames is de-interleaved prior to the TNS filtering either before or after the de-whitening process with SNS.

6.3.4.6 Frequency-to-time domain transformations

As a final decoding step, the time-to-frequency transformation is applied on each channel to obtain the final time-domain signals of the left and right channels respectively. In clause 6.2.4 of [3], the details of the processing of the inverse MDCT, overlap-add and windowing are described.

6.3.5 Switching between stereo modes

6.3.6 DTX operation

6.3.6.1 DTX in Unified stereo

6.3.6.1.1 General

If a Unified stereo CNG frame is received, the decoder generally operates as in DFT-based stereo mode as described in 6.3.3.2. The main difference is that the stereo CNG decoder extracts the spectral shape represented by the decoded CNG frame and uses this in a stereo CNG generation, explained in 6.3.6.1.2. In addition, there are two differences compared to the EVS CNG decoding, matching the differences as described in 5.3.3.5.1.2.

- Core codec reset in active frame onset
- Energy scaling of CNG

The DFT-based stereo parameters $ITD(m),\ gIPD(m),\ g_S[m,b]$ are decoded from the bitstream as described in 6.3.6.1.3 with the band resolution set by the encoder. In addition, the stereo coherence is decoded as described in 6.3.6.1.4 and a slightly modified DFT-based stereo synthesis is done as described in 6.3.6.1.5.

6.3.6.1.2 Stereo CNG spectral shape extraction

The EVS CNG decoder operates in two modes, LP-CNG and FD-CNG. In Unified stereo CNG, the CNG is generated in DFT domain based on the spectral shape of the decoded EVS CNG frame. For the LP-CNG, the filter LP synthesis filter $\hat{A}(z)$ described in [3] 6.7.2.1.4 is convolved with the denominator of the de-emphasis filter $[1 \quad -\beta]$ as described in [3] 6.4, to produce a deemphasized synthesis filter

$$\hat{A}_{demph}(n) = A(n) * [1 \quad -\beta] \qquad (6.3\text{-}103)$$

where '$*$' denotes convolution. $\hat{A}_{demph}(n)$ is then transformed to DFT domain where the synthesis shape $1/\hat{A}_{demph}(n)$ is formed according to

$$\begin{aligned} N_{LP-CNG}(k) &= \frac{g_{scale}}{|N_{LPinv}(k)|} \\ N_{LPinv}(k) &= \sum_{n=0}^{L_{FFT}-1} \hat{A}_{demph}(n) e^{-j2\pi kn/N} \\ g_{scale} &= 2\sqrt{\frac{E_{CN}}{0.5L_{FFT}}} \\ k &= 0,1,\dots,L_{FFT}-1 \end{aligned} \qquad (6.3\text{-}104)$$

where E_{CN} is a low-pass filtered energy of the low-band excitation signal and L_{FFT} is the FFT length. For a core coding of $F_s = 12.8$ kHz, $L_{FFT} = 256$ and for $F_s = 16$ kHz, $L_{FFT} = 320$. These steps are repeated to obtain the high band using the LP-filter for SHB-CNG as defined in [3] 6.7.2.1.7.

When switching from TD stereo active mode to LP-CNG during first side frame. The background noise spectral shape is adapted to give smooth transition from active coding to CNG generation in FD domain. After first SID frame, coherence is initialized based on the left and right output of the previous frame. First a correlation is calculated according to the equation below

$$C_{LR\_TD} = \frac{enr_L\, enr_R}{\sqrt{enr_L\, enr_R}} \qquad (6.3\text{-}105)$$

Where enr_L and enr_R are the energy of the output signals from the previous active frame. A low pass filtering of the interchannel correlation is performed according to the equation below.

$$\hat{C}_{LR\_TD}[m] = \alpha_{corr} C_{LR\_TD}[m] + (1-\alpha_{corr})\hat{C}_{LR\_TD}[m-1] \qquad (6.3\text{-}106)$$

Coherence for all the bands is initialized as follows for the first SID after a TD active frame

$$\hat{C}_{band}^{CNG}[m,b] = \begin{cases} \hat{C}_{LR\_TD}[m] * \hat{C}_{LR\_TD}[m], & firstSidTD = 1 \\ \hat{C}_{band}[m,b], & firstSidTD = 0 \end{cases} \qquad (6.3\text{-}107)$$

Where $\hat{C}_{band}[m,b]$ is defined by equation (6.3-130)and $firstSidTD$ is a flag indicating that the last active frame was in done in TD mode. α_{corr} is set to 0.8.

For subsequent inactive the frames the coherence is updated as follows

$$\hat{C}_{band}^{CNG}[m,b] = \begin{cases} \hat{C}_{LR\_TD}[m] * \hat{C}_{LR\_TD}[m], \ if \ 8 < i_{tdCntr} < 50 \wedge i_{sidCntr} < 6 \\ (1-\alpha_{coh})\hat{C}_{band}[m,b] + \alpha_{coh}\hat{C}_{band}^{CNG}[m-1,b], otherwise \end{cases} \quad (6.3\text{-}108)$$

Where i_{tdCntr} is the number of TD frames and $i_{sidCntr}$ is the number of SID frames received by the decoder. α_{coh} is set to 0.8.

For transitions between active TD coding to CNG generation a crossfade is performed between two noise spectra, one being the background noise estimation during active TD frame on the decoder and the other based on parameters from the SID frame.

For LP CNG the crossfade is based on the noise in active TD frames $N_{CNA}(k)$ described in clause X, here denoted an $N_{CNA}^{lastActive}(k)$. First a crossfade length is determined in the first SID after TD according to

$$L_{xfade} = \begin{cases} -M_{xfade}r_1 + M_{xfade}, \ if \ r_1 < 1 \\ -M_{xfade}\left(\frac{1}{r_1}\right) + M_{xfade}, \ otherwise \end{cases} \quad (6.3\text{-}109)$$

Where M_{xfade} is the maximum crossfade or transition length allowed.
r_1 is the energy ratio of the two background estimates determined by

$$r_1 = \sqrt{\frac{\sum_{k=k_0}^{k=b_{N-1}} N_{CNA}^{lastActive}(k)}{\sum_{k=k_0}^{k=k_{N-1}} N'_{LP-CNG}(k)}} \quad (6.3\text{-}110)$$

Where $N'_{LP-CNG}(k)$ is background noise equivalent to before g_{scale} is applied equivalent to

$$N'_{LP-CNG}(k) = \frac{1}{g_{scale}} N_{LP-CNG}(k) \quad (6.3\text{-}111)$$

The background noise spectra are updated according to

$$\hat{N}_{LP-CNG}(k) = \begin{cases} \left(1 - \frac{i_{inactive}}{k}\right)\left(\alpha_{avg}N_{CNA}^{lastActive}(k)\right) + \frac{i_{inactive}}{k} N_{LP-CNG}(k), \ i_{inactive} < L_{xfade} \\ N'_{LP-CNG}(k), otherwise \end{cases} \quad (6.3\text{-}112)$$

Where $i_{inactive}$ is the number of inactive frames and α_{avg} is used to scale the energy of the background noise estimate in TD the active mode to match that of $N_{LP-CNG}(k)$. The scaling parameter is per band is computed according to equation(6.3-113), (frame and band indices have been skipped for clarity).

$$\alpha[b] = \frac{1}{2}\sqrt{\frac{1 + c + 2\sqrt{c \cdot C_{LR\_TD}}}{c \cdot ratio_{LR}{}^2 + (1-ratio_{LR})^2 s_{right}^2 + 2ratio_{LR}(1-ratio_{LR})s_{right}\sqrt{c \cdot C_{LR\_TD}}}} \quad (6.3\text{-}113)$$

where $ratio_{LR}$ are the TD downmixing parameters and s_{right} is the gain applied to the right channel during downmixing as described in clause X, is given by equation (6.3-114).

$$c = \frac{(1+g)^2 + \gamma^2}{(1-g)^2 + \gamma^2} \quad (6.3\text{-}114)$$

The scaling parameter α_{avg} is computed by summing $\alpha[b]$ over all bands and dividing by the number of bands.

For low band LP CNG, a random noise generator is used to generate noise for the real and imaginary parts. The random noise is scaled with $\hat{N}_{LP-CNG}(k)/2$ and g_{scale}.

For high band LP CNG, denoted as $N_{LP-CNG}^{SHB}(k)$, a target gain received by the decoder is used to compute a scale factor for the high band CNG.

$$scale_{target} = \frac{\bar{\bar{E}}_{hs,i}}{N_{LP-CNG}^{SHB}(0) * N_{LP-CNG}^{SHB}(0)} \quad (6.3\text{-}115)$$

Where $\bar{\bar{E}}_{hs,i}$ is defined by equation 1971 in reference [3]. A random noise generator is used to generate noise for the real and imaginary parts. The random noise is scaled with a flipped spectrum of $\hat{N}_{LP-CNG}(k)/2$ and $scale_{target}$.

Low band and high band noise generation is repeated to generate two uncorrelated noise spectra $N_{CNG-0}(k)$ and $N_{CNG-1}(k)$ in the case of LP CNG.

For FD CNG the background noise estimate at the decoder is updated in a similar way as in clause 6.7.3.2.3.2 in reference [3], with the modification provided in equation (6.3-116). The modifications provide a smooth transition when switching from TD active mode to FD CNG.

$$\bar{N}_{FD-CNG}^{full}(k) = \begin{cases} N_{FD-CNG}^{[shappin,LR]}(k)\beta \text{ }, lastActiveTD = 0 \\ N_{FD-CNG}^{[shappin,LR]}(k)\left(\alpha_{avg} + \frac{1}{M_{xfade}}\left(\beta(k) - \alpha_{avg}\right) L_{xfade}\right), \\ lastActiveTD = 1 \wedge i_{inactive} < M_{xfade} \end{cases} \tag{6.3-116}$$

Where $lastActiveTD$ flag indicates the last frame was TD mode and β is evaluated as follows

$$\beta(k) = \frac{N_{FD-CNG}^{[SID]}(k)}{N_{FD-CNG}^{[shapping,LR]}(k)} \tag{6.3-117}$$

For the first part of the CNG spectrum the frequency resolution of the noise shaping function $N_{FD-CNG}^{full}(k)$ is twice the DFT-based stereo resolution since DFT-based stereo uses two subframes for each speech frame. The CNG spectrum $N_{FD-CNG}(k)$ is formed by averaging the bins two by two according to

$$\begin{gathered} \widehat{N}_{FD-CNG}(k) = \frac{\bar{N}_{FD-CNG}^{full}(2k) + \bar{N}_{FD-CNG}^{full}(2k+1)}{2} \\ k = 0,1,\dots,L_{FFT} - 1 \end{gathered} \tag{6.3-118}$$

Two uncorrelated noise spectra $N_{CNG-0}(k)$ and $N_{CNG-1}(k)$ are generated as in [3] 6.7.3.3.2 based on pseudo-random Gaussian noise with different random seeds, scaled with the magnitude of $\widehat{N}_{FD-CNG}(k)/2$ in case of FD-CNG decoding.

6.3.6.1.3 Stereo CNG side gain, ITD and IPD decoding

The side gain $\hat{g}_S[m,b]$ is decoded the same way as in active frames, but with the band resolution as described in Table 5.3-17. The decoded side gain parameters are low-pass filtered during CNG frames according to

$$\hat{g}_{S,LP}[m,b] = 0.2\hat{g}_S[m,b] + 0.8\hat{g}_{S,LP}[m-1,b] \tag{6.3-119}$$

except for the first CNG frame after active coding where $\hat{g}_{S,LP}[m,b]$ is directly set to $\hat{g}_S[m,b]$.

The ITD parameter is decoded similar to the active frames but without the option of Huffman coding and included an extra step due to the reduced resolution.

$$ITD_{SID}(m) = 2I_{ITD} + 256 \cdot sign \tag{6.3-120}$$

where I_{ITD} is the received ITD index and $sign \in \{-1,1\}$ is the decoded sign bit.

Whether residual encoding is enabled or not during the active encoding mode indicates whether the foreground and background signals are efficiently separated for active frames, and if it is enabled, $ITD_{SID}(m)$ is used directly for synthesis in the CNG encoding mode, i.e. $ITD_{syn}(m) = ITD_{SID}(m)$. However, for CNG stereo synthesis where the active frames are encoded at a bitrate $\leq 24.4\ kbps$, where residual coding is not utilized, the ITD is adjusted by a gradual fading from the ITD of the previous frame towards the received ITD target, $ITD_{target} = ITD_{SID}(m)$, as

$$ITD_{syn}(m) = ITD_{syn}(m-1) + ITD_{step} \quad if \quad itd\_xfade\_counter < L_{xfade} \tag{6.3-121}$$

where $itd\_xfade\_counter$ is a counter of number of frames for which the fade has been performed and $L_{xfade} = 100$ is an upper threshold for the number of fading frames. $ITD_{syn}(m-1)$ denotes the ITD of the previous frame, being the latest ITD value of the fading and starting from the ITD of the last active frame prior the CNG period. The size of the steps taken towards the target ITD is set in the beginning of the CNG period, and updated whenever a new ITD_{target} is received, according to

$$ITD_{step} = \frac{ITD_{target} - ITD_{syn}(m-1)}{L_{xfade} - itd\_xfade\_counter} \tag{6.3-122}$$

The fading counter $itd\_xfade\_counter$ is increased by one for each frame the fade is being performed and reset to zero when there has been at more than $N_{xfade\_reset} = 2$ active frames. Following segments of at most N_{xfade_reset} active frames, the counter is not reset and the ITD fade is resumed from the ITD of the previous CNG frame instead of being restarted from the last active frame ITD, i.e.

$$ITD_{syn}(m) = ITD_{prev} + ITD_{step} \quad if \quad itd\_xfade\_counter < L_{xfade} \tag{6.3-123}$$

where ITD_{prev} is the latest ITD value of the gradual fade from the previous CNG period. If a new ITD target is received, the step size is updated as

$$ITD_{step} = \frac{ITD_{target} - ITD_{prev}}{L_{xfade} - itd\_xfade\_counter} \tag{6.3-124}$$

The IPD is decoded according to

$$gIPD_{SID}(m) = \frac{I_{IPD}\pi}{2} - \pi \tag{6.3-125}$$

where I_{IPD} is the decoded IPD index. As for the ITD, if residual coding is enabled for the active frames, $gIPD_{SID}(m)$ is used directly for stereo CNG synthesis, $gIPD_{syn}(m) = gIPD_{SID}(m)$. However, for active frame bitrates $\leq 24.4\ kbps$ where residual coding is not utilized, the IPD is adjusted by a gradual fading towards $gIPD_{target} = gIPD_{SID}(m)$, as

$$gIPD_{syn}(m) = gIPD_{syn}(m-1) + gIPD_{step} \quad if \quad ipd\_xfade\_counter < L_{xfade} \tag{6.3-126}$$

where $ipd\_xfade\_counter$ is a counter of number of frames for which the fade has been performed and $L_{xfade} = 100$ is an upper threshold for the number of fading frames. $gIPD_{syn}(m-1)$ denotes the IPD of the previous frame, being the latest IPD value of the fading and starting from the IPD of the last active frame prior the CNG period. The size of the steps taken towards the target IPD is set in the beginning of the CNG period, and updated whenever a new $gIPD_{target}$ is received, according to

$$gIPD_{step} = \frac{gIPD_{target} - gIPD_{syn}(m-1)}{L_{xfade} - ipd\_xfade\_counter} \tag{6.3-127}$$

The fading counter $ipd\_xfade\_counter$ is increased by one for each frame the fade is being performed and reset to zero when there has been at more than $N_{xfade\_reset} = 2$ active frames. Following segments of at most N_{xfade_reset} active frames, the counter is not reset and the IPD fade is resumed from the IPD of the previous CNG frame instead of being restarted from the last active frame IPD, i.e.

$$gIPD_{syn}(m) = gIPD_{prev} + gIPD_{step} \quad if \quad ipd\_xfade\_counter < L_{xfade} \tag{6.3-128}$$

where $gIPD_{prev}$ is the latest IPD value of the gradual fade from the previous CNG period. If a new IPD target is received, the step size is updated as

$$gIPD_{step} = \frac{gIPD_{target} - gIPD_{prev}}{L_{xfade} - ipd\_xfade\_counter} \tag{6.3-129}$$

6.3.6.1.4 Stereo CNG coherence decoding

The intra-frame predictor index q is obtained from the bitstream and the intra-frame predictor $p^{(q)}[b,i]$ is selected. Based on the available bit budget for the encoded stereo coherence B_C for the current frame m, the weighting factor α is obtained according to Table 5.3-18, where the decoded bit now indicates whether to select α_{low} or α_{high}. The coherence for each band b is obtained according to

$$\hat{C}_{band}[m,b] = \alpha \hat{C}_{intra}^{(q)}[m,b] + (1-\alpha)\hat{C}_{band}[m-1,b] + \hat{C}_{res}[m,b] \tag{6.3-130}$$

where the coherence prediction residual $\hat{C}_{res}[m,b]$ is now obtained from the bitstream and decoded according to Table 5.3-19. The intra-frame prediction $\hat{C}_{intra}^{(q)}[m,b]$ and the inter-frame prediction $\hat{C}_{band}[m-1,b]$ are obtained in the same way as in (5.3-239), using the previously reconstructed coherence values. The coherence $\hat{C}_{band}[m,b]$ is used together with the remaining decoded stereo parameters and the decoded CNG down-mix signal to produce a stereo CNG synthesis as described in 6.3.6.1.5.

6.3.6.1.5 Stereo CNG synthesis

The stereo CNG synthesis is done in frequency domain based on the stereo parameters, as described in 6.3.6.1.3 here omitting the frame index m, and the uncorrelated noise spectra $N_{CNG-0}(k)$ and $N_{CNG-1}(k)$, as generated in clause 6.3.6.1.2. Compared to the active frame stereo upmix, see 6.3.3.2.7, the processing is done for a coarser frequency resolution as described in Table 5.3-17, and with the same stereo parameters for both synthesis subframes i, as

$$\hat{L}_i[k] = N_{CNG-0}(k)\left(1 + \hat{g}_{S,LP}[b]\right) + \gamma[b]\, N_{CNG-1}(k) \tag{6.3-131}$$

and

$$\hat{R}_i[k] = N_{CNG-0}(k)\left(1 - \hat{g}_{S,LP}[b]\right) - \gamma[b]\, N_{CNG-1}(k) \tag{6.3-132}$$

for k ∈ I_b, where I_b is the set of coefficients k belonging to frequency band b and $\gamma[b]$ is based on the coherence as

$$\gamma[b] = \begin{cases} \sqrt{\frac{\hat{C}_{band}[b]}{1-\hat{C}_{band}[b]} + 1 - \hat{g}_{S,LP}^2[b]} - \sqrt{\frac{\hat{C}_{band}[b]}{1-\hat{C}_{band}[b]}} & if \quad \hat{C}_{band}[b] < 0.9 \\ 0 & otherwise \end{cases} \tag{6.3-133}$$

The channels are then rotated for reinjecting the global IPD:

$$\begin{cases} \hat{L}_i[k] = \hat{L}_i[k]e^{j\ gIPD_{syn}} \\ \hat{R}_i[k] = \hat{R}_i[k] \end{cases} \tag{6.3-134}$$

The left and right channel waveforms are then generated by applying STFT synthesis to the reconstructed left and right spectra as specified in clause 6.3.3.2.8. Finally, the ITD between the channels is synthesized as described in clause 6.3.3.1.1 using ITD_{syn} as obtained in 6.3.6.1.3.

6.3.6.2 DTX in MDCT-based stereo

6.3.6.2.1 General overview

In MDCT-based stereo, the SID payload consists of parametric noise data for the two input channels in a mid/side representation, a coherence value indicating how correlated the background noise seen at the encoder is and a one-bit value indicating whether the energy of the side representation of the noise data is lower than a threshold (the "no-side flag").

6.3.6.2.2 Decoding of parametric noise data

Decoding of the noise data follows the MSVQ decoding approach described in clause 5.3.3.5.2.4. For decoding the mid noise data, all 6 stages of the MSVQ are used while for the side noise data only the first four are used. The no-side flag is represented by a single bit and thus read directly from the bitstream. If the value of the no-side flag is 1, the mid noise data vector is set to zero. Then, the MSVQ-decoded parametric noise data is reconverted from a mid/side representation to a left/right representation. Thus, the reconstructed parametric noise data for each channel is calculated as

$$N_{L,FD-CNG}^{SID}(i) = \begin{cases} N_{M,FD-CNG}^{SID}(i)\ if\ no-side\ is\ 1, \\ N_{M,FD-CNG}^{SID}(i) + N_{S,FD-CNG}^{SID}(i)\ if\ no-side\ is\ 0. \end{cases} \tag{6.3-135}$$

$$N_{R,FD-CNG}^{SID}(i) = \begin{cases} N_{M,FD-CNG}^{SID}(i)\ if\ no-side\ is\ 1, \\ N_{M,FD-CNG}^{SID}(i) - N_{S,FD-CNG}^{SID}(i)\ if\ no-side\ is\ 0. \end{cases} \tag{6.3-136}$$

Before generating the actual stereo CNG signal, processing steps described in clauses 6.7.3.1.2 and 6.7.3.1.3 of [3] are applied separately on the noise data for each channel.

6.3.6.2.3 Stereo CNG

The two channels in the CNG output signal of high-rate stereo are generated using a stereo signal generator which consists of three uncorrelated gaussian noise sources and a mixer to mix the generated noise signals controlled by the coherence value decoded from the last SID.

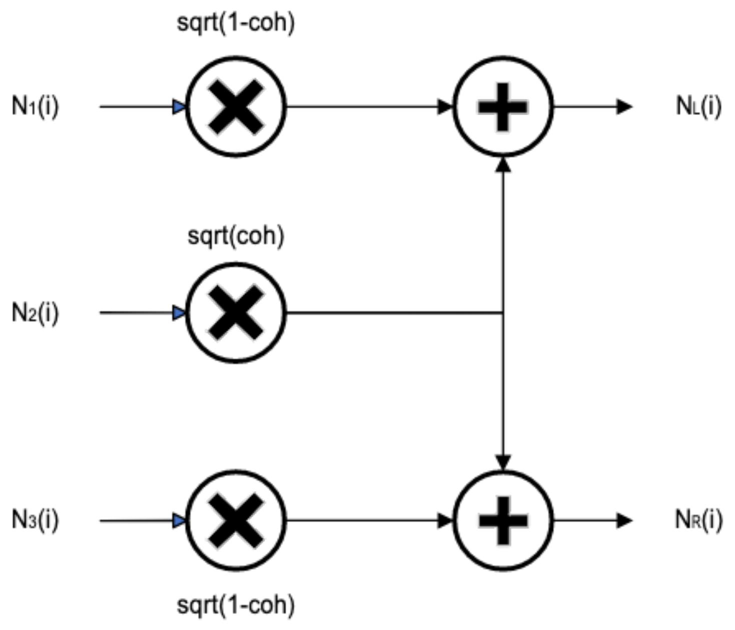


Figure 6.3-4: Stereo noise signal generator for CNG in MDCT-based Stereo DTX

The comfort noise generation process is illustrated in Figure 6.3-4. For each of the two channels, a separate gaussian noise source generates a separate noise signal (*$N_1(i)$* or *$N_3(i)$*, respectively, the "channel noise signals") with both noise signals being decorrelated between each other. A third gaussian noise source (the "mixing noise source", *$N_2(i)$*) generates a third noise signal that is also decorrelated from each of the two other noise signals. The two channel noise signals are then mixed with the mixing noise signal, i.e., the mixing noise signal is added to both channel noise signals after weighting all of the signals. The coherence value decoded from the last received SID frame serves as a control parameter for the weighting. With *$N_l(i)$* being the noise signal for the left channel, *$N_r(i)$* the noise signal for the right channel and *coh* being the coherence value, mixing of the noise signals is done like so:

$$\mathrm{N_L(i)} = \sqrt{1-\mathrm{coh}} \times \mathrm{N_1(i)} + \sqrt{\mathrm{coh}} \times \mathrm{N_2(i)} \tag{6.3-137}$$

$$\mathrm{N_R(i)} = \sqrt{1-\mathrm{coh}} \times \mathrm{N_3(i)} + \sqrt{\mathrm{coh}} \times \mathrm{N_2(i)} \tag{6.3-138}$$

A higher coherence between the channels thus leads to more correlated noise being generated in the channels, while a lower coherence value leads to a higher amount of uncorrelated noise in the stereo output. This way, the inter-channel coherence in the comfort noise signal follows the coherence of the background noise seen in the encoder and a similar spatial expression is achieved.

Next, both channel noise signals are spectrally shaped by employing FD-CNG as described in the EVS specification [3] clause 6.7.3.3. This is done separately on both channels.

6.3.6.2.4 Smoothing of transitions from DTX frame to active TCX

Especially for very low-frequency background noises with high spectral tilt such as car noise, there can be transition artifacts at the border between an inactive frame and an active frame coded with TCX. To mitigate these artifacts, a mean filter is applied on the output signal around the transition borders by crossfading between the unprocessed output signal and the mean-filtered output signal. This way, the maximum smoothing effect is applied on the transition part only and no adjacent signal parts are affected by the lowpass effect of the mean filter. The smoothing is applied after step TODO: needs reference to correct ivas core switching code!

First, the unprocessed output signal s is filtered using a mean filter:

$$s_{filtered}(i) = \frac{1}{w}\sum_{j=i-\left\lfloor\frac{L}{2}\right\rfloor}^{i+\left\lfloor\frac{L}{2}\right\rfloor} s(j),\ i = 0,\dots,2d \tag{6.3-139}$$

where d is the delay compensation length in samples and $s(j) = s(0), if\ j < 0$. L denotes the length of the filter memory. It depends on the output sampling rate and is 15 for 16kHz output sampling rate, 31 for 32kHz output sampling rate and 47 for 48kHz output sampling rate. This filtered signal is then crossfaded with the unprocessed signal so that the filtered signal is faded in completely at the frame transition border and faded out again afterwards:

$$s_{smooth}(i) = fade(i)\ s_{filtered}(i) + \left(1 - fade(i)\right) s(i), i = 0, \dots, 2d \tag{6.3-140}$$

The crossfade function is defined as:

$$fade(i) = \begin{cases} \frac{i}{d}, for\ 0 < i \le d \\ 2 - \frac{i}{d}, for\ d < i \le 2d \end{cases} \tag{6.3-141}$$

6.3.7 Stereo output format conversion

6.3.7.1 Stereo-to-Mono output

6.3.7.1.1 DFT stereo mono output

6.3.7.1.1.1 13.2 – 24.4 kbps: direct output of decoded core signal

In most cases, outputting a mono signal is trivial in DFT Stereo. Since it is a parametric approach to stereo that relies on a single downmix channel and parametric side information for upmixing back to stereo output, a suitable mono signal is already available after the core decoder. For DFT Stereo bitrates <= 24.4 kbps, this mono downmix signal was generated at the encoder entirely via an active downmixing scheme (as described in 5.3.3.2.2.9) and can be used as mono output directly, bypassing the need for any stereo upmix processing in DFT domain.

6.3.7.1.1.2 32 kbps: conversion of residual downmix to active downmix

For the bitrate of 32 kbps, the DFT Stereo processing is not entirely parametric, anymore. While the upper part of the downmix spectrum (above 1 kHz) is still generated with the aforementioned active downmixing scheme, the lower part of the spectrum is downmixed via an M/S transform. The mid-signal of this transform becomes the downmix signal in this frequency range while the side (or residual) signal of the M/S downmix is separately coded and also transmitted in the bitstream. This is described at length in 5.3.3.2.2.11.

This means that for 32 kbps a mono signal suitable for direct decoder output is not available, as the transmitted core signal was generated by 2 different downmixing schemes. Particularly, the mid-signal used in the lower part of the spectrum does not always give the best quality for playback compared to the active downmix of the upper part of the spectrum. Therefore, it is desirable to harmonize all parts of the spectrum of the decoded core signal by also converting the lower part to an active downmix. The necessary decoder processing for this conversion is illustrated in Figure 6.3-5.

Unlike the lower bitrates – DFT domain processing cannot be skipped at 32 kbps. Instead, both the decoded core signal and the decoded residual signal (see 6.3.3.2.4) have to be converted to DFT domain. There, the part of the spectrum that uses the residual signal is upmixed in the same way as for regular stereo output, as described in clause 6.3.3.2.7. After obtaining the stereo representation, the according frequency bands can be downmixed again using the active downmixing scheme (the same way as it is done at the encoder for the higher bands, see again clause 5.3.3.2.2.9). This results in a final harmonized downmix signal where the combined lower and upper parts of the spectrum are based on the same downmixing scheme. This signal is now transformed back to time-domain via inverse DFT and rendered as mono output.

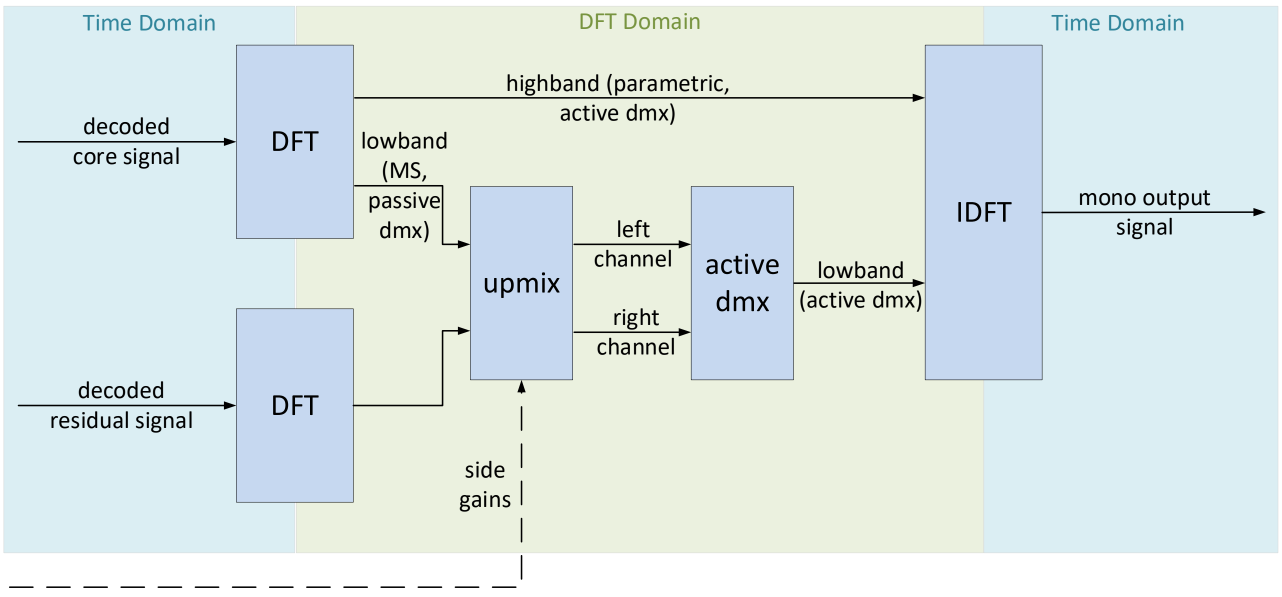


Figure 6.3-5: Mono output at 32 kbps DFT Stereo

6.3.7.1.2 MDCT stereo mono output

6.3.7.1.2.1 Introduction

For generating mono output from an MDCT Stereo bitstream, a dedicated downmixing scheme is employed towards the end of the MDCT Stereo decoding process (see clause 6.3.4). To benefit the quality of the mono output and to avoid the issues coming with a simple passive downmix, the downmix is done in an active manner by estimating and applying adaptive bandwise weights for the decoded stereo channels in MDCT domain. The actual downmix to a single mono channel is done later after transforming back to time-domain since direct downmixing in spectral domain cannot be done if the time resolution between the channels is different (TCX20 vs TCX5/10). The general scheme of this hybrid approach is illustrated in Figure 6.3-6.

6.3.7.1.2.2 Computation of downmix weights

The first step of this downmixing process (estimating and applying the downmix weights) happens after all the core decoder and stereo decoder processing is done and directly before the IMDCT back transform. The weights are computed based on a given target energy corresponding to the energy of the phase-rotated mid-channel (in a manner similar to the one employed in the downmix of the DFT-based stereo encoder (as described in 5.3.3.2.2):

$$E_{target} = \left|\frac{L+Re^{-j\varphi}}{2}\right|^2 = \frac{\langle L,L\rangle+\langle R,R\rangle+2|\langle L,R\rangle|}{4} = \frac{|L|^2+|R|^2+2|\langle L,R\rangle|}{4}, \qquad (6.3\text{-}142)$$

where L and R represent the left and right channel spectral magnitudes. Based on this target energy the channel weights can be computed for each spectral band as follows:

$$w_R = \frac{1}{2\sqrt{2}}\frac{\sqrt{|L|^2+|R|^2+2|\langle L,R\rangle|}}{|L|+|R|} \qquad (6.3\text{-}143)$$

and

$$w_L = w_R + 1 - \frac{|L+R|}{|L|+|R|}. \qquad (6.3\text{-}144)$$

These weights or band-wise weighting values wR and wL are computed per spectral band with each band encompassing several MDCT bins where the size of the bands increase towards higher frequencies. Concretely, the same bitrate-dependent band configurations as in the regular MDCT Stereo En-/Decoding (see Table 5.3-14) are used also for the mono downmixing.

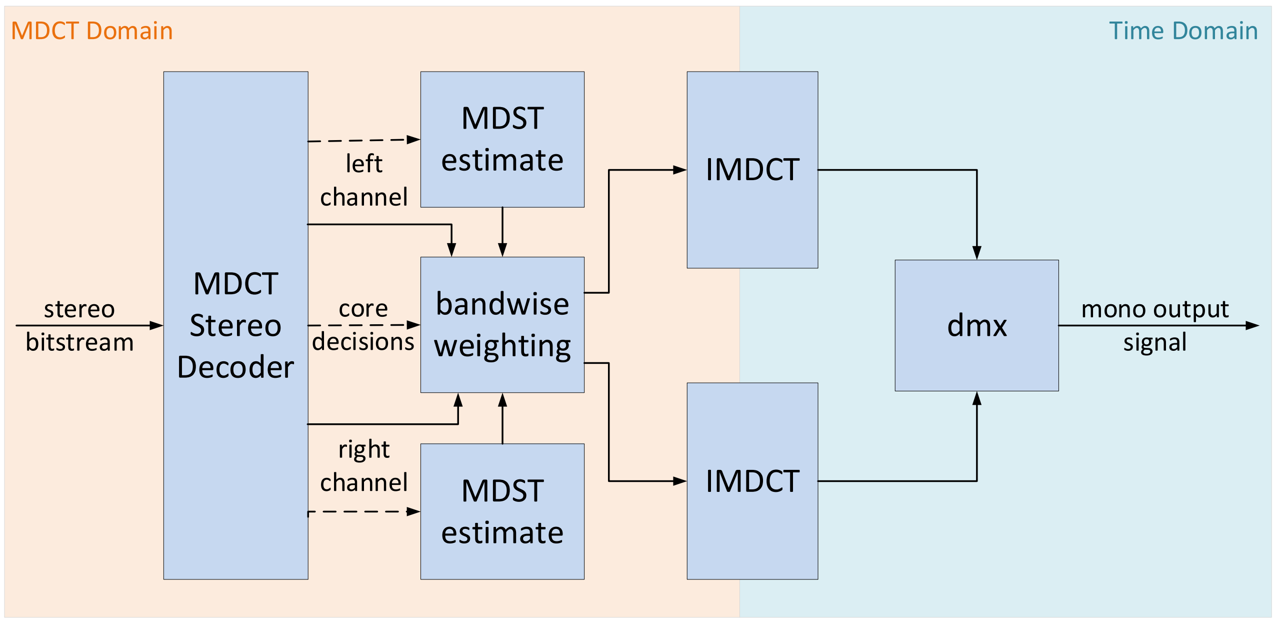


Figure 6.3-6: Mono output for MDCT Stereo

As the transmitted MDCT coefficients are only real-valued, the complementary MDST values that are required for energy-preserving weighting are obtained for each channel by the estimate

$$MDST_i = MDCT_{i+1} - MDCT_{i-1}, \tag{6.3-145}$$

where i specifies the spectral bin number.

Using this estimate |L| and |R| are computed for each band i as

$$|L| = \sqrt{\Sigma_{i\ in\ b}(MDCT_{i,l}{}^2 + MDST_{i,l}{}^2)}\ ,\ |R| = \sqrt{\Sigma_{i\ in\ b}(MDCT_{i,r}{}^2 + MDST_{i,r}{}^2)} \tag{6.3-146}$$

$|L+R|$ is computed as

$$|L+R| = \sqrt{|L|^2 + |R|^2 + 2\left(\Sigma_{i\ in\ b}(MDCT_{i,l}MDCT_{i,r} + MDST_{i,l}MDST_{i,r})\right)^2} \tag{6.3-147}$$

and $|\langle L,R\rangle|$ is computed as the magnitude or absolute value of the complex dot product

$$|\langle L,R\rangle| = \sqrt{\left(\Sigma_{i\ in\ b}(MDCT_{i,l}MDCT_{i,r} + MDST_{i,l}MDST_{i,r})\right)^2 + \left(\Sigma_{i\ in\ b}(MDST_{i,l}MDCT_{i,r} - MDCT_{i,l}MDST_{i,r})\right)^2} \tag{6.3-148}$$

where i specifies the bin number inside spectral band b.

6.3.7.1.2.3 Weighting of stereo channels

The calculated weights $w_{L,b}$ and $w_{R,b}$ are then applied to every MDCT bin *i* inside their respective band *b*, thereby creating weighted MDCT spectra of the stereo channels:

$$MDCT_{i,l,weighted} = w_{L,b} MDCT_{i,l} \tag{6.3-149}$$

and

$$MDCT_{i,r,weighted} = w_{R,b} MDCT_{i,r} \tag{6.3-150}$$

6.3.7.1.2.4 Time-domain downmix

In a second step, the two weighted channels *L* and *R* are then transformed back to time domain via IMDCT and downmixed to a single downmix channel *D* by simple summing and scaling of each time-domain sample *i* of *L* and *R*:

$$D_i = \frac{1}{\sqrt{2}}(L_i + R_i) \qquad (6.3\text{-}151)$$

6.3.7.1.2.5 Conversion of MDCT coefficients to higher resolution

For the special case of different cores in the two channels the computation of the cross-spectra correlation as part of the weight calculation has to be slightly adapted. Due to the different frequency and time resolutions of TCX20 and TCX10, directly calculating the dot product between left and right is not possible. In order to make it possible, the spectrum of the (sub)frame with the lower spectral resolution is converted into an approximation of a spectrum with twice the spectral resolution by computing:

$$MDCT_{2i,k0} = \frac{1}{2}\left(MDCT_{i,k1} + (-1)^i MDCT_{i,k0}\right) \qquad (6.3\text{-}152)$$

and

$$MDCT_{2i+1,k0} = \frac{1}{2}\left(MDCT_{i,k1} - (-1)^i MDCT_{i,k0}\right) \qquad (6.3\text{-}153)$$

where i specifies the spectral bin number and k0 and k1 the subframes with lower resolution. These additions and subtractions can be seen as high- and lowpass filtering operations that split one lower-resolution bin into two higher-resolution bins where the filtering depends on whether the bin number i is even or odd (starting with i = 0 for the lowest bin).

This means that if one channel is TCX20, the other channel is converted to the same spectral resolution. If one or both of the subframes of the other channel are subdivided again into two TCX5 "sub-subframes", these are first converted to TCX10 resolution by the same filtering before splitting again to arrive at the final TCX20 representation. An example of such a conversion of bins with lower time resolution to higher resolution is shown in Figure 6.3-7.

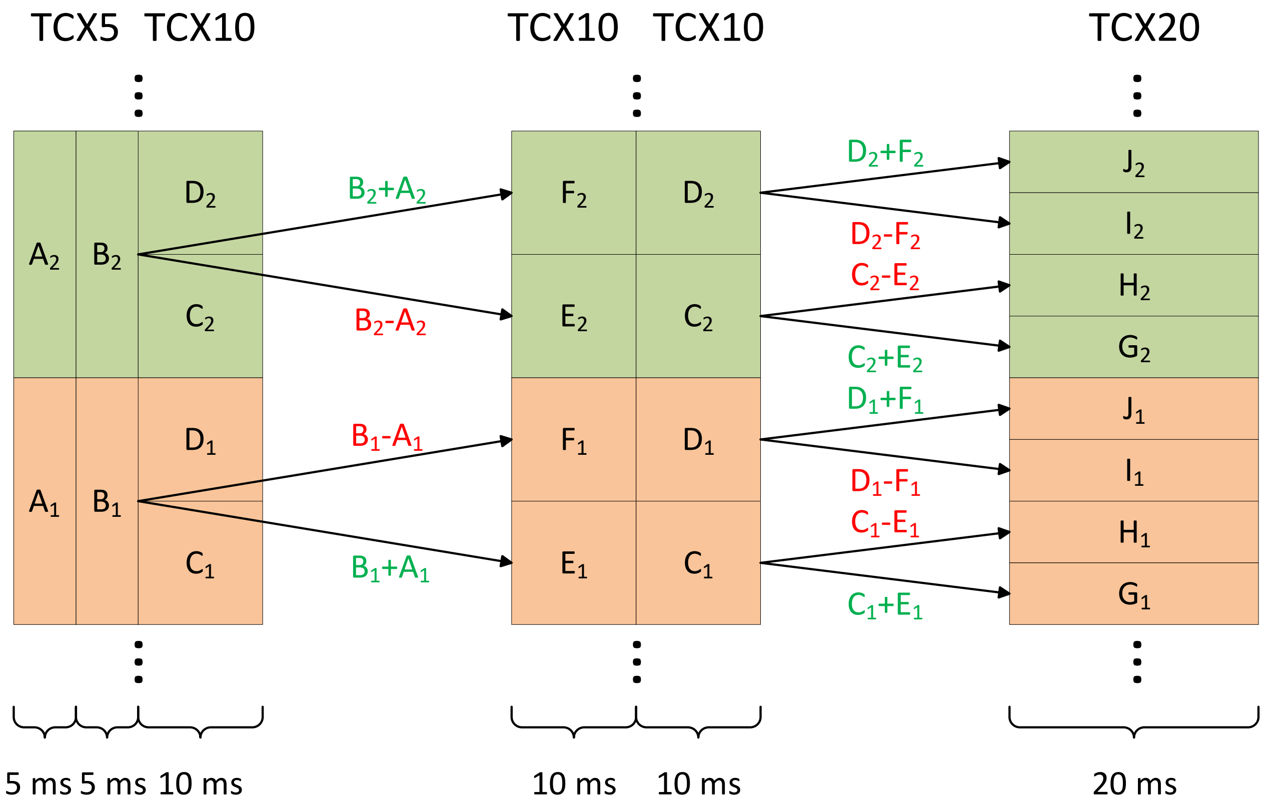


Figure 6.3-7: Example of conversion of MDCT coefficients

Even if none of the channels is TCX20, conversion to the higher resolution may still be necessary for one or both subframes in case there is TCX10 in one channel and TCX5 in the other. As an example, if the left channel is TCX10 in subframe A and 2 x TCX5 in subframe B, while the right channel is 2 x TCX5 in subframe A and TCX10 in subframe

B, both channels will be converted to have TCX10 resolution in both subframes (convert subframe B for left channel, A for right channel). If in the same example the right channel is also TCX 10 for subframe A and 2 x TCX5 for B, then no conversion is done; i.e. subframe A will be downmixed with TCX10 resolution, B with TCX5.

The MDST estimates and the final channel weights are then computed using these converted spectra. The weights themselves are applied to the original input spectra which means that in case of a conversion each computed weight is applied to all bins covering the same frequency range in the original lower resolution for every subframe.

6.3.7.1.2.6 CNG for mono output

In CNG operation, instead of always generating two channels of comfort noise and downmixing to a single channel afterwards, only one channel of comfort noise is generated in the first channel and shaped using an average noise parameter data vector. Additionally, some buffers are synchronized or passively downmixed. In transition frames (inactive frames following an active frame or active frames following an inactive frame), a crossfade is performed between the passive downmix of the two channels and the mono comfort noise signal in the first channel.

In a transition-to-CNG frame, two channels of comfort noise are still generated and the passive downmix is applied as in active frame decoding. Afterwards, a passive downmix with equal weighting is applied to the noise parameter data vector to generate a mid channel parametric noise data vector which will be used for CNG in this inactive frame period:

$$N^{SID}_{M,FD-CNG}(i) = \frac{N^{SID}_{L,FD-CNG}(i)+N^{SID}_{R,FD-CNG}(i)}{2}. \qquad (6.3\text{-}154)$$

For all following SID frames in this inactive frame period, this conversion is done before CNG operation. The CNG operation itself in all following frames of this inactive-frame period is done only once in the first channel to generate a single output channel using the mid channel parametric noise data vector. This process is the same as FD-CNG for mono as described in EVS. To keep buffers consistent in both channels for possibly switching back to active frame decoding in the next frame, the output buffers of the first channel are copied over to the second channel after CNG operation is finished for the current frame. This way, output buffers for the second channel always contain meaningful data even though CNG synthesis is only done in the first channel.

In transition frames, decoding operation switches between generating only one channel (in mono CNG) and generating two channels and applying the downmix. To achieve smooth transitions between these two output scenarios, a crossfade is performed in every transition frame which blends between the single output channel of the CNG operation and the downmix of the actively decoded channels. For transitions **to** active decoding frames, the following equation applies.

$$D_{out,i} = \begin{cases} D_{CNG,i} \times \frac{i}{L_{crossfade}} + D_i \times \left(1 - \frac{i}{L_{crossfade}}\right), for\ i < L_{crossfade} \\ D_i, for\ i \geq L_{crossfade} \end{cases} \qquad (6.3\text{-}155)$$

where $L_{crossfade}$ depends on the output sampling rate and is 96 for 48kHz, 64 for 32kHz and 32 for 16kHz. For transitions from active decoding frames, the following equation is used

$$D_{out,i} = \begin{cases} D_{CNG,i} \times \left(1 - \frac{i}{L_{crossfade}}\right) + D_i \times \frac{i}{L_{crossfade}}, for\ i < L_{crossfade} \\ D_{CNG,i}, for\ i \geq L_{crossfade} \end{cases}. \qquad (6.3\text{-}156)$$

Here, $L_{crossfade}$ is set to a quarter of the current frame length and is 240 for 48kHz, 160 for 32kHz and 80 for 16kHz. In both equations, D_i denotes the passively downmixed signal as given by equation ().

6.4 Scene-based audio (SBA) decoding

6.4.1 Combined DirAC and SPAR based SBA decoding overview

Figure 6.4-1 shows the overview of SBA decoding in IVAS. The decoder receives IVAS bitstream as described in subclause 8.3. From IVAS bitstream, the SBA decoder reads core coder bits, which can correspond to SCE, CPE or MCT. The selection of core coder is as described in Table 5.4-1. Core coder bits are decoded by corresponding core decoder outputting downmix signal $s^{DMX}(n)$ with N_{dmx} channels. From IVAS bitstream, the SBA decoder reads DirAC and SPAR metadata bits and the spatial metadata decoder decodes DirAC and SPAR metadata. The downmix signal $s^{DMX}(n)$, DirAC and SPAR metadata are then upmixed and rendered to one of the supported output formats in IVAS.

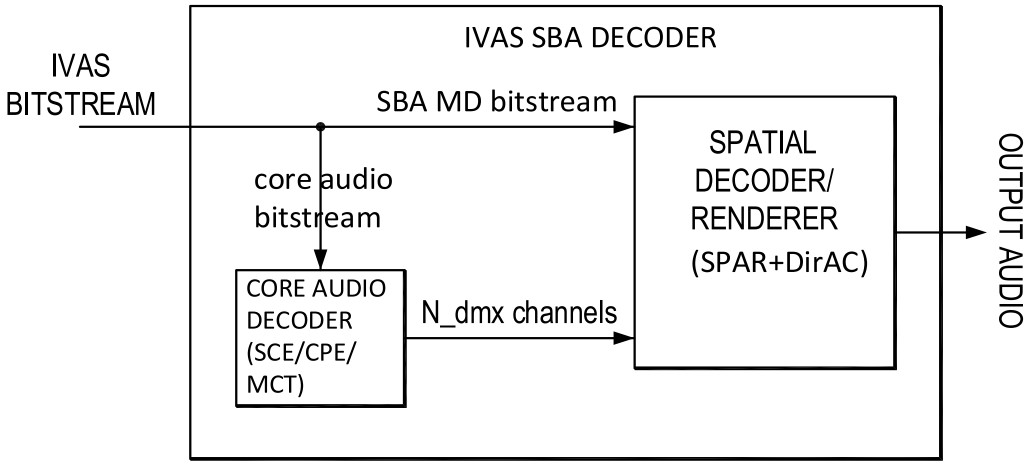


Figure 6.4-1: SBA decoding architecture in IVAS

6.4.1.1 FOA signal decoding at all bitrates and HOA2, HOA3 signal decoding at bitrates below 256 kbps

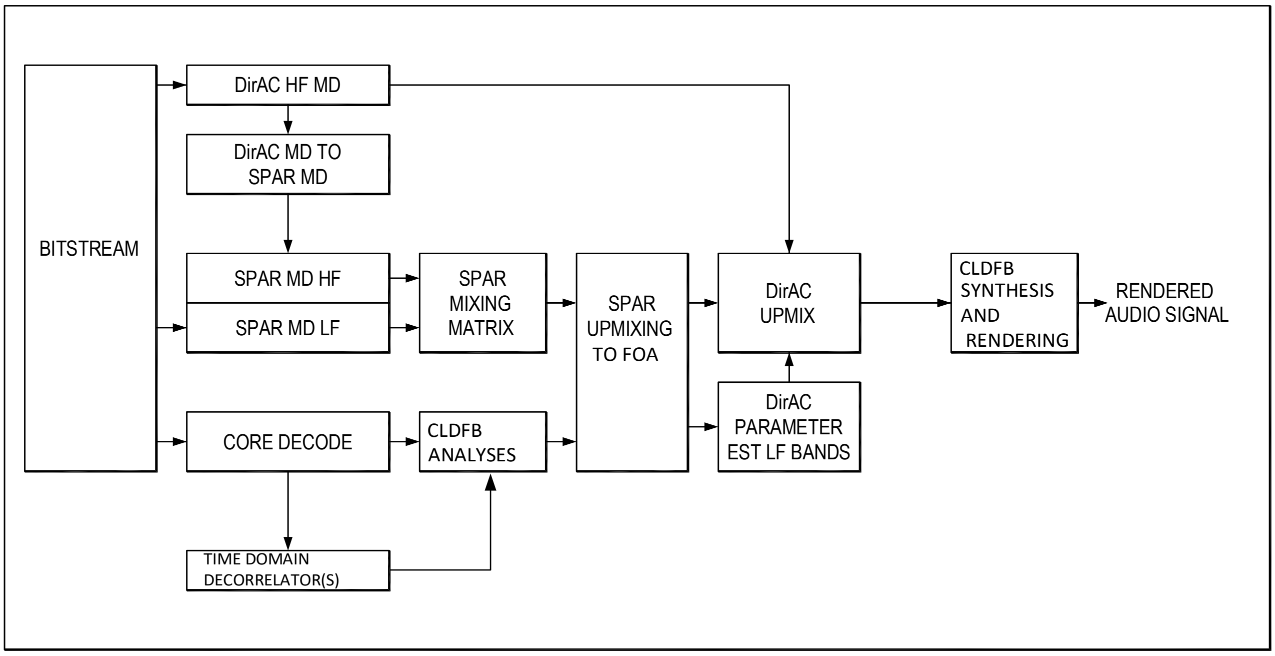


Figure 6.4-2: FOA signal decoding at all bitrates and HOA2, HOA3 signal decoding at bitrates below 256 kbps

Figure 6.4-2 shows the block diagram of SBA decoding of FOA signal at all bitrates and HOA2, HOA3 signal at bitrates below 256 kbps. The decoding is done in following stages.

- Metadata decoding
- Core coder decoding
- SBA upmix and rendering

<….>

6.4.1.1.1 Metadata decoding

DirAC metadata decoder block (DirAC HF MD) reads DirAC metadata bits from IVAS bitstream and decodes frequency banded DirAC parameters in high frequency range f^{DirAC}, such that f^{DirAC} > 4.4 KHz, as described in subclause 6.4.2. The decoded DirAC parameters are processed through DirAC to SPAR converter, as described in subclause 6.4.4.3, to generate frequency banded SPAR parameters in high frequency range f^{DirAC}.

SPAR metadata decoder block (SPAR MD LF) reads SPAR metadata bits from IVAS bitstream and decodes frequency banded SPAR parameters in low frequency range f^{SPAR}, such that f^{SPAR} <= 4.4 KHz, as described in subclause 6.4.3. SPAR parameters in low and high frequency bands are then converted into the upmix matrix $umx_{[N_{spar_{ana}} x \, N_{dmx}]}$ as described in subclause 6.4.5 where SPAR metadata analysis channels, N_{spar_ana}, are described in Table 5.4-2 and downmix channels N_{dmx} are described in Table 5.4-1.

6.4.1.1.2 Core coder decoding

Core coder decoder block (CORE DECODE) reads core coder bits from IVAS bitstream and decodes SBA downmix signal S'_{dmx} using either SCE, CPE or MCT decoder based on the mapping described in Table 5.4-1. S'_{dmx} is then high pass filtered as per subclause 6.2.1.1 and outputs signal S_{dmx}^{hp20} .

6.4.1.1.3 SBA upmix and rendering

The high pass filtered downmix signal S_{dmx}^{hp20} , upmix matrix $umx_{[N_{spar_{ana}} x \, N_{dmx}]}$ and decoded metadata are then processed by SBA upmix and rendering signal chain to generate IVAS output signal S_{out}. SBA upmix and rendering signal chain includes following steps.

First, the primary downmix channel W'' of downmix signal S_{dmx}^{hp20} is processed by Time Domain Decorrelator block, as described in detail in subclause 6.4.6.4.1, which generates N_{decorr} channel signal S_{decorr} that is decorrelated with respect W'', here $N_{decorr} = N_{spar\_ana} - N_{dmx}$. Then CLDFB analysis is done on S'_{dmx} and S_{decorr} , as described in subclaus e6.2.5.2, that generates CLDFB domain signal S^{cldfb}. SPAR upmixing block then applies the upmix matrix $umx_{[N_{spar_{ana}} x \, N_{dmx}]}$ to the CLDFB signal S^{cldfb} and outputs FOA signal $S_{umx,foa}^{cldfb}$ in CLDFB domain, the upmixing process is described in detail in subclause 6.4.6.4.4If the output format is FOA then $S_{umx,foa}^{cldfb}$ is synthesized as described in subclause 6.2.5.3 to generate IVAS output signal S_{out}. For output formats other than FOA or MONO or STEREO, SPAR upmixed signal $S_{umx,foa}^{cldfb}$ is processed through DirAC decoder block which does higher order ambisonics processing <…..> and generates IVAS output signal S_{out}.

For MONO and STEREO output formats, the primary downmix channel W'' of downmix signal S_{dmx}^{hp20} and upmix matrix $umx_{[N_{spar_{ana}} x \, N_{dmx}]}$ is processed by format conversion as described in subclause 6.4.10.

6.4.1.2 HOA2 and HOA3 signal decoding at bitrate 256 kbps

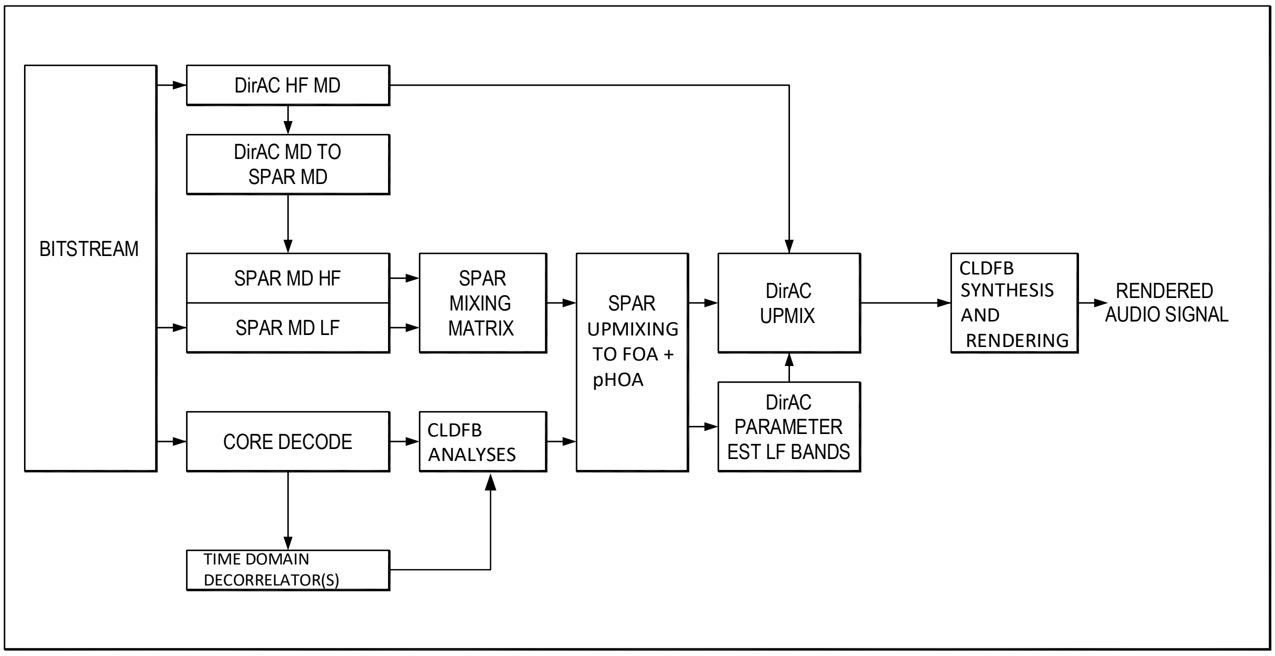


Figure 6.4-3: HOA2 and HOA3 signal decoding at bitrate 256 kbps

Figure 6.4-3 shows the block diagram of SBA decoding of HOA2 and HOA3 signal at IVAS bitrate 256 kbps. The decoding is done in following stages.

- Metadata decoding
- Core coder decoding
- SBA upmix and rendering

<…>

6.4.1.2.1 Metadata decoding

DirAC metadata decoder block (DirAC HF MD) reads DirAC metadata bits from IVAS bitstream and decodes frequency banded DirAC parameters in high frequency range $\boldsymbol{f}^{\boldsymbol{DirAC}}$, such that $\boldsymbol{f}^{\boldsymbol{DirAC}}$ > 4.4 KHz, as described in subclause 6.4.2. The decoded DirAC parameters are processed through DirAC to SPAR converter, as described in subclause 6.4.4.3, to generate frequency banded SPAR parameters corresponding to FOA channels in the high frequency range $\boldsymbol{f}^{\boldsymbol{DirAC}}$.

SPAR metadata decoder block (SPAR MD LF) reads SPAR metadata bits from IVAS bitstream and decodes frequency banded SPAR parameters, as described in subclause 6.4.3. Decoded SPAR parameters include SPAR parameters corresponding to FOA channels in the low frequency range $\boldsymbol{f}^{\boldsymbol{SPAR}}$, such that $\boldsymbol{f}^{\boldsymbol{SPAR}}$ <= 4.4 KHz, and SPAR parameters corresponding to HOA channels in both $\boldsymbol{f}^{\boldsymbol{SPAR}}$ and $\boldsymbol{f}^{\boldsymbol{DirAC}}$ frequency ranges. SPAR parameters in low and high frequency bands are then converted into the upmix matrix $umx_{[N_{spar_{ana}} x\, N_{dmx}]}$ as described in subclause 6.4.5, here SPAR metadata analysis channels, N_{spar_ana} , are described in Table 5.4-2 and downmix channels N_{dmx} are described in Table 5.4-1.

6.4.1.2.2 Core coder decoding

Core coder decoder block (CORE DECODE) reads core coder bits from IVAS bitstream and decodes SBA downmix signal S'_{dmx} using MCT decoder based on the mapping described in Table 5.4-1. S'_{dmx} is then high pass filtered as per subclause 6.2.1.1 and outputs signal S_{dmx}^{hp20} .

6.4.1.2.3 SBA upmix and rendering

The high pass filtered downmix signal S_{dmx}^{hp20} , upmix matrix $umx_{[N_{spar_{ana}} x\, N_{dmx}]}$ and decoded metadata are then processed by SBA upmix and rendering signal chain to generate IVAS output signal S_{out}. SBA upmix and rendering signal chain includes following steps.

First, the primary downmix channel W'' of downmix signal S_{dmx}^{hp20} is processed by Time Domain Decorrelator block, as described in detail in subclause 6.4.6.4.1which generates N_{decorr} channel signal S_{decorr} that is decorrelated with respect W'', here $N_{decorr} = N_{spar\_ana} - N_{dmx}$. Then CLDFB analysis is done on S_{dmx}' and S_{decorr} , as described in subclause 6.2.5.2 that generates CLDFB domain signal S^{cldfb}. SPAR upmixing block then applies the upmix matrix $umx_{[N_{spar_{ana}} x\, N_{dmx}]}$ to the CLDFB signal S^{cldfb} and outputs FOA and planar HOA signal $S_{umx,hoa}^{cldfb}$ in CLDFB domain, the upmixing process is described in detail in subclause 6.4.6.4.4. If the output format is FOA then FOA component of $S_{umx,hoa}^{cldfb}$ is synthesized as described in subclause 6.2.5.3 to generate IVAS output signal S_{out}. For output formats other than FOA or MONO or STEREO, SPAR upmixed signal $S_{umx,hoa}^{cldfb}$ is processed through DirAC decoder block which does higher order ambisonics processing <…..> and generates IVAS output signal S_{out}.

For MONO and STEREO output formats, the primary downmix channel W'' of downmix signal S_{dmx}^{hp20} and upmix matrix $umx_{[N_{spar_{ana}} x\, N_{dmx}]}$ is processed by format conversion as described in subclause 6.4.10.

6.4.1.3 HOA2 and HOA3 signal decoding at bitrates 384 kbps

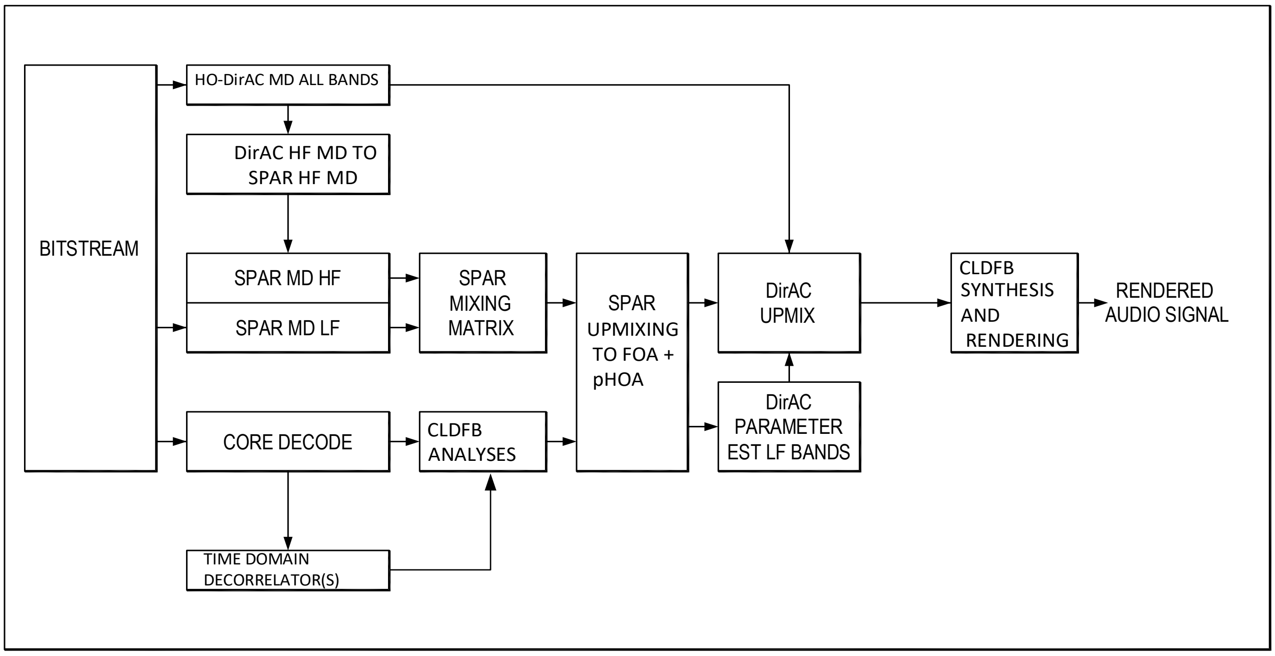

BITSTREAM
HO-DirAC MD ALL BANDS
DirAC HF MD TO SPAR HF MD
SPAR MD HF
SPAR MD LF
SPAR MIXING MATRIX
SPAR UPMIXING TO FOA + pHOA
DirAC UPMIX
DirAC PARAMETER EST LF BANDS
CLDFB SYNTHESIS AND RENDERING
RENDERED AUDIO SIGNAL
CORE DECODE
CLDFB ANALYSES
TIME DOMAIN DECORRELATOR(S)

Figure 6.4-4: HOA2 and HOA3 signal decoding at bitrate 384 kbps

Figure 6.4-4 shows the block diagram of SBA decoding of HOA2 and HOA3 signal at IVAS bitrate 384 kbps. The decoding is done in following stages.

- Metadata decoding
- Core coder decoding
- SBA upmix and rendering

<….>

6.4.1.3.1 Metadata decoding

Higher Order DirAC metadata decoder block (HO- DirAC MD) reads DirAC metadata bits from IVAS bitstream and decodes frequency banded DirAC parameters in in both $\boldsymbol{f^{SPAR}}$ and $\boldsymbol{f^{DirAC}}$ frequency ranges, as described in subclause 6.4.2. The decoded DirAC parameters in $\boldsymbol{f^{DirAC}}$ frequency range are processed through DirAC to SPAR converter, as described in subclause 6.4.4.3 to generate frequency banded SPAR parameters corresponding to FOA channels in the high frequency range $\boldsymbol{f^{DirAC}}$.

SPAR metadata decoder block (SPAR MD LF) reads SPAR metadata bits from IVAS bitstream and decodes frequency banded SPAR parameters, as described in subclause 6.4.3 and 6.4.5, here SPAR metadata analysis channels, N_{spar_ana}, are described in Table 5.4-2 and downmix channels N_{dmx} are described in Table 5.4-1.

6.4.1.3.2 Core-coder decoding

Core coder decoder block (CORE DECODE) reads core coder bits from IVAS bitstream and decodes SBA downmix signal S'_{dmx} using MCT decoder based on the mapping described in Table 5.4-1. S'_{dmx} is then high pass filtered as per subclause 6.2.1.1 and outputs signal S_{dmx}^{hp20} .

6.4.1.3.3 SBA upmix and rendering

The high pass filtered downmix signal S_{dmx}^{hp20} , upmix matrix $umx_{[N_{spar_{ana}} x\, N_{dmx}]}$ and decoded metadata are then processed by SBA upmix and rendering signal chain to generate IVAS output signal S_{out}. SBA upmix and rendering signal chain includes following steps.

First, the primary downmix channel W'' of downmix signal S_{dmx}^{hp20} is processed by Time Domain Decorrelator block, as described in detail in subclause 6.4.6.4.1, which generates N_{decorr} channel signal S_{decorr} that is decorrelated with respect W'', here $N_{decorr} = N_{spar\_ana} - N_{dmx}$. Then CLDFB analysis is done on S'_{dmx} and S_{decorr} , as described in subclause 6.2.5.2, that generates CLDFB domain signal S^{cldfb}. SPAR upmixing block then applies the upmix matrix $umx_{[N_{spar_{ana}} x\, N_{dmx}]}$ to the CLDFB signal S^{cldfb} and outputs FOA and planar HOA signal $S_{umx,hoa}^{cldfb}$ in CLDFB domain, the upmixing process is described in detail in subclause 6.4.6.4.4. If the output format is FOA then FOA component of $S_{umx,hoa}^{cldfb}$ is synthesized as described in subclause 6.2.5.3 to generate IVAS output signal S_{out}. For output formats other than FOA or MONO or STEREO, SPAR upmixed signal $S_{umx,hoa}^{cldfb}$ is processed through DirAC decoder block which does higher order ambisonics processing <…..> and generates IVAS output signal S_{out}.

For MONO and STEREO output formats, the primary downmix channel W'' of downmix signal S_{dmx}^{hp20} and upmix matrix $umx_{[N_{spar_{ana}} x\, N_{dmx}]}$ is processed by format conversion as described in subclause 6.4.10.

6.4.1.4 HOA2 and HOA3 signal coding at bitrates 512 kbps

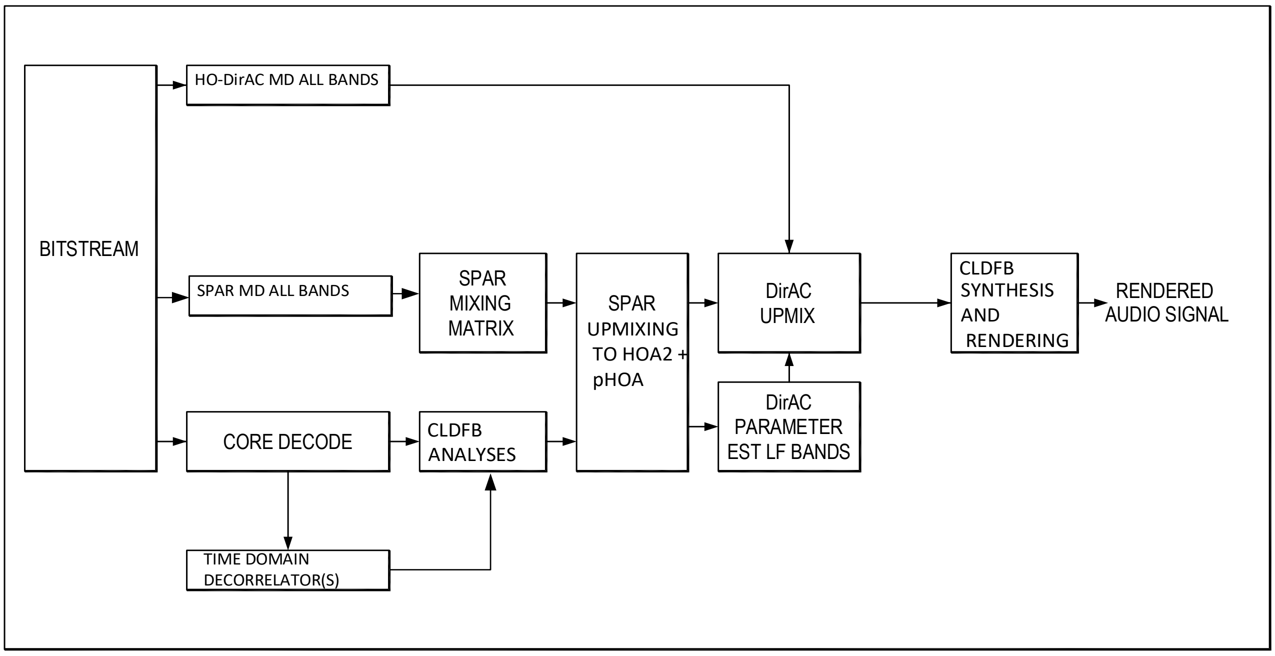


Figure 6.4-5: HOA2 and HOA3 signal decoding at bitrate 512 kbps

Figure 6.4-5 shows the block diagram of SBA decoding of HOA2 and HOA3 signal at IVAS bitrate 512 kbps. The decoding is done in following stages.

- Metadata decoding
- Core coder decoding
- SBA upmix and rendering

<….>

6.4.1.4.1 Metadata decoding

Higher Order DirAC metadata decoder block (HO- DirAC MD) reads DirAC metadata bits from IVAS bitstream and decodes frequency banded DirAC parameters in in both $\boldsymbol{f}^{\boldsymbol{SPAR}}$ and $\boldsymbol{f}^{\boldsymbol{DirAC}}$ frequency ranges, as described in subclause 6.4.2.

SPAR metadata decoder block (SPAR MD) reads SPAR metadata bits from IVAS bitstream and decodes frequency banded SPAR parameters corresponding to HOA2 and planar HOA3 channels in both $\boldsymbol{f}^{\boldsymbol{SPAR}}$ and $\boldsymbol{f}^{\boldsymbol{DirAC}}$ frequency ranges as described in subclause 6.4.3. SPAR parameters are then converted into the upmix matrix $umx_{[N_{spar_{ana}} x\, N_{dmx}]}$ as described in subclause 6.4.5here SPAR metadata analysis channels, N_{spar_ana} , are described in Table 5.4-2 and downmix channels N_{dmx} are described in Table 5.4-1.

6.4.1.4.2 Core coder decoding

Core coder decoder block (CORE DECODE) reads core coder bits from IVAS bitstream and decodes SBA downmix signal S'_{dmx} using MCT decoder based on the mapping described in Table 5.4-1. S'_{dmx} is then high pass filtered as per subclause 6.2.1.1 and outputs signal S_{dmx}^{hp20} .

6.4.1.4.3 SBA upmix and rendering

The high pass filtered downmix signal S_{dmx}^{hp20} , upmix matrix $umx_{[N_{spar_{ana}} x\, N_{dmx}]}$ and decoded metadata are then processed by SBA upmix and rendering signal chain to generate IVAS output signal S_{out}. SBA upmix and rendering signal chain includes following steps.

First, the primary downmix channel W'' of downmix signal S_{dmx}^{hp20} is processed by Time Domain Decorrelator block, as described in detail in subclause 6.4.6.4.1, which generates N_{decorr} channel signal S_{decorr} that is decorrelated with respect W'', here $N_{decorr} = N_{spar\_ana} - N_{dmx}$. Then CLDFB analysis is done on S'_{dmx} and S_{decorr} , as described in subclause 6.2.5.2, that generates CLDFB domain signal S^{cldfb}. SPAR upmixing block then applies the upmix matrix $umx_{[N_{spar_{ana}} x\, N_{dmx}]}$ to the CLDFB signal S^{cldfb} and outputs HOA2 and planar HOA3 signal $S_{umx,hoa}^{cldfb}$ in CLDFB domain, the upmixing process is described in detail in subclause 6.4.6.4.4. If the output format is FOA then the FOA component of $S_{umx,hoa}^{cldfb}$ is synthesized as described in subclause 6.2.5.3 to generate IVAS output signal S_{out}. For output formats other than FOA or MONO or STEREO, SPAR upmixed signal $S_{umx,hoa}^{cldfb}$ is processed through DirAC decoder block which does higher order ambisonics processing <…..> and generates IVAS output signal S_{out}.

For MONO and STEREO output formats, the primary downmix channel W'' of downmix signal S_{dmx}^{hp20} and upmix matrix $umx_{[N_{spar_{ana}} x\, N_{dmx}]}$ is processed by format conversion as described in subclause 6.4.10.

6.4.2 DirAC parameter decoding

SBA parameters are decoded based on metadata active/inactive mode flag W_{vad} that is read from the bitstream. W_{vad} flag indicates whether metadata was coded in active or inactive mode.

<……>

6.4.3 SPAR parameter decoding

SBA ambisonics order, that is stored as a 2 bit field in SBA header section of IVAS bitstream as described in subclause 8.3 and SBA planar mode, that is stored as a 1 bit field in SBA header section of IVAS bitstream as described in subclause 8.3, are decoded. Based on IVAS bitrate and SBA order, SBA analysis order is set as per Table 5.4-2. Then SPAR bitrate distribution table row index is computed based on IVAS bitrate and SBA analysis order as per 5.4-4.

6.4.3.1 Set active W prediction flag

Dynamic active W flag, $Dyn_{activeW}$, and active W residual index, Res_{ind}^{bs}, are decoded from metadata section of IVAS bitstream as described in Table 8.3-2 Table 8.3-3. The active W prediction flag is then set as follows.

$$Flag_{activeW} = \begin{cases} 1 & , if\ IVAS\ bitrates \leq 32kbps \\ Dyn_{activeW} & , if\ 48kbps \leq IVAS\ bitrates \leq 192kbps \\ 0 & , if\ IVAS\ bitrates \geq 256kbps \end{cases} \quad (6.4\text{-}1)$$

$$Res_{ind} = \begin{cases} Res_{ind}^{bs} & , if\ \boldsymbol{N_{dmx}} = 2 \\ 3 & , if\ \boldsymbol{N_{dmx}} = 3 \end{cases} \quad (6.4\text{-}2)$$

6.4.3.2 Decode quantization and coding strategy bits

If MD_{active} is set to one, then SPAR quantization strategy index and coding strategy index are decoded from IVAS bitstream from the SPAR quantization strategy section and SPAR coding strategy section as described in subclause 8.3. Number of SPAR quantization strategy bits, $spar_{qbits}$, depend on IVAS bitrate as given below.

$$spar_{qbits} = \begin{cases} 2 & , if\ ivas\ bitrate < 256\ kbps \\ 1 & , otherwise \end{cases}$$

Number of SPAR coding strategy bits, $spar_{CSbits}$, are fixed as given below.

$$spar_{CSbits} = 3$$

From the SPAR bitrate distribution table row index and quantization strategy index, quantization strategy is decoded as $\{qlvl_{pr}, qlvl_{cp}, qlvl_d, 1\}$, where $qlvl_{pr}$ is the number of quantization points for PR coefficients, $qlvl_{cp}$ is the number of quantization points for CP coefficients, $qlvl_D$ is the number of quantization points for D coefficients. Based on the values of $qlvl_{pr}, qlvl_{cp}, qlvl_d$ and SPAR coding strategy, corresponding Arithmetic and Huffman coder probability distribution models are read as described in subclause <…>. SPAR indices corresponding to PR, CP and D coefficients are then decoded with either arithmetic or Huffman decoder as described in subclause 6.4.3.5 and subclause 6.4.3.6. Number of PR, CP and D indices to be read from bitstream, depend on SPAR analysis channels. For FOA channels, SPAR parameters are coded for low frequency bands with band index ranging from 1 to nB_{spar}^{foa}, where $nB_{spar}^{foa} = \frac{8}{bw_{bands}}$ and bw_{bands} is given in Eqn (6.4-3), SPAR parameters for frequency band index greater than nB_{spar}^{foa} are computed from DirAC parameters as described in subclause 6.4.4.3. For HOA2 and HOA3 channels, SPAR parameters are coded for all frequency bands with band index ranging from 1 to nB_{spar}^{hoa}, where $nB_{spar}^{hoa} = \frac{12}{bw_{bands}}$ and bw_{bands} is given in Eqn (6.4-3). Number of PR, CP and D indices are given as follows.

$$N_{PR}^{indices} = 3\, nB_{spar}^{foa} + (\boldsymbol{N_{spar_{ana}}} - 4)(nB_{spar}^{hoa})$$

$$N_{CP}^{indices} = \begin{cases} (\boldsymbol{N_{dmx}} - 1)\,(\boldsymbol{N_{spar_{ana}}} - \boldsymbol{N_{dmx}})nB_{spar}^{foa} & , if\ \boldsymbol{N_{spar\_ana}} = 4 \\ (\boldsymbol{N_{dmx}} - 1)\,(\boldsymbol{N_{spar_{ana}}} - \boldsymbol{N_{dmx}})nB_{spar}^{hoa} & , if\ \boldsymbol{N_{spar_{ana}}} > 4 \end{cases}$$

$$N_{D}^{indices} = \begin{cases} (\boldsymbol{N_{spar_{ana}}} - \boldsymbol{N_{dmx}})nB_{spar}^{foa} & , if\ \boldsymbol{N_{spar\_ana}} = 4 \\ (\boldsymbol{N_{spar_{ana}}} - \boldsymbol{N_{dmx}})nB_{spar}^{hoa} & , if\ \boldsymbol{N_{spar_{ana}}} > 4 \end{cases}$$

where bw_{bands} is computed as

$$bw_{bands} = \begin{cases} 4 & , if\ W_{vad} = 0 \\ 2 & , if\ W_{vad} = 1\,, ivas\ bitrate \leq 16.4\ kbps \\ 1 & , otherwise \end{cases} \quad (6.4\text{-}3)$$

The decoded SPAR indices are then de-indexed and converted to quantized SPAR parameters as per Eqn 5.4-64).

6.4.3.3 Time differential decoding

If SPAR coding strategy is one of the four time differential coding schemes as described in Table 5.4-9 and IVAS bitrate is greater than 16.4 kbps, then the time-differentially coded bands are decoded with time-differential decoding method. Time differential decoding where quantization strategy is same between previous frame and current frame is described in subclause 6.4.3.3.1. Time differential decoding where quantization strategy is different between previous frame and current frame is described in subclause 6.4.3.3.2.

6.4.3.3.1 Time differential decoding across same quantization strategies

When quantization strategy is same between previous frame and current frame, non-differential index is computed from time-differential index in three steps. First the time-differential index is decoded from the bitstream, then the time-differential index is added to the previous frame's non-differential index as follows.

$$A_{current}^{index} = A_{TD}^{index} + A_{previous}^{index} \quad (6.4\text{-}4)$$

Then modulo $qlvl_A$ is performed to keep the indices range as $\{A_{min}^{index}, A_{max}^{index}\}$

$$A_{current}^{index} = \begin{cases} A_{current}^{index} + qlvl_A & if\ A_{current}^{index} < A_{min}^{index} \\ A_{current}^{index} - qlvl_A & if\ A_{current}^{index} > A_{max}^{index}\ , \\ A_{current}^{index} & otherwise \end{cases} \quad (6.4\text{-}5)$$

where $A_{min}^{index} = round\left(\frac{((qlvl_A - 1) * A_{min})}{A_{max} - A_{min}}\right), A_{max}^{index} = round\left(\frac{((qlvl_A - 1) * A_{max})}{A_{max} - A_{min}}\right)$, A refers to PR, CP and D coefficients and $qlvl_A$ is the number of quantization points as per the current frame's quantization strategy read from SPAR bitrate distribution table. $A_{max} and\ A_{min}$ are the minimum and maximum quantized values as per the current frame's quantization strategy as per Eqn 5.4-60, 5.4-61) and 5.4-62).

6.4.3.3.2 Time differential decoding across different quantization strategies

When quantization strategy is different between previous frame and current frame, non-differential index is computed from time-differential index in four steps. First the time-differential index is decoded from the bitstream, then the previous frame's non-differential index is mapped to current frame's quantization strategy as per Eqn (5.4-69), then the time-differential index is added to the mapped previous frame's non-differential index as follows.

$$A^{index}_{current} = A^{index}_{TD} + A^{index}_{previous,mapped} \qquad (6.4\text{-}6)$$

Then modulo $qlvl_A$ is performed as described in Eqn (6.4-5).

6.4.3.4 Band interleaved decoding at 13.2 kbps and 16.4 kbps

If SPAR coding strategy is one of the four time differential coding schemes as described in Table 5.4-9 and IVAS bitrate is either 13.2 kbps or 16.4 kbps then that indicates SPAR parameters coding with 40ms metadata update rate using the band interleaved method described in subclause 5.4.4.3. In this case, the number of bands to be read from bitstream are $\frac{nB^{foa}_{spar}}{2}$ and depending on the coding scheme (4a, 4b, 4c or 4d), these parameters are either read to band indices that are labelled A or band indices that are labelled B in Table 5.4-11 and Table 5.4-12. The remaining band indices use the quantized parameters from previous frame.

6.4.3.5 Arithmetic decoder

If the coding strategy is either BASE non-differential entropy coding scheme OR one of the four time differential coding schemes, then that indicates Arithmetic coding of SPAR parameters as described in subclause 5.4.3.6.15.

For each type of SPAR parameter (PR, CP or D) a 2 bit probability distribution model index is read from the bitstream and then the corresponding probability distribution model is read from Table 5.4-13, Table 5.4-14, Table 5.4-15 depending on the coding strategy and quantization strategy. Metadata bitstream is then decoded with Arithmetic decoder.

<…Add reference to Arithmetic coder code TBD.>

6.4.3.6 Huffman decoder

If the coding strategy is BASE_NOEC, then that indicates Huffman coding of SPAR parameters as described in subclause 5.4.3.6.16. Huffman codebook is read from Table 5.4-16, Table 5.4-17 and Table 5.4-18 depending on quantization strategy. Metadata bitstream is then decoded with Huffman decoder.

<…Add reference to Huffman coder code TBD.>

6.4.4 SPAR and DirAC parameter merge

6.4.4.1 SBA mono handling

Decoder Mono Detector/Preserver

The mono detector/preserver in the decoder looks at the signalling through the bitstream (or implicit signalling through the metadata). If mono is signalled, then the mechanism sets the DIRAC energy ratio to be zero and diffuseness to be one for every frequency band and subframe in the decoder. This ensures that the W channel energy is not spread across the other channels and the signal remains as mono content.

The metadata values that the mono detector/preserver examines are:

- SPAR metadata
    - pred_re
    - C_re
    - P_re
- DiRAC metadata
    - energy_ratio
    - azimuth

- elevation

The detector/preserver looks for these 6 values in each band, if any of the values in any band do not match the mono expected values, then the whole block is declared non-mono. Otherwise the block is declared mono. All the expected values for mono are zero, except for energy_ratio which is expected to be < 0.15 due to quantisation of that value.

When mono is detected in the decoder, the energy_ratio values are reset to zero for all bands and time slices in the block, and diffuseness is set to one. This results in the correct behaviour in the deocoder for mono content contained within an ambisonics format.

6.4.4.2 SPAR to DirAC parameter conversion

In bitrate switching scenarios when bitrate is switched between IVAS bitrate greater than or equal to 384 kbps and IVAS bitrate less than 384 kbps, then in the transition frame, DirAC parameters are estimated from quantized SPAR parameters, in bands 1 to nB_{spar}^{foa} , using SPAR to DirAC metadata converter. The direction vector is generated as given below.

$$dv_x = \frac{PR_x}{\sqrt{(PR_x^2 + PR_y^2 + PR_z^2)}}$$

$$dv_y = \frac{PR_y}{\sqrt{(PR_x^2 + PR_y^2 + PR_z^2)}}$$

$$dv_z = \frac{PR_z}{\sqrt{(PR_x^2 + PR_y^2 + PR_z^2)}}$$

where PR_x, PR_y and PR_z are SPAR prediction coefficients corresponding to X, Y and Z channel respectively.

Azimuth and elevation angles are then generated as

$$Az = \arctan\left(\frac{dv_y}{dv_x}\right) * \frac{180}{\pi} \qquad (6.4\text{-}7)$$

$$El = \arctan\left(\frac{dv_z}{\sqrt{dv_x^2 + dv_y^2}}\right) * \frac{180}{\pi} \qquad (6.4\text{-}8)$$

Energy ratio computation depends on number of downmix channels $\boldsymbol{N_{dmx}}$ as given Eqn (6.4-10) and

Number of downmix channels greater than one

In this case energy ratio and diffuseness parameters are computed directly from SPAR prediction coefficients as follows.

$$en_i = \left(PR_x^2 + PR_y^2 + PR_z^2\right)$$

$$en_{lt} = 0.25en_i + 0.75en_{lt,-1} \qquad (6.4\text{-}9)$$

$$en_{ratio} = \sqrt{en_{lt}} \qquad (6.4\text{-}10)$$

where en_{lt} is long term average of en_i, $en_{lt,-1}$ is previous frame's en_{lt}.

Diffuseness parameter is then computed as

$\psi = 1 - en_{ratio}$ (6.4-11)

Number of downmix channels equal to one

In this case energy ratio and diffuseness parameters are computed directly from SPAR active W prediction coefficients and decorrelation coefficients as follows.

$$en_i = \left(PR_x^2 + PR_y^2 + PR_z^2\right)$$

where PR_x, PR_y and PR_z are SPAR prediction coefficients corresponding to X, Y and Z channel respectively.

$$diff_i = \left(D_x^2 + D_y^2 + D_z^2\right)$$

where D_x, D_y and D_z are SPAR decorrelation coefficients corresponding to X, Y and Z channel respectively.

$$en_{lt} = 0.25en_i + 0.75en_{lt,-1}$$

where en_{lt} is long term average of en_i, $en_{lt,-1}$ is previous frame's en_{lt}.

$$pow_i = 0.5((1 - f_s en_i)^2 + en_i + diff_i$$

$$pow_{lt} = 0.25pow_i + 0.75pow_{lt,-1}$$

where pow_{lt} is long term average of pow_i, $pow_{lt,-1}$ is previous frame's pow_{lt}. f_s is given in Eqn XX).

$$en_{ratio} = \frac{\sqrt{en_{lt}}}{pow_{lt}} \qquad (6.4\text{-}12)$$

Diffuseness parameter is then computed as

$$\psi = 1 - en_{ratio} \qquad (6.4\text{-}13)$$

6.4.4.3 DirAC to SPAR parameter conversion

For FOA channels, SPAR parameters are coded for low frequency bands with band index ranging from 1 to nB_{spar}^{foa}, where $nB_{spar}^{foa} = \frac{8}{bw_{bands}}$ and bw_{bands} is given in Eqn (6.4-3), SPAR parameters for frequency band index greater than nB_{spar}^{foa} are computed from decoded DirAC parameters. DirAC to SPAR parameter conversion is done differently for parameters corresponding to downmix channels, that is channel index i with $1 \le i \le \boldsymbol{N_{dmx}}$, and for parameters corresponding to parametric FOA channels, that is channel index i with $\boldsymbol{N_{dmx}} \le i \le 4$ as follows.

DirAC to SPAR conversion for parameters corresponding to downmix channels

SPAR parameters corresponding to downmix channels are only PR coefficients and they are generated with 20 ms time resolution as done in encoder as described in subclause 5.4.3.7.2 and subclause 5.4.3.7.4.

DirAC to SPAR conversion for PR, CP and D coefficients corresponding to parametric channels

SPAR parameters are generated with same time resolution a DirAC parameter by modifying Eqn (5.4-75) as follows:

$$\gamma_i = \gamma_{m,i} \qquad (6.4\text{-}14)$$

where $1 \le i \le 4$, $\gamma_{m,1} = 1$, $\gamma_{m,2} = \sin\theta_m \cos\emptyset_m$, $\gamma_{m,3} = \sin\emptyset_m$, $\gamma_{m,4} = \cos\theta_m \cos\emptyset_m$, m is the subframe index with in a frame, θ_m is quantized DirAC azimuth angle in subframe m, and $\emptyset_m$ is quantized DirAC elevation angle in subframe m.

γ_i is then used in Eqn (5.4-72), (5.4-73) and (5.4-74) to generate covariance matrix in subframe m. The covariance matrix in each subframe is then converted to SPAR parameters as described in subclause 5.4.3.7.4.

6.4.4.4 DirAC model for the SPAR parameters

The inter-channel covariance matrix employed in the calculation of the upmix matrix is derived from the acoustic model parameters in the same way as on the encoder side in subclause xxx. The calculation is repeated independently based on the model parameters received from the bitstream. In the reference implementation, the same code is run on both the encoder and decoder side.

6.4.5 Upmix matrix calculation

6.4.5.1 SPAR matrix generation (P and C matrices)

SPAR first generates P and C matrices, where P matrix is applied to decorrelator output and C matrix is applied to downmix signal during the upmixing process as described in subclause 6.4.6.4. These matrices are computed as follows.

First prediction upmix matrix u created as follows.

$$u_{[\boldsymbol{N_{spar_{ana}}} x \boldsymbol{N_{spar_{ana}}}]}(m,b) = \begin{pmatrix} s(1-\ f_s g^2) & -f_s\, PR^{quantized}_{[\boldsymbol{N_{PR}}\, x\, 1]}(m,b)^H \\ PR^{quantized}_{[\boldsymbol{N_{PR}}\, x\, 1]}(m,b) & I_{[\boldsymbol{N_{PR}} x\ \boldsymbol{N_{PR}}]} \end{pmatrix} \left(remix_{\left[\boldsymbol{N_{spar_{ana}}} x \boldsymbol{N_{spar_{a\square\square a}}}\right]}\right)^{-1} \quad (6.4\text{-}15)$$

where m is the subframe index in a frame with $1 \leq m \leq 4$, b is band index with $1 \leq b \leq nB^{hoa}_{spar}$, s is scaling factor is given in Eqn (6.4-17), f_s is active W scaling factor as given in Eqn (6.4-16), $PR^{quantized}_{[\boldsymbol{N_{PR}}\, x\, 1]}$ are the decoded SPAR prediction coefficients, $remix_{[\boldsymbol{N_{spar_{ana}}} x \boldsymbol{N_{spar_{ana}}}]}$ matrix is as given in Eqn (5.4-52).

$$f_s = \begin{cases} 0 & , if\ Flag_{activeW} = 0 \\ 0.25 & if\ ivas\ bitrate \leq 16.4\ kbps\ OR\ \boldsymbol{W_{vad}} = 0 \\ 0.5 & otherwise \end{cases} \quad (6.4\text{-}16)$$

$$s = \begin{cases} 1 & , if\ Flag_{activeW} = 0 \\ 0.0039 & , if\ Flag_{activeW} = 1, IVAS\ bitrates \geq 48kbps \end{cases} \quad (6.4\text{-}17)$$

Then decorrelation upmix matrix d and cross prediction upmix matrix v are created as follows.

$$d_{i,j}(m,b) = \begin{cases} D_j(m,b) & , if\ i > N_{dmx}, j = i - N_{dmx} \\ 0 & , otherwise \end{cases} \quad (6.4\text{-}18)$$

$$v_{i,k}(m,b) = \begin{cases} CP_{i-N_{dmx}, k-1}(m,b) & , if\ i > N_{dmx}, k > 1 \\ 1 & , if\ i = k \\ 0 & , otherwise \end{cases} \quad (6.4\text{-}19)$$

where $i, j\ and\ k$ are channel indices with $1 \leq i \leq \boldsymbol{N_{spar_{ana}}}$, $1 \leq j \leq \boldsymbol{N_{dec}}$ and $1 \leq k \leq \boldsymbol{N_{dmx}}$.

SPAR P and C matrices are then created as follows.

$$P''_{[\boldsymbol{N_{spar_{ana}}} x \boldsymbol{N_{dec}}]}(m,b) = u(m,b)\ d(m,b) \quad (6.4\text{-}20)$$

$$C''_{[\boldsymbol{N_{spar_{ana}}} x \boldsymbol{N_{dmx}}]}(m,b) = u(m,b)\ v(m,b) \quad (6.4\text{-}21)$$

When active W is enabled, upmix matrix is further modified to just scale the W channel. This results in a different upmix strategy at the decoder than the downmix strategy at the encoder. The modifications to P and C matrix are follows.

$$P''_{i,j}(m,b) = \begin{cases} 0 & , if\ Flag_{activeW} = 0, i = 1 \\ P''_{i,j}(m,b) & , otherwise \end{cases} \quad (6.4\text{-}22)$$

$$C''_{i,j}(m,b) = \begin{cases} \max\left(0, C''_{i,j}(m,b)\right) & , i = j = 1 \\ C''_{i,j}(m,b) & , otherwise \end{cases} \quad (6.4\text{-}23)$$

Then band unmixing is performed as follows.

$$P(m, i + (b1 - 1) * bw_{bands}) =\ P''(m, b1) \quad (6.4\text{-}24)$$

$$C(m, i + (b1 - 1) * bw_{bands}) =\ C''(m, b1) \quad (6.4\text{-}25)$$

where $1 \leq b1 \leq\ nB^{foa}_{spar}$, $1 \leq i \leq\ bw_{bands}$.

6.4.5.2 Upmix matrix generation

The SPAR upmix matrix is generated as follows.

$$U_{i,j}(m,b) = \begin{cases} C_{i,j}(m,b) & , j \leq N_{dmx} \\ P_{i,j}(m,b) & , N_{dmx} \leq j \leq \boldsymbol{N_{spar_{ana}}} \end{cases}$$

where $i\ and\ j$ are channel indices with $1 \leq i \leq \boldsymbol{N_{spar_{ana}}}$, $1 \leq j \leq \boldsymbol{N_{spar_{ana}}}$, P matrix is given in Eqn (6.4-24) and C matrix is given in Eqn(6.4-25).

This upmix matrix U is applied to the downmix signal and decorrelated signal to generated SPAR upmix signal as described in Clause 6.4.6.4.

6.4.6 Decoded audio processing

6.4.6.1 Downmix signal decoding (core decoding)

6.4.6.2 AGC

The decoder counterpart of AGC performs an inverse gain operation to the decoded downmixed audio signal $s_i^{DMX\_dec}(n)$ following subclause 6.4.6.2, to be subsequently upmixed following 6.4.6.4. This inverse operation is a lossless operation provided that a lossless core codec is used and that all overload conditions are properly compensated. This operation requires an inverse operation of the gain transition function used at the encoder side as described in subclause 5.4.7. As previously described, the following information is needed to initially construct the gain transition function, i.e., the gain scaling factor $e(j) - e(j-1)$, gain transition step size $DBSTEP$ and the prototype smoothing function $p(\cdot)$. They are derived and obtained from the decoded AGC metadata bitstream (i.e., the gain parameters) and predefined functions/values.

The inverse gain transition function is then formulated as:

$$t^{-1}(l,j) = \begin{cases} 1, & l = 0 \dots D-1 \\ p(l-D,j)^{-(e(j)-e(j-1))}, & l = D \dots L-1, \end{cases}$$

Note that the inverse gain transition function still retains the properties of the original function, i.e., it comprises a transitory portion (lower part) and a steady-state portion (upper part). Moreover, the length of the transitory portion is limited by an overall core codec delay $D = 12ms$ or 576 samples at $48kHz$ sampling frequency. Figure 6.4.6-1 depicts the inverse gain transition function at $48kHz$ sampling frequency having the same scaling factors specified in Figure 5.4.7-1 at the encoder counterpart.

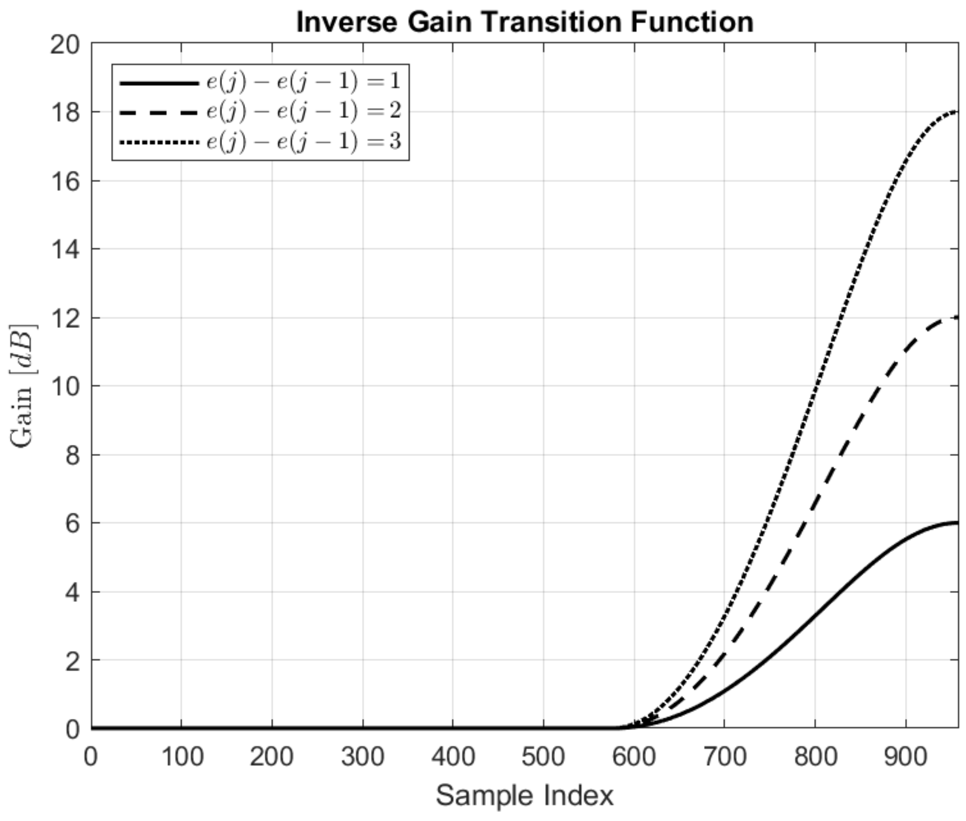


Figure 6.4.6-1 Inverse gain transition function having the scaling factor $e(j) - e(j-1)$ set to 1, 2 and 3.

The final construct of the inverse gain transition function showing all the dependencies is as follows.

$$t^{-1}(DBSTEP, l, j) = t^{-1}(l, j) \cdot t^{-1}(DBSTEP, L-1, j-1), \quad l = 0 \cdots L-1,$$

where $t^{-1}(DBSTEP, L-1, j-1)$ is initialized to 1.

Setting the sample index l to n, and introducing the decoded downmix channel index $i = 1$, indicating a single downmixed channel, gives the following formulation:

$$s_{i=1}^{AGC\_dec}(n) = t^{-1}(DBSTEP, n, j) \cdot s_{i=1}^{DMX\_dec}(n).$$

The gain adjusted frame $s_{i=1}^{AGC\_dec}(n)$ is then subject for upmixing.

The gain parameter for constructing the inverse gain transition function is obtained from the bitstream. Upon receiving the encoded AGC metadata bitstream, the following steps are performed. Firstly, 1 bit is read indicating whether the gain control is applied to the current frame or not. If it is set to 0, a unity gain is applied to the current frame. If it is set to 1, β_E bits are read from the bitstream. These β_E bits are binary decoded to yield the gain parameter $e(j)$. For a given $\beta_E = 2$ bits, the β_E bits having the binary code from the set $\{(00),(01),(10),(11)\}$ are converted to $e(j)$ having a value of $\{0,1,2,3\}$.

When dealing with a single or multiple frames packet loss, the decoder reconstructs the lost frames and AGC defaults the gain parameter of the lost frame to that of the last received (good) frame having no gain control applied. Upon receiving an actual (good) frame, AGC internal state is updated with the information corresponding to the decoded AGC metadata bitstream. This mechanism is considered as the best effort gain control handling ensuring a smooth transition of gain parameters in the event of packet loss.

6.4.6.3 PCA

6.4.6.4 SPAR upmix processing

The SPAR upmixing process is as follows. First, the downmix signal is converted to the filter bank domain using CLDFB analysis. Then the slow-fluctuating decorrelated signal is converted to filterbank domain using CLDFB analysis. Then SPAR upmix matrix elements, comprising wet parameters (or P matrix as given in Eqn (6.3-24)) and dry parameters (or C matrix as given in Eqn (6.3-25)), are mapped to CLDFB filterbank banding. The mapped wet parameters are applied to the decorrelator output to create the wet upmix and the mapped dry parameters are applied to the downmix signal to create the dry upmix. The slow-fluctuating wet upmix is then added to the dry upmix, which contains a transient component, to reconstruct the SBA signal with all FOA channels and selected HOA channels as given in Table 5.3-2.

6.4.6.4.1 Time domain decorrelator

6.4.6.4.2 CLDFB analysis

The decoded time domain transport channels u_d are transformed into the CLDFB domain *U* as:

$$U_l^d(k) = \sum_{n=0}^{N-1} p(n) u_d(kS-n) \exp\left(j\frac{\pi}{L}(l+0.5)(n-\frac{D}{2})\right) , \qquad (6.4\text{-}26)$$

Using the following notation:

$l = 0, 1, \ldots, L-1$ is the CLDFB frequency band index where L = 60

$N = 600$ is the length of the FIR prototype filter *p.*

$S = 60$ is the processing stride in samples.

k is the CLDFB domain time slot index.

N is the input time sample index.

D = 299 is the analysis-synthesis (sample-by-sample) delay in samples.

D = 0,1,2,3 is the transport channel index.

6.4.6.4.3 CLDFB upmixing

The number of transport channels depends on the target bitrate. In case of one transport channel, a modified version of the FOA W signal is transmitted. Otherwise, the first transport channel corresponds to the unmodified W signal. Other transport channels correspond to residuals with respect to the FOA signals X, Z, Y. Missing transport channels will be approximated by running the decoded W signal through a time domain decorrelator as described in section [CrossRef: "Time domain decorrelator"].

Based on the transport channels and transmitted metadata, FOA signals will be reconstructed.

The metadata time resolution depends on the SPAR frequency band. For SPAR bands 0, 1, …, 7 SPAR metadata at 20 ms time resolution (one frame) is transmitted. For SPAR bands 8, 9, 10, 11, DirAC metadata at 5ms time resolution is transmitted. However, to reconstruct FOA signals from transport channels the DirAC metadata is subsampled to 20 ms time resolution similarly at the encoder and the decoder allowing for waveform reconstruction. For missing transport channels and SPAR bands 8, 9, 10, 11, parametric reconstruction works at 5 ms time resolution using DirAC metadata.

Therefore, the reconstruction of FOA signals is described in terms of 5 ms subframes corresponding to 4 CLDFB time slots each and associated upmix matrices. If the effective metadata time resolution is 20 ms, then the associated subframe upmix matrices are held constant over the period of one frame (4 subframes).

For the reconstruction of the FOA signals, the 4 transport channels (or its replacements from the decorrelator processing) are run through the CLDFB analysis resulting in 4 complex-valued signals $U_l^{in\_ch}$, each consisting of 60 frequency bands and 16 time slots for one audio frame. Each frame is reconstructed based on one previous frame and 4 current frame upmix-matrices which are derived from the transmitted SPAR and DirAC metadata as described in section [CrossRef]. The subframe metadata sets are stored in a delay line.

The FOA signals $V_l^{out\_ch}$ are computed as

$$V_l^{out\_ch}(k) = \sum_{in\_ch=0}^{3} U_l^{in\_ch}(k) M_{out_{ch},in_{ch},l}^{CLDFB}(k) \quad . \tag{6.4-27}$$

The upmix matrix $M_{out_{ch},in_{ch},l}^{CLDFB}$ in the CLDFB domain is computed as

$$M_{out_{ch},in_{ch},l}^{CLDFB}(k) = \sum_{b=0}^{11} H_l^b \left((1 - G(k)_{out_{ch}}) M_{out_{ch},in_{ch},b}^{sf(k)-mdi_{out_{ch}}} + G(k)_{out_{ch}} M_{out_{ch},in_{ch},b}^{sf(k)} \right) \tag{6.4-28}$$

where the subframe index *sf* is computed as $sf(k) = INT(k/4)$.

The helper bool variable *isResChan* indicates if a certain output channel has an associated residual transport channel. It is defined as a function of number of transport channels (which depends on bitrate) as shown in 6.4-1.

The metadata update interval *mdi* is given as

$$mdi_{out_{ch}} = \begin{cases} 4 & \text{if } isResChan(out_{ch}) = 1 \\ 1 & \text{if } isResChan(out_{ch}) = 0 \end{cases} \quad . \tag{6.4-29}$$

Table 6.4-1: Bool variable isResChan

#Transport Channels	Output channel			
	0 (W)	**1 (Y)**	**2 (Z)**	**3 (X)**
1	0	0	0	0
2	0	1	0	0
3	0	1	0	1
4	0	1	1	1

Note: Indication of residual transport channels dependent on the number of transport channels. 1 indicates residual transport channel. 0 indicates not a residual transport channel. Note also that W is always a transport channel but not a residual transport channel.

The cross-fading weights $G(k)_{out_{ch}}$ are defined in Table 6.4-2, depending on the bool variable *IsResChan*. Details on how the cross-fading weights are computed are given in subclause xxx.

Table 6.4-2: Cross-fading weights G(k) used to interpolate between subframe upmix matrices

Time slot index k	*isResChan*	
	0	**1**
0	0.00	0.00
1	0.25	0.00
2	0.50	0.00
3	0.75	0.00
4	0.00	0.00
5	0.25	0.00
6	0.50	0.00
7	0.75	0.00
8	0.00	0.00
9	0.25	0.00
10	0.50	0.00
11	0.75	0.00
12	0.00	0.16102859
13	0.25	0.65157765
14	0.50	1.00
15	0.75	1.00

The SPAR band contribution weights H_l^b to a particular CLDFB band are obtained through table lookup. They are illustrated in Figure 6.4.6-2. Details how the table values are derived are given in subclause 6.4.6.4.4.

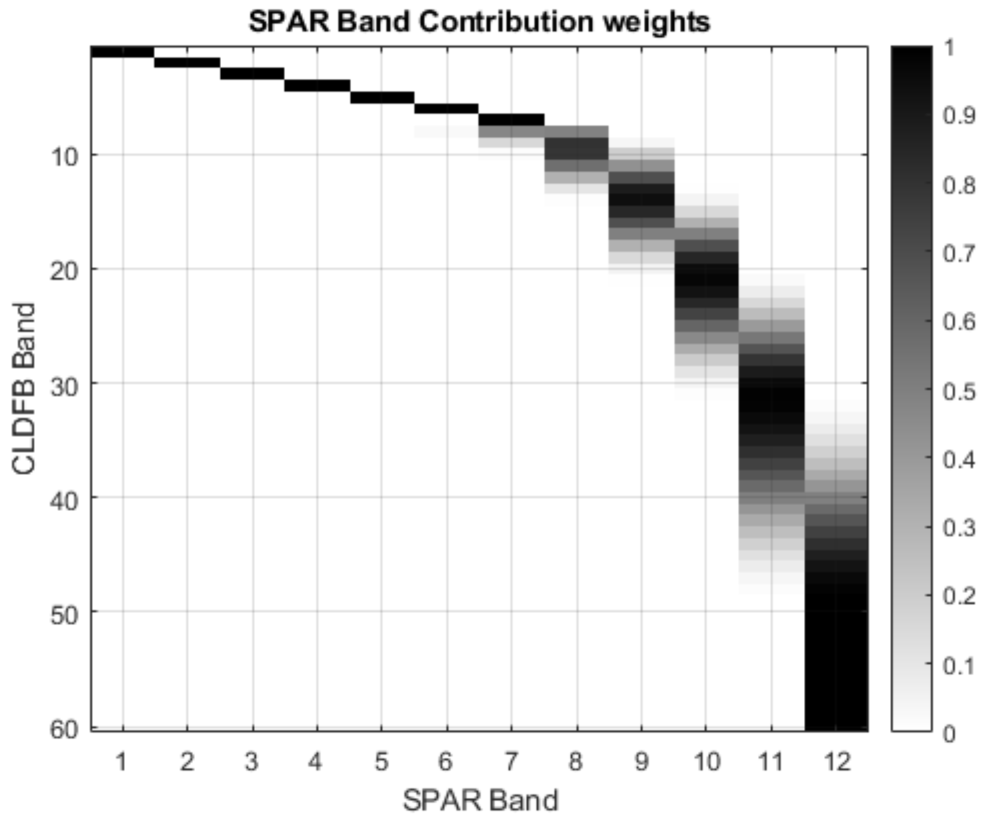


Figure 6.4.6-2: SPAR Band contributions H_l^b

6.4.6.4.4 MDFT filterbank banded upmix matrix to CLDFB banded upmix matrix conversion

The upmix matrix used to reconstruct one FOA signal from the decoded signals, for a particular CLDFB band is derived as a linear combination of upmix matrices corresponding to contributions from multiple SPAR bands. This is due to the overlapping nature of the SPAR filterbank bands and since the SPAR filterbank synthesis consists of the summation over all SPAR filter outputs. Ideally, for best waveform reconstruction, the SPAR filters would be run in the CLDFB domain as complex multi-tap FIR filters to best match the encoder processing. Since the computational complexity of running multi-tap filters in the CLDFB domain is high, the SPAR filters are approximated by real-valued single tap filters H_l^b for CLDFB band l as follows.

$$H_l^b = \frac{M_l^b}{\sum_{b=0}^{11} M_l^b} \quad , \tag{6.4-30}$$

with

$$M_l^b = \sqrt{\sum_{f=lN/L}^{(l+1)N/L} \left| \sum_{n=0}^{N_h-1} h_b(n) e^{-j2\pi(f+0.5)n/N} \right|^2} \quad . \tag{6.4-31}$$

In the above equation, h_b is the SPAR FIR filter for band b of length N_h samples, N is the size of the MDFT of the SPAR FIR filter (960, 640 or 320 for sampling frequencies of, respectively, 48000 Hz, 32000 Hz or 16000 Hz) and L is the number CLDFB bands (60).

For certain CLDFB bands it is observed that the frequency responses of SPAR Filters and CLDFB bands match sufficiently well. This is the case for the first 7 CLDFB bands such that only single SPAR bands contribute to CLDFB bands

$$M_l^b = \begin{cases} 0 & \text{if } b = l \\ 1 & \text{if } b \neq l \end{cases} \qquad l = 0 \ldots 6 \quad . \tag{6.4-32}$$

6.4.6.4.5 Crossfading in CLDFB

For transmitted residual transport channels, the goal in the decoder is to perform best possible waveform reconstruction based on the transmitted metadata given the known encoder prediction (residual) processing using the SPAR (encoder) filterbank and time cross-fade between frames. Theoretically the encoder processing can be undone perfectly on the decoder side if the same SPAR encoder filterbank and time domain cross-fade were applied. However, due to system and complexity reasons, the encoder operation is reversed in the CLDFB domain and cross-fading between frames is performed by cross-fading the corresponding CLDFB domain parameters. The cross-fading weights $G(k)$ for that operation are given in the third column in Table 6.4-2. These weights were derived such that after CLDFB synthesis they optimally (in a least mean squares sense) approximate the fading operation in the SPAR encoder.

6.4.6.4.6 Additional VLBR smoothing

6.4.6.4.6.1 CLDFB upmixing

6.4.6.5 DirAC synthesis and rendering

6.4.6.5.1 Overview

At the decoder side, the SPAR upmix according to subclause xxx produces a larger second set of transport channels from a smaller first set of transport channels, which are decoded using the core decoder tools according to subclause xxx.

The channels that comprise this second set are listen in Table 6.4-3 as a function of the total IVAS bitrate.

Table 6.4-3: XX

IVAS total bitrate [kbps]	Transport channels in the second set
< 256	0,1,2,3 (w, x, y, z)
256	0,1,2,3,4,8,9,15
384	0,1,2,3,4,8,9,15
512	0,1,2,3,4,5,6,7,8,9,15

The numbers refer to the real spherical harmonics (spatial basis functions) in ACN order. Figure 6.4-3 shows sketches of these functions.

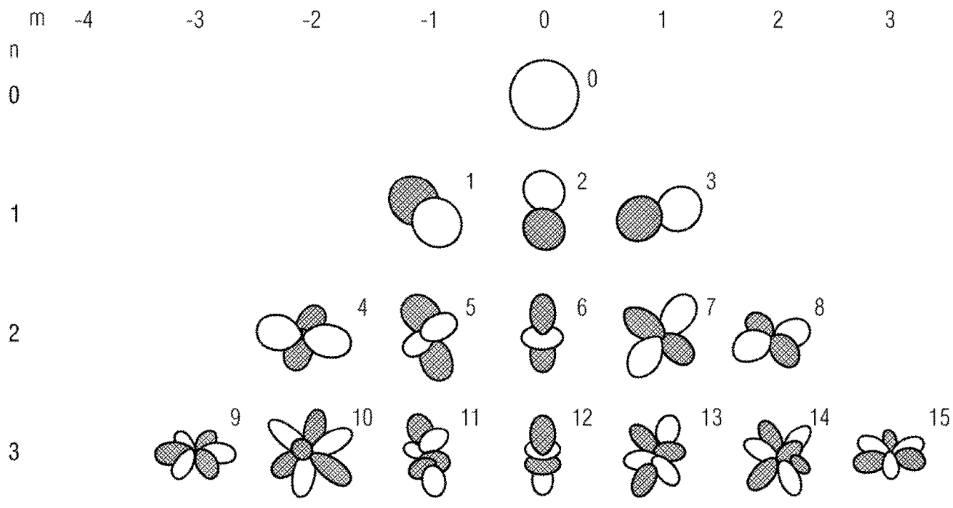


Figure 6.4-4 Representation of the Ambisonic components up to third order. Shaded portions represent regions where the polarity is inverted.

These channels are obtained directly in the domain of the CLDFB, and the synthesis of the output signal is performed in the same domain, thereby avoiding the need for extra synthesis and analysis steps.

The remaining ambisonics channels are then reconstructed (synthesized) using the DirAC method.

6.4.6.5.2 Mode selection

For the two highest bitrates (384 kbps and 512 kbps), the high-order mode of the DirAC decoder is activated. It is based on higher-order directional audio coding (HO-DirAC). In this mode, two sector decoding paths are activated. For lower bitrates, the low-order mode (first-order DirAC) is activated. Then only one sector decoding path is activated and the other one is deactivated.

6.4.6.5.3 High-order operation mode

In the high-order operation mode, the IVAS SBA decoder unit processes multiple directional sector signals and a global diffuse signal. It generates a decompressed ambisonics spatial audio signal representation from a compressed ambisonics spatial audio signal representation, which includes 4 or more transport channels (see above) and side information.

The side information includes sound field parameters: directional parameters providing information on a direction of arrival in each spatial sector, sector diffuseness parameter for each spatial sector, and a global diffuseness parameter.

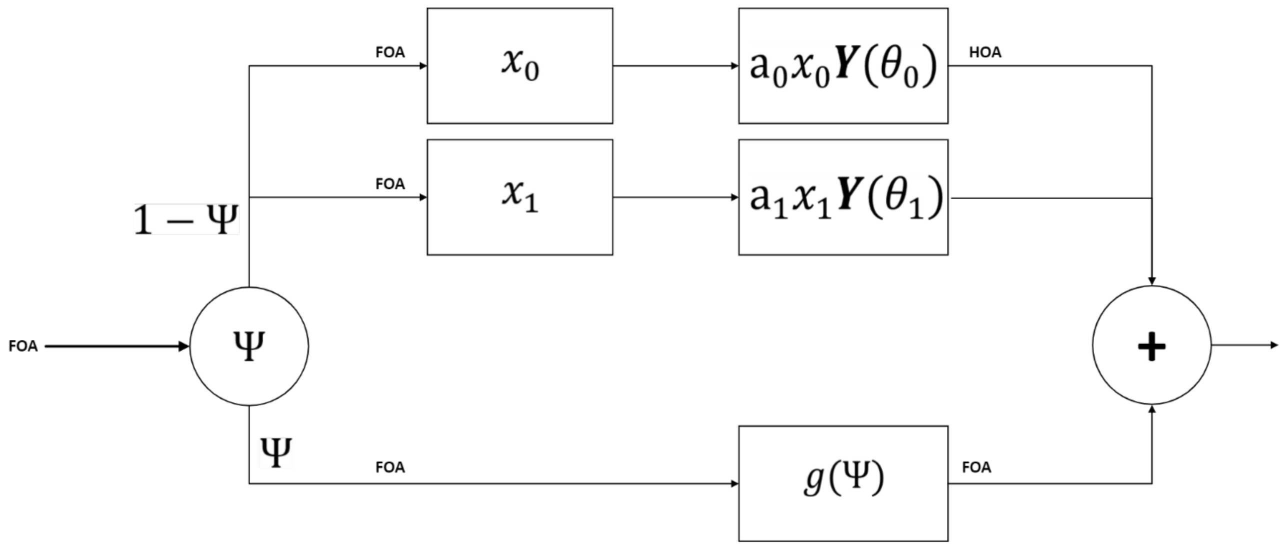

FOA
x_0
$a_0 x_0 \boldsymbol{Y}(\theta_0)$
HOA
FOA
x_1
$a_1 x_1 \boldsymbol{Y}(\theta_1)$
$1-\Psi$
FOA
Ψ
+
Ψ
FOA
$g(\Psi)$
FOA

Figure 6.4-2: Block diagram of the DirAC decoder in the high-order operation mode

####### 6.4.6.5.3.1 Directional sector decoding paths

A set of sector decoding paths, one for each spatial sector, decodes the directional sector signals into the decompressed ambisonic signal representation (see figure 3). In IVAS, two spatial sectors are used. Each path derives the corresponding sector signal from the transport channels by means of spatial filtering. The directional (DoA) parameters and sector diffuseness parameters are applied to weight and pan this signal into the ambisonic output stream:

$$\boldsymbol{x}_{lm,s} = a_s \boldsymbol{Y}(\theta_s) x_s = \ a_s [x_s Y_{00},\ x_s Y_{1-1}, x_s Y, x_s Y_{11}, \dots], \tag{6.4-33}$$

Where x_s is the scalar-valued sector pressure signal. It results from the inner product of the beamforming matrix element of the spatial sector s, $\boldsymbol{w}_s^T$, with the FOA signal $\boldsymbol{x}_{FOA}$:

$$x_s = \boldsymbol{b_s} \cdot \boldsymbol{x}_{FOA} \tag{6.4-2}$$

$\boldsymbol{x}_{FOA}$ is the vector of the FOA channels in the second set of transport channels, and $\boldsymbol{b_s}$ is the vector of the beamformer weights. The matrix can be understood as the projector on this vector, i.e., $\boldsymbol{w}_s^T = \boldsymbol{b_s b_s^T}$. In IVAS, the two sector pressure signals are effectively calculated as

$$x_0 \ = \ 1.772454\, x_{FOA,w} \ + \ 1.023327\, x_{FOA,x} \ , \tag{6.4-3}$$

$$x_1 = 1.772454\, x_{FOA,w} - 1.023327\, x_{FOA,x} \ , \qquad (6.4\text{-}4)$$

$\boldsymbol{Y}(\theta_s)$ is the vector of the real spherical harmonic up to the configured output order evaluated θ_s, the DoA angle of the sector s. $a_s$is the relative directionality of the spatial sector s defined in clause 5.4.3.4.1. Thereby the sound field components described by the higher-order (non-planar) ambisonics channels are obtained by evaluating the second set of transport channels (direct sound signals) the SPAR upmix (reference sound signals) and the above response functions.

This panning is performed for each time-frequency tile of the CLDFB, which are grouped into the parameter bands according to table II and subframes of 5 ms length. For each group of time-frequency tiles, the real spherical harmonic functions (response functions) $\boldsymbol{Y}(\theta_s)$ are calculated from the sound direction (DoA) in the sector s.

In order to avoid highly complex calculations of trigonometric functions and Legendre polynomials, the spherical harmonics (spatial basis functions) are evaluated by table look up. The tables are indexed by the degree and index l and m, and the azimuth and elevation angles corresponding to the sound direction of arrival (DoA) parameters of the acoustic model.

6.4.6.5.3.2 Global Diffusenes decoding

The global diffuseness decoding path generates the global diffuseness signal. This is achieved by applying the global diffuseness parameter Ψ to the second set of transport channels from the SPAR upmix. Specifically, the subset of the transport channels that represents the first order of the ambisonic output signal is amplified to generate the correct amount of diffuse energy.

All diffuse energy is exclusively contained in the first ambisonics order. The directional paths synthesize only directional components into higher ambisonics orders. To compensate for the absence of diffuse energy in the directional path (see Fig. 3), an adaptive gain is applied according to the formula

$$g(\Psi) = \sqrt{1 + \Psi * (f_{comp} - 1)} \ , \qquad (6.4\text{-}5)$$

with the diffuse compensation factor f_{comp} . With this gain, the correct ratio between directional and diffuse energy in the higher-order ambisonics output signal is restored.

f_{comp} is derived from the energies of the sound components in the first and second group (ambisonics channels of first and higher orders). Specifically, the energy in the first group (low-order channels) is given by

$$E_L = \sum_{l=0..L} \frac{1}{\sqrt{2*l+1}}, \qquad (6.4\text{-}6)$$

and, analogously for the second group (high order channels)

$$E_H = \sum_{l=0..H} \frac{1}{\sqrt{2*l+1}}, \qquad (6.4\text{-}7)$$

where H is the ambisonics order of the output signal and L that of the second set of transport channels (SPAR upmix). The diffuse compensation factor is then calculated as

$$f_{comp} = \frac{E_H}{E_L}, \qquad (6.4\text{-}8)$$

This means that the gain increases when the output order H (number of components in the second group) increases and decreases when the input order L increases (number of sound field components in the first group). It also increases when the diffuses increases. Therefore, the diffuse energy missing in the reconstructed higher-order channels is compensated for in the FOA channels. The energy-compensation gain is restricted to a certain range of values to avoid too strong deviations from the reference signal. The minimum threshold is set to 0.99 and the maximum threshold is set to 2.0;

In case the first set of transport channels does not contain all channels of the first group of sound field components (first ambisonics order), the SPAR decoder generates the sound field components of the first group by upmixing from the first set of transport channels, i.e., forming weighted combinations of the input channels decoded with MCT and the decorrelator channels.

To obtain a perceptually faithful reproduction of the diffuse sound components in the scene, it is required to have the necessary diffuse signals in the first set of transport channels. Therefore, additional channels are generated to complement this first set, when the number of channels in the second set is greater than in the first set. Specifically, decorrelation is applied to the channels in the first set. These decorrelated channels are then mixed up to the second set of channels in SPAR (cf. clause xxx).

6.4.6.5.3.3 Signal inserter

The global diffuseness signal is then inserted into the output ambisonic output stream together with the directional signals, obtaining the final higher-order ambisonics output signal

$$\boldsymbol{x}_{HOA} = (1-\Psi)\sum_s \boldsymbol{x}_s + g(\Psi)\boldsymbol{x}_{FOA} \tag{6.4-9}$$

The addition of the higher-order signal $\boldsymbol{x}_s$ and the first-order signal $\boldsymbol{x}_{FOA}$ is understood such that the components of the latter are added to the corresponding vector components of the former while nothing is added to its other vector components.

In other words, the SBA decoder generates two groups of sound field components. The first group includes the channels of the first ambisonics order. It contains both direct and diffuse sound field components. The second group comprises the channels of the higher ambisonics orders 2 and 3. It contains only direct sound field components. To mitigate the absence of diffuse sound field components in the second group, an energy compensation is performed based on the diffuseness parameter and the ambisonics order in the first (L) and second (H) group.

6.4.6.5.3.4 Gain smoothing

In order to avoid artifacts, a gain smoothing is applied to the gains in Eqs. (6.4-34) and (6.4-5). Specifically, and onset filter and an interpolation between the gains of two subsequent subframes over the time slots of the CLDFB filter bank are employed.

The onset filter controls the weight of the gain of the previous subframe

$$w_1 = 0.3679 + \text{onset\_filter} * (0.1175 - 0.3679) \tag{6.4-10}$$

and that of the current subframe

$$w_2 = 1 - w_1. \tag{6.4-11}$$

onset_filter is equal to 1 during normal operation of the codec and, hence, the update rate is very slow. When an onset is detected, the weight of the current gains is increased, leading to a faster update and avoid a smearing out of the onset. The onset filter and the gain weights are evaluated individually for each band of the CLDFB.

The filtered gains are then applied to the signals in each time slot of the CLDFB. As length of this time slot is 1.25 ms but the resolution of the DoA and diffuseness information is 5 ms (one subframe), the gains are linearly interpolated over the time slots in a subframe.

6.4.6.5.4 Low-order operation mode

In the low-order operation mode, one sector decoding path is deactivated and only one is activated. Hence, the block diagram of the DirAC decoder simplifies to that in right hand-side of figure 2. In the left-hand side of Fig. 2, additional processing is described that is required to save on the metadata bitrate. Specifically, a hybrid decoding is employed based on a split of the processing in accordance with the core coder (cf. clause 5.4.3.4.2).

The encoded representation of the first portion (the lowest 8 parameter bands defined in table II.) of the first set of transport channels is decoded in a wave-form preserving way by the core decoder. No sound field parameters (DirAC) are received from the input interface (bitstream) for this portion. Instead, this portion is fed into the SPAR upmixer (cf subclause xxx) to obtain the vector of the components of the first-order ambisonics signal $\boldsymbol{x}_{FOA}$.

6.4.6.5.4.1 Hybrid decoding

A spatial parameter estimator (input signal analyzer), including a direction determiner and a diffuseness estimator, then obtains the sound field parameters (including direction and diffuseness) for each CLDFB time-frequency tile in the low-frequency bands (first portion) from $\boldsymbol{x}_{FOA}$ (second set of transport channels). The necessary conversion of the sound field components into the time-frequency representation is already performed prior to the SPAR upmix. The time and frequency resolution of the estimated DirAC parameters is that of the CLDFB itself. No parameter grouping or quantization is applied for the parameters estimated in the decoder.

The encoded representation of the second portion of the first set transport channels is decoded using parametric methods (cf. clause xxx) and upmixed to obtain the high-frequency portion of $\boldsymbol{x}_{FOA}$. The DirAC metadata decoder decodes the DirAC parameters (DoA and diffuseness for the 0th sector) for the second portion (high frequency bands). The DoA and diffuseness values corresponding to the parameter bands transmitted are expanded to the TF tiles that fall into these groups and subframes. The upmix to $\boldsymbol{x}_{FOA}$ is computed based on the SPAR (low frequencies) and DirAC (high frequencies) parameters.

This hybrid decoding method avoids the redundant transmission of SPAR and DirAC parameters for the low parameter bands, while the transmission of the SPAR parameters can be avoided via the use of the model covariance according to clauses 5.4.3.7.2.1 and 0 Therefore, the total amount of metadata bits is reduced.

A spatial renderer (see figure 2) then generates a sound field description in the form of higher-order ambisonics channels from components of the sound field represented by the first order ambisonics channels (second larger set of transport channels) from the SPAR upmix, $\boldsymbol{x}_{FOA}$. After the upmixing, the signal is present in the CLDFB domain. For the low frequencies (first) portion, the DirAC parameters are derived from the parameter estimator. For the high-frequencies (second portion), the DirAC parameters, sound direction (DoA) and diffuseness, are decoded from the bitstream.

The spatial rendering to higher-order ambisonics is described in details in the following. It consists of the generation the high-order components above the first order components inserting only the directional contribution. The diffuseness contribution is not generated above the first order, but only its energy is compensated by scaling the already generated first order components. As stated above, the first order components are divided in a first and second set of transport channels. The first set is directly derived directly from coded and transmitted transport channels and represent a low-order of sound field, while the second set, extended by SPAR parameters and DirAC DoAs and diffuseness, is a first order representation of the sound field, comprising both directional and diffuse contributions.

6.4.6.5.4.2 Directional sound field decoding

The directional sound field components of the higher ambisonics orders are reconstructed via the panning according to Eq. (1). However, there is only one sector pressure signal. As this one sector spans the whole sphere, the sector pressure signal x_1 is equal to the w component of the first-order ambisonics signal from the first set of transport channels $x_{FOA,w}$.

Hence, the sound field components described by the higher-order (non-planar) ambisonics channels are obtained by evaluating the direct sound signals from the SPAR upmix (reference sound signals) and the response functions (spherical harmonics). The response functions are obtained via table look up.

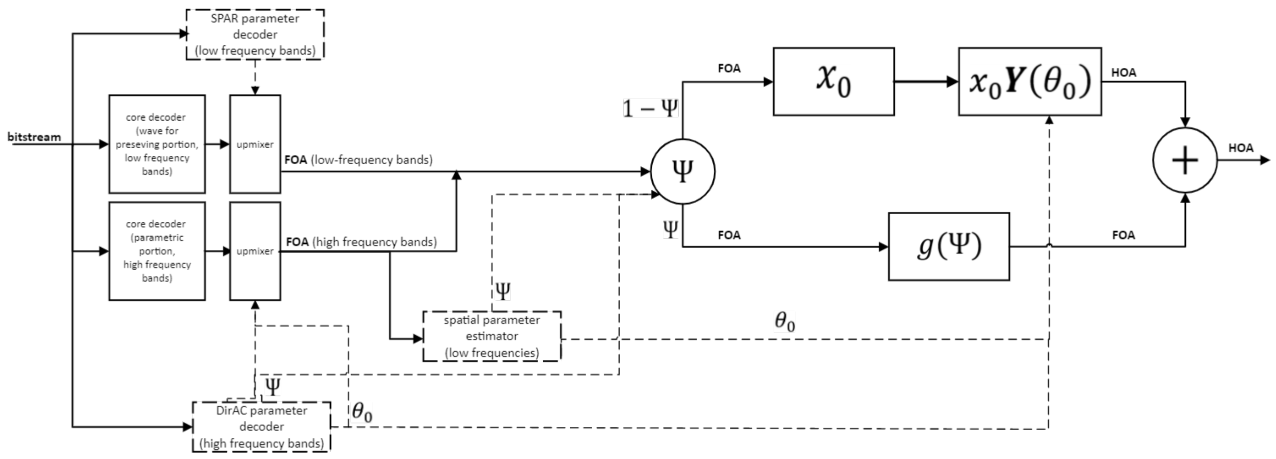

SPAR parameter decoder (low frequency bands)
bitstream
core decoder (wave for preseving portion, low frequency bands)
upmixer
FOA (low-frequency bands)
core decoder (parametric portion, high frequency bands)
upmixer
FOA (high frequency bands)
spatial parameter estimator (low frequencies)
DirAC parameter decoder (high frequency bands)
Ψ
θ_0
$1-\Psi$
FOA
x_0
$x_0\boldsymbol{Y}(\theta_0)$
HOA
$g(\Psi)$
HOA

Figure 6.4-3: Block diagram of the DirAC decoder in the low-order mode

6.4.6.5.4.3 Signal inserter

The directional and diffuseness paths are then added by a signal inserter, which sums up the directional and diffuse components to obtain the final output signal at the configured output order (see figure 2) in the spatial basis of the real spherical harmonics. The block diagram of the DirAC decoder in the low-order mode is depicted in figure 2.

6.4.6.5.4.4 Diffuseness decoding path

The weights of the directional and diffuse paths are, in this case, calculated as $\sqrt{1-\Psi}$ and $\sqrt{\Psi}$, respectively. Therefore, the vectors of the expansion coefficients of the sound field in the spatial basis of spherical harmonics is given by

$$\boldsymbol{x}_{HOA} = \sqrt{(1-\Psi)} + \boldsymbol{x}_{FOA}^{diff} \tag{6.4-12}$$

with the diffuse signal component

$$\boldsymbol{x}_{FOA}^{diff} = \begin{bmatrix} g_w(\Psi)\, x_{FOA,w} \\ g_x(\Psi)\, x_{FOA,x} \\ g_y(\Psi)\, x_{FOA,y} \\ g_z(\Psi)\, x_{FOA,z} \end{bmatrix} \tag{6.4-13}$$

The diffuse gains are calculated as

$$g_w(\Psi) = \sqrt{1+\Psi*(f_{comp}-1)}, \tag{6.4-14}$$

and

$$g_{x;y;z}(\Psi) = \sqrt{\Psi\,\mathrm{c} + (1-\Psi)\mathrm{Y}_{x;y;z}^{2}(\theta_0) b_{x;y;z}}, \tag{6.4-15}$$

The variables b and c are defined as $b_{x;y;z} = \frac{E}{E_{x;y;z}+\epsilon}$ and $c = 1 + \frac{1}{2}\,(f_{comp} - 1)$. E is the signal energy of all channels in $\boldsymbol{x}_{FOA}$in the current frame and $E_{x;y;z}$ that of the respective channel x, y, or z.

6.4.6.5.4.5 Gain smoothing

To all gains, in both the directional and diffuse paths, the gain smoothing is applied in the same was as in the high-order operation mode.

6.4.6.5.4.6 Summary

In summary, the DirAC decoder calculates the sound field components described by the higher-order (non-planar) ambisonics channels by evaluating the direct and diffuse sound signals from the SPAR upmix (reference sound signals) and the above response functions.

6.4.6.5.5 Bitrate switching

The IVAS SBA decoding unit can operate at different bitrates and dynamically switch between them in case of varying bandwidth of the connection. Specifically, it can switch between a low-order operation mode and a high-order operation mode.

In the low-order operation mode, only one sector decoding path is activated and the other one is deactivated. In this case, the side information does not contain the sound field parameters for the deactivated sector. The active sector then spans the whole sphere.

In the high-order operation mode, all two sector paths are activated, and the side information contains the sound field parameters for both sectors and the global diffuseness parameters.

6.4.6.5.6 Binaural rendering

6.4.6.5.6.1 Rendering modes

Binaural rendering of decoded SBA output signals is realized using different methods. These are selected depending on Output configuration and bitrate.

Table 6.4-4: XX

Bitrate [kbps] /output configuration	Binaural	Binaural room
< 96	Parametric renderer	Parametric room renderer
>= 96	Fastconv	Fastconv with room effect

The parametric renderer directly evaluates the first-order ambisonics signal from second set of transport channels and the DoA and diffuseness parameters. The sector decoding and diffuseness paths according to clauses 6.4.6.5.3 and 6.4.6.5.4 are not used. The parameter estimator (spatial analyzer) is not run. Instead, the DirAC parameters are derived from the SPAR parameters. A detailed description of this renderer can be found in clause 6.1.4 of [3GPP TS 26.253 V0.0.1 (2023-11)].

The FastConv based renderers process the third-order ambisonics signal generated by DirAC decoder and computes a convolution with a HRIR in the spherical harmonics domain. A detailed description of this renderer can be found in clause 6.1.3 of [3GPP TS 26.253 V0.0.1 (2023-11)].

6.4.6.5.6.2 Scene rotation

When head-tracking is enabled, the sound scene is rotated to account for rotations of the listeners head to prove a realistic VR experience. The rotation of the third-order ambisonics signal resulting from the DirAC synthesis can be achieved by applying the appropriate rotation matrix in the spherical-harmonic domain. This requires, however, a matrix multiplication with a high computational complexity.

The DirAC decoder supports a low-complexity head tracking mode to reduce this complexity. Only to the vector those channels belonging to the lowest ambisonic order that contains transport channels from the second larger set are pre-multiplied by the rotation matrix in the spherical harmonics domain.

The channels belonging to higher ambisonics orders are rotated more efficiently: the parameters of the psychoacoustic model are transformed by applying the rotation matrix to the DoA.

First the direction vector pointing to the source direction is constructed

$$\boldsymbol{r}_{DoA} = \begin{pmatrix} \cos(\phi)\cos(\theta) \\ \sin(\phi)\cos(\theta) \\ \sin(\theta) \end{pmatrix}.$$

Then the rotation is applied

$$\boldsymbol{r}_{DoA} \rightarrow \boldsymbol{R}\,\boldsymbol{r}_{DoA}.$$

Finally, the vector is converted back to the direction angles:

$$\phi = \operatorname{atan}\left(\frac{r_{DoA,y}}{r_{DoA,x}}\right)$$

and

$$\theta = \operatorname{atan}\left(\frac{\mathrm{r}'_{\mathrm{DoA,z}}}{\sqrt{\mathrm{r}^2_{\mathrm{DoA,x}} + r^2_{DoA,y}}}\right) .$$

The DirAC reconstruction then automatically yields the channels of the rotated scene.

6.4.6.5.7 Loudspeaker rendering

The rendering of the first set of transport channels requires the upmixing to the second set of transport channels (SPAR). This second set is a multi-channel representation of the sound scene in the spatial basis of the spherical harmonics. The channels in this representation are listed on table 6.3-1.

For loudspeaker output, a conversion of this multi-channel representation into a different output multi-channel representation is required. Specifically, this new multi-channel representation comprises one channel to be output on each loudspeaker of the configured output layout. The conversion is achieved by a parameter analyzer and a signal composer.

In accordance with clause 6.3.6.5.2, the spatial analyzer (parameter estimator) derives a parametric representation of the scene comprising DoA and diffuseness information in addition to the downmix channels (first set of transport channels).

The signal composer generates the output multi-channel representation of the spatial audio signal in the domain of the configured loudspeaker layout, as opposed to the domain of the real spherical harmonics in the input representation. Specifically, an upmix is performed from the second set of transport channels to the output loudspeaker layout using the spatial parameters from the intermediate representation. The method used for the upmixing is selected based on the total IVAS bitrate. Table 6.4-5 shows the conversion method for each bitrate.

Table 6.4-5: XX

IVAS total bitrate [kbps]	multi-channel conversion method
< 96	SHD
96 – 256	PSD
384	SHD
512	SHD

The spherical harmonics domain (SHD) synthesis is always performed in the same way as in clauses 6.4.6.5.3 and 6.4.6.5.4 . The resulting third-order ambisonics signal is then decoded to the configured loudspeaker layout according to clause 6.4.6.5.7.1. The power spectral density (PSD) synthesis directly generates a multi-channel signal in the spatial domain of the output loudspeaker layout according to clause 6.4.6.5.7.2.

6.4.6.5.7.1 ALLRAD decoding

The method SHD is based on a first synthesis to an ambisonics signal of order 3 with subsequent conversion of the multi-channel ambisonics signal to the spatial domain of the configured loudspeaker layout. This conversion is performed using matrix multiplication of a decoder matrix to the multi-channel vector of the ambisonics signal $\boldsymbol{x}_{HOA}$:

$$\boldsymbol{x}_{LS} = \boldsymbol{D}^{ALLRAD}\boldsymbol{x}_{HOA} \,. \qquad (6.4\text{-}16)$$

The decoder matrix $\boldsymbol{D}^{ALLRAD}$ is calculated once during decoder initialization or bitrate switching and stored in memory.

6.4.6.5.7.2 PSD rendering

6.4.6.5.7.2.1 Upmixing

As a first step, the reference power of the first-order ambisonics signal from the SPAR upmix (second set of transport channels), $\boldsymbol{x}_{FOA}$, is calculated according to

$$P_l^{\text{ref}} = \sum_{i=w,x,y,z}\left\|x_{FOA,i}^l\right\|^2. \qquad (6.4\text{-}17)$$

This is done for each frequency band of the CLDFB denoted by the band index l.

Then $\boldsymbol{x}_{FOA}$ decoded to the output loudspeaker layout with a decoding matric according to Eq. (6.4-16).

From this signal, the prototype power is computed according to

$$P_l^{\text{proto}} = \sum_{i=w,x,y,z}\left\|x_{LS,i}^l\right\|^2. \qquad (6.4\text{-}18)$$

Finally, a per-channel gain is multiplied to the vector of the multi-channel signal $\boldsymbol{x}_{LS}$, i.e., the components of output multi-channel signal become

$$x_{LS,i}^l \rightarrow g_{i,l}x_{LS,i}^l \qquad (6.4\text{-}19)$$

for each channel in the loudspeaker domain indexed by i and each CLDFB band indexed by l.

6.4.6.5.7.2.2 Gain calculation

The gains comprise a directional and a diffuse part:

$$g_{i,l} = g_{i,l}^{\text{dir}} + g_{i,l}^{\text{diff}}. \tag{6.4-20}$$

The gains g_i^l are determined by VBAP panning of the DoA in the CLDFB band l:

$$g_{i,l}^{\text{dir}} = \sqrt{P_l^{\text{proto}} * (1 - \overline{\Psi_l})^2 * P_l^{\text{ref}} * (\text{VBAP}(\theta_l, \boldsymbol{r}_i))^2}$$

The function $\text{VBAP}(\theta^l, \boldsymbol{r}_i)$ denotes the VBAP panning gains of a signal from the direction θ_l to the loudspeaker at the position $\boldsymbol{r}_i$. Ψ_l is the diffuseness in the CLDFB band l . The DoA and diffuseness values is obtained from a direction determiner for low frequency bands, and from the decoding of the bitstream for the high frequency bands (cf. clause 6.4.6.5.4). The bar $\bar{}$ denotes a smoothing over time.

Similarly, the diffuse gains are given by

$$g_l = \sqrt{P_l^{\text{proto}} * \overline{\Psi_l}^2 * P_l^{\text{ref}} * h_i^2}$$

with the diffuse response function $h_i = \frac{1}{\sqrt{N_{LS}}}$, where N_{LS} is the number of loudspeakers in the configure output layout.

6.4.6.5.8 Stereo output

6.4.6.5.8.1 Overview

This subclause describes an optimized decoding path for stereo output from an encoded SBA input. It is activated when stereo output is requested and there are 1 or 2 channel in the first set of transport channels. This applies to the SBA bitrates 13.2 – 32 kbps and 48 – 80 kbps, respectively.

Specifically, these optimizations reduce the delay and computational complexity. The CLDFB analysis and synthesis steps and the time-domain decorrelation can be avoided. Instead, the processing of the upmix of the one transport channel is performed in the DFT domain of the DFT stereo decoder, which is described in clause xxx.

The encoded audio scene comprises the transport channel signals (first set) and a first set of parameters. The transport channels are encoded using DFT stereo. The parameters consist of SPAR parameters for the low frequency bands and DirAC parameters, which describe a DoA and a diffuseness of the scene at the virtual listener position, for the high frequency bands.

This first set of parameters is converted to a second set of parameters. These parameters are employed for generating a representation of the scene by audio channels for reproduction at predefined spatial positions of a stereo loudspeaker of headphone setup. The conversion is described in clause 6.4.6.5.8.1.1

An output interface generates the processed audio scene using the second set of parameters and the first set of transport-channel signals by means of an upmix to stereo. This is described in clause 6.4.6.5.8.1.2. The complete processing diagram of the SBA-to-stereo decoding is shown in figure 6.3-3 and figure 6.3-4 for ACELP and TCX frames, respectively.

6.4.6.5.8.1.1 Parameter conversion

The parameters received and decoded from the bitstream comprise two different types: SPAR parameters (prediction and decorrelation) and DirAC parameters (DoA and diffuseness). The former are available for the 8 lower parameter bands, whereas the latter are available for each of the 4 higher parameter bands. A parameter conversion is performed in order to obtain a second set of parameters to describe the scene in the form of stereo audio channels. This conversion is achieved in the following steps.

6.4.6.5.8.1.1.1 Calculation of the model covariance

For the high frequency bands the inter-channel covariance matrix is estimated from the DirAC parameters according to clause 0 and 5.4.3.7.2.1.

6.4.6.5.8.1.1.2 Calculation of the SPAR parameters

From the model covariance for the high-frequency bands, the SPAR parameters comprising prediction and decorrelation coefficients are calculated according to clause xxx.

6.4.6.5.8.1.1.3 Calculation of the SPAR upmix matrix

From the combined input parameters from the bitstream and clauses 6.4.6.5.8.1.1.1 and 6.4.6.5.8.1.1.2, the upmix matrix is calculated according to clause xxx. It is of the form

$$\boldsymbol{U} = \begin{pmatrix} P_w & 0 & 0 & 0 \\ P_x & C_x & 0 & 0 \\ P_y & 0 & C_y & 0 \\ P_z & 0 & 0 & C_z \end{pmatrix}, \tag{6.4-23}$$

Where P_w is the inverse of the active W weighting, and the parameters P_x, P_y, P_z are the residual prediction gains to restore the first-order ambisonics components that are correlated with the W channel in the encoder input signal. The matrix **U** is computed for each parameter band, but the band index is omitted from the notation here.

Pre-Multiplication of the internal channel vector

$$x_{\text{int}} = \begin{pmatrix} x_w \\ d_x \\ d_y \\ d_z \end{pmatrix}, \tag{6.4-24}$$

By the matrix $\boldsymbol{U}$ for each parameter band yields the first order ambisonics signal $\boldsymbol{x}_{FOA}$. x_w is the audio signal from the one transport channel in the first set. In the full SPAR decoding path, $d_{x;y;z}$ are decorrelator signals. In the optimized stereo output mode, they are replaced by the allpass-filtered signals from the Enhanced stereo filling according to clause xxx.

6.4.6.5.8.1.1.4 Adaptive smoothing

To the matrix $\boldsymbol{U}$ an adaptive smoothing is applied. Specifically, the smoothed upmix matrix is obtained recursively from an averaging filter according to

$$\overline{\boldsymbol{U}} = \beta\overline{\boldsymbol{U}} + (1-\beta)\boldsymbol{U}_{\text{delayed}} \tag{6.4-25}$$

in each DFT-stereo subframe (10 ms).

$\boldsymbol{U}_{\text{delayed}}$ is a delayed version of the matrix $\boldsymbol{U}$ for the current frame, i.e., values from a preceding time frame, which is obtained from a delay line in the decoder. The time resolution of the parameters encoded in this matrix is 1 frame (20 ms) in the 8 lower parameter bands and 1 subframe (5 ms) in the higher 4 parameter bands. It is evaluated in each output DFT-stereo subframe (10 ms).

In this way, the parameters for the high parameter bands are combined, providing an effective time resolution of 20 ms for the low and 10 ms for the high frequency parameter bands. Hence, the smoothed second set of parameters has a time resolution higher than one IVAS input frame but lower than one IVAS subframe.

The length of the delay line in IVAS subframes (5 ms) is

$$N_{delay} = \frac{t_{delay}}{\frac{L_{\text{frame}}}{N_{\text{sf}}}} \tag{6.4-26}$$

with the delay of the audio transport channel in nanoseconds, t_{delay}, the length of a frame in nanoseconds, L_{frame}, and the number of subframes per frame in the IVAS framework, $N_{\text{sf}} = 4$.

The delay for the optimized stereo output mode is given by

$$t_{\text{delay}} = t_{delay}^{SPAR} - t_{delay}^{CLDFB}. \tag{6.4-27}$$

Hence, a reduction of the codec delay by the contribution of the CLDFB filterbank is achieved as compared to the full SBA decoding (t_{delay}^{SPAR}). The lower delay with the optimized stereo output is 32 ms.

The adaptivity of the smoothing enters via the smoothing factor (update rate) β, which is dynamically calculated for each DFT-stereo subframe based on the signal energy. It is given by

$$\beta = \min\left(1, \frac{E_{\text{long}}}{E_{\text{short}}}\right). \tag{6.4-28}$$

The long and short energies E_{long} and E_{short} are obtained by averaging the signal energy E in the one transport channel in the first set over a longer and a shorter time interval. The longer time interval is $N_{\text{long}} = 10$ subframes (100 ms) and the shorter is $N_{\text{short}} = 3$ subframes (30 ms). Eqs. (6.4-25) and (6.4-28) are evaluated separately for each parameter band group.

The factor β is then re-mapped to the range of [0,1] using the formula

$$\beta \rightarrow \max\left(1, \beta - \frac{N_{\text{short}}}{N_{\text{long}}}\right) * \left(\frac{N_{\text{long}}}{N_{\text{long}} - N_{\text{short}}}\right).$$

For less extreme cases, the smoothing is now reduced excessively, so the factor is compressed with a root function (compression function) towards value 1. As stability is particularly important, the in lowest bands. the 4th root is used in bands b=0 and b=l

$$\beta \rightarrow \sqrt[4]{\beta},$$

while all other bands are compressed by a square root

$$\beta \rightarrow \sqrt{\beta}.$$

This way the compression is stronger for the lower frequency band than for the higher frequency bands. Extreme cases remain close to 0 while a less rapid increase in energy does not decrease smoothing so strongly.

Finally, the maximum smoothing is set depending on the band (a factor of 1 would simply repeat the previous value):

$$\beta \rightarrow \min(\beta, \beta_{\max}).$$

The value of $\beta_{\max}$ for each band is listed in Table 6.4-6.

Table 6.4-6: XX

band index	β_{max}
0	0.98
1	0.97
2	0.95
3-12	0.90

The adaptive smoothing according to Eq. (6.3.28), Eq. (6.3-25) describes a weighted combination of the parameters of different subframes, where the weighting factors depend on the signal energy (and amplitude) in the transport channel. The weighting factor of $\boldsymbol{U}_{\text{delayed}}$ (update rate) in Eq. (6.3-25) is greater in a subframe with a higher energy in the transport signal compared to a subframe with a lower signal energy.

Therefore, the weighted combination has the property that a relatively strong smoothing is acquired for some portions of the audio scene, while a relatively weak smoothing is acquired for other portions of the scene. The first portion consists of the slowly varying time intervals of the signal, whereas the second consists of the more rapidly varying ones.

6.4.6.5.8.1.1.5 Derivation of the W and Y channel weights

From the smoothed matrix $\overline{\boldsymbol{U}}$ mixing weights can be obtained to describe the audio scene in the form of stereo output channels depending on the one transport channel x_w and the Enhanced-stereo-filling signal d . The left and right stereo channels x_L and x_R follow from the internal ambisonics channels x_w and x_y as

$$x_L = x_w + x_y, \tag{6.4-29}$$

$$x_R = x_w - x_y. \tag{6.4-30}$$

Evaluating the x- and y-rows of the matrix vector product $\boldsymbol{U}\,\boldsymbol{x}_{\text{int}}$ and inserting them into (6.4-29) and (6.4-30), the stereo channel signals become

$$x_L = \bar{P}_x x_w + \bar{P}_y x_w + \bar{C}_y d, \tag{6.4-31}$$

$$x_R = \bar{P}_x x_w - \bar{P}_y x_w - \bar{C}_y d. \tag{6.4-32}$$

Hence, the converted parameters in the second set are the smoothed matrix elements $\bar{P}_x$, $\bar{P}_y$, and $\bar{C}_y$. They describe a representation of the encoded audio scene in terms of two stereo output channels using the audio channels x_w and d from the first set of transport channels.

The parameters $\bar{P}_x$, $\bar{P}_y$, and $\bar{C}_y$ refer to the input parameter bands given in table 5.3-2. As the channel signals refer to the DFT bins (output parameter bands), the parameters are expanded onto to DFT bins such that they apply approximately to the same frequency ranges.

6.4.6.5.8.1.2 Stereo output processing

The actual processing of the stereo output signal depends on the bitrate and the number of transport channels in the first set (1 or 2).

The processing also depends on the type of core coder that is activated in the SCE (see figure 6.3-3). In the case of 1 transport channel, depending on the signal characteristics of the channel, each frame can be coded with either ACELP or TCX according. This is described in detail in clauses xxx and xxx. In the case of 2 transport channels, only TCX coding is employed.

Generally, Eqs. (6.4-31) and (6.4-32) are evaluated in the DFT domain. The differences arise from the sources of the signals x_w and d .

6.4.6.5.8.1.2.1 ACELP frames

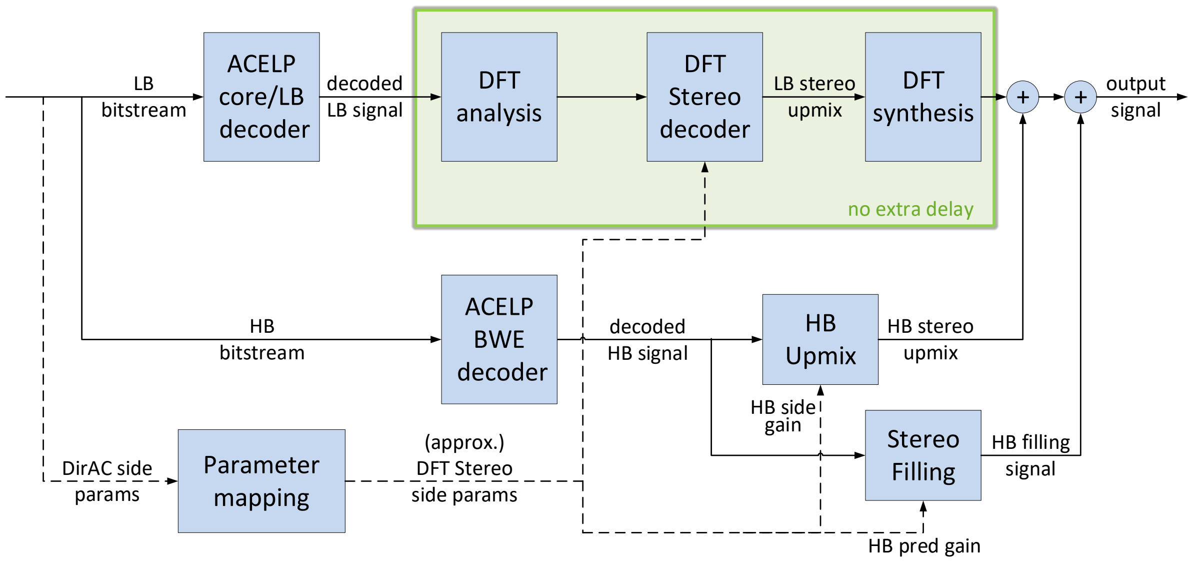


Figure 6.4-3: XX

The decoding of the one transport channel consists of an ACELP core and a bandwidth extension (BWE) (see figure 6.3-4). The generation of the BWE signal is performed in parallel to the DFT-Stereo processing for the raw representation (transport channel and artificial residual signal). Thus, the delays incurred by both algorithms do not accumulate, but only the given delay incurred by one algorithm will be the final delay.

The decoded transport signal from the ACELP core is fed into two branches: one for the processing of the low-frequency portion (upmix to the output signal) and one for the generation and upmix of the enhancement signals for the high frequency portion.

The low-frequency portion of the transport signal x_w is generated by the ACELP core decoder. The low-frequency portion of d comes from the enhanced stereo filling with the low-frequency portion of x_w. These two signals are fed to the upmixer (output interface) in the DFT domain. After the upmixing, the output stereo signal is transformed back to the TD.

The high-frequency portion of x_w is generated in TD by the BWE. The high-frequency portion of d comes from the stereo filling in TD. These two high-frequency signals are upmixed in the time domain. The parameters $\bar{P}_x$, $\bar{P}_y$, and $\bar{C}_y$ for this broadband TD mixing obtained by averaging the values from the parameter bands 8,9, and 10 (cf. table 5.2). Then the stereo decoding can be processed with a delay of 32 ms.

The low- and high-frequency (enhancement) portions of the output stereo signal are then combined in the TD.

6.4.6.5.8.1.2.2 TCX frames

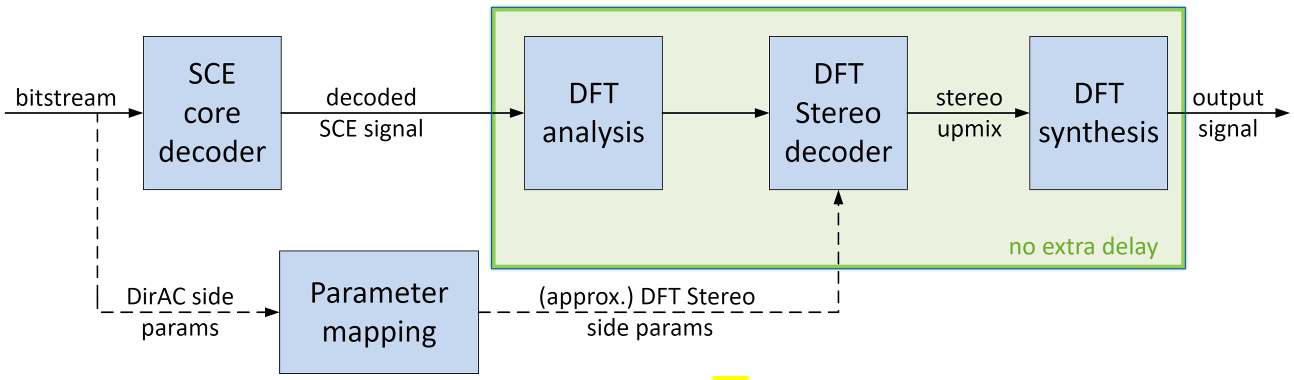


Figure 6.4-4: XX

The decoding simplifies for TCX frames. The processing diagram is shown in figure 6.3-4 for the case of 1 transport channel. All frequencies are processed in the same way. The transport-channel signal x_w is decoded by the TCX core for all DFT bins up to configured audio bandwidth. The residual signal d is obtained from enhanced stereo filling for all DFT bins. The bandwidth of the residual signal is limited to (**insert frequency here**) for perceptual reason.

In the case of 2 transport channels, these are decoded with a CPE. The second transport channel then contains the residual signal d, such that no enhanced stereo filling is required.

The stereo signal is generated by upmixing according to Eqs. (6.4-31) and (6.4-32) in each DFT bin of the spectral representation. It is then transformed back to the TD.

6.4.6.5.8.1.3 Mono output processing

6.4.6.5.8.2 Passive W

With passive W mixing, mono output becomes trivial. The W channel is output as the mono channel. It is already contained in the first set of transport channels.

6.4.6.5.8.3 Active W

To restore the W channel of the ambisonics input approximately, the band-weighting due to the active-W downmix must be reverted on the decoder side. For this purpose, the same processing as for stereo output is used. This is described in clause 6.4.6.5.8. However, only the W channel in Eqs. (6.4-31) and (6.4-32) is computed. Consequently, it is not necessary to run the stereo-filling processing.

6.4.6.5.9 High-order operation mode

6.4.6.5.10 Low-order operation mode

6.4.6.5.11 Bitrate switching

6.4.6.5.12 Head tracking

6.4.6.6 CLDFB synthesis

6.4.7 DTX operation

6.4.7.1 DTX in DirAC

The audio scene analysis done in DirAC is performed for both active and inactive frames and produce two different sets of spatial parameters. Active and inactive frames are categorized using the Voice Activity Detection (VAD) performed on the transport channels. However, the spatial DirAC parameters for the SID frames of the inactive phases are fewer and quantized coarser than the spatial parameters of the active phase. Once, the DirAC parameters are derived as for the active frame for a given parameter band partition, the grouping, the quantization and the coding of the DirAC parameters (diffuseness and directions of arrival (DoAs)) are done as described in clause xxx.

6.4.7.2 DTX in SPAR

The current frame is decoded as active or inactive frame based on active/inactive MD bit in SBA bitstream. This bit sets W_{vad} flag.

In active MD frames, SPAR PR and D coefficients are read from the bitstream as per the bse2 and pair-wise coding described in subclause 5.4.9.2.2. SPAR parameters in DirAC bands are generated as per the subclause 6.4.4.3.

From the coefficients, P and C matrices are generated as described in subclause 6.4.5.1. If the current frame is SID or NO-DATA frame, then further smoothing is applied to the P and C matrices. The P and C matrices that are generated with the SPAR MD received in the current SID frame are applied to the subsequent NO-DATA frames with a linear ramp function.

$$P_{curr} = P_{prevSID} + ramp(P_{currSID} - P_{prevSID})$$

Where, P_{curr} is the current SID or NO-DATA frame, $P_{currSID}$ is the P matrix as per latest SID frame, $P_{prevSID}$ is the P matrix as per the previous SID frame, $ramp = \frac{counter}{8}$, $counter$ is the number of frames since latest SID frame.

C matrix is ramped in a similar way

$$C_{curr} = C_{prevSID} + ramp(C_{currSID} - C_{prevSID})$$

The ramped P and C matrices are then converted to upmix matrix as per Eqn xxx).

6.4.8 SBA PLC

6.4.8.1 PLC in DirAC

During packet loss, the DirAC rendering and upmixing is unchanged, except that the DirAC parameters cannot be read from the bitstream. Instead, previously well received DirAC parameters, i.e., DoAs and diffuseness, are used and hold for generating the output for the current lost frame. However, when the sound field is more diffuse, the directions are less meaningful and can be considered as the realization of a stochastic process. Dithering then helps make more natural and more pleasant rendered sound field by injecting a random noise to the previous directions before using it for the lost frames. The inject noise and its variance σ_{azi} or σ_{ele} are function of the previous well-received diffuseness index, and the current direction is computed as:

$$\begin{cases} azi'[i] = azi[b] + rand\_tri() * \sigma_{\text{azi}}(\text{diff\_idx[b]}), \forall\, i \in I_b \\ ele'[i] = ele[b] + rand\_tri() * \sigma_{\text{ele}}(\text{diff\_idx[b]}), \forall\, i \in I_b \end{cases}$$

where b is the parameter band index, azi and ele are the previously well received and decoded direction angles, and $rand\_tri()$ is simulate random process following the triangle probability density using the following pseudocode:

```
rand_tri()
{
   rand_val = uniform_random();
   if( rand_val <= 0.0f )
   {
      return 0.5f * sqrt(rand_val + 1.0f) - 0.5f;
   }
   else
   {
      return 0.5f - 0.5f * sqrt(1.0f - rand_val);
   }
```

The variance of the injected noise is defined in tables given in Table 6.4-7:

Table 6.4-7: Variance of the dithering noise in function of the diffuseness index, for the azimuth and for the elevation

$\sigma_{\text{azi}}[]$	$\sigma_{\text{ele}}[]$
6.716062e-01, 1.011837e+00, 1.799065e+00, 2.824915e+00, 4.800879e+00, 9.206031e+00, 1.469832e+01, 2.566224e+01	6.716062e-01, 1.011804e+00, 1.796875e+00, 2.804382e+00, 4.623130e+00, 7.802667e+00, 1.045446e+01, 1.379538e+01

For HO-Dirac, no dithering is performed, and only the replacement of the current directions and diffuseness by the previously well-received directions and diffuseness is done.

6.4.8.2 PLC in SPAR

6.4.8.2.1 SPAR processing of erroneous frames

Packet loss concealment of SPAR parameters is applied to avoid a sudden change in spatial image during a sequence of one or more lost frames and to enable a quick recovery when receiving valid frames afterwards.

SPAR decoding of frames marked with the BFI flag as lost or erroneous comprises the following modifications.

When one or more lost frames are encountered by the decoder, which is indicated by the BFI flag, there will be no update of the SPAR parameters. Instead, the parameters of the last frame are kept and the SPAR upmix matrix remains unchanged.

The number of consecutive lost frames *num_lost_frames* is counted by increasing a corresponding counter if the BFI flag is raised. Otherwise, the counter is reset.

If the counter indicates 9 or more lost frames in a row, the following processing takes place:

- The spatial image will be faded over a period of 9 frames towards the predefined spatial configuration of a direction independent spatial image by muting all channels except the W channel. With

$$num\_fade\_frames = \min\left(num\_lost\_frames - 9, 0\right) \tag{6.4-35}$$

the channel dependent gain factor that is applied to the SPAR upmix matrix (see 6.4.5) is calculated according to

$$gain\_fade\_target(ch) = \begin{cases} 1, & \text{if } ch = 0 \\ 1 - min\left(\frac{num\_fade\_frames}{9}, 1\right), & \text{for all other channels} \end{cases} \tag{6.4-36}$$

- At the same time, a fade out of the decoded output is started. This is achieved by applying another gain $gain\_fade\_out$ to the SPAR upmix matrix. The ramp down extends over 33 frames and during that period the gain is decreased by 3dB per frame. The gain parameter is calculated as follows:

$$gain\_fade\_out = 10^{-\min\left(num\_fade\_frames, 33\right)\frac{3}{20}} \tag{6.4-37}$$

6.4.8.2.2 Recovery after one or more lost frames

As SPAR uses different time differential coding schemes for the SPAR parameters depending on the bitrate, slightly different SPAR parameter recovery strategies for the bands with directly coded SPAR parameters are defined after a sequence of one or more lost frames.

SPAR parameters that are derived from a DirAC to SPAR conversion are always marked as valid in good frames and no particular recovery method is needed for that case.

6.4.8.2.3 SPAR parameter recovery for 24.4 kbps and above

Given the time differential coding scheme for SPAR parameters which is defined in 5.4.3.6.1.3 for bitrates of 24.4 kbps and above, after the loss of one or more frames, the decoder will get at least ¼ valid parameters with the first good frame. If the last known valid parameter of a given band is more than 3 frames old, the time differentially coded parameters cannot be reconstructed and are consequently recovered by interpolation from adjacent bands with valid parameters. In case that for a specific band the last known value of the parameter is not older than three frames, this last good value will be kept.

6.4.8.2.4 SPAR parameter recovery for 13.2 kbps and 16.4 kbps

Low bitrates of 13.2 kbps and 16.4 kbps use a method of band interleaved 40ms update rate transmission for efficient parameter coding as described in 5.4.3.6.1.4. Despite the total number of bands is 6 for these configurations, packet loss concealment operates on the normal SPAR banding of 12 bands. For this, the array indicating the bands with valid parameters in a frame is generated by converting the 6 bands representation to an upsampled version with 12 bands. This is, the 12-band parameters for band $2k - 1$ and band $2k$ are obtained from the respective 6-band parameters for band k, where $k = 1 \dots 6$.

6.4.8.2.5 Interpolation of missing SPAR parameters from current existing SPAR parameters

To discriminate between good and not yet known parameters after bad frames, the decoder keeps track of the status with a 12-band variable of the valid bands. In addition, to identify bands that are still invalid, the number of frames since the last received set of valid parameters is counted.

Interpolation of missing SPAR parameters from current existing SPAR parameters then works as follows:

- Starting from zero, for all bands that contain invalid parameters with a band index below the first band containing valid parameters, the SPAR parameters of this first valid band are copied over to the invalid bands.
- Bands with invalid parameters that are located in between a lower band and a higher band with valid parameters are updated with a linear interpolation of the two valid band parameters. If id_0 is the band index of the lower band with valid parameters and id_1 corresponds to the index of the next higher band with valid parameters, the interpolation factor w for parameters of band b is:

$$w = \frac{b - id_0}{id_1 - id_0} \tag{6.4-38}$$

The missing SPAR parameter *p(b)* is interpolated with the help of the valid parameters *p(id₀)* and *p(id₁)* by:

$$p(b) = (1 - \mathrm{w})\, p(id_0) + w\, p(id_1) \tag{6.4-39}$$

- The parameters of all bands that contain invalid parameters with a band index higher than the last band containing valid parameters, are set to these valid parameters.

6.4.9 SBA bitrate switching

6.4.10 SBA output format conversion

6.5 Metadata-assisted spatial audio (MASA) decoding

6.5.1 Overview

6.5.2 Decoding MASA configuration

6.5.3 MASA metadata decoding

6.5.3.1 Energy ratio decoding

6.5.3.1.1 Energy ratio decoding for bitrates up to 256kbps

6.5.3.1.2 Energy ratio decoding for 384kbps and 512kbps

6.5.3.2 Direction decoding

6.5.3.2.1 Direction decoding for bitrates up to 256kbps

6.5.3.2.2 Direction decoding for 384kbps and 512kbps

6.5.3.3 Spread coherence decoding

6.5.3.3.1 Spread coherence decoding for bitrates up to 256kbps

6.5.3.3.2 Spread coherence decoding for 384kbps and 512kbps

6.5.3.4 Surround coherence decoding

6.5.3.4.1 Surround coherence decoding for bitrates up to 256kbps

6.5.3.4.2 Surround coherence decoding for 384kbps and 512kbps

6.5.3.5 Decoding of the second direction parameters

6.6 Object-based audio (ISM) decoding

6.6.1 Discrete ISM coding mode

Figure 6.6-1 is a schematic block diagram illustrating the ISM decoder in the DiscISM mode.

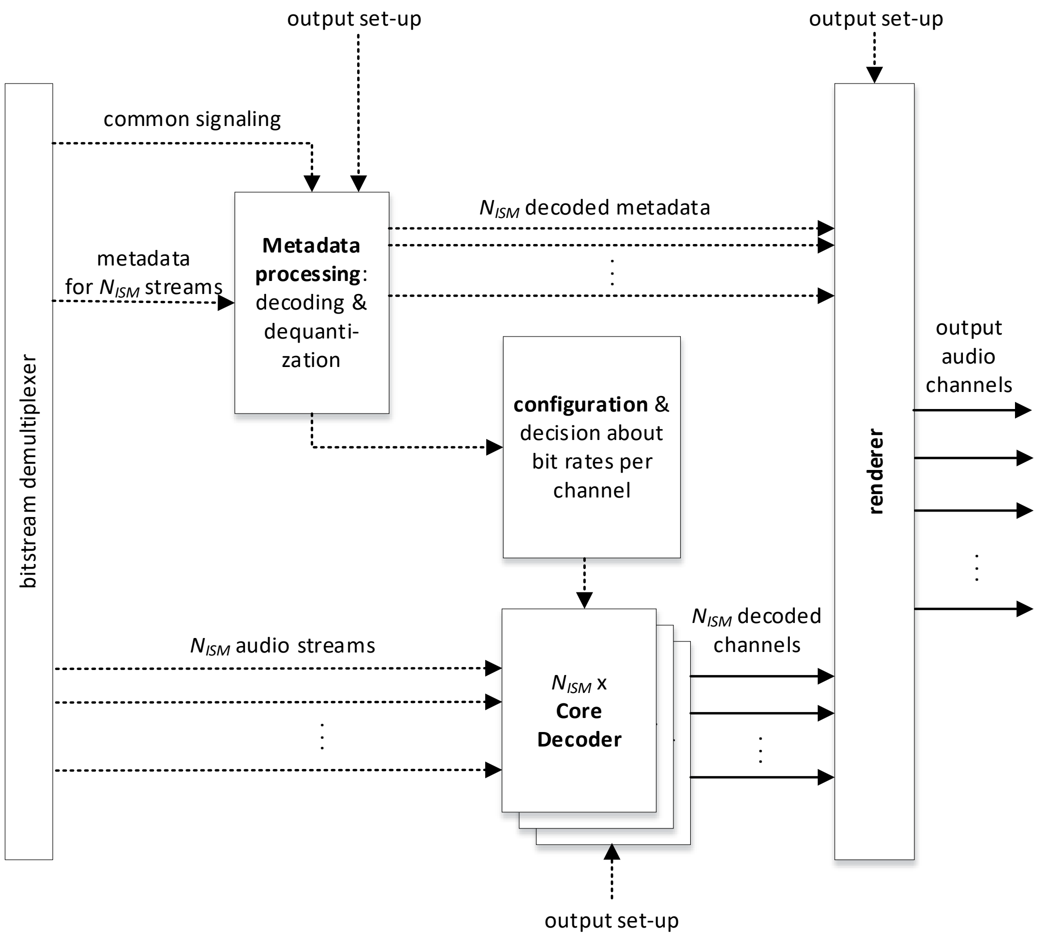

output set-up
output set-up
common signaling
metadata for N_{ISM} streams
Metadata processing: decoding & dequantization
N_{ISM} decoded metadata
configuration & decision about bit rates per channel
bitstream demultiplexer
N_{ISM} audio streams
N_{ISM} x Core Decoder
N_{ISM} decoded channels
renderer
output audio channels
output set-up

Figure 6.6-1: Block diagram of the DiscISM decoder

The bitstream demultiplexer receives a bitstream which is in the structure from Figure 5.6-3. When the IVAS format corresponds to the ISM format, the following is read from the bitstream in a sequential order: a) ISM common signaling incl. the number of audio streams, N_{ISM}, ISM importance classes, $class_{ISM}[n]$, and metadata presence flags, $flag_{meta}[n]$, $n = 0, \ldots, N_{ISM} - 1$, b) the coded metadata for N_{ISM} streams, c) core-coder payloads for N_{ISM} streams. It is noted that the ISM mode is not part of the bitstream in active frames but it is derived from the number of coded streams N_{ISM} and the *ism_total_brate* parameter.

One the metadata are decoded, the information about respective bitbudgets and ISM classes per stream are supplied from the metadata processing module to the configuration module which comprises the bitbudget allocator. The bitbudget allocator at the decoder uses the same procedure as in the bitbudget allocator at the encoder of Figure 5.6-1 to determine the core-decoder bitrates (see Clause 5.6.2.2).

Next, the N_{ISM} transport channels from the bitstream demultiplexer are sequentially decoded using N_{ISM} fluctuating bitrate SCE core-decoders (Clause 6.2.3.1). These core-coder channels (corresponding to the transport channels) are finally supplied to the renderer.

It is noted that Figure 6.6-1 contains arrows indicating “output set-up” parameters. These parameters are e.g. output audio configuration, output sapling rate, etc. and they are used for simplifying some steps during the decoding process.

6.6.2 Parametric ISM coding mode

6.6.2.1 General

In ISM mode, if the bitrate was configured as 24.4 kbit/s or 32 kbit/s, the input audio objects were encoded in Parametric ISM (ParamISM) mode. Per each time frame, the ParamISM decoder decodes an encoded audio signal comprising two transport channels and direction information for three or four audio objects as well as parameter data for each audio object. The parameter data is used along with the transport channels to render the original audio objects to a target output format. To that end, the two transport channels are transformed into a spectral representation and then rendered into a number of audio channels using the direction information (azimuth and elevation), contained in the transmitted parameter data. The direction information is based on the two most relevant objects of each frequency bin. The parameter includes object indices indicating the two most relevant objects per parameter band as well as a power ratio between the two relevant objects. Figure 6.6-2 shows an overview of the ParamISM decoder.

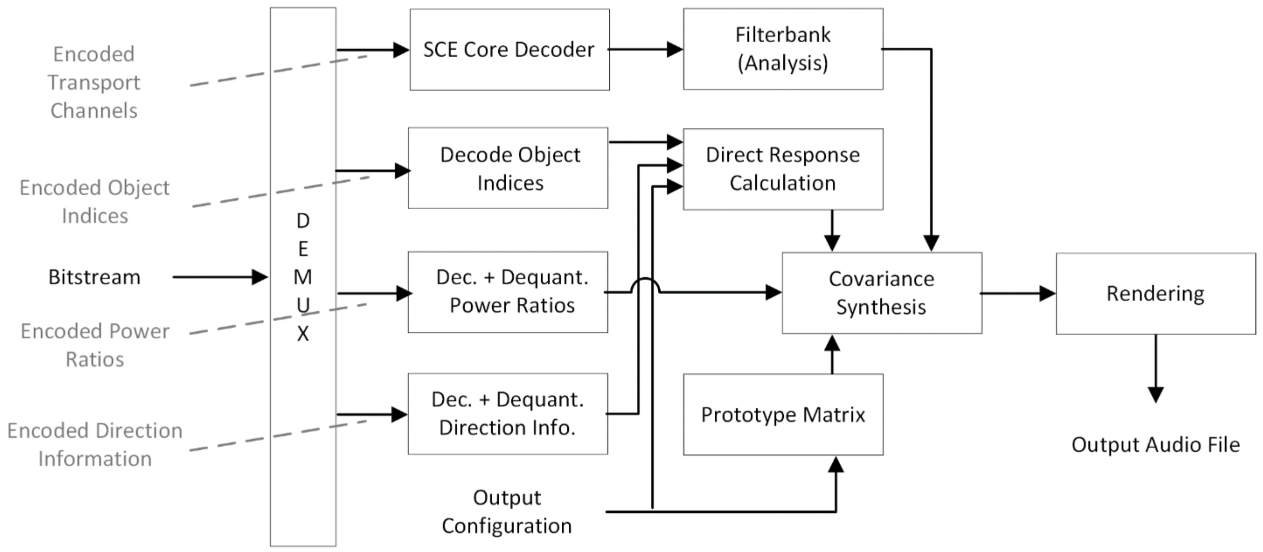


Figure 6.6-2: ParamISM Decoder Overview

6.6.2.2 Parameters

The parametric side information is decoded from the bitstream for each frame. The decoded azimuth and elevation indices are mapped to their corresponding quantized azimuth and elevation values as described in 6.6.3.

The decoded power ratio index of each parameter band is mapped to its corresponding quantized power ratio value according to

$$\hat{r}_1 = idx/14 + 0.5, \tag{6.6-1}$$

and the second, not transmitted power ratio, is obtained by

$$\hat{r}_2 = 1 - \hat{r}_1, \tag{6.6-2}$$

where $idx \in \{0, 1, 2, 3, 4, 5, 6, 7\}$ so that the resulting values of $\hat{r}_1$ always range between 0.5 and 1 and the values of $\hat{r}_2$ always range between 0 and 0.5.

The obtained parametric data then comprises, per each time frame: azimuth and elevation information for each original input object, as well as, per parameter band, the object indices indicating the two most relevant objects and the power ratios indicating the contribution of each of the two relevant objects in terms of signal power. As per the encoder-side definition of the parameter bands, there are 22 object indices per original time frame as well as 11 power ratios. During rendering, these parameters are expanded from the parameter band resolution (1 time slot by 11 frequency bands) to the filterbank resolution (16 time slots by 60 frequency bins) by means of repetition. Futhermore, the noisy speech flag is decoded for each frame, indicating whether the encoded audio content as classified as noisy speech.

6.6.2.3 Downmix

The two transport channels are decoded from the bitstream via the SCE decoder described in 6.2.3.1.

For the further processing steps, the decoded time-domain transport channels are transformed into a spectral representation by means of a CLDFB filterbank as defined in 6.2.5.2, resulting in a time-frequency resolution of, e.g., 16 time slots and 60 frequency bins per transport channel in the case of 48 kHz input.

[Editor’s note: Energy compensation to be elaborated on in next version.]

6.6.2.4 ParamISM Decoding and Rendering

6.6.2.4.1 Mono, stereo, and binaural output

For mono output, rendering to a mono downmix is employed [ref].

For stereo output, the decoded transport channels are output.

For binaural output configurations, the parametric binaural renderer is used [ref]. Prior to rendering, a loudness correction is employed, in the same way as done in the SBA mode [ref].

6.6.2.4.2 Loudspeaker and Ambisonics output

To render the decoded transport channels to a loudspeaker or Ambisonics configuration, EFAP described in 7.2.1.3 is used in a first step to calculate a direct response information for each object. Based on the transmitted and dequantized direction information (azimuth and elevation) of the original input objects as well as information on the target loudspeaker layout, the EFAP direct response values are determined in the form of direct response vectors $\boldsymbol{dr}_i$, with i the object index, resulting in one vector per object with each vector containing one value per each output channel (excluding any LFE loudspeaker channels). These values are specific to the target layout, i.e., the number and location of the loudspeakers and serve as the panning gains. The direct response vectors are calculated once per frame and object.

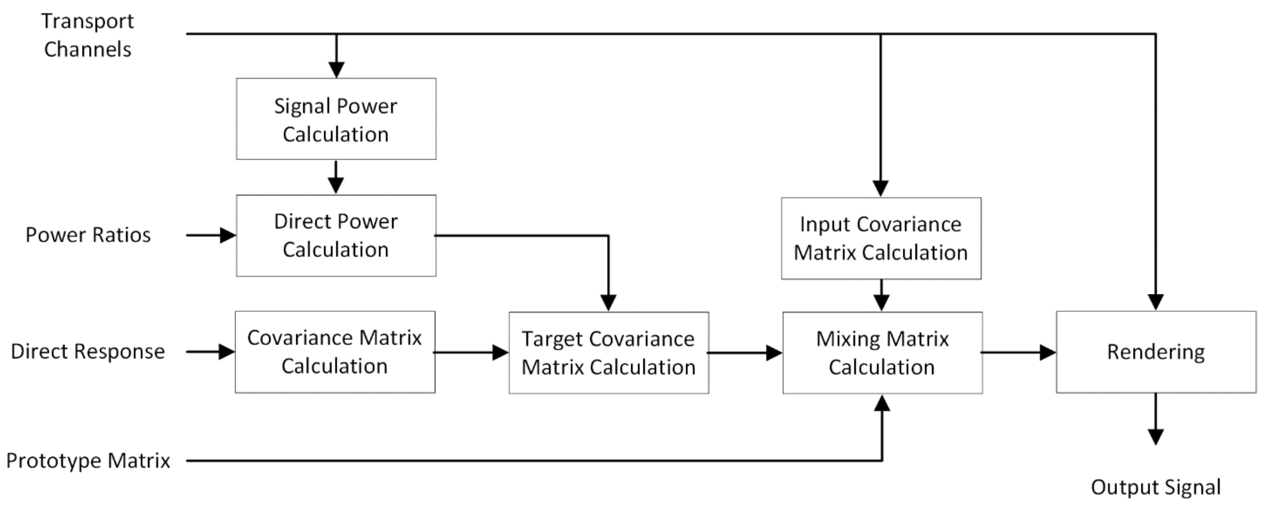


Figure 6.6-3: Overview of covariance synthesis

For rendering, covariance synthesis is employed using the direct response information as well as the information on the number of audio channels to acquire the output audio channels. Figure 6.6-3 shows an overview of the covariance synthesis. Specifically, a mixing matrix $\boldsymbol{M}$ is calculated for each frequency bin of the current frame and multiplied to the two decoded transport channels $\boldsymbol{x} = \boldsymbol{x}(k,n) = [\boldsymbol{X}_1, \boldsymbol{X}_2]^{\mathrm{T}} = [X_1(k,n), X_2(k,n)]^{\mathrm{T}}$, resulting in the output audio channels $\boldsymbol{y} = \boldsymbol{y}(k,n) = [Y_1(k,n), Y_2(k,n), Y_3(k,n), \ldots]^{\mathrm{T}}$:

$$\boldsymbol{y} = \boldsymbol{M}\boldsymbol{x} \qquad (6.6\text{-}3)$$

The mixing matrix is calculated from the direct response values, the prototype matrix, the input covariance matrix $\boldsymbol{C}_x$ of the transport channels, the reference power of the transport channels and the target covariance matrix $\boldsymbol{C}_y$ of the output channels.

The input covariance matrix is given by:

$$\boldsymbol{C}_x = \begin{bmatrix} \boldsymbol{X}_1\boldsymbol{X}_1 & 0 \\ 0 & \boldsymbol{X}_2\boldsymbol{X}_2 \end{bmatrix} \qquad (6.6\text{-}4)$$

Only the main diagonal elements are used, all other matrix elements are explicitly set to zero.

The prototype matrix is defined as:

$$M_{\text{proto}}(j,i) = \begin{cases} 1 \ \text{ if}\left(i = 0 \ \wedge \ \theta_j > 0\right) \vee (i = 1 \wedge \theta_j < 0) \\ 0 \ \text{ if}\,(i = 0 \wedge \theta_j < 0) \vee (i = 1 \wedge \theta_j > 0) \\ \qquad\qquad 0.5 \ \text{ if}\,\theta_j = 0 \end{cases} \qquad (6.6\text{-}5)$$

where $i \in \{0, 1\}$ denotes the transport channel index, $j \in \{0, 1, \ldots, N\}$ denotes the output channel index, with N the number of output channels, and θ_j denotes the azimuth value of loudspeaker j.

The reference power per transport channel is obtained by:

$$P_i\,(k,n) = |X_i(k,n)|^2, \qquad (6.6\text{-}6)$$

where X_i denotes the i-th transport channel in the spectral domain, k denotes the frequency bin index, and n denotes the time slot index. Since the rendering is done using the entire frame (as opposed to subframes), the signal powers of all time slots are accumulated. Furthermore, both transport channels are added to obtain the total signal power of the downmix:

$$P_{\text{DMX}}(k) = \sum_{i=0}^{1} \sum_{n=0}^{15} P_i(k,n) \qquad (6.6\text{-}7)$$

To calculate the target covariance matrix $\boldsymbol{C_y}$ for one frequency bin, the transmitted object indices and power ratios are required. For each of the 11 parameter bands, the two corresponding object indices, indicating the two relevant objects of each parameter band, are used to determine which direct response vectors to include in the direct response matrix.

The target covariance matrix for the frequency bin k is obtained as:

$$\boldsymbol{C_y} = \boldsymbol{RER}^H, \qquad (6.6\text{-}8)$$

with the direct response matrix $\boldsymbol{R} = [\boldsymbol{dr}_1 \ \boldsymbol{dr}_2]$ that contains the previously obtained direct response vectors of the two relevant objects (as indicated by the transmitted object indices) and a diagonal matrix $\boldsymbol{E}$, with $e_{i,i} = E_i$ and

$$E_i = DP_i(k) \qquad (6.6\text{-}9)$$

that contains the direct powers

$$DP_i(k) = \hat{r}_i * P_{\text{DMX}}(k) \qquad (6.6\text{-}10)$$

Here, i denotes the relevant object index. $\hat{r}_i$ is the quantized power ratio value of the i-th relevant object and is valid for all frequency bins that are contained in the parameter band that $\hat{r}_i$ was calculated for.
Finally, the mixing matrix is obtained by [Editor's note: optimized implementation to be elaborated on in next version]

After applying the mixing matrix, the resulting output channels are converted into the time domain by means of a CLDFB synthesis filterbank as defined in 6.2.5.3 to acquire the final output audio channels in the time domain.

6.6.2.4.3 EXT output

For EXT output, no rendering to a specific output format is conducted. Instead, the encoded objects are output again as objects so that the number of input channels matches the number of output channels. Furthermore, the original associated metadata (if present) is output in quantized form according to 6.6.3.

The direct response values in this case are not obtained via EFAP, but explicitly set to:

$$R(j,i) = \begin{cases} 1 & if \ i = j \\ 0 & if \ i \neq j \end{cases} \qquad (6.6\text{-}11)$$

where i denotes the object index and j denotes the output channel index.

The prototype matrix is defined as:

$$M_{\text{proto}}(j,i) = \begin{cases} 1 \ \text{ if}\left(i = 0 \ \wedge \ \theta_j > 0\right) \vee (i = 1 \wedge \theta_j < 0) \\ 0 \ \text{ if}\,(i = 0 \wedge \theta_j < 0) \vee (i = 1 \wedge \theta_j > 0) \\ \qquad\qquad 0.5 \ \text{ if}\,\theta_j = 0 \end{cases} \qquad (6.6\text{-}12)$$

where $i \in \{0,1\}$ denotes the transport channel index, $j \in \{0,1,\dots,N\}$ denotes the output channel index, with N the number of input objects, and θ_j denotes the azimuth value of object j. In the case of the external output configuration, the number of input objects/channels is equal to the number of output objects/channels.

The remaining rendering steps corresponds to those of loudspeaker rendering.

6.6.3 ISM metadata decoding

The metadata decoding and dequantization process is the same regardless of the ISM mode being DiscISM or ParamISM and starts with decoding the ISM common signaling including the number of audio streams, N_{ISM}, and metadata presence flags, $flag_{meta}[n]$, $n = 0, \dots, N_{ISM} - 1$.

The metadata are then decoded and processed in the metadata processing module (see Figure 6.6-1) where the metadata are decoded and dequantized for audio streams with active content. The decoding and dequantization performed by the metadata processing module at the decoder are the inverse of the quantization and coding performed by the metadata processing block at the encoder from Figure 5.6-1 and Clause 5.6.4. The produced decoded and dequantzied metadata for the N_{ISM} audio streams are then supplied to the renderer or a metadata writer in case of an external rendering.

6.6.4 DTX operation decoding

The ISM format decoder in the DTX operation follows the structure of the ISM decoder in active frames from Figure 6.6-1 though there are a few differences when SID or NO_DATA frames are decoded. In case of the SID frame, which has a structure from Figure 5.6-6, the bitstream demultiplexer extracts a) the number of streams N_{ISM}, b) spatial information including the ISM mode information, c) ISM common signalling, d) metadata, and e) CNG core-coder parameters. More details are provided in following Clauses.

6.6.4.1 Spatial information

The spatial information in the SID frame correspond to the ISM CNG parameters.

6.6.4.2 Metadata Dequantization and Decoding

The metadata processing module is supplied with the coded metadata of the transmitted N_{ISM} audio streams and the ISM common signaling and dequantize the metadata for the audio streams. At the decoder, an inverse processing to the encoder processing from Clause 5.6.6.3 is used while a certain adjustment to the decoded metadata values is used.

At the ISM decoder, the SID frames are received at a certain rate (8 frames by default), resulting in a possibility that received MD parameter values are changed between SID frames with a large step. In order to avoid subjective artifacts from this, the metadata processing module adjusts the MD parameter values at the ISM decoder such that the MD parameter value differences are lowered between frames. Specifically, an interpolation between the true decoded and dequantized current frame MD parameter value and the previous frame MD parameter value is applied in certain frames following the SID frame. This results in the MD parameter values evolving more smoothly while the smoothing is applied in several CNG frames, or several active frames, or several CNG and active frames following an SID frame.

The adjustment, or smoothing, is applied on each MD parameter such that the maximum difference (step) of a MD parameter between two adjacent frames is not more than a given threshold. Let's suppose the current decoded and dequantized MD parameter value of azimuth being θ_{dec} for one stream and the maximum smoothing step for azimuth difference being Δ_θ. Note that the stream index n in the metadata parameter is omitted in this Clause for simplification while it is supposed that the same adjustment is done for every metadata parameter (i.e. azimuth and elevation included in the SID payload) of every stream. Then:

$$\theta_{dec} = \begin{cases} \theta_{last} + \mathrm{sgn}(\theta_{true} - \theta_{last}) \cdot \Delta_\theta & \text{if } |\theta_{true} - \theta_{last}| > \Delta_\theta \\ \theta_{true} & \text{otherwise} \end{cases} \qquad (6.6\text{-}13)$$

where θ_{true} is the transmitted quantized azimuth (a quantized MD parameter value in general) in the SID frame and θ_{last} is the azimuth (a MD parameter value in general) in the preceding frame and updated at the end of each frame decoding as $\theta_{last} = \theta_{dec}$. The azimuth smoothing step value is set to $\Delta_\theta = 5$ and the elevation smoothing step value to $\Delta_\varphi = 5$. Further, the operation sgn(x) in Equation (6.6-13) is a mathematical expression

$$\mathrm{sgn}(x) = \begin{cases} 1, & \text{if } x \geq 0 \\ -1 & \text{otherwise} \end{cases} \quad (6.6\text{-}14)$$

Further, the maximum number of frames to apply the smoothing step is set to a given threshold value D_{max}. Specifically, it is not limited in inactive segments but limited in active segments to D_{max} = 5 frames while the same value of parameter D_{max} is used for all metadata parameters.

Moreover, the smoothing step is skipped in active frames when the absolute value of the difference between the current frame quantized MD parameter value θ_{true} and the previous frame decoded MD parameter value θ_{last} is higher than the smoothing step value Δ_θ multiplied by the threshold value D_{max}. For example, in case of the azimuth MD parameter, it means that:

$$\text{if} \quad |\theta_{true} - \theta_{last}| > D_{max} \cdot \Delta_\theta \quad \text{then skip the smoothing} \quad (6.6\text{-}15)$$

The relation (6.6-15) also applies to the elevation MD parameter and it is used to prevent wrong smoothed MD parameter estimation when the true value of the MD parameter is changing significantly from frame to frame and the number of frames between the last SID frame and the active frame is low (in general lower than D_{max}).

6.6.4.3 ISM DTX core decoding

In case of SID frame, the core-decoder SID bitrate is assigned to one SCE core-decoder in which a FD-CNG from EVS (see Clause 6.7.3 in [3]) is used to generate the output signal while bitrate of 0 kbps is assigned to the other core-decoders. Thus, the other core-decoders are fed by their respective CN parameters and spatial information SID. Obviously, in case of NO_DATA frame, the bitrate of 0 kbps is assigned to all core-decoders.

6.6.5 ISM bitrate switching

6.6.5.1 Metadata handling in bitrate switching

The direction metadata is encoded independent of the bitrate conditions. However, gain metadata coding operation is only active for bitrates higher than 64kbps. Therefore, a mechanism is introduced to ensure smooth transition between the detail level of metadata in case of bitrate switching. Two detail levels are described for this operation.

In the case of level 1 detail of metadata, only the common metadata (azimuth and elevation) decoder is active. If the second or extended level of metadata detail is used, the radius r and the orientation angles yaw and $pitch$ are also decoded with extended metadata decoder. Note that the first level of detail, azimuth and elevation is a subset of the extended level of detail. The level of detail for metadata to be decoded is dependent on the variable received via bitstream, bt_{extmd}. If bt_{extmd} is set, the encoded metadata includes the extended metadata, level 2 detail.

A variable to track the state of the level of metadata is introduced as ac_{extmd}. Active state variable, ac_{extmd}, determines the detail level of the metadata to be used for rendering. At the beginning of the decoding process, ac_{extmd} is set to -1 to indicate the decoding process of the first frame.

A counter cnt_{extmd} is introduced to keep track of the number of received frames from different levels of detail. The level 1 detail of metadata is always encoded and decoded independent of the transmission conditions. However, some parts of the level 2 detail are only encoded for high bitrates (>=64kbps). In the event of the changes in the bitrate conditions, the metadata change counter, cnt_{extmd} keeps track of the duration for such changes before updating the active state, ac_{extmd}. The counter, cnt_{extmd} controls the status of the metadata detail state and allows the updates in the metadata memory accordingly via the variable ac_{extmd}.

Before the decoding process, all metadata parameters (level 1 and level 2) are set to their default values and stored in the metadata memory.

For the first received frame, indicated by $ac_{extmd} = -1$, cnt_{extmd} is set to 'zero' and the state ac_{extmd} is updated with the bitstream level, bt_{extmd}:

$$cnt_{extmd} := 0 \quad (6.6\text{-}16)$$

$$ac_{extmd} := bt_{extmd} \quad (6.6\text{-}17)$$

If bt_{extmd} indicates level 2 detail, the metadata (both level 1 and 2) is decoded according to the level encoded, their values are used for rendering, and stored in the memory.

In the case of bt_{extmd} indicating level 1 detail, the level 2 metadata parameters (yaw, pitch, and radius) are reset to their default values to be used in the rendering process. The metadata memory is updated with the default values of level 2 metadata in this case. The level 1 metadata parameters are updated according to the bitstream in the memory and those values are used for rendering.

For the static bitrate condition where the metadata detail state currently being used is the same with the bitstream metadata detail level, i.e., $ac_{extmd} == bt_{extmd}$, the operations follow the logic in the previous paragraph with Equations (6.6-16) and (6.6-17).

If the received metadata detail level is different from the current state, $ac_{extmd} \neq bt_{extmd}$ when $ac_{extmd} \neq -1$, it corresponds to the change in the bitstream conditions. In this case, cnt_{extmd} is incremented by one:

$$cnt_{extmd} := cnt_{extmd} + 1. \qquad (6.6\text{-}18)$$

Following the incrementation, cnt_{extmd} is subjected to a threshold of 5 to see if the change is persistent over a period (5 frames in this case). If counter $cnt_{extmd} < 5$, meaning the change has been effective for less than 5 frames, the metadata values from the memory are used for rendering. The update on the level 2 metadata parameters is withheld for a short time window. When the counter cnt_{extmd} reaches the threshold, Equations (6.6-16) and (6.6-17) are applied followed by the same operations in the first frame. The renderer uses the received and decoded values of level 2 detail metadata when bt_{extmd} indicates level 2 detail. Otherwise, the received and decoded values of level 1 metadata, and the default values of level 2 metadata are used for rendering. The metadata memory is updated with the metadata values used for rendering.

6.6.5.2 Codec reconfiguration in bitrate switching

6.6.6 ISM output format conversion

6.7 Multi-channel audio decoding

6.7.1 MC format decoding

6.7.1.1 LFE channel decoding

6.7.1.1.1 MDCT LFE decoder

6.7.1.1.2 MDCT Coefficients decoding

6.7.1.1.2.1.1 Arithmetic decoder

6.7.1.1.2.1.2 De-indexing

6.7.1.1.2.2 Inverse MDCT

6.7.1.1.2.3 output LPF and delay adjustment

6.7.1.1.2.4 LFE PLC

LFE frames marked by the BFI flag as lost or unusable are not decoded using the regular LFE decoding scheme. Instead, a packet loss concealment mechanism for the LFE channel is applied following a linear predictive approach in time domain. The general concept is to reconstruct the samples of a substitution frame by operating an all-pole filter that is derived through bandwidth sharpening from an LPC synthesis filter. The filter is tuned in on a portion of the previously reconstructed signal and is then operated as a resonator by keeping it ringing for the samples of the substitution frame. To reduce complexity of the computations and memory needs, operations are done in down-sampled domain at a sampling frequency of 1600 Hz. For the case of burst errors with multiple successive frame losses, an attenuation/muting mechanism is provided.

6.7.1.1.2.4.1 Burst loss determination

In case a frame is marked with the BFI flag, an LFE burst loss counter is incremented. If, after the increment, the counter value is less than a threshold of LFE_PLC_MUTE_THR=10, then PLC operation without muting behavior takes place. Otherwise, a slightly modified PLC procedure with muting behavior is applied with an attenuation of approximately 3dB per frame effective starting from the 10th successive frame marked with the BFI flag.

Frames not marked with the BFI flag cause the LFE burst loss counter to be set to zero. Otherwise, if the LFE burst loss counter exceeds the threshold of 10, it is set to 10.

6.7.1.1.2.4.2 LFE substitution frame processing

6.7.1.1.2.4.2.1 Buffering of previously synthesized LFE channel

PLC for the LFE channel relies on a buffer of 240 samples of the previously synthesized LFE signal in down-sampled domain. To maintain that buffer in a prepared state for the case the next frame is marked as lost or unusable, each synthesized LFE frame, either obtained through regular LFE decoding or LFE PLC, is firstly decimated to the sampling frequency of 1600 Hz and then pushed from right-hand side into the buffer while discarding the corresponding number of 32 oldest samples. As the LFE channel contents can assumed to be sufficiently band limited, the decimation merely takes every *n*th sample of the LFE signal at original sampling rate, starting from the zeroth sample. The decimation factor *n* equals the original sampling rate (48000, 32000 or 16000 Hz) divided by 1600 Hz.

6.7.1.1.2.4.2.2 Resonator filter calculation

Upon raised BFI flag, the substitution frame processing begins with the calculation of an initial set of LPC synthesis filter coefficients. To that end, the 240 samples of the previously synthesized and down-sampled LFE signal are first windowed with a 240-point symmetric Hamming window. Next, the autocorrelations of the windowed signal $s_w(n)$ are computed by

$$r_c(k) = \sum_{n=k}^{239} s_w(n) s_w(n-k)\,, \quad k = 0 \ldots 20 \ . \qquad (6.7\text{-}1)$$

If $r_c(0)$ is below a threshold of 0.0000024, then a buffer prepared to store the substitution frame is zeroed and the procedure continues as described in clause <td alias signal calculation>.

Otherwise, the procedure continues with solving the following equation system using the Levinson-Durbin recursion:

$$\sum_{k=1}^{20} a_k r(|i-k|) = -r(i), \quad i = 1 \ldots 20 \ . \qquad (6.7\text{-}2)$$

The Levinson-Durbin algorithm is described in clause <EVS specification: 5.1.9.4>.

The resulting LPC synthesis filter is subsequently modified by bandwidth sharpening. To that end and in case no attenuation/muting takes place, the filter coefficients are modified using a bandwidth sharpening factor γ according to

$$a_{\gamma,k} = a_k \gamma^k, k = 1 \dots 20 \ . \qquad (6.7\text{-}3)$$

6.7.1.1.2.4.2.3 Calculation of bandwidth sharpening factor

The bandwidth sharpening is controlled by the sharpening factor $\gamma = 1 + \delta$, where δ is used as a helper variable. δ is successively iterated using a nested interval technique such that the resulting modified filter is as close as possible at the stability limit but still stable.

6.7.1.1.2.4.2.3.1 Stability test of modified filter

Part of the procedure is the determination if the all-pole filter with modified coefficients is stable. The corresponding test relies on determining if the all-pole filter with coefficients modified according to equation (6.7-3) and a specific value of δ_{test} is stable. To that end, the modified filter coefficients are firstly calculated according to

$$a_{test,k} = a_k (1 + \delta_{test})^k, k = 1 \dots 20 \ . \qquad (6.7\text{-}4)$$

Subsequently, stability is checked by first converting the linear prediction coefficients $a_{test,k}$ to reflection coefficients $refl_{test,k}$ using a backwards Levinson Durbin recursion as described in clause <EVS specification: 5.2.6.1.4.1>. If the absolute value of any of the reflection coefficients is equal or greater than 1, the stability test declares the filter instable, otherwise it is declared stable.

In the following description, the stability test for a set of filter coefficients $a_k, k = 1 \dots 20$ and a sharpening parameter δ is shortly referred to as check_stab($\boldsymbol{a}, \delta$). It returns the value True in case the filter is declared stable. Otherwise, it returns the value False.

6.7.1.1.2.4.2.3.2 Nested interval search

The nested interval search to find the maximum δ for which the modified filter is still stable is done using the following iteration.

In preparation of the iteration, firstly, the initial interval size for the nested interval search is determined.

Starting with an initial value of $\epsilon = 0.01$, it is checked using check_stab($\boldsymbol{a}, \delta_{test}$) with $\delta_{test} = \epsilon$ if the corresponding all-pole filter is stable. While this is true, ϵ is doubled and the check is repeated. As this loop ends with an ϵ value resulting in an instable filter, the latest doubling is subsequently reverted. To catch the case where already the initial value of $\epsilon = 0.01$ leads to an instable filter, for that case, ϵ is successively halved followed by a stability check until the resulting filter is stable. The result of this preparation procedure is an iteration start value of $\delta_0 = \epsilon$ for which a stable filter is ensured and for which a modified filter with $\delta = 2\epsilon$ is unstable.

The nested interval search is then carried out according to the following pseudo-code with the iteration start value $\delta_{test} = \delta_0$ and initial interval size of $\epsilon = \delta_0/2$. The iteration stops with $\delta = \delta_{test}$ if the latest sharpening parameter δ_{test} provides a stable filter and when the interval size is below an end value of $\epsilon_{stop} = 10^{-5}$.

```
while ( True )
    {
        $\delta_{test} = \delta_{test} + \epsilon$
        stable = check_stab( $\boldsymbol{a}, \delta_{test}$ )
        if ( !stable )
        {
            if ( $|\epsilon| > \epsilon_{stop}$ )
            {
                $\epsilon = -|\epsilon|/2$
            }
            else
            {
                $\epsilon = -|\epsilon|$
            }
        }
        else
        {
            if  ($|\epsilon| < \epsilon_{stop}$)
            {
                break
            }
            $\epsilon = |\epsilon|/2$
        }
    }
$\delta = \delta_{test}$
```

6.7.1.1.2.4.2.4 Modification of bandwidth sharpening factor in case of attenuation/muting

In case attenuation/muting takes place, the filter coefficients are modified using a modified bandwidth sharpening factor that additionally imprints the attenuation behavior. While the previously calculated bandwidth sharpening factor γ essentially results in that the modified LPC synthesis filter produces a sustained substitution signal, for attenuation/muting, γ is additionally scaled to perform attenuation. Scaling γ has the effect that the poles of the modified synthesis filter are moved by the scaling factor inwards the unit circles and that the filter response decays accordingly. To achieve a target attenuation of 3dB per frame (0.02s), given the sampling rate of 1600, the bandwidth sharpening parameter γ is modified with the following factor:

$$\alpha_{mute} = \sqrt[0.02\cdot 1600]{10^{-3[dB]/20}} = 0.9892647 \ . \qquad (6.7\text{-}5)$$

Accordingly, the resulting sharpening factor to be used replaces the previously calculated sharpening factor according to $\gamma = \alpha_{mute}\gamma$.

It is notable that α_{mute} remains unchanged for any subsequent substitution frames of a given frame loss burst. Further attenuation and ultimately muting is accomplished since a new substitution frame is generated based on the already attenuated previous substitution signal.

6.7.1.1.2.4.2.5 Generation of substitution signal

With the coefficients of the all-pole resonator filter available, the samples of the substitution frame in subsampled time domain are generated by IIR filtering according to the following filter recursion:

$$\hat{s}(n) = \sum_{k=1}^{20} a_k \gamma^k \hat{s}(n-k) \ , \ n = 0 \dots L-1 \quad . \qquad (6.7\text{-}6)$$

Note that prior to the filtering, the filter memories are initialized with the most recent samples of the buffered previously synthesized LFE channel signal in down-sampled domain:

$$\hat{s}(n) = s(240 - 20 + n) \, , \ n = -20 \dots -1 \quad . \qquad (6.7\text{-}7)$$

The filtering is carried out for L=58 samples in down sampled domain, which accommodates for 32 samples of a frame, 16 samples needed for overlap-add processing in the LFE MDCT decoding/reconstruction framework and 10 samples delay of the lowpass filter used in up-sampling to a sampling rate of 48000 kHz.

After generation of the substitution signal in subsampled domain, it is resampled to a sampling frequency of 48000. This is done in the numerically efficient polyphase implementation described in clause <IVAS specification: modify_Fs> using an up-sampling factor of $30 = 48000/1600$. The interpolation lowpass filter is designed using a Kaiser window FIR filter design technique with a cutoff frequency of 800 Hz at the sampling frequency of 48000 Hz. The filter delay is 300 samples at 48000 Hz, corresponding to 10 samples in 1600 Hz down sampled domain. Note, to properly warm up the interpolation filter state memory, it is needed to start the up-sampling 30 samples prior to the end of the previously synthesized and down-sampled LFE signal. In case the output audio sampling frequency is other than 48000 Hz, the substitution signal is subsequently further decimated to that respective sampling frequency.

Subsequently, the substitution signal is windowed and time-domain aliased as described in <Ref LFE encoder>. The two resulting windowed and time-domain aliased signal buffers are then inversely time-domain aliased, windowed, and overlap-added with the previously synthesized frame as described in <Ref LFE decoder>. Finally, the synthesized LFE channel signal is buffered for the event of a future frame loss, as described in subclause 6.6.1.1.2.4.3.1.

6.7.1.1.2.4.3 General

6.7.2 Multi-channel MASA (McMASA) decoding mode

6.7.2.1 McMASA decoding mode overview

6.7.2.2 McMASA metadata decoding

6.7.2.2.1 Spatial metadata decoding

6.7.2.2.2 LFE-to-total energy ratio decoding

6.7.2.3 McMASA transport audio signal decoding

6.7.2.4 McMASA synthesis

6.7.2.4.1 McMASA synthesis overview

6.7.2.4.2 McMASA LFE synthesis

6.7.2.4.3 McMASA loudspeaker synthesis optimisation

6.7.3 Parametric MC decoding mode

6.7.3.1 ParamMC Overview

The ParamMC decoder and synthesizer generates the multi-channel synthesized signal from the downmix signal using the received transport signal (i.e. the downmix signal) with 2 or 3 channels and the received side information containing the channel level and correlation information of the original signal.

6.7.3.2 ParamMC Parameter Decoding

6.7.3.2.1 Common ParamMC parameter decoding

- f_{LFE} with 1 bit
- the encoded band width with 2 bits
- the parameter frame indicator with 1 bit
- the transient flag with 1 bit, signalling the occurrence of a transient in the frame
    - if the transient flag is 1, the transient position with 3 bits, signalling in which slot the transient has occured

6.7.3.2.2 ICC and ICLD Parameter decoding

6.7.3.3 ParamMC Transport Audio Signal Decoding

6.7.3.4 ParamMC Synthesis

6.7.3.4.1 ParamMC Target Covariance Calculation

6.7.3.4.2 ParamMC Mixing Matrix Calculation

6.7.3.4.2.1 Overview

6.7.3.4.2.2 Direct Mixing Matrix

6.7.3.4.2.3 Residual Mixing Matrix

6.7.3.4.3 Decorrelation

6.7.3.4.4 Upmix

6.7.4 Parametric upmix coding mode for 7.1.4

6.7.4.1 General

A schematic of the decoder of the parametric upmix coding mode for 7.1.4 at 160 kbps is shown in the following Figure.

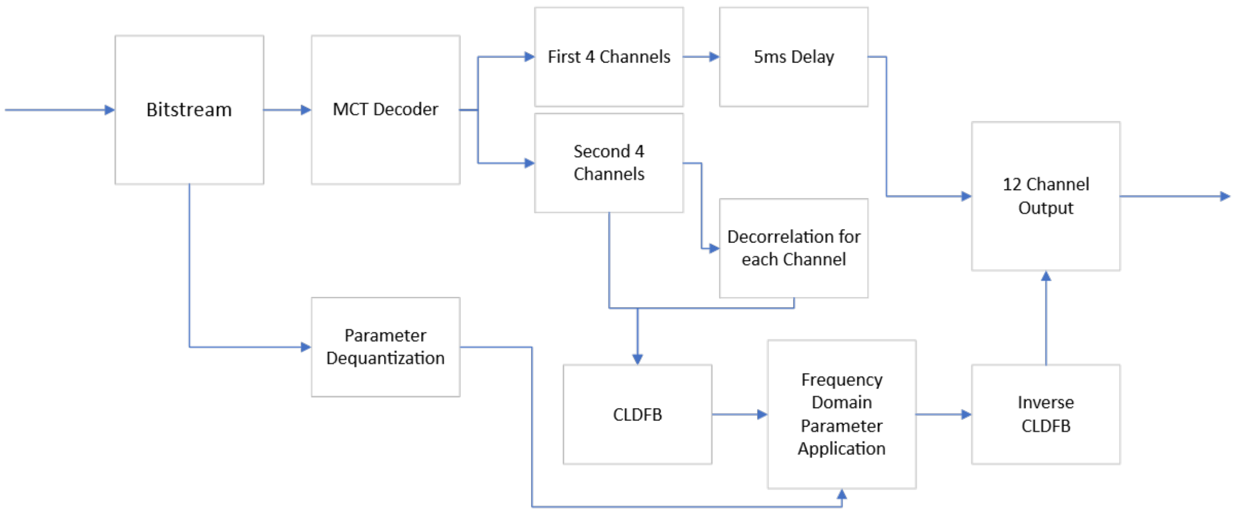


Figure 6.7.4-1: Schematic of parametric upmix decoder

The input to the ParamUpmix portion of the decoder consists of the 7 channels which have been decoded by the MCT (reference) decoder, and the bitstream elements representing the alpha and beta parameters needed for upmixing to the desired 12 channels. The first 4 channels (including LFE, decoded separately (Reference), call D1. The second 4 channels call D2, will be expanded to 8 channels.

6.7.4.2 Special case of Stereo and Mono rendering configurations

If rendering is set to Stereo or Mono then the following parametrix upmix decoding procedure is skipped and D1 and D2 are propagated to the rendering procedure in a 5.1.2 configuration, however, after each D2 channel has a gain of 2.0 applied to match levels when they would have been processed by ParamUpmix.

6.7.4.3 Bitstream decoding

The bitstream elements are decoded as follows.

First a single bit is read which indicates whether the Huffman codes are in delta time or delta frequency configuration (1 = delta time, 0 = delta frequency).

Next 12 alpha and beta quantized indices, 1 per frequency band, are Huffman decoded as in the following pseudocode with inputs bit_stream and current bitstream position (next_bit_pos):

```
If delta_time:
  for ( iv = 0; iv < 12; iv++ )
  {
     parameter[iv] = Huffman_decode(bitstream) + prev_quant_value[iv] – quantization_table_size - 1
  }
If delta_frequency:
  parameter[0] = Huffman_decode(bitstream)
  for ( iv = 1; iv < 12; iv++ )
  {
     parameter[iv] = Huffman_decode(bitstream) + prev_freq_value[iv] – quantization_table_size - 1
  }
```

6.7.4.4 Dequantization

Alpha and Beta quantized parameters (12 of each) are then dequantized. Alpha parameters are dequantized with a simple lookup table using the tables in the Quantization section. Beta parameters are dequantized in a reverse process to the quantization step: alpha quantized parameters are used as a lookup in the beta quantization table index to find the appropriate Beta Quantization table then the quantized beta parameter is used to look up the unquantized values.

6.7.4.5 Decorrelation

The second 4 channels (indices 4-7) of the 8 channels of PCM output from the MCT decoder are separated out and each channel is passed through the Decorrelator block (Reference) to form 8 channels.

The 8 channels are paired into 4 stereo pairs, each original channel with its decorrelated counterpart. The undecorrelated channel will be the "Mid" channel and the decorrelated the "Side" channel.

6.7.4.6 Upmixing operation

Each of the 8 paired channels is transformed using the CLDFB block (Ref) in order to be able to apply alpha and beta parameters to corresponding frequency bands, given below.

6.7.4.6.1 Apply Parameters

The below table is used to map the 60 CLDFB bands to the 12 parameter bands.

Table 6.7-1: Mapping between Parameter Bands and CLDFB Bands

Parameter Band	CLDFB Bands	Parameter Band	CLDFB Bands	Parameter Band	CLDFB Bands
0	0	4	4	8	12-15
1	1	5	5-6	9	16-21
2	2	6	7-8	10	22-31
3	3	7	9-11	11	32-59

The below pseudocode describes the interpolation of alpha and beta parameters across time slots of the CLDFB representation, as well as the determination of the output 4 CLDFB decorrelated channels (side channels) in a linear combination with the interpolated parameters.

```
num_columns = 16
for ( qmf_band = 0; qmf_band < 60; qmf_band ++ )
```

```
   {
      param_band = qmf_to_par_band[qmf_band];
      alpha1 = alpha_previous;
      beta1 = beta_previous;
      col = 0;
      alpha2 = alphas[channel_number];
      beta2 = betas[channel_number];
      alpha_smp = alpha1[param_band];
      beta_smp = beta1[param_band];
      delta_alpha = ( alpha2[param_band] - alpha1[param_band] ) / num_columns;
      delta_beta = ( beta2[param_band] - beta1[param_band] ) / num_columns;
      for ( nc = 0; nc < num_columns; nc++ )
      {
         alpha_smp += delta_alpha;
         beta_smp += delta_beta;
         mid_real = qmf_mod_real[col][qmf_band];
         mid_imag = qmf_mod_imag[col][qmf_band];
         side_real = qmf_side_re[col][qmf_band];
         side_imag = qmf_side_im[col][qmf_band];
         qmf_side_real[col][qmf_band] = alpha_smp * mid_real + beta_smp * side_real;
         qmf_side_imal[col][qmf_band] = alpha_smp * mid_imag + beta_smp * side_imag;
         col++;
      }
      alpha1 = alpha2;
      beta1 = beta2;
   }
The each of the 4 pairs of Mid/Side channels are then converted to Left/Right channels.
for ( i = 0; i < 16; i++ )
{
   for ( k = 0; k < 60; k++ )
   {
      Cldfb_Left_Real[i][k]  = mid_real[i][k] + side_real[i][k];
      Cldfb_Left_Imag[i][k]  = mid_imag[i][k] + side_imag[i][k];
      Cldfb_Right_Real[i][k] = mid_real[i][k] - side_real[i][k];
      Cldfb_Right_Imag[i][k] = mid_imag[i][k] - side_imag[i][k];
   }
}
```

Next each of the 8 paired channels is transformed using the inverse CLDFB block (Ref).

Because of the CLDFB processing of the D2 channels expanded to 8 channels, they are delayed relative to the 4 D1 channels. Therefore, 5ms of delay is applied to the first 4 channels.

The 12 time-domain signals (delayed D1 and D2 expanded to 8 channels) then exit the ParamUpmix block to be rendered.

6.7.5 Discrete MC decoding mode

6.7.6 MC bitrate switching

6.7.7 MC format conversion

6.8 Combined Object-based audio and SBA (OSBA) decoding

6.8.1 Low-bitrate pre-rendering coding mode

In the low-bitrate mode, the OSBA processing is on the decoder side is mostly identical to that of regular SBA. All objects are rendered into the SBA input signal on the encoder side. Therefore, it is sufficient to decode and render the SBA signal in the same way as in SBA mode.

However, the external output format supported additionally. This format consists of the object audio channels plus metadata files and the SBA output channels. In the low bitrate mode, the correct number of output channels and metadata files are produced by the encoder. However, the contents of the objects are part of the SBA output channels.

The object channels are output empty, and the metadata are default values. This output format enables compatibility with the same postprocessing or external rendering as in the high-bitrate mode. It also avoids the need to implement a bitrate dependent post-processing or rendering of the external output.

6.8.2 High-bitrate discrete coding mode

In the high-quality high-bitrate mode, the SBA and ISM metadata decoders run concurrently, each of them separately decoding their metadata. The MCT decodes the jointly coded audio channels of the objects and the SPAR downmix. The SPAR upmix and the DirAC synthesis then run as in regular SBA. The objects are rendered to the selected output configuration. A factor of 0.5 is multiplied to the outputs in order to keep the loudness at approximately the same level as if both formats are encoded and decoded with two separate instances of the codec.

In external output mode, the decoded ISM metadata are written to one output file per object. The audio output files contain the object audio channels and the SBA output channels.

6.8.3 OSBA bitrate switching

6.9 Combined Object-based audio and MASA (OMASA) decoding

6.9.1 Overview

6.9.2 Low-bitrate pre-rendering mode decoding

6.9.3 One object with MASA representation mode decoding

6.9.4 One object with parametric representation mode decoding

6.9.4.1 Overview

6.9.4.2 MASA to total ratios decoding

6.9.4.3 ISM ratios decoding

6.9.5 Discrete mode decoding

6.10 EVS-compatible mono audio decoding

7 Description of the rendering modes and rendering control

7.1 Overview

Editor's note: Describe decoding (as much as needed) and rendering according to rendering mode/type (loudspeaker rendering, binaural rendering, different binaural renderers). Clause 6 describes decoding and specific rendering algorithms for each operation (audio input format), clause 7 gives the overall renderer point of view. In addition, clause 7 provides the rendering control aspects (head-tracking, etc., HRIR sets, etc., room acoustic parameters).

7.2 Rendering modes

7.2.1 Rendering for loudspeaker reproduction

7.2.1.1 Overview

Digital audio decoded by IVAS decoder can be rendered for loudspeaker reproduction. The process of rendering depends on the decoded audio format. In case of multi-channel formats, the decoded format can match the loudspeaker configuration or output channels can be generated by application of multi-channel conversion gains from conversion tables. For SBA, MASA and ISM formats the spatial audio needs to be mapped to the loudspeaker positions of the loudspeaker setup (e.g., 5.1 or 7.1.4). Depending on the decoded format amplitude panning is employed, with either the vector-base amplitude panning (VBAP) scheme (using triangles), described in detail in clause 7.2.1.2, or an improved edge-fading amplitude panning (EFAP) scheme (using polygons), described in detail in clause 7.2.1.3. Specifically for SBA audio, the AllRAD loudspeaker decoding scheme [ref?] is used with EFAP.

7.2.1.2 Vector-base amplitude panning (VBAP)

7.2.1.2.1 VBAP initialization

7.2.1.2.1.1 VBAP initialization overview

The input to the processing is the loudspeaker positions as azimuth $\theta_{LS}(i)$ and elevation $\phi_{LS}(i)$ angles, and the number of loudspeakers N.

First, it is checked if there is a need for virtual loudspeakers, using the method presented in clause 7.2.1.2.1.2. The check is performed for the need of the virtual bottom, top, and back loudspeakers.

Based on the checks, a speaker node set $\theta(i), \phi(i)$ is determined. It contains the input loudspeaker nodes $\theta(i) = \theta_{LS}(i), \phi(i) = \phi_{LS}(i)$. In addition, it contains the virtual loudspeakers, if it was determined that there is a need for them. The locations of the virtual loudspeakers (if they are needed) are: bottom: $\theta = 0, \phi = -90$, top: $\theta = 0, \phi = 90$, and back: $\theta = 180, \phi = 0$. Thus, the resulting speaker node set $\theta(i), \phi(i)$ contains within a 3D space $N_{node} = N + 0 \ldots 3$ speaker nodes depending on the need of the virtual loudspeakers.

Furthermore, for each virtual loudspeaker, the virtual loudspeaker type is determined by the method in clause 7.2.1.2.1.2. The types are: DISTRIBUTE, DISCARD.

Then, the speaker node data is initialized, using the method presented in clause 7.2.1.2.1.3. Furthermore, as an outcome, a group is obtained for each speaker node (HORIZONTAL, BOTTOM_HALF, TOP_HALF) or alternatively all nodes are assigned with group ALL, depending on the loudspeaker node configuration.

Then, a set of non-crossing planes is determined, using the method presented in clause 7.2.1.2.1.4.

Then, the connections between the speaker nodes are determined, using the method presented in clause 7.2.1.2.1.5. In this clause, the determination of the connections depends on the group. If the nodes were determined to be in the ALL group, the speaker nodes are grouped to a single group. If the speaker nodes were determined to not be in the ALL group, the speaker nodes are grouped to bottom half speaker nodes, top half speaker nodes, and horizontal nodes, and the determination of the node connections is performed in these groups.

Next, the rule to distribute virtual speaker node gains to the actual speaker nodes is presented in clause 7.2.1.2.1.8.

Then, the virtual surface triplets are determined based on the determined connections as presented in clause 7.2.1.2.1.9. The outcome of the initialization is one or two search structures, each containing the virtual surface triplets (and their speaker nodes and inverse matrices), and initial search indices for fast searching and panning.

7.2.1.2.1.2 Checking for the need of virtual loudspeakers

The basic operating principle of vector base amplitude panning (VBAP) is to formulate, for any panning azimuth and elevation, three non-negative amplitude panning gains. For a given direction, a base of three unit vectors pointing towards a subset of three loudspeakers is selected, and gains are formulated for these vectors so that the weighted sum of them points towards the panning target. For such non-negative gains to be formulable, there are requirements for the loudspeaker arrangement. For example, if there are loudspeakers only at the front half sphere, the panning gains cannot be determined for rear directions with such methodology. Similarly, if there are only horizontal and elevated loudspeakers, non-negative panning gains cannot be determined for negative elevations.

Therefore, the present VBAP algorithm performs checks to add virtual loudspeakers to the loudspeaker arrangement. In specific, it is checked if any of a virtual bottom, virtual top, or virtual back loudspeaker needs to be added. If any of these virtual loudspeakers are added to the loudspeaker configuration, they are considered at the VBAP triangulation and triplet gain formulation as regular loudspeakers. The triangulation refers to determining a non-overlapping virtual surface arrangement covering all 3D sphere of directions. Each of the virtual surfaces in the arrangement is triangle-shaped and has corners positioned at three different speaker nodes.

When determining the final panning gains for actual loudspeakers only, any positive gains assigned to the virtual loudspeakers will be discarded or distributed to adjacent actual loudspeakers at the setup. The rule to decide discarding or distributing is detailed in clause 7.2.1.2.1.2.

First, it is checked if a virtual bottom and top directions are needed. This is checked by first finding the maximum and minimum elevation value from the existing loudspeaker setup. If the minimum elevation $\phi_{min} \geq -45°$, then a bottom (-90 degrees elevation) virtual loudspeaker is added to the loudspeaker setup. If the maximum elevation $\phi_{max} \leq 45°$, then a top (90 degrees elevation) virtual loudspeaker is added.

Next it is checked if a virtual back direction is needed. It is checked if there is at least one loudspeaker position with $|\phi(i)| < 45°$ and $|\theta(i)| > 91°$, when azimuth values are considered in range -180 … 180 degrees. If such a loudspeaker is not found, then the loudspeaker setup is added with an additional virtual loudspeaker directly behind (180 degrees azimuth, 0 degrees elevation).

Since the order of the added virtual loudspeakers matters in clause 7.2.1.2.1.8, the order of adding them is defined here. If any of the virtual loudspeakers are added, they are added in order: bottom, top and back. They are added to the end of the loudspeaker setup arrangement. In other words, the indices of the actual loudspeakers do not change.

Furthermore, in case a virtual loudspeaker has been added at bottom and/or at top, then a further check is performed: For the top, if $\phi_{max} \geq 20°$ (and correspondingly for the bottom $\phi_{min} \leq -20°$), then a flag DISTRIBUTE is set for that virtual loudspeaker. Otherwise, a flag DISCARD is set for that virtual loudspeaker. In case a back virtual loudspeaker has been added, then a flag DISTRIBUTE is set for that virtual loudspeaker.

The added virtual bottom and/or top and/or rear loudspeakers are in the following considered as actual loudspeaker nodes as if there were loudspeaker(s) in those positions, unless specifically mentioned otherwise.

7.2.1.2.1.3 Speaker node data initialization

The loudspeaker positions (including any virtual loudspeakers) are referred to as “speaker nodes” or simply “nodes”. The nodes are defined by their azimuth and elevation. In this clause, pre-processing operations to the node data is described. First, any nodes that have the elevation at range $-5° \leq \phi(i) \leq 5°$ is set to have $\phi(i) = 0°$

Next, the largest azimuth angle gap of the nodes with $\phi(i) = 0°$ is determined. This is formulated by finding the azimuth angle difference of each of the nodes at $\phi(i) = 0°$ to the next node at $\phi(i) = 0°$, and finding the largest of these differences. The angle difference is formulated for the shortest path, for example, nodes with azimuths $179°$ and $-179°$ would have a $2°$ azimuth angle difference.

If the largest gap for the horizontal nodes is 170 degrees or less, then the following steps are applied: Each horizontal node is associated with a label HORIZONTAL; Each node with positive elevation is associated with a label TOP_HALF; Each node with negative elevation is associated with a label BOTTOM_HALF. In other words, when the condition is met, the speaker node set is divided into top and bottom parts by a defined virtual plane that is the horizontal plane in the 3D space, and an additional part having nodes at the horizontal plane.

If the above condition is not met, all nodes are associated with a label ALL.

7.2.1.2.1.4 Determining non-crossing planes

For the given set of loudspeaker nodes, a set of non-crossing planes are defined. All unique elevations $\phi(i)$ of the loudspeaker nodes that do not have the label HORIZONTAL are checked. For each of these elevations, the following steps are performed: Find the largest gap of the nodes having the same elevation value (the operation of finding the largest gap is the same as in clause 7.2.1.2.1.3 for horizontal nodes); If the largest gap is equal or smaller than 140 degrees, then the corresponding elevation ϕ is set as a non-crossing plane. The resulting set of non-crossing planes are denoted as $\phi_{nc}(p)$ where p is the index of the non-crossing plane. The number of non-crossing planes can be zero or higher, depending on the loudspeaker arrangement. The non-crossing plane information is used in clause 7.2.1.2.1.7, and its purpose is to prefer triangulation where the triangle edges are along these non-crossing planes, instead of crossing them.

7.2.1.2.1.5 Determining speaker node connections

The term connection refers to a pair of nodes (a, b) where a and b can refer to any node index i for $a \neq b$. When a set of connections is determined, the resulting set are the VBAP triangulation edges. In other words, determining a suitable set of connections therefore determines a non-overlapping arrangement of triangle-shaped virtual surfaces encompassing the 3D directions. These connections are the virtual surface sides or edges, and the nodes are the virtual surface corners.

There are two ways the VBAP can determine the set of connections for the given loudspeaker node arrangements (where some of the nodes may be virtual).

The first initialization mode is set if the speaker nodes were associated with label ALL. In this mode, all loudspeaker nodes are considered in one group. The speaker node connections are determined using the 3D method as described in clause 7.2.1.2.1.7.

Otherwise, the second initialization mode is set, where the speaker node connections are determined in the following steps: First, top half connections are determined using the 3D method as described in clause 7.2.1.2.1.7; Next, horizontal connections are determined using a horizontal method as described in clause 7.2.1.2.1.6; Next, the bottom half connections are determined using the 3D method as described in clause 7.2.1.2.1.7. After determining all connections (which are the virtual surface edges) for these three parts (top, horizontal, bottom), the resulting set of connections are combined as one unified set. These connections then are the edges of the triangle-shaped virtual surfaces within the non-overlapping virtual surface arrangement encompassing the 3D directions. When the connections are defined in three sets (top, horizontal, bottom), the resulting triangulation thus is such that the edges of the triplet surfaces do not intersect the horizontal plane (and thus also the triplet surfaces themselves do not intersect the horizontal plane.

7.2.1.2.1.6 Determining speaker node connections for horizontal node groups

The horizontal connections are determined by finding for each node a that has label HORIZONTAL the next node b that has a label HORIZONTAL. The “next node” here refers to the finding the node b (that has a label HORIZONTAL) which is most adjacent to the node a in positive azimuth direction. The adjacency accounts for any wrapping of the angle, so that if the wrapping point is at $\theta = 180°$, then for example for a node at $\theta(a) = 170°$, a further node at $-170°$ may be considered adjacent (unless there is further node in between) and $20°$ degrees at the positive azimuth direction from $170°$.

The result of this operation is a set of as many virtual connections (a, b) at the horizontal plane as there are nodes with label HORIZONTAL.

7.2.1.2.1.7 Determining speaker node connections for 3D node groups

In this clause, it is defined how to define a set of connections for a node group which includes at least one non-horizontal node. The operations in this clause can be called in different ways: for entire node set, for top node set, or for bottom node set.

First, if the processing is for the top node set, all nodes having a label BOTTOM_HALF are discarded from the following operations. And, if the processing is for the bottom node set, all nodes having a label TOP_HALF are discarded from the following operations.

Next, all the pairs (a, b) where $a \neq b$ in the remaining nodes are defined as potential connections.

Next, all connections where both nodes have the label HORIZONTAL are discarded.

Next is to check if any of the connections passes the origin closely. Denoting $\mathbf{u}_a$ and $\mathbf{u}_b$ as column unit vector pointing towards nodes a and b, this is checked by finding if $|\mathbf{u}_a^T\mathbf{u}_b + 1| < 0.001$. If this condition is met, the connection is discarded. The unit vectors are the speaker node positions in 3D space where the node distance from origin is assumed to be 1 unit.

Next is to check if there is a node behind any of the connections. This is checked for all remaining connections (a, b) by assuming a line between $\mathbf{u}_a$ and $\mathbf{u}_b$, and then checking all unit vectors $\mathbf{u}_c$, where c is a further node index so that $c \neq a \neq b$, as follows: For each $\mathbf{u}_c$, a unit vector $\mathbf{u}_{ab}$ is defined that, with a condition that it intersects the line between $\mathbf{u}_a$ and $\mathbf{u}_b$, the spatial angle $\cos^{-1}(\mathbf{u}_{ab}\mathbf{u}_c)$ is minimized. If this angle is less than 1 degree, the connection between a and b is discarded.

Furthermore, for all remaining connections (a, b) it is checked if $|\mathbf{u}_a - \mathbf{u}_b| > 5|\mathbf{u}_c - \mathbf{u}_{ab}|$ for any $c \neq a \neq b$. If this condition is met, the connection is assigned with a label THIN_TRIANGLE_CONNECTION.

For the connections (a, b) that have not been discarded due to the above steps, an arc value is determined by $\alpha_{ab} = \cos^{-1}(\mathbf{u}_a^T\mathbf{u}_b)$. Furthermore, a weighted arc $\hat{\alpha}_{ab}$ is determined that is $\hat{\alpha}_{ab} = \alpha_{ab,mod1}\alpha_{ab,mod2}\alpha_{ab}$, where α_{mod1} is 1, except if the connection has been assigned with the label THIN_TRIANGLE_CONNECTION, and in that case $\alpha_{ab,mod1} = 2$. The second modifier $\alpha_{ab,mod2}$ is 1, except if the elevation crosses a non-crossing plane (defined in clause 7.2.1.2.1.4), i.e., for $\phi(a)$ and $\phi(b)$ one is at least 1 degree larger than $\phi_{nc}(p)$ and the second one is at least 1 degree smaller than $\phi_{nc}(p)$, for any p. If this condition is met, then $\alpha_{ab,mod2} = 2$.

Next, the connections (a, b) are sorted with respect to their $\hat{\alpha}_{ab}$, from smallest to largest.

Then, an empty set of valid connections is defined. From first to last, all sorted connections (a, b) are processed with the following steps.

For each connection (a, b), it is checked that if within the currently existing set of valid connections there are any connection that meets each of these criteria: The existing valid connection has nodes that are different from both a and b; From the viewpoint of the origin, a line between node pair (a, b) crosses a line between the nodes of any of the connections at the set of valid connections. If these conditions are met, the new connection (a, b) is not valid and is discarded. Otherwise, the new connection (a, b) is valid and is added to the set of valid connections.

The above crossing in of two different connections (a_1, b_1) and (a_2, b_2) for $a_1 \neq b_1 \neq a_2 \neq b_2$ may be formulated by defining two planes: First plane is defined by origin and points defined by the unit vectors $\mathbf{u}_{a_1}$ and $\mathbf{u}_{b_1}$, and second plane is defined by origin and points defined by the unit vectors $\mathbf{u}_{a_2}$ and $\mathbf{u}_{b_2}$. The intersection of these planes is formulated, and the result is a line in 3D space. If this line passes between $\mathbf{u}_{a_1}$ and $\mathbf{u}_{b_1}$, then a crossing is registered. If the planes are the same, then the intersection is not a line but a plane, and in this case the crossing is not registered.

When all sorted connections (a, b) are exhausted as in the foregoing, the resulting set of valid connections is the determined set of node connections in the 3D method.

7.2.1.2.1.8 Determining virtual loudspeaker distribution gains

For any virtual loudspeaker node (back, bottom, top) added to the loudspeaker setup, a set of division gains are defined. This means, for any of the back, bottom, top virtual loudspeakers a gain distribution column vector $\mathbf{g}_{\text{dist,back}}$, $\mathbf{g}_{\text{dist,bottom}}$, $\mathbf{g}_{\text{dist,top}}$ is defined. The vector has as many elements as there are (non-virtual) loudspeakers in the loudspeaker arrangement. For any of the virtual loudspeakers where a flag DISCARD is set, the corresponding gain distribution vector $\mathbf{g}_{\text{dist,x}}$, where x is back, bottom or top, is set as zeros. For any of the virtual loudspeakers where a flag DISTRIBUTE is set, the following determination of gains is performed.

For any of the virtual loudspeaker nodes, the connection set determined in clause 7.2.1.2.1.5 is searched through for any connections for any connections (a, b) where either a or b is the virtual loudspeaker node in question. With this method, all loudspeaker nodes i can be found that are connected to the virtual loudspeaker node in question. Then, the column gain vector $\mathbf{g}_{\mathrm{dist,x}}$ values are set so that the row value that corresponds to any of the indices i that is connected to the virtual loudspeaker node are set as $1/\mathrm{N}_{dist,x}$ where $\mathrm{N}_{dist,x}$ is the number of nodes connected to the virtual loudspeaker node in question, and other values of $\mathbf{g}_{\mathrm{dist,x}}$ are zero.

An exception to the above is if the IVAS format is MASA_ISM_FORMAT, and then the non-zero gains are set as $\left(\frac{1}{\mathrm{N}_{dist,x}}\right)^{0.8}$.

Furthermore, a converter matrix $\mathbf{C}$ is defined that maps any gains determined for a loudspeaker arrangement with potential added virtual loudspeakers, to a set of gains where the gains are defined for only the real loudspeakers. The matrix $\mathbf{C}$ is defined as follows: First, an N times N identity matrix is defined, and then it is appended with columns at the right edge if there are virtual loudspeakers added. Notice from clause 7.2.1.2.1.2 that any virtual loudspeakers were added in order (bottom, top, back), and therefore, if any of these were added, the matrix $\mathbf{C}$ is appended with columns $\mathbf{g}_{\mathrm{dist,bottom}}$, $\mathbf{g}_{\mathrm{dist,top}}$, $\mathbf{g}_{\mathrm{dist,back}}$, in this order, if any one or more them needs to be added.

7.2.1.2.1.9 Determining virtual surface triplets

The virtual surface triplets are determined based on the connections determined in clause 7.2.1.2.1.5. The connections are the triplet edges, and the triplets are found by finding all sets of three connections which share a set of three loudspeaker nodes. When the nodes are labelled ALL, then there is one set of virtual surface triplets determined that is the entire virtual surface arrangement. Otherwise, two sets of virtual surface triplets are defined: One has all the virtual surfaces where the nodes have labels TOP_HALF and HORIZONTAL. Another has all the virtual surfaces where the nodes have labels BOTTOM_HALF and HORIZONTAL. In other words, in that case there are two sets, divided by the horizontal plane.

Each triplet includes three (real or virtual) nodes (a, b, c) where $a \neq b \neq c$, and the set of (a, b, c) is different for each triplet. For each triplet an associated 3x3 inverse matrix is defined by the unit 3x1 vectors

$$\mathbf{R}^{-1} = [\mathbf{u}_a \quad \mathbf{u}_b \quad \mathbf{u}_c]^{-1}$$

The purpose of such a matrix is to find a 3x1 panning gain vector $\mathbf{g}_{\mathrm{tmp}}$ that satisfies the condition that $\mathbf{u}_{\mathrm{target}} = \mathbf{R}\mathbf{g}_{\mathrm{tmp}}$ where $\mathbf{u}_{\mathrm{target}}$ is a unit vector pointing towards the desired panning direction. Notice that $\mathbf{g}_{\mathrm{tmp}}$ is a temporary triplet gain vector, not the final gain vector.

The following operations in this clause are performed independently for each of the determined virtual surface sets (which can be one set, or bottom and top halves separately).

The virtual surface triplets are arranged to an ordered virtual surface set based on the triplet azimuth angles $\theta_{triplet}(i_{tr})$ (i_{tr} is the reference index of the triplet) in an ascending order, where these triplet azimuth angles are determined by summing together the three unit vectors contributing to the triplet (i.e., $\mathbf{u}_{abc} = \mathbf{u}_a + \mathbf{u}_b + \mathbf{u}_c$), where (a, b, c) refer to the node indices within the triplet and are thus different for each triplet, and finding the azimuth angle by $\theta_{triplet}(i_{tr}) = atan2(u_{abc,y}, u_{abc,x})$, where $u_{abc,y/x}$ refer to the horizontal y/x entries of $\mathbf{u}_{abc}$.

Then, initial search indices are determined based on the sorted triplet azimuth angles $\theta_{triplet,sort}(i_{tr})$ (that correspond to the sorted triplets). Four search sectors are determined, where each sector occupies a range of azimuth angles defined by start and end azimuth angles (in degrees), which are determined for each sector $j = 0 \ldots 3$ by

$$\alpha_{start}(j) = 90j$$

$$\alpha_{end}(j) = 90(j + 1)$$

Then, the sector reference azimuth angles are determined by

$$\alpha_{ref}(j) = \frac{\alpha_{start}(j) + \alpha_{end}(j)}{2}$$

Then, for each of these sectors, one of the virtual surface triplets of the ordered set is associated to the search sector, by selecting the triplet having the azimuth angle $\theta_{triplet,sort}(i_{tr})$ the closest to the sector reference azimuth angle $\alpha_{ref}(j)$. This is set as the initial triplet for the search sector, i.e., $\xi(j) = i_{tr}$.

7.2.1.2.2 VBAP gain determination

7.2.1.2.2.1 VBAP gain determination overview

The input to the processing is a target panning direction comprising a target azimuth angle θ and target elevation angle ϕ.

First, the correct search structure is determined. If there are two search structures and if the elevation angle is larger than zero (i.e., $\phi > 0$), the second search structure is used (containing virtual surfaces associated with TOP_HALF and HORIZONTAL nodes). Otherwise, the first search structure is used (containing virtual surfaces associated with BOTTOM_HALF and HORIZONTAL, or ALL, nodes).

Then, the best virtual surface triplet from the determined virtual surface arrangement is selected and the corresponding panning gains are generated based on it using the method described in clause 7.2.1.2.2.2. In other words, triplet gains $\mathbf{g}_{\text{tmp}}$ and the triplet to be used with respect to the present azimuth θ and elevation ϕ are formulated. The triplet node indices (a, b, c) for which the triplet gains $\mathbf{g}_{\text{tmp}}$ are associated with are thus also determined.

The next step is to normalize the gains by

$$\mathbf{g}_{\text{norm}} = \begin{bmatrix} g_a \\ g_b \\ g_c \end{bmatrix} = \frac{\mathbf{g}_{\text{tmp}}}{\sqrt{\mathbf{g}_{\text{tmp}}^{\text{T}}\mathbf{g}_{\text{tmp}}}}$$

Note that in the above, any of the indices (a, b, c) may correspond to real or virtual loudspeaker node. These normalized triplet gains $\mathbf{g}_{\text{norm}}$ are converted to a panning gain vector $\mathbf{g}_{\text{all}}$ that is a column vector that has N elements, i.e., as many elements as there are actual loudspeakers. This is achieved by first determining a $\mathbf{g}_{\text{all,tmp}}$ which is a column vector with as many entries as there are in total real or virtual nodes, where each a:th, b:th and c:th entry is set to respectively have values g_a, g_b, g_c, and other values are zero. Next the converter matrix $\mathbf{C}$ defined in clause 7.2.1.2.1.8 is used to convert the gains to the final amplitude panning gains for the given loudspeaker setup by

$$\mathbf{g}_{\text{all}} = \mathbf{C}\mathbf{g}_{\text{all,tmp}}$$

These are the panning gains output by the VBAP algorithm. They are used for positioning sound in 3D space. For example, with the OMASA format, these gains are applied to an audio object signal, and the gained audio object signals are forwarded to corresponding loudspeaker channels, as a result causing that object audio signal to be positioned in the desired direction within the 3D space.

7.2.1.2.2.2 Determining the best virtual surface triplet and the panning gains

First, the correct search sector from the four search sectors is determined based on the target azimuth angle θ by

$$if\ 0 \le \theta \le 90 \rightarrow j = 0$$

$$if\ 90 < \theta \le 180 \rightarrow j = 1$$

$$if - 180 < \theta < -90 \rightarrow j = 2$$

$$if - 90 \le \theta < 0 \rightarrow j = 3$$

Then, the first virtual surface triplet is determined by $\xi(j)$, where $\xi(j)$ is the initial triplet for sector j determined in clause 7.2.1.2.1.9.

The next step is to iteratively find the correct virtual surface triplet (i.e., triangle). Starting from the virtual surface triplet associated with the correct search sector, the ordered virtual surface set is searched to determine a virtual surface that encloses the target panning direction. This is done by the following steps:

1. Set the initial search index as the current triplet index $i_{tr} = \xi(j)$.
2. Set triplet counter $m = 0$.
3. Check if the current triplet is the correct triplet (see details below).
4. If not, increase the counter m by one, and calculate new current triplet index, according to the equation $i_{tr} = \text{mod}\left(\xi(j) + \text{floor}\left(\frac{m+1}{2}\right)(-1)^m, N\right)$, where N is the number of triplets in the ordered virtual surface set,

mod is a modulo function, and floor a flooring function. In practice, the function results in a pattern of $(i_{tr}, i_{tr}-1, i_{tr}+1, i_{tr}-2, i_{tr}+2, \ldots)$ until all triplets have been checked.

5. If yes, then the current triplet is the correct one, and the triplet index i_{tr} is output (alongside the triplet gains $\mathbf{g}_{\mathrm{tmp}}$).

In step 3, the checking if the current triplet is the correct triplet is performed as follows. As shown in clause 7.2.1.2.1.9, each triplet is associated with a preformulated inverse matrix $\mathbf{R}^{-1}$ that is unique to each triplet. Triplet panning gains are obtained by $\mathbf{g}_{\mathrm{tmp}} = \mathbf{R}^{-1}\mathbf{u}_{\mathrm{target}}$. If all gains are non-negative, the triplet is determined to be a correct triplet (i.e., the virtual surface triplet that encloses the target panning direction), and otherwise not. When the triplet is deemed correct, the formulated $\mathbf{g}_{\mathrm{tmp}}$ are the triplet panning gains that are further processed in clause 7.2.1.2.2.1 to obtain the final gains for the loudspeakers.

7.2.1.3 Edge Fading Amplitude Panning (EFAP)

The Edge Fading Amplitude Panning (EFAP) algorithm is used to compute panning gains to position a sound source at a requested point on a given set of loudspeaker nodes.

EFAP is used to determine panning gains which are used in the following rendering paths of the decoder:

- Parametric ISM rendering to loudspeaker or ambisonics (7_1_4 used as an intermediate layout for loudspeaker) output
- Multichannel rendering to a user-specified arbitrary loudspeaker layout
- Rotating a multichannel scene on the same loudspeaker layout for combined head and/or orientation tracking for a subsequent binaural output
- Rendering of virtual loudspeaker positions to a real loudspeaker layout in the ALLRAD (All-round ambisonic decoding) rendering of ambisonics to loudspeaker output

Panning with EFAP consists of two steps; first determining the panning gains for the input audio channels and then applying these gains via multiplication to the input audio samples to produce the output audio samples. This operation can also be performed iteratively on all input channels as a matrix multiplication where the matrix is populated using EFAP for each of the individual input channels.

The output audio channels are each associated with a loudspeaker position which can either be one of the supported output loudspeaker configurations in IVAS (REFERENCE NEEDED) or a user-specified custom loudspeaker layout.

The input audio channels are each associated with a panning position on the spherical surface specified by the azimuth and elevation coordinates. A unit radius is implicit in this setup.

The process of gain computation with EFAP consists of two phases.

The first phase is an initialization phase, executed ideally prior to runtime in which a 3D convex hull is constructed from the set of output loudspeaker positions.

The second phase can be executed after the initialization of the module. In this phase the convex hull structure computed at initialization is used to find a subset of the output loudspeaker positions according to the panning position. This subset is essentially a two-dimensional polygon on the spherical surface and encloses the panning position or is coincident or colinear with it on one of its vertices or edges.

Editor's Note: A block diagram is expoected to inserted that illustrates the two steps required to obtain the output audio channels from the input audio channels. As mentioned above, the panning gains may be a vector for one input channel, or a matrix for multiple input channels.

Since the spherical surface is mapped to a two-dimensional plane, comparison of the coordinates of the panning position and loudspeakers, along with computation of Barycentric coordinates can be used to find the facet of the convex hull which contains the panning position.

Once the correct speaker subset has been determined, the gains for each individual loudspeaker in the subset are derived by:

- Sub-dividing the subset polygon into triangles that contain the relevant loudspeaker as a vertex

- Determining whether the panning position lies inside this triangle by performing a check on the Barycentric coordinates
- Finally, the crossfading gain for the relevant loudspeaker is computed by using the normal vector of the triangle

The check for whether the panning position lies inside a triangle can be performed by computing the coefficients λ and μ:

$$[\lambda, \mu]^T = [b - a, c - a]^{-1}(p - a), \tag{7.2-1}$$

where a, b and c are the coordinate pairs (azimuth, elevation) of the loudspeakers in the given triangle and p is the coordinate pair for the panning position.

If the following conditions are satisfied, the panning position p is located inside the triangle:

$$\lambda \geq 0 \tag{7.2-2}$$

$$\mu \geq 0 \tag{7.2-3}$$

$$\lambda + \mu \leq 1\,. \tag{7.2-4}$$

The crossfading gain g for a loudspeaker in a triangle can be determined as follows:

$$\mathrm{g} = 1 - \mathrm{n}(\mathrm{p} - \mathrm{a}) \tag{7.2-5}$$

where n is the normal vector of the given triangle, a is the coordinate pair of the relevant loudspeaker and p is coordinate pair of the panning position.

Once the crossfading gains for all the loudspeakers have been computed, a Euclidean norm is performed to obtain the final amplitude panning gains.

EFAP also supports an intensity panning mode which is used for ALLRAD (REFERENCE NEEDED). In this case the square root of the sum-normalized amplitude panning gains is used.

Since the panning position is time-dependent, these gains are computed and applied every frame.

7.2.2 Rendering for binaural headphone reproduction

7.2.2.1 Time Domain binaural renderer

7.2.2.1.1 General

The time domain (TD) renderer operates on signals in time domain. In the IVAS decoder it is used for binaural rendering of discrete ISM, where each audio signal is encoded and decoded with a dedicated SCE module. This covers all ISM bit rates, except 3-4 objects for bit rates 24.4 kbps and 32 kbps. Further it is used in the decoder for binaural rendering of 5.1 and 7.1 signals when headtracking is enabled. In the external renderer it is used for all ISM configurations and all multichannel loudspeaker configurations, both with and without headtracking enabled. An overview of the TD binaural renderer is found in Figure 7.2-1 below. An HRIR model accepts the object position metadata along with the headtracking data and generates an HRIR filter pair. The ITD may be modelled as a part of the HRIR, or it may be modelled as a separate parameter. In case an ITD parameter is output, the ITD is synthesis is performed in the ITD synthesis stage. The time aligned signals are then convolved with the HRIR filter pair to form a binauralized signal.

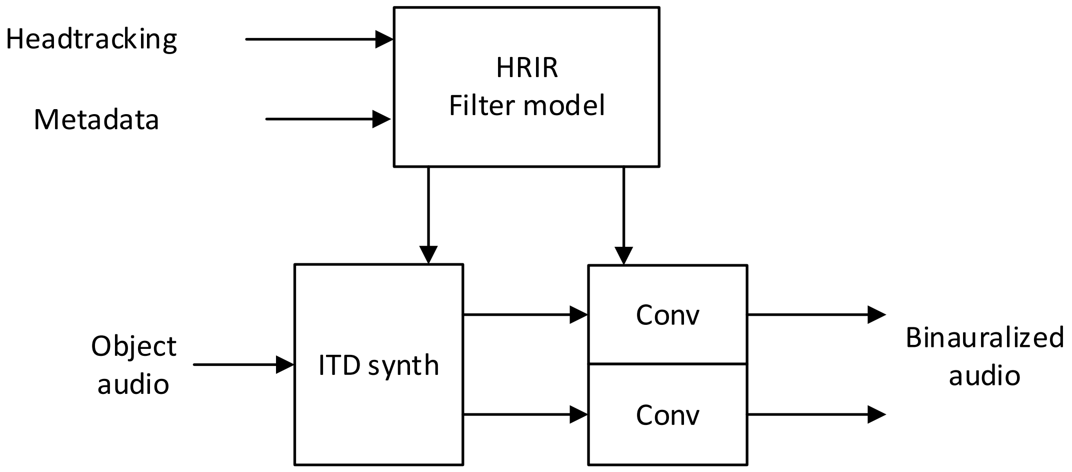


Figure 7.2-1: Overview of TD binaural renderer

7.2.2.1.2 HRIR model

The HR filters for a certain rendering position, specified by an elevation angle ϑ and an azimuth angle φ, are generated based on model parameters representing the HR filter set. The ITD may be modelled as part of the HRIR model or modelled separately, describing the time difference between the left and right HR filters generated from the HRIR model.

The HRIR of the left and right channels are represented as

$$\hat{\boldsymbol{h}}^{l,r}(\vartheta,\varphi)=\sum_{p=1}^{P}\sum_{q=1}^{Q_p}\sum_{k=1}^{K}\alpha_{p,q,k}^{l,r}\Theta_p(\vartheta)\Phi_{p,q}(\varphi)\boldsymbol{e}_k, \qquad (7.2\text{-}6)$$

where

$\alpha_{p,q,k}^{l}$ for $p=1,\dots,P,\ q=1,\dots,Q_p$, and $k=1,\dots,K$ is a set of left filter model scalar parameters,
$\alpha_{p,q,k}^{r}$ for $p=1,\dots,P,\ q=1,\dots,Q_p$, and $k=1,\dots,K$ is a set of right filter model scalar parameters,
$\Theta_p(\vartheta)$ for $p=1,\dots,P$ defines the set of elevation basis function values at the elevation angle ϑ, and
$\Phi_{p,q}(\varphi)$ for $p=1,\dots,P$ and $q=1,\dots,Q_p$ defines P sets of Q_p azimuth basis function values at the azimuth angle φ; and $\boldsymbol{e}_k$ for $k=1,\dots,K$ is a set of canonical orthonormal basis vectors of length N.

In matrix form the HRIR representation can be written as

$$\hat{h}_k^{l,r}(\vartheta,\varphi)=\underbrace{\begin{pmatrix}\boldsymbol{b}(\vartheta_1,\varphi_1)\\ \vdots\\ \boldsymbol{b}(\vartheta_M,\varphi_M)\end{pmatrix}}_{\boldsymbol{B}}\boldsymbol{\alpha}_k^{l,r}=\boldsymbol{B}\boldsymbol{\alpha}_k^{l,r} \qquad (7.2\text{-}7)$$

where

$$\begin{aligned}\boldsymbol{b}(\vartheta_m,\varphi_m) &= \left(\Theta_p(\vartheta_m)\Phi_{p,q}(\varphi_m)\colon p=1,\dots,P; q=1,\dots,Q_p\right)_{row\ vector}\\ \boldsymbol{\alpha}_k^{l,r} &= \left(\alpha_{p,q,k}^{l,r}:p=1,\dots,P; q=1,\dots,Q_p\right)_{column\ vector}\end{aligned} \qquad (7.2\text{-}8)$$

The length of the represented filters K may typically be shorter than the length N of HR filters being modelled, i.e. $K<N$. The basis vectors $\boldsymbol{e}_k$ are chosen as the canonical orthonormal basis vectors of length N

$$\boldsymbol{e}_1=\begin{pmatrix}1\\0\\0\\ \vdots\\0\end{pmatrix}, \quad \boldsymbol{e}_2=\begin{pmatrix}0\\1\\0\\ \vdots\\0\end{pmatrix}, \quad \dots\ \boldsymbol{e}_N=\begin{pmatrix}0\\ \vdots\\0\\0\\1\end{pmatrix}. \qquad (7.2\text{-}9)$$

However, for the IVAS default HR filter set, see clause [xx], only the K first basis vectors are used to represent the HR filter set. The elevation basis functions $\Theta_p(\vartheta)$ and the elevation dependent azimuth basis functions $\Phi_{p,q}(\varphi)$ are cubic B-spline functions of order $J = 4$. The elevation basis functions are standard B-spline functions while the azimuth basis functions are periodic (circular) with a period of 360 degrees.

For the elevation, the univariate B-spline basis functions of order J over the variable ϑ, where ϑ is in the interval $\theta_A \leq \vartheta \leq \theta_B$, is a set of piecewise polynomial functions of degree $J-1$ defined over that interval. The ranges over which the functions are polynomials are specified with the so-called knot sequence $\boldsymbol{\theta} = (\theta_1, \dots, \theta_U)$, where the sub-intervals over which the functions are polynomials are $\theta_u \leq \vartheta \leq \theta_{u+1}$ for $u = 1, \dots, U-1$. In each sub-interval the basis function is a polynomial function of degree $J-1$, i.e.

$$\Theta_p(\vartheta) = \sum_{j=0}^{J-1} \gamma_{j,u,p}^{\Theta} \vartheta^j, \qquad \text{for} \quad \theta_u \leq \vartheta < \theta_{u+1}. \tag{7.2-10}$$

The smoothness at the knot-points of a function that is a linear sum of the B-spline basis functions is controlled by the so-called multiplicity sequence $\boldsymbol{m} = (m_1, \dots, m_U)$, which is a sequence of integers greater than 0 where the value $m_u = i$ means that the $(J-i)$-th derivative at knot-point θ_u is continuous. This means that $i = 1$ gives maximum smoothness, while $i = J$ only gives 0-th derivative continuity. Given the knot sequence and the multiplicity sequence the polynomial model coefficients $\boldsymbol{\gamma}^{\Theta} = \{\gamma_{j,u,p}^{\Theta}: j = 0, \dots, J; u = 1, \dots, U-1; p = 1, \dots, P\}$ may be obtained iteratively starting with the 0-th degree polynomials using recursion, as described by [12].

Similarly, the azimuth B-spline basis functions of order J over the variable φ are represented as

$$\Phi_q(\varphi) = \sum_{j=0}^{J-1} \gamma_{j,l,q}^{\Phi} \varphi^j, \phi_l \leq \varphi < \phi_{l+1} \tag{7.2-11}$$

for the knot sequence $\{\phi_l: l = 1, \dots, L\}$.

The azimuth angles are periodic (e.g., circular) in the meaning that the azimuth angle of φ degrees is the same point in space as the azimuth angle of $\varphi + \kappa * 360$ degrees for any integer valued κ. To obtain efficient modelling in the azimuth dimension it is important to use basis functions that are periodic in the same way, such that $f(\varphi) = f(\varphi + \kappa * 360)$. An example of such periodic azimuth basis function is shown in figure 7.2-6.

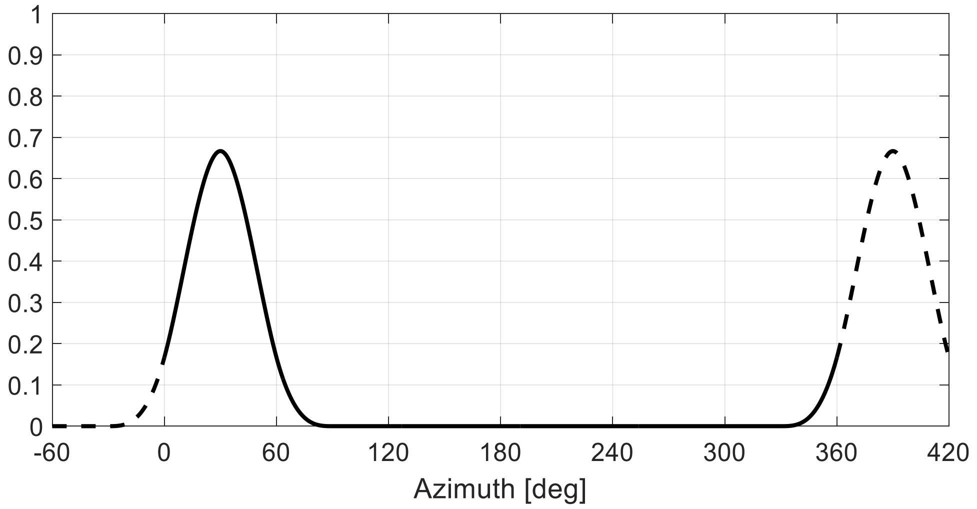


Figure 7.2-2: Example of periodic azimuth basis function.

Figure 7.2-3 and figure 7.2-4 show example standard and periodic B-spline functions used for elevation and azimuth respectively.

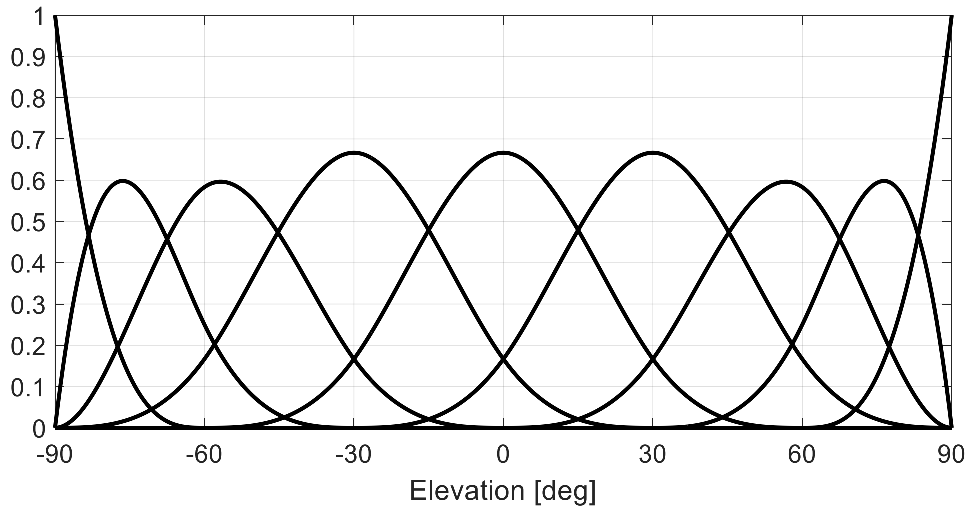


Figure 7.2-3: Example of standard B-spline basis functions for elevation.

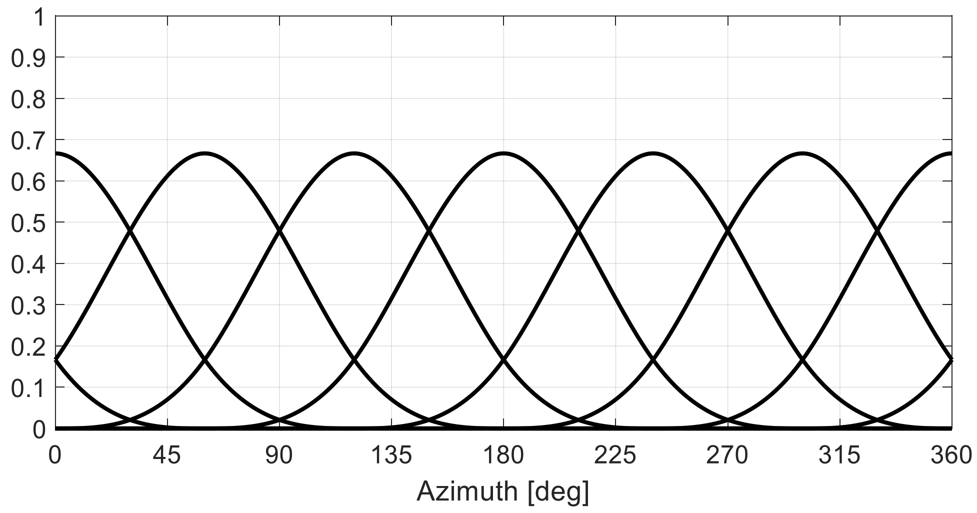


Figure 7.2-4: Example of periodic B-spline functions for azimuth.

In order to efficiently generate HR filters from the model representation, compact representations of the elevation and azimuth basis functions are utilized. The compact representations typically correspond to certain non-zero parts of sampled versions of the elevation and azimuth and basis functions. Shape metadata and basis function shape data, which identifies the compact representations or converted versions of the basis functions, is obtained, either from the ROM tables for the default HR filter set, or from the HR filter binary representation, see [HR data binary file]. The shape metadata indicates whether to obtain a converted version of compact representations of the basis functions for the filter generation, and consists of

- The number of basis functions (the number of the azimuth basis functions may be different for different elevations);
- Starting point of each basis function (within the modelling interval);
- Shape indices per basis function (identifying which of the stored shapes to use for the basis function);
- A shape resampling factor per basis function; and
- A flipping indicator per basis function (indicating whether or not to flip the compact representation (stored shape data) for that specific basis function).

The first stage of the filter generation is to obtain the basis function shape data and evaluate the value of the at most four active elevation basis functions for the requested elevation angle ϑ. The non-zero elevation basis function values are obtained according to clause 7.2.2.1.3, with the shape metadata and basis function shape data (e.g. $Kseq$, $N_{BF}, L_{BS}, B_{sh}, B_{start}$) of the elevation basis functions. For each of the non-zero elevation basis functions, an azimuth angle within the modelled azimuth range is obtained as

$$\varphi = \text{mod}(\varphi + Kseq[i_p, 0], 360) \tag{7.2-12}$$

If $\varphi < 0$, $\varphi := \varphi + 360$ and then

$$\varphi := \varphi - Kseq[i_p, 0] \tag{7.2-13}$$

If there is a constant azimuth basis function for a certain elevation, e.g. at the poles, the number of azimuth basis functions is $N_q[p] = 1$, the index for the azimuth basis function $I_q[p] = 0$, and $B_f = \{1,0,0,0\}$. Otherwise, the values of the non-zero azimuth basis function values are obtained from the basis function shape data for the requested elevation angle φ in accordance with clause 7.2.2.1.4, with the elevation-dependent shape metadata and basis function shape data (e.g. $Kseq$, $N_{BF}, L_{BS}, B_{sh}, B_{start}$) of the azimuth basis functions.

Once the sample values of the non-zero elevation and azimuth basis functions are determined, the non-zero elements of the matrix $\boldsymbol{B}$, see equation (7.2-7), are computed as

$$B[p,q] = B_f^{elev}[p]\, B_f^{azim}[q]\ \forall\, p \in \{1, \ldots, N_{idx}^{elev}\}, q \in \{1, \ldots, N_{idx}^{azim}\} \tag{7.2-14}$$

where B_f^{elev} and N_{idx}^{elev} is the elevation basis functions shape values and number of non-zero basis functions, and B_f^{azim} and N_{idx}^{azim} is the azimuth basis functions shape values and number of non-zero basis functions. $\boldsymbol{B}$ only depends on the requested elevation and azimuth angle and is equal for the left and right HR filters. The matrix $\boldsymbol{B}$ may be represented by a vector, keeping track of which index the non-zero p, q corresponds to.

The HRIR are estimated in N_S HR filter sections $\widehat{\boldsymbol{h}}_s$ by the most important components, in terms of a determined energy metric value, for each of the sections. The obtained model parameters $\boldsymbol{\alpha}^{l,r}$, being $N_{PQ} = \sum_p Q_p$ basis vectors of length K, is partitioned over k of the matrix $\boldsymbol{\alpha}^{l,r}$of $\boldsymbol{\alpha}_k^{l,r}, k = 1, \ldots, K$, into $N_S = 3$ sections, corresponding to the HR filter sections $\widehat{\boldsymbol{h}}_s$, where the first two sections are of length (number of columns) $L_{sec} = \left\lfloor \frac{K}{N_S} \right\rfloor$, and the last one includes the remaining coefficients (which might be $> L_{sec}$). Each section consists of the N_{PQ} sub-vectors (rows) corresponding to each of the $\sum_p Q_p$ elevation and azimuth B-spline products $B[p,q]$ determining the weight of each $\boldsymbol{\alpha}^{l,r}[s]$ row sub-vector for the HR filter generation. An energy metric is determined for the row sub-vectors of $\boldsymbol{\alpha}^{l,r}$, corresponding to the non-zero components of $\boldsymbol{B}$ in the filter generation, as specified in equation (7.2-7) as

$$E_B^{l,r}[p,q] = B[p,q]^2\, E_\alpha^{l,r}[p,q] \tag{7.2-15}$$

where $E_\alpha[p,q]$ is pre-computed energies of the row sub-vectors of $\boldsymbol{\alpha}^{l,r}$. The total energy of the left and right filter components is computed per section s as

$$E_{tot}^{l,r}[s] = \sum_{p,q} E_B^{l,r}[p,q] \tag{7.2-16}$$

The maximum number of sub-vectors used per section is predetermined as $N_c[s] = \{13,12,11\}$. The maximum number of non-zero components is 16, which is much less than the total number N_{PQ} of sub-vectors (rows) of $\boldsymbol{\alpha}^{l,r}$, but it would be further reduced at the knot points of the B-spline functions, see equations (7.2-24) and (7.2-37). Consequently, the predetermined number of vectors to be used is limited according to

$$N_c[s] := \min\left(N_{idx}^{elev}\, N_{idx}^{azim}, N_c[s]\right) \tag{7.2-17}$$

The $N_c[s]$ largest components of $E_B^{l,r}[p,q]$ is determined as $\hat{E}_S^{l,r}[p,q]$ for each section s. To compensate account for lost energy, scale factors $f_E^{l,r}$ are determined as

$$f_E^{l,r}[s] = \sqrt{\frac{E_{tot}^{l,r}[s]}{E_{used}^{l,r}[s]}} \tag{7.2-18}$$

where

$$E_{used}^{l,r}[s] = \sum_{p,q} \hat{E}_s^{l,r}[p,q] \tag{7.2-19}$$

Finally, estimated HR filters for the left and right channel are produced for the requested elevation and azimuth angle (ϑ, φ), separately computed for each of the HR filter sections per filter tap k, as

$$\hat{h}_k^{l,r}(\vartheta,\varphi) = f_E^{l,r}[s] \sum_{p[s],q[s]} B[p,q]\ \alpha_{p,q,k}^{l,r} \qquad \forall\, k \in K_s \tag{7.2-20}$$

for $p[s], q[s]$ being the $N_c[s]$ determined most important components of each section, where K_s includes the filter taps (columns of $\boldsymbol{\alpha}^{l,r}$) of the section s.

7.2.2.1.3 Obtain standard spline sample values

The length of the knot intervals is identified as

$$L_k = \frac{Kseq[N_{BF}-3] - Kseq[0]}{N_{BF}-3} \tag{7.2-21}$$

where N_{BF} is the number of basis functions, and $Kseq$ is the sequence of knot points, including multiplicities. The index of the closest sample point within the identified knot interval is determined as

$$i_0 = \left[\!\!\left[\frac{t - Kseq[0]}{L_k/N_k} \right]\!\!\right] \tag{7.2-22}$$

Where $[\![\cdot]\!]$ denotes rounding and N_k is the number of samples per segment (knot interval). The index of the knot interval is determined as

$$i_k = \lfloor i_0/N_k \rfloor \tag{7.2-23}$$

As the B-splines are of order 4, there are at most four non-zero basis functions, However, at the knot points the last basis function is zero, i.e. the number of non-zero basis functions N_{idx} is

$$\begin{cases} N_{idx} = 3 & if \quad i_0 \bmod N_k = 0 \\ N_{idx} = 4 & otherwise \end{cases} \tag{7.2-24}$$

Subsequently, the basis function shape data is obtained based on the shape metadata. For each of the N_{idx} non-zero basis functions, the shape index and the start index are determined as

$$start\_idx = \max(0, i_{nz} + i_k - 3)\ \ \forall\, i_{nz} \in \{0,1,\dots,N_{idx}-1\} \tag{7.2-25}$$

$$shape\_idx = \min(i_{nz} + i_k, \min(3, N_{BF} - 1 - (i_{nz} + i_k)))\ \ \forall\, i_{nz} \in \{0,1,\dots,N_{idx}-1\} \tag{7.2-26}$$

Then the offset from the starting point of each basis function i_d is determined as

$$i_d = i_0 - start\_idx\ N_k \tag{7.2-27}$$

and further based on the shape metadata

$$\begin{array}{lll} i_d := L_{BS}[shape\_idx] - 1 - i_d & if & reverse\_full\_shape \\ i_d := 2\,(L_{BS}[shape\_idx] - 1) - i_d & if & reverse\_half\_shape \end{array} \tag{7.2-28}$$

where the flipping indicators

$$reverse\_full\_shape = \begin{cases} true & if \quad i_{nz} + i_k > N_{BF} - 4 \\ false & \qquad otherwise \end{cases} \tag{7.2-29}$$

$$reverse\_half\_shape = \begin{cases} true & if \quad i_{nz} > L_{BS}[shape\_idx] - 1 \\ false & \qquad otherwise \end{cases} \tag{7.2-30}$$

$\forall\, i_{nz} \in \{0,1,\dots,N_{idx}-1\}$.

The evaluated value of the basis function shape data for each non-zero basis function is determined as

$$B_f[i_{nz}] = B_{sh}[B_{start}[shape\_idx] + |i_d|] \quad (7.2\text{-}31)$$

where B_{sh} is the compact representation of the basis functions, $B_{start}[shape\_idx]$ identifies which of the compact representations to be used, and i_d defines the index to read from the compact representation to obtain the compact representation or a converted version of the compact representation in determining each sample $B_f[i_{nz}]$.

The indices of the non-zero basis functions are determined as

$$I_f[i_{nz}] = i_{nz} + i_k \quad (7.2\text{-}32)$$

7.2.2.1.4 Obtain periodic spline sample values

The length of the knot intervals is identified as

$$L_k = \frac{Kseq[N_{BF}] - Kseq[0]}{N_{BF}} \quad (7.2\text{-}33)$$

where N_{BF} is the number of basis functions, and $Kseq$ is the sequence of knot points, not including multiplicities.

The number of samples per segment (knot interval) is determined as

$$N_k := N_k / f_{sub} \quad (7.2\text{-}34)$$

where f_{sub} is the subsampling (shape resampling) factor as indicated by the shape metadata.

The index of the closest sample point within the identified knot interval is determined as

$$i_0 = \left[\!\left[\frac{t - Kseq[0]}{L_k / N_k} \right]\!\right] \quad (7.2\text{-}35)$$

where $[\![\cdot]\!]$ denotes rounding and N_k is the number of samples per knot interval. The index of the knot interval is determined as

$$i_k = \lfloor i_0 / N_k \rfloor \quad (7.2\text{-}36)$$

As the B-splines are of order 4, there are at most four non-zero basis functions, However, on the knot points the last basis function is zero, i.e. the number of non-zero basis functions N_{idx} is

$$\begin{cases} N_{idx} = 3 & if \quad i_0 \bmod N_k = 0 \\ N_{idx} = 4 & otherwise \end{cases} \quad (7.2\text{-}37)$$

With a uniform knot sequence, the azimuth basis functions are symmetric, and the basis function shape data obtained based on the shape metadata only depends on the subsampling factor and the offset within the knot interval determined as

$$i_d[i_{nz}] = i_0 - (i_{nz} + i_k - 1)\, N_k \quad (7.2\text{-}38)$$

The evaluated value of the basis function shape data for each non-zero basis function is determined as

$$B_f[i_{nz}] = B_{sh}[|i_d[i_{nz}]|\, f_{sub}] \quad (7.2\text{-}39)$$

$\forall\, i_{nz} \in \{0, 1, \ldots, N_{idx} - 1\}$,

where B_{sh} is the compact representation of the basis functions, and i_d and f_{sub} define the index to read from the compact representation to obtain the compact representation or a converted (sub-sampled) version of the compact representation in determining each sample $B_f[i_{nz}]$.

The indices of the non-zero basis functions are determined as

$$I_f[i_{nz}] = (i_{nz} + i_k) \bmod N_{BF} \quad (7.2\text{-}40)$$

7.2.2.1.5 ITD model

For the case a separate ITD model is used to represent the HR filter set, i.e., the HRIR model represents only the zero-time delay (minimum phase) filters and an ITD model is used to generate the time difference between the left and right channels. This time difference corresponds to the inter-aural time difference.

The ITD model has the same basic structure as the HRIR model, cubic B-spline functions of order $J = 4,$ see clause 7.2.2.1.2. The ITD is obtained as a linear combination of model parameters and B-spline basis function weights, evaluated for the requested elevation angle ϑ and an azimuth angle φ, according to

$$\hat{\tau}(\vartheta,\varphi) = \sum_{\tilde{p}=1}^{\tilde{P}} \sum_{\tilde{q}=1}^{\tilde{Q}_{\tilde{p}}} c_{\tilde{p},\tilde{q}} \tilde{\Theta}_{\tilde{p}}(\vartheta) \tilde{\Phi}_{\tilde{p},\tilde{q}}(\varphi) \tag{7.2-41}$$

$\{\tilde{\Theta}_{\tilde{p}}(\vartheta): \tilde{p} = 1,\dots,\tilde{P}\}$ and $\{\tilde{\Phi}_{\tilde{p},\tilde{q}}(\varphi): \tilde{q} = 1,\dots,\tilde{Q}_{\tilde{p}}\}$ are the B-spline basis functions over the elevation angles and the azimuth angles, respectively. $\{c_{\tilde{p},\tilde{q}}\}$ is a set of model parameters.

In matrix form the ITD representation can be written as

$$\hat{\tau}(\vartheta,\varphi) = \underbrace{\begin{pmatrix} \tilde{\boldsymbol{b}}(\vartheta_1,\varphi_1) \\ \vdots \\ \tilde{\boldsymbol{b}}(\vartheta_M,\varphi_M) \end{pmatrix}}_{\tilde{\boldsymbol{B}}} \boldsymbol{c} = \tilde{\boldsymbol{B}}\boldsymbol{c} \tag{7.2-42}$$

where

$$\begin{aligned} \tilde{\boldsymbol{b}}(\vartheta_m,\varphi_m) &= \left(\tilde{\Theta}_{\tilde{p}}(\vartheta_m)\tilde{\Phi}_{\tilde{p},\tilde{q}}(\varphi_m):\ \tilde{p} = 1,\dots,\tilde{P}; \tilde{q} = 1,\dots,\tilde{Q}_{\tilde{p}}\right)_{row\ vector} \\ \boldsymbol{c} &= \left(c_{\tilde{p},\tilde{q}} : \tilde{p} = 1,\dots,\tilde{P}; \tilde{q} = 1,\dots,\tilde{Q}_{\tilde{p}}\right)_{column\ vector} \end{aligned} \tag{7.2-43}$$

As ITDs at azimuths between 180 to 360 degrees can be considered a mirror of those at azimuths between 0 to 180 degrees, mirrored B-spline functions are used for azimuth angles of the two intervals, [0, 180] and [180, 360], i.e.

$$\tilde{\Phi}_{\tilde{q}}(\varphi) = \begin{cases} \displaystyle\sum_{j=0}^{J-1} \gamma_{j,l,\tilde{q}}^{\tilde{\Phi}} \varphi^j, & \varphi \le 180 \text{ and } \tilde{\phi}_l \le \varphi < \tilde{\phi}_{l+1} \\ \displaystyle\sum_{j=0}^{J-1} \gamma_{j,l,\tilde{q}}^{\tilde{\Phi}} (360-\varphi)^j, & \varphi > 180 \text{ and } \tilde{\phi}_l \le 360-\varphi < \tilde{\phi}_{l+1} \end{cases} \tag{7.2-44}$$

for the knot sequence $\{\tilde{\phi}_l: l = 1,\dots,L_{ITD}\}$. Figure 7.2-5 shows an example of B-spline functions as used for the azimuth modelling of ITD. For the elevation, standard B-spline functions may be used, see example in figure 7.2-3, being polynomial functions

$$\tilde{\Theta}_{\tilde{p}}(\vartheta) = \sum_{j=0}^{J-1} \gamma_{j,u,\tilde{p}}^{\tilde{\Theta}} \vartheta^j, \quad \tilde{\theta}_u \le \vartheta < \tilde{\theta}_{u+1}. \tag{7.2-45}$$

for the knot sequence $\{\tilde{\theta}_u: u = 1,\dots,U_{ITD}\}$.

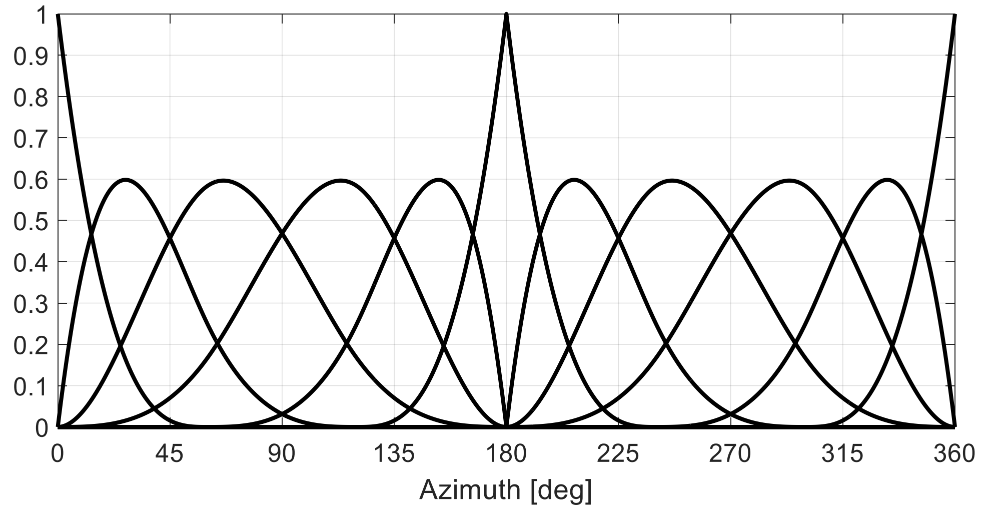


Figure 7.2-5: Example of mirrored standard B-spline functions for azimuth.

To generate an ITD value of the requested elevation angle ϑ and azimuth angle φ, the azimuth is wrapped to the supported range as specified in equation (7.2-21)-(7.2-22). The non-zero samples of the elevation basis functions are determined according to clause 7.2.2.1.3 with the shape metadata and basis function shape data (e.g. $Kseq$, $N_{BF}, L_{BS}, B_{sh}, B_{start}$) of the elevation basis functions for the ITD model. Similarly, the values of the non-zero basis functions for azimuth are obtained from the basis function shape data in accordance with clause 7.2.2.1.4 with the elevation-dependent shape metadata and basis function shape data (e.g. $Kseq$, $N_{BF}, L_{BS}, B_{sh}, B_{start}$) of the azimuth basis functions for the ITD model. Note, due to the mirroring $N_{BF} \coloneqq (N_{BF}+1)/2$ and $\varphi := 360 - \varphi$ if $\varphi > 180$.

At the poles $\vartheta = \{-90, 90\}$, there is one elevation basis function equal to one and a constant azimuth basis function such that

$$B[\tilde{p}, \tilde{q}] = 1 \tag{7.2-46}$$

At other positions, not being at the poles, the elements of the matrix $\widetilde{\boldsymbol{B}}$, are computed by

$$\tilde{B}[\tilde{p}, \tilde{q}] = B_{f,ITD}^{elev}[\tilde{p}]\, B_{f,ITD}^{azim}[\tilde{q}]\ \forall\, \tilde{p} \in \{1, \ldots, N_{idx,ITD}^{elev}\}, \tilde{q} \in \{1, \ldots, N_{idx,ITD}^{azim}\} \tag{7.2-47}$$

Finally, the ITD is computed as

$$\hat{t}(\vartheta, \varphi) = -\left[\!\!\left[f_{resamp} \sum_{\tilde{p},\tilde{q}} \tilde{B}[\tilde{p}, \tilde{q}]\, c_{\tilde{p},\tilde{q}} \right]\!\!\right] \tag{7.2-48}$$

for $\tilde{p}, \tilde{q}$ corresponding to the non-zero B-spline functions of the requested elevation angle ϑ and an azimuth angle φ. The matrix $\widetilde{\boldsymbol{B}}$ may be represented by a vector, keeping track of which index the non-zero $\tilde{p}, \tilde{q}$ corresponds to. The negation comes from different conventions between model and renderer. f_{resamp} is a resampling factor in case the rendering is performed with a different sampling rate than the ITD model is representing, typically 48 kHz.

7.2.2.1.6 ITD synthesis

The ITD synthesis adjusts the timing of the signals such that the desired ITD is achieved in the rendered signal. For audio segments where the ITD remains the same, this can is realized by buffering and delaying the signal with the later time of arrival. To keep the delay at a minimum, the signal with the later time of arrival is delayed while the other channel is rendered with zero delay. When the ITD value changes, a time scaling operation needs to be performed in order to change the alignment of the channels. In case the ITD value changes sign, this means the delayed channel needs to be adjusted to zero delay and the other channel is adjusted to be delayed with the new target ITD.

For each object signal $\hat{s}_i(n)$, either input from the decoder or fed to the external renderer, the new object signal frame is fed into a processing buffer. The L_{ITDmem} samples of memory from the preceding frame is appended in front of the new signal frame. First, a buffer length L_3 is calculated to leave look-ahead room for the resampling filter according to

$$\begin{cases} L_3 = \max(0, L_{sinc} - |ITD(m)|) \\ L_{tot} = L_{sf} - L_3 \end{cases} \tag{7.2-49}$$

where $L_{sinc} = 5$ is the number of samples used in the polyphase resampling stage, $ITD(m)$ is the ITD value of the current subframe m, L_{tot} is the total transition time in samples and L_{sf} is the length of the 5 ms subframe. At 48 kHz sampling rate, $L_{sf} = 240$. If the sign of the ITD did not change from the previous subframe, or one of them is zero $ITD(m-1)ITD(m) \geq 0$, the time scaling operation only needs to be done on one channel. In that case the number of transition times in samples $L_1$and L_2 are calculated according to

$$\begin{cases} L_1 = L_{tot} \\ L_2 = L_{tot} - L_1 = 0 \\ n_1 = -|ITD(m-1)| \\ L_{in,1} = L_1 + |ITD(m-1)| - |ITD(m)| \\ n_2 = L_1 - |ITD(m)| \\ L_{in,2} = 0 \\ n_3 = L_{tot} \end{cases} \tag{7.2-50}$$

where n_1, n_2, n_3 denote the starting indices of each time shift segment assuming that the current input subframe starts at $n = 0$, $L_{in,1}, L_{in,2}$ is the length of the resampling segments 1 and 2. If the previous and current subframe ITD is non-zero and changes sign $ITD(m-1)ITD(m) < 0$ the transition times L_1 and L_2 are calculated according to

$$\begin{cases} L_1 = \left[\frac{L_{tot}|ITD(m-1)|}{|ITD(m-1)|+|ITD(m)|}\right] \\ L_2 = L_{tot} - L_1 \\ n_1 = -|ITD(m-1)| \\ L_{in,1} = L_1 + |ITD(m-1)| \\ n_2 = L_1 \\ L_{in,2} = L_2 - |ITD(m)| \\ n_3 = L_{tot} - |ITD(m)| \end{cases} \tag{7.2-51}$$

where $[\cdot]$ denotes rounding to the nearest integer. Next, the output buffers A and B are assembled using the time-shifting the signal using the transition lengths L_1 and L_2 as illustrated in 7.2-65.3-15. First, a resampling is done of the buffer starting from n_1 of length $L_{in,1}$ to the first L_1 samples of output buffer A. The last $L_{sf} - L_1$ samples are populated by copying the remainder of the input buffer to output buffer A. Then, the first samples of the input buffer starting from index 0 are copied to the first L_1 samples of output buffer B. The next L_2 samples of output buffer B are created by resampling the samples starting from n_2 in the input buffer of length $L_{in,2}$. Finally, the last L_3 samples of the input buffer are copied to output buffer B. The last L_{ITDmem} samples of the input frame is stored in memory for processing the next subframe. Output buffers A and B are assigned to the output channels 0 and 1 for left and right HRIR filtering respectively. The assignment of the output channels is done based on the signs of $ITD(m)$ and $ITD(m-1)$ as follows,

$$\begin{cases} (A,B) \rightarrow (1,0), \; ITD(m-1) = 0 \wedge ITD(m) > 0 \; \vee ITD(m-1) > 0 \\ (A,B) \rightarrow (0,1), \; otherwise \end{cases} \tag{7.2-52}$$

where $\wedge$ denotes logical AND and $\vee$ denotes inclusive OR. The resampling operations are implemented using a polyphase filter with a sinc function from a lookup table.

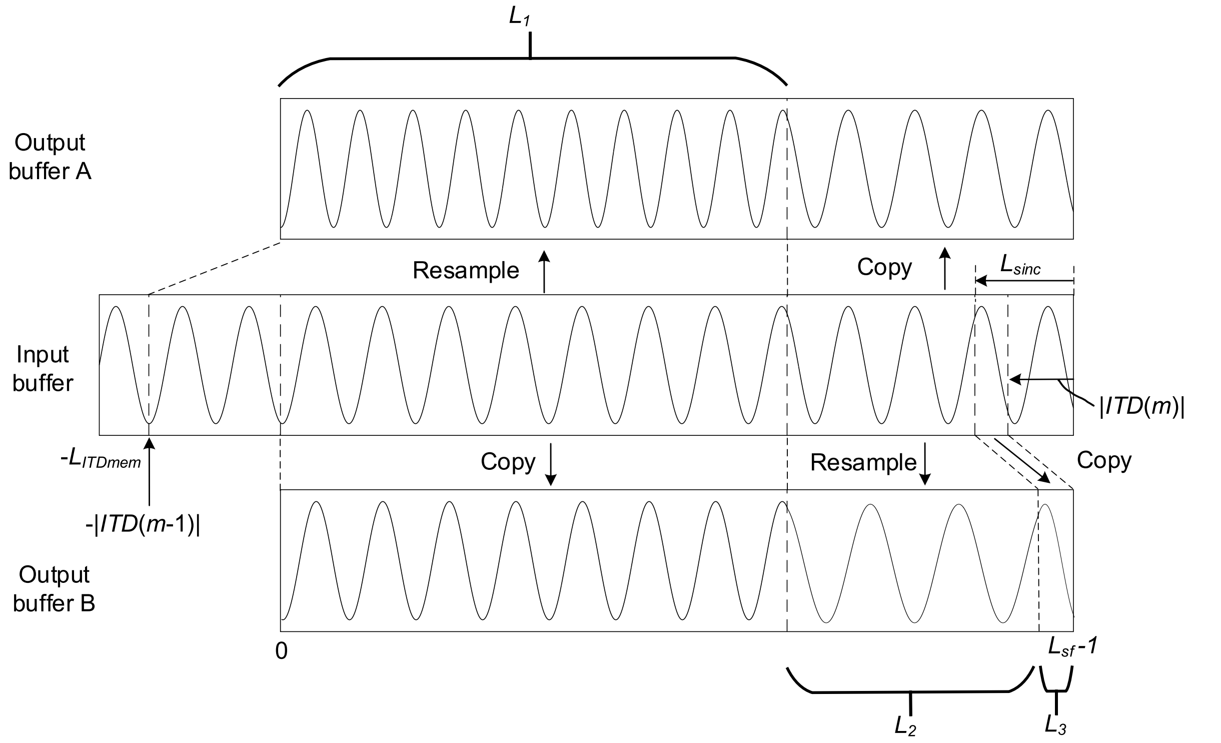


Figure 7.2-6: Resampling and copying operations when changing ITD.

7.2.2.1.7 Distance and Direction Gain

7.2.2.1.8 HRIR convolution

If the difference from the previous frame for azimuth or elevation is higher than zero, an interpolation count variable, cnt_{intp} is set and subjected to a threshold of $thr = 12$:

$$cnt_{intp} = \begin{cases} \Delta_{angle}, & \Delta_{angle} < 12 \\ 12, & else \end{cases} \tag{7.2-53}$$

$$\Delta_{angle} = \max(|\Delta_{azimuth}| * 100, |\Delta_{elevation}| * 100) \tag{7.2-54}$$

$$\Delta_{azimuth} = azimuth - azimuth^{[-1]} \tag{7.2-55}$$

$$\Delta_{elevation} = elevation - elevation^{[-1]} \tag{7.2-56}$$

If cnt_{intp} is bigger than '*zero*' and subframe update flag is set, the right and left HR filter interpolation delta is calculated.

$$\Delta h_{left} = \frac{\hat{\boldsymbol{h}}_l - \hat{\boldsymbol{h}}_l^{[-1]}}{cnt_{intp}}, \qquad \Delta h_{right} = \frac{\hat{\boldsymbol{h}}_r - \hat{\boldsymbol{h}}_r^{[-1]}}{cnt_{intp}} \tag{7.2-57}$$

Following the application of ITD to $\hat{h}_k^{l,r}(\vartheta, \varphi)$ by delaying the late channel as described in clause 7.2.2.1.6, a gain $g_{tmp}(m;n)$ is applied to the convolution of the HR filter for the audio signals of each of the left (l) and right (r) channel. Gain is calculated based on the relative distance and directivity of the listener based on the input from head tracking and metadata. The resulting signal $s_{l,r}(m;n)$ is then:

$$s_{l,r}(m;n) = \left(s_{in}(m;n) * \tilde{h}_{l,r}(m;n)\right) \times g_{tmp}(m;n) \tag{7.2-58}$$

where $*$ and $\times$ stand for convolution and multiplication respectively, and $s_{in}(m;n)$ denotes the audio signal being filtered. $\tilde{h}_{l,r}(m;n)$ is calculated based on the generated pair of HR filters $\hat{h}_k^{l,r}(\vartheta, \varphi)$, adjusted for ITD, by a linear interpolation using $\Delta h_{left,right}$ from equation (7.2-57) as a step for the samples $n < cnt\_intp$. The gain $g_{tmp}(m;n)$

is also applied via linear interpolation to assure a smooth transition in case of abrupt gain changes. The gain difference between two subsequent subframes is divided to the length of subframe to calculate the increment step.

7.2.2.2 Parametric binauralizer and stereo renderer

7.2.2.2.1 Overview

The parametric binauralizer and stereo renderer operates on the following IVAS formats and operations: MASA, OMASA, multi-channel (in McMASA mode), SBA, OSBA, and ISM, i.e., the input to the encoder has been audio signals (and potentially spatial metadata) in one of these formats, and it is now being rendered to binaural or stereo output. The IVAS format being processed (i.e., whether operating on MASA, OMASA, multi-channel (in McMASA mode), SBA, OSBA, or ISM format) is obtained from TODO: ADD REF TO CONFIG.

The parametric binaural (and stereo) rendering is performed in subframes, where m denotes the subframe index. A subframe contains N_{slots} CLDFB slots (in non-JBM operation, $N_{slots} = 4$, in JBM operation $N_{slots} = 1 \ldots 7$). The data determined at previous calls (i.e., subframes m-1 and earlier) affects the rendering of the present subframe m due to temporal averaging and interpolation. The binaural renderer system described in the following is also capable to render a stereo signal instead of a binaural signal. Also, binaural sound with and without room effect can be reproduced.

As an input, the renderer obtains (or receives) a spatial audio signal containing one or two transport audio signals and associated spatial metadata. The number of transport audio signals is one, when TODO: ADD REF TO CONFIG. The number of transport audio signals is two, when TODO: ADD REF TO CONFIG. The spatial metadata obtained (or received) by the renderer contains the following parameters: azimuth $\theta(b,m,i)$, elevation $\phi(b,m,i)$, direct-to-total energy ratio $r_{dir}(b,m,i)$, spread coherence $\zeta(b,m,i)$, and surround coherence $\gamma(b,m)$. In case of the SBA and OSBA formats, the spatial metadata contains SPAR metadata. The audio signals and the spatial metadata are used for providing spatial audio reproduction (i.e., to enable the rendering of spatial audio).

In addition, the input to the renderer includes a separated centre channel audio signal when the IVAS format is multi-channel and operating in McMASA “separate channel” mode. In addition, the input to the renderer contains a separated object audio signal and associated object metadata when the IVAS format is OMASA and operating in “MASA one object coding mode” or “parametric one object coding mode”. In addition, the input to the renderer contains all object audio signals and associated object metadata when the IVAS format is OMASA and operating in “discrete coding mode”.

Furthermore, head orientation data and external orientation data may be received. Head tracking is described in clause 7.4.2, external orientation input is described in clause 7.4.4, and rotation combining functionality for determining the final rotation data to be used in the binaural rendering is described in clause 7.4.5. Head tracking (and external orientation information) should be used if they are available for improved spatial audio experience. In the following it is referred to head orientation and head tracking for simplicity regardless of which components the orientation data is originally composed of, and which processing has been applied to it prior to the rendering step.

In addition, the renderer (when rendering binaural output) obtains a room effect control indication TODO: ADD REF TO CONFIG, and based on this indication, it is determined whether to apply a room effect to the input spatial audio signal or to not apply the room effect. For rendering binaural audio with a room effect, two data sets related to binaural rendering are obtained. The first is a pre-defined data set containing the HRTFs as spherical harmonics to binaural conversion matrices. The second data set contains binaural room responses as reverberation early part energy correction gains (which are used for modifying the resulting spectrum that is obtained from the rendering according to the first data set), and late reverberation energy correction gains and reverberation times. Thus, the binaural output signals are generated from the transport audio signal(s) and associated metadata based on a combination of these two data sets.

The fetching of the temporally correct spatial metadata parameter values for the current subframe m so that they are in sync with the audio signals is handled in clause XXX. TODO: ADD REF. It operates differently for the JBM and non-JBM use. Fetching the correct spatial metadata values is not discussed in the following, it is assumed that it has already been correctly performed, as described in the aforementioned clause.

The binaural (or stereo) rendering is based on using covariance matrices. The transport signal covariance matrices are measured, and the target covariance matrices are determined. Based on these covariance matrices, processing matrices are determined to process the transport audio signals to a determined target. The details of this operation are described in the following.

First, the transport audio signals are transformed to the time-frequency domain with a 60-bin complex low-delay filter-bank (CLDFB) (see clause XXX for details TODO: ADD REF), resulting in $S(k,n,i)$, where k is the frequency bin index, n is the CLDFB temporal slot index, and i is the transport audio signal channel index. In case there is only one transport audio signal, the time-frequency transformed signal $S(k,n,1)$ is duplicated to a dual-mono signal where $S(k,n,2)=S(k,n,1)$. Any object audio signals, as well as the potential separated centre channel signal, are also transformed to a time-frequency representation using the same filter bank.

If the IVAS format is SBA or OSBA, MASA metadata is determined from the SPAR metadata as detailed in clause XXX. TODO: ADD REF, and the prototype audio signals are determined as detailed in clause XXX. TODO: ADD REF.

If the head tracking is enabled, and if the number of transport audio signals is two, the transport audio signals are processed for improved rendering quality using the methods described in clause 7.2.2.2.2.

Then, the input and target covariance matrices are determined based on the transport audio signal(s) and the spatial metadata as described in clause 7.2.2.2.3.

Then, the processing matrices are determined based on the determined input and target covariance matrices as described in clause 7.2.2.2.4.

And, finally, the binaural signals (with or without the room effect) or the stereo signals are rendered to reproduce the sound scene depicted by the input audio signals and the spatial metadata by processing the input audio signals using the determined processing matrices as described in clause 7.2.2.2.5.

7.2.2.2.2 Pre-processing the transport audio signals based on head orientation

When there are two transport audio signals (i.e., not duplicated mono transport audio signal), the transport audio signals are adapted based on the head orientation parameter(s). This is performed by analysing the two channel transport audio signals to determine an inter-channel property and based on that property and the obtained head orientation parameter(s) determining mixing information to process the two transport audio channels to two processed transport audio channels. These processed transport audio channels then replace the originals in the following clauses and are used (together with the head orientation parameter(s) and spatial metadata) for rendering the output binaural audio signals.

In case head orientation data is not available, or if there was only one transport audio channel, or if stereo output is used, none of the following steps in this clause are performed.

First part of the pre-processing of the transport audio signals is to adaptively cross-mix them in a condition when the user is facing side directions, using the following steps.

First, the rotation matrix $\mathbf{R}$ is obtained based on the head orientation TODO: ADD REF. From the rotation matrix $\mathbf{R}$, the second-row second-column entry is selected, which is denoted r_y. The subscript y refers to the entry being the y-axis-to-y-axis processing element of the rotation matrix. Then, a first mono factor is determined based on the head orientation by

$$f_{\text{mono,R}} = trunc_{0,1}\left(\frac{0.6-|r_y|}{0.4}\right)$$

where $trunc_{0,1}$ is an operation that sets values smaller than 0 to 0, and values larger than 1 to 1. f_{mono} is the same for all bands but may vary over time.

Then, the transport audio signals are analysed to determine an inter-channel property (namely, the inter-channel level difference) based on which the transport audio signals are to be mixed. The following processing steps are performed in frequency bands according to MASA_band_grouping_24 TODO: ADD REF. For each band index b in that grouping, there is a lower and upper bin index $k_{\text{low}}(b)$ and $k_{\text{high}}(b)$.

First, the channel energies are formulated in bands b by

$$E_{in}(b,m,i) = \sum_{k=k_{\text{low}}(b)}^{k_{\text{high}}(b)}\left(\sum_{n=n_{first}(m)}^{n_{last}(m)} |S(k,n,i)|^2\right)$$

where $n_{first}(m)$ and $n_{last}(m)$ are the first and last temporal slots of the current subframe m, and i is the transport audio signal channel index. Then, temporal averaging is performed by

$$E'_{in}(b,m,i) = \alpha_{E_{in}} E'_{in}(b,m-1,i) + (1-\alpha_{E_{in}})E_{in}(b,m,i)$$

where $\alpha_{E_{in}} = 0.81450625$ and where $E'_{in}(b,0,i) = 0$ (for the first subframe of the first frame). Then, the inter-channel level difference (ILD) and the absolute value of it are determined by

$$ILD(b,m) = \left| 10\log_{10}\frac{E'_{in}(b,m,1)}{E'_{in}(b,m,2)} \right|$$

Then a louder and softer channel indices l and s are defined so that if $E'_{in}(b,m,2) > E'_{in}(b,m,1)$, then $l = 2$ and $s = 1$, and otherwise $l = 1$ and $s = 2$.

Then a second mono factor is determined based on the ILD by

$$f_{\mathrm{mono,ILD}} = trunc_{0,1}\left(\frac{ILD(b,m)-1}{3}\right)$$

Next, mixing information is determined by combining the two mono factors

$$f_{\mathrm{mono}} = f_{\mathrm{mono,R}} f_{\mathrm{mono,ILD}}$$

Then, for each bin from $k_{\mathrm{low}}(b)$ to $k_{\mathrm{high}}(b)$ and from each slot from $n_{first}(m)$ to $n_{last}(m)$ the transport signal is mixed by

$$S_{mix}(k,n,s) = f_{\mathrm{mono}}\big(S(k,n,1)+S(k,n,2)\big) + (1-f_{\mathrm{mono}})S(k,n,s)$$

$$S_{mix}(k,n,l) = S_{mix}(k,n,l)$$

Energy after mixing is formulated

$$E_{mix}(b,m,i) = \sum_{k=k_{\mathrm{low}}(b)}^{k_{\mathrm{high}}(b)} \left(\sum_{n=n_{first}(m)}^{n_{last}(m)} |S_{mix}(k,n,i)|^2 \right)$$

This is temporally averaged

$$E'_{mix}(b,m,i) = \alpha_{E_{in}} E'_{mix}(b,m-1,i) + (1-\alpha_{E_{in}})E_{mix}(b,m,i)$$

where $E'_{mix}(b,0,i) = 0$ (for the first subframe of the first frame).

Finally, the transport signals are equalized, resulting in adapted transport audio signals, which are set to replace the original transport audio signals, for each bin from $k_{\mathrm{low}}(b)$ to $k_{\mathrm{high}}(b)$ and from each slot from $n_{first}(m)$ to $n_{last}(m)$, by

$$S(k,n,i) \coloneqq S_{mix}(k,n,i)\min\left(1, \sqrt{\frac{E'_{in}(b,m,1)+E'_{in}(b,m,2)}{E'_{mix}(b,m,1)+E'_{mix}(b,m,2)}}\right)$$

Next, for the transport audio signals, it is checked if the transport audio signals need to be switched (so that left and right channels replace each other). Such processing is performed so that when the user faces rear directions, the transport audio signal aligns better in the left-right axis for the current head orientation. The processing has an interpolation factor $f_{interp}(n)$ so that $f_{interp}(0) = 0$.

The left-right channel switching process has a state, and this state can be one of these four: State **A** for "switching", state **B** for "not switching", and then there can be two transition states, denoted **A→B** and **B→A**.

As a first step, it is checked if the system is at one of the transition states. If not, then following steps are performed: If in state **A** and $r_y < -0.17$, then change state to **A→B**; If in state **B** and $r_y > 0.17$, then change state to **B→A**.

Then, if the system is in state **A**, no switching is done to the transport audio signals $S(k,n,i)$ for any of the indices n in the current subframe m, and the following operations are not performed.

If the system is in state **B**, transport audio signals are, for all indices n in the current subframe m, switched so that the data at $S(k,n,1)$ and $S(k,n,2)$ replace each other. In this condition, the following operations are not performed.

Then, the following steps are performed when the system is at either of the transition states **A→B** or **B→A**. The following steps are performed in this order for each slot n of the present subframe m.

First, the interpolation value is updated by

$$f_{interp}(n) = f_{interp}(n-1) + 0.0025$$

Next, if $f_{interp}(n) > 0.999$, then $f_{interp}(n)$ is set to 0 and if in mode **A→B** the state is set to **B**; and if in state **B→A** the mode is set to **A**. If either of these state updates is performed, the current and the remaining slots n within the subframe are processed as previously described for states **A** or **B**, depending on which state was now set, and then the following operations are not performed.

Two further factors are formulated. In case of **A→B**

$$f_{switch}(n) = \sqrt{f_{interp}(n)}$$

$$f_{orig}(n) = \sqrt{1 - f_{interp}(n)}$$

And in case of **B→A**

$$f_{switch}(n) = \sqrt{1 - f_{interp}(n)}$$

$$f_{orig}(n) = \sqrt{f_{interp}(n)}$$

Then, mixing is performed by

$$\begin{bmatrix} S_{mix}(k,n,1) \\ S_{mix}(k,n,2) \end{bmatrix} = \begin{bmatrix} f_{orig}(n) & f_{switch}(n) \\ f_{switch}(n) & f_{orig}(n) \end{bmatrix} \begin{bmatrix} S(k,n,1) \\ S(k,n,2) \end{bmatrix}$$

Then, equalization gains for the channels $i = 1,2$ are formulated by

$$g_{eq}(k,n,i) = \min\left(4, \sqrt{\frac{\left|S(k,n,i)f_{orig}(n)\right|^2 + |S(k,n,2-i)f_{switch}(n)|^2}{|S_{mix}(k,n,1)|}}\right)$$

Finally, the transport audio signals are replaced with

$$S(k,n,i) \coloneqq S_{mix}(k,n,i)g_{eq}(k,n,i)$$

The above transition-state **A→B** or **B→A** procedures are performed for each slot n until the slots n of the subframe are exhausted.

7.2.2.2.3 Determination of input and target covariance matrices

A target covariance matrix contains properties that the output signal is targeted to attain, and it is determined in frequency bins. Considering a binaural or stereo output, a target covariance matrix contains the channel energies and the cross-correlations for the output. The cross-correlation amplitude determines the coherence between the output channels and the phase determines the phase-difference. The energies, phases, and coherences are controlled since they are key contributors to the spatial audio perception. For the renderer, there is always two output channels (binaural or stereo), and therefore the target covariance matrix is of shape 2x2 for each frequency bin.

Correspondingly, the input covariance matrix has the information of these energetic and correlation properties of the received transport audio channels. Any additional object nor centre channels are not considered in the input covariance matrix. Therefore, also the input covariance matrices are always of shape 2x2 for each frequency bin.

Many of the described operations are identical in case of binaural and stereo outputs. Any differences are specifically mentioned.

The input covariance matrix $\mathbf{C}_{in}(k,m)$ and subframe energy $E_{in}(k,m)$ are determined for the current subframe m as follows. First, the input signal is expressed in a column vector form

$$\mathbf{s}(k,n) = \begin{bmatrix} S(k,n,1) \\ S(k,n,2) \end{bmatrix}$$

Then,

$$\mathbf{C}_{in}(k,m) = \sum_{n=n_{first}(m)}^{n_{last}(m)} \mathbf{s}(k,n)\,\mathbf{s}^H(k,n)$$

$$E_{in}(k,m) = tr(\mathbf{C}_{in}(k,m))$$

Where $n_{first}(m)$ and $n_{last}(m)$ are the first and last temporal slots of the current subframe m, and $tr\big(\mathbf{C}_{in}(k,m)\big)$ denotes the matrix trace of $\mathbf{C}_{in}(k,m)$.

In case the low-bitrate EQ is enabled (i.e., if the bitrate is 13.2 or 16.4 kbps and the IVAS format is MASA or multi-channel), the subframe energy $E_{in}(k,m)$ is modified using it

$$E_{in}(k,m) \coloneqq E_{in}(k,m) g_{lbr,EQ}(k)$$

where $g_{lbr,EQ}(k)$ is the low-bitrate EQ TODO: ADD REF TO THE TABLE. If it is not enabled, the subframe energy modification is not performed.

In addition, if the IVAS format is SBA or OSBA and if the number of transport audio signals is 2, the subframe energy $E_{in}(k,m)$ is modified by

$$E_{in}(k,m) := \max\left(\frac{E_{in}(k,m)}{2}, \sum_{n=n_{first}(m)}^{n_{last}(m)} |S(k,n,1) + S(k,n,2)|^2\right)$$

For the other formats, and for the other number of transport audio signals, this modification is not performed.

The target covariance matrices are described in the following, based on the spatial metadata and the subframe energy $E_{in}(k,m)$.

First, the mean energy per output channel $E_{meanOut}(k,m)$ is formulated by

$$E_{meanOut}(k,m) = 0.5 g_{early}(k)\, E_{in}(k,m)$$

where $g_{early}(k)$ is an early part energy correction gain for the binaural rendering. When the room effect is not enabled, $g_{early}(k) = 1$. When the room effect is enabled, the values for $g_{early}(k)$ are obtained from the data set in TODO: ADD REF TO THE TABLE.

The spatial metadata that the renderer obtains may have one or two simultaneous "parametric" directions (associated with the MASA, SBA, multichannel, or ISM streams), and from one to four simultaneous "object" directions (associated with the objects part of the OMASA stream). I.e., there may be from one to six simultaneous directions. Each direction is associated with an azimuth, elevation, direct-to-total energy ratio and spread coherence parameter. The number of simultaneous directions is denoted by D, and d refers to the direction index for $1 \leq d \leq D$.

Before formulating the rendering gains, if operating in the separateCenterChannelRendering mode (of McMASA) TODO: ADD REF TO CONFIG), the spread coherence is modified by

$$\zeta_d(k,m) := \max\left(\zeta_d(k,m), 1 - \frac{\cos^{-1}(\cos(\theta_d(k,m))\cos(\varphi_d(k,m)))}{30^\circ}\right)$$

to account for the missing centre channel in the analysis of the spread coherence in the encoder.

In processing the target covariance matrix, the next step is to determine the direct part gain vector $\mathbf{g}_{direct,d}(k,m)$, which is a 2x1 column vector. It is formulated based on the metadata azimuth $\theta_d(k,m)$ and elevation $\varphi_d(k,m)$ for each d, as described in clause 7.2.2.2.6, where it is also described how the spread coherence value $\zeta_d(k,m)$ is used in determining the gains. The gains $\mathbf{g}_{direct,d}(k,m)$ may be complex valued in case of a binaural output (due to needed phase differences in the binaural output) and are real valued for stereo output.

The direct part target covariance matrix is then

$$\mathbf{C}_{out,dir}(k,m) = E_{meanOut}(k,m)\sum_{d=1}^{D}\mathbf{g}'_{direct,d}(k,m)\mathbf{g}'^{H}_{direct,d}(k,m)r_d(k,m)$$

where $r_d(k,m)$ is the direct-to-total energy ratio of direction index d.

The non-direct part of the target covariance matrix is determined as follows. First, the diffuse-to-total energy ratio (also called diffuseness, in short) is formulated as

$$r_{diff}(k,m) = \max\left(0,\left(1-\sum_{d=1}^{D}r_d(k,m)\right)\right)$$

Also, a diffuseness value for decorrelation reduction is formulated as

$$r_{diff,dec}(k,m) = \max\left(0,\left(1-\sum_{d=1}^{D}(1-0.5*ism(d))r_d(k,m)\right)\right)$$

where ism(d) is a function that is 1 when the direction d corresponds to an ISM direction indicated in the spatial metadata, and 0 otherwise. The value $r_{diff}(k,m)$ therefore describes the energy of the non-directional sound energy, whereas $r_{diff,dec}(k,m)$ is a related control parameter that is later used to configure the level of decorrelated energy.

Next, the per-output-channel diffuse sound energy, and a related control parameter, are formulated by

$$E_{diff}(k,m) = r_{diff}(k,m)E_{meanOut}(k,m)g_{diffMod}(k,m)$$

$$E_{diff,dec}(k,m) = r_{diff,dec}(k,m)E_{meanOut}(k,m)g_{diffMod}(k,m)$$

where $g_{diffMod}(k)$ is an energy modifier value which is 1 otherwise, except in case of the output being binaural and the IVAS format being multichannel, in which case it has the following value

$$g_{diffMod}(k,m) = \left(1-\gamma(k,m)\right)+\gamma(k,m)E_{\gamma}(k)$$

where $0 \leq \gamma(k,m) \leq 1$ is the surround coherence value and $E_{\gamma}(k)$ is a spectral modifier that affects the binaural spectrum of surround-coherent sound. TODO: ADD REF TO TABLE

In case the output is stereo (not binaural), the non-direct part of the target covariance matrix then is

$$\mathbf{C}_{out,nonDir}(k,m) = (\gamma(k,m)+(1-\gamma(k,m)\mathbf{I})E_{diff}(k,m)$$

Where $\mathbf{I}$ denotes a 2x2 identity matrix.

In case the output is binaural, the non-direct part of the target covariance matrix is

$$\mathbf{C}_{out,nonDir}(k,m) = \left(\gamma(k,m)+(1-\gamma(k,m)c_{diffBin}(k)\mathbf{I}\right)E_{diff}(k,m)$$

where $c_{diffBin}(k)$ is the binaural diffuse field coherence.

If the IVAS format is SBA or OSBA, the binaural diffuse field coherence rendering information is determined based on the directional distribution information for the indirect audio (the diffuse ratios in x, y, and z directions)

$$c_{diffBin}(k,m) = \varsigma_x(k,m)c_{diffBin,x}(k)+\varsigma_y(k,m)c_{diffBin,y}(k)+\varsigma_z(k,m)c_{diffBin,z}(k)$$

where $\varsigma_{x/y/z}(k,m)$ is the diffuse ratio in x/y/z-directions (see clause XXX TODO: ADD REF) and $c_{diffBin,x/y/z}(k)$ is a binaural diffuse field coherence in the x/y/z-directions. TODO: ADD REFERENCES TO THE TABLES WITH VALUES AND/OR EQUATIONS. The determined binaural diffuse field coherence enables rendering binaural audio with the correct directional distribution of the indirect audio.

If the IVAS format is not SBA nor OSBA, the binaural diffuse field coherence $c_{diffBin}(k,m)$ is fixed (i.e., $c_{diffBin}(k,m) = c_{diffBin}(k)$) and can be obtained from TODO: ADD REF TO TABLE OR EQUATIONS.

Then, the target covariance matrix is formed by combining the target covariance matrices for the direct and the non-direct parts

$$\mathbf{C}_{out}(k,m) = \mathbf{C}_{out,dir}(k,m) + \mathbf{C}_{out,nonDir}(k,m)$$

The above processing was for the current subframe. Next, the covariance matrices are temporally smoothed. An averaging value is formulated by

$$\alpha_{IIR}(m) = \beta_{IIR} + (1 - q(n))$$

where $\beta_{IIR} = 16$ if the IVAS format is MASA and the bitrate is smaller than 24.4 kbps and $\beta_{IIR} = 8$ otherwise. $q(n)$ is the encoding quality based smoothing factor, which is determined by

$$q(n) = \nu^2$$

where ν is an encoding metric indicating the quality of encoding TODO: ADD REFERENCE TO THE DETERMINATION OF NU. The determined averaging value $\alpha_{IIR}(m)$ is then used for temporally smoothing the covariance matrices.

Furthermore, a temporal energy ratio parameter is determined

$$E_{in,r}(k,m) = \frac{tr\left(\mathbf{C}_{in}(k,m)\right)}{tr(\mathbf{C}_{in}(k,m-1))}$$

Then an IIR energy limiter factor is determined

$$\alpha_{lim}(k,m) = \min\left(1, E_{in,r}(k,m)\alpha_{IIR}(m)\right)$$

The temporally smoothed covariance matrices are obtained by

$$\mathbf{C}'_{in}(k,m) = q(n)\mathbf{C}_{in}(k,m) + \alpha_{lim}(k,m)\mathbf{C}'_{in}(k,m-1)$$

$$\mathbf{C}'_{out}(k,m) = q(n)\mathbf{C}_{out}(k,m) + \alpha_{lim}(k,m)\mathbf{C}'_{out}(k,m-1)$$

where $\mathbf{C}'_{in}(k,0) = \mathbf{C}'_{out}(k,0) = \mathbf{0}$ for the first subframe of the first frame.

The matrices $\mathbf{C}'_{in}(k,m)$ and $\mathbf{C}'_{out}(k,m)$ are then the formulated and temporally averaged input and target covariance matrices.

7.2.2.2.4 Determining processing matrices based on input and target covariance matrices

In this clause, processing parameters called processing matrices are determined based on the input and target covariance matrices. Using these processing matrices and the transport audio signals, the binaural or stereo audio signals are generated in clause XXX. TODO: ADD REF.

First, the output covariance matrix is Cholesky decomposed to $\mathbf{K}_{out}(k,m)$ so that $\mathbf{C}'_{out}(k,m) = \mathbf{K}_{out}(k,m)\mathbf{K}^H_{out}(k,m)$.

Then, the input covariance matrix is Eigen decomposed to $\mathbf{U}_{in}(k,m)$ and $\mathbf{S}_{in}(k,m)$ so that $\mathbf{C}'_{in}(k,m) = \mathbf{U}_{in}(k,m)\mathbf{S}_{in}(k,m)\mathbf{U}^H_{in}(k,m)$.

Then, a matrix is formulated $\mathbf{K}_{in}(k,m) = \mathbf{U}_{in}(k,m)\mathbf{S}_{in,sq}(k,m)$ where $\mathbf{S}_{in,sq}(k,m)$ is a diagonal matrix with the diagonal entries that are the square root of $\mathbf{S}_{in}(k,m)$.

Then, $\mathbf{S}_{in,sq}(k,m)$ is modified using a control parameter called the regularization factor. This is done by modifying the entries of $\mathbf{S}_{in,sq}(k,m)$ so that its diagonal values are not smaller than the regularization factor f_{reg} times the maximum diagonal value of $\mathbf{S}_{in,sq}(k,m)$, resulting in a modified $\mathbf{S}_{in,sq}(k,m)$ which is used instead of the original $\mathbf{S}_{in,sq}(k,m)$ (the same term is used here for simplicity). The regularization factor f_{reg} is determined as presented in clause 7.2.2.2.7.

Then, a matrix $\mathbf{A}(k,m)$ is formulated by

$$\mathbf{A}(k,m) = \mathbf{K}^H_{out}(k,m)\widehat{\boldsymbol{G}}(k,m)\mathbf{Q}\mathbf{K}_{in}(k,m)$$

Where $\widehat{\boldsymbol{G}}(k,m)$ is a diagonal matrix where the diagonal entries are the diagonal entries of $\mathbf{C}'_{out}(k,m)$ divided by the diagonal entries of $\mathbf{C}'_{in}(k,m)$ and $\mathbf{Q} = \begin{bmatrix} 1 & 0.05 \\ 0.05 & 1 \end{bmatrix}$.

Then, a matrix $\mathbf{P}(k,m)$ is determined that is the nearest orthonormal matrix to $\mathbf{A}(k,m)$ by

$$\boldsymbol{P}(k,m) = \mathbf{A}(k,m)\left(\mathbf{A}^{H}(k,m)\mathbf{A}(k,m)\right)^{-\frac{1}{2}}$$

Then, a processing matrix is formulated by

$$\boldsymbol{M}(k,m) = \boldsymbol{K}_{out}(k,m)\boldsymbol{P}(k,m)\mathbf{S}_{in,sq}^{-1}(k,m)\boldsymbol{U}_{in}^{H}(k,m)$$

The effect of applying the processing matrix to the input signals is formulated by

$$\widehat{\boldsymbol{C}}'_{out}(k,m) = \boldsymbol{M}(k,m)\boldsymbol{C}'_{in}(k,m)\boldsymbol{M}^{H}(k,m)$$

In case k is smaller than the maximum decorrelation frequency bin K_{maxdec} (see clause 7.2.2.2.8), a residual covariance matrix $\boldsymbol{C}'_{res}(k,m)$ is then determined as follows, and otherwise $\boldsymbol{C}'_{res}(k,m) = \mathbf{0}$.

$$\boldsymbol{C}'_{res}(k,m) = \left(\boldsymbol{C}'_{out}(k,m) - \widehat{\boldsymbol{C}}'_{out}(k,m)\right) f_{dec}(k,m)$$

where $f_{dec}(k,m)$ is a decorrelation reduction factor, which is determined by the following rules:

- If the IVAS format is MASA_FORMAT, and the IVAS total bit rate is less than 24.4 kbps, then

$$f_{dec}(k,m) = 1$$

- Otherwise, if the IVAS format is MC_FORMAT or the IVAS format is MASA_FORMAT and the number of transport channels (prior creating dual-mono) is 1, then

$$f_{dec}(k,m) = \sqrt{\frac{E_{diff,dec}(k,m)}{E_{meanOut}(k,m)}}$$

- Otherwise, if the IVAS format is SBA_FORMAT or SBA_ISM_FORMAT, and the number of transport channels (prior creating dual-mono) is 1, then

$$f_{dec}(k,m) = 1$$

- Otherwise

$$f_{dec}(k,m) = \frac{E_{diff,dec}(k,m)}{E_{meanOut}(k,m)}$$

The residual covariance matrix $\boldsymbol{C}'_{res}(k,m)$ can be seen as a control parameter configured to control the amount of decorrelated audio within the two output audio signals for stereo or binaural audio reproduction. It is based on the input and target covariance matrices so that necessary amount of decorrelation is provided to the output audio signals to generate the spatial perception according to the spatial metadata, but it is further controlled by the spatial metadata to suppress the decorrelation when needed to minimise the decorrelation artefacts.

Then, a residual mixing matrix $\boldsymbol{M}_{res}(k,m)$ is formulated. It is formulated by the same steps as described in the foregoing for determining $\boldsymbol{M}(k,m)$, however, with the following differences: In place of the input covariance matrix $\boldsymbol{C}'_{in}(k,m)$, the same matrix but where all non-diagonal entries are zeroed is used; in place of $\boldsymbol{C}'_{out}(k,m)$ the matrix $\boldsymbol{C}'_{res}(k,m)$ is used; and the regularization factor is set to 0.2. If $\boldsymbol{C}'_{res}(k,m) = \mathbf{0}$ then $\boldsymbol{M}_{res}(k,m) = \mathbf{0}$.

Then, the missing energy due to potential regularizations is formulated by

$$\boldsymbol{E}_{miss}(k,m) = \max\left(0, tr\left(\boldsymbol{C}'_{out}(k,m)\right) - tr\left(\boldsymbol{C}'_{res}(k,m) + \widehat{\boldsymbol{C}}'_{out}(k,m)\right)\right)$$

And, an equalization gain to compensate for the missing energy is formulated by

$$g_{miss}(k,m) = \min\left(4, \sqrt{\frac{tr\left(\boldsymbol{C'}_{res}(k,m)\right) + \boldsymbol{E}_{miss}(k,m)}{tr\left(\boldsymbol{C'}_{res}(k,m)\right)}}\right)$$

Then, the processing matrix is gain-compensated by

$$\boldsymbol{M}'(k,m) = \boldsymbol{M}(k,m)g_{miss}(k,m)$$

Furthermore, if the stream contains a separate center channel (when operating in the multi-channel format) or if the stream contains separated object channels (when operating in the OMASA format), 2x1 processing gain vectors $\boldsymbol{m}_i'(k,m)$ are determined, where i is the separate channel/object index. These are formulated by first using the method in clause 7.2.2.2.6 to obtain gains $\mathbf{g}_{direct,sep,i}(k,m)$, and then processing them as $\boldsymbol{m}_i'(k,m) = \mathbf{g}_{direct,sep,i}(k,m)g_i$, where g_i is a gain value that is 0.7943 when the IVAS format is MASA_ISM_FORMAT and 0.8414 otherwise.

7.2.2.2.5 Processing audio signals with the processing matrices

The binaural (or stereo) audio output is produced by generating different binaural audio signal parts (direct input signal part, decorrelated input signal part, separated centre/object channel part, and room-effect part) and combining the different binaural audio signal parts to generate a combined binaural audio signal (which is the output of the processing)

$$\mathbf{s}_{\text{out}}(k,n) = \mathbf{s}_{\text{out,1}}(k,n) + \mathbf{s}_{\text{out,2}}(k,n) + \mathbf{s}_{\text{out,3}}(k,n) + \mathbf{s}_{\text{out,4}}(k,n)$$

where

$$\mathbf{s}_{\text{out,1}}(k,n) = (\boldsymbol{M}'(k,m)p(m,n) + \boldsymbol{M}'(k,m-1)(1-p(m,n)))\mathbf{s}(k,n)$$

$$\mathbf{s}_{\text{out,2}}(k,n) = (\boldsymbol{M}_{res}(k,m)p(m,n) + \boldsymbol{M}_{res}(k,m-1)(1-p(m,n)))\mathbf{s}_{\text{decorr}}(k,n)$$

$$\mathbf{s}_{\text{out,3}}(k,n) = \sum_i (\boldsymbol{m}'_i(k,m)p(m,n) + \boldsymbol{m}'_i(k,m-1)(1-p(m,n)))\mathbf{s}_{sep}(k,n)$$

where $\mathbf{s}(k,n)$ is the transport audio signals, $\mathbf{s}_{sep}(k,n)$ is the one or more separate channel/object signals (if there is none of them, it is set to zero), $\mathbf{s}_{\text{decorr}}(k,n)$ is a decorrelated version of $\mathbf{s}(k,n)$, the processing coefficients in $\boldsymbol{M}'$, $\boldsymbol{M}_{res}$ and $\boldsymbol{m}'_i$ are based on the spatial metadata as described in the foregoing clauses, and $p(m,n)$ is an interpolation value determined by

$$p(m,n) = \frac{(n - n_{first}(m) + 1)}{(n_{last}(m) - n_{first}(m) + 1)}$$

As an exception, $p(m,n) = 1$ when the IVAS format is OMASA, and the ISM mode is ISM_MASA_MODE_MASA_ONE_OBJ or ISM_MASA_MODE_PARAM_ONE_OBJ. Otherwise $p(m,n)$ is determined using the equation above.

The decorrelated audio signals $\mathbf{s}_{\text{decorr}}(k,n)$ are generated by processing the transport audio signal(s) $\mathbf{s}(k,n)$ as described in clause 7.2.2.2.8.

$\mathbf{s}_{\text{out,4}}(k,n)$ is generated by processing the input signals with a room effect when the room effect indication says to apply it. Applying the room effect is described in TODO: ADD REF TO REVERB. The reverberator is initialized based on the late reverberation energy correction gains and reverberation times obtained from TODO: ADD REF TO TABLE. When the room effect indication says to not apply a room effect, $\mathbf{s}_{\text{out,4}}(k,n)$ is set to zero.

7.2.2.2.6 Determining direct part gains

In the following, it is described how direct part gains are formulated for binaural or stereo output for a given direction. For determining the gains, the direction parameter is denoted azimuth θ and elevation φ, and the output gains are denoted as a 2x1 column vector $\mathbf{g}$. In this clause, the time and frequency indices, and most of the subscripts, are omitted for clarity of the equations.

First, the determining of stereo gains is described. In that operating mode, the azimuth and elevation are first mapped to a horizontal azimuth by

$$\theta_{map} = arctan2\left(y, \sqrt{1-y^2}\right)$$

where

$$y = \sin(\theta)\cos(\varphi)$$

When azimuth is between -$\theta_{ref} < \theta_{map} < \theta_{ref}$, where θ_{ref} is 30 degrees, then the panning gains are formulated by

$$a_1 = \frac{\tan\,(\theta_{map})}{\tan\,(\theta_{ref})}$$

$$a_2 = \frac{a_1 - 1}{\max\,(0.001, a_1 + 1)}$$

$$a_3 = \frac{1}{a_2^2 + 1}$$

$$\mathbf{g}(k,m) = \begin{bmatrix} \sqrt{a_3} \\ \sqrt{1-a_3} \end{bmatrix}$$

When $\theta_{map} \geq \theta_{ref}$, then $\mathbf{g} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$. When $\theta_{map} \leq -\theta_{ref}$, then $\mathbf{g} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$.

In case the operating mode is providing binaural output, the above gains are not used, but the following formulas are used instead.

If the head orientation is tracked, the direction parameters azimuth θ and elevation φ are converted to rotated direction parameters. The rotation is performed by converting the azimuth θ and elevation φ values to a 3x1 unit vector $\mathbf{u}_{\theta,\varphi}$ pointing towards the azimuth and elevation, and then rotating it with a 3x3 rotation matrix $\mathbf{R}$ TODO: ADD REF so that a rotated unit vector is $\mathbf{u}_{\theta,\varphi,rot} = \mathbf{R}\mathbf{u}_{\theta,\varphi}$. If head orientation data is not available or not used, then $\mathbf{u}_{\theta,\varphi,rot} = \mathbf{u}_{\theta,\varphi}$.

As a second step, third order spherical harmonic 16x1 gain vector $\mathbf{g}_{HOA3,\theta,\varphi,rot}$, which corresponds to the direction of $\mathbf{u}_{\theta,\varphi,rot}$, is determined using the method described in clause XXX. TODO: ADD REF. The binaural gains are then obtained by

$$\mathbf{g} = \boldsymbol{M}_{HOA3bin}\mathbf{g}_{HOA3,\theta,\varphi,rot}$$

where $\boldsymbol{M}_{HOA3bin}$ is a 2x16 spherical-harmonic-to-binaural processing matrix TODO: ADD REF TO WHERE THESE MATRICES ARE DEFINED. Note that $\boldsymbol{M}_{HOA3bin}$ is dependent on the frequency bin k. The determined gains $\mathbf{g}$ are effectively head-related transfer functions (HRTFs) and are typically complex-valued at least in low/mid frequencies. Since the same system is used for rendering binaural and stereo, the vector $\mathbf{g}$ values are not referred to as HRTFs but more generally as "gains".

In case the rendering output is binaural (i.e., not stereo), the gains $\mathbf{g}$ are further affected by the spread coherence value ζ associated with the current azimuth θ and elevation φ. If the output is stereo, or if $\zeta = 0$ the spread-coherence-related processing in the following is not applied.

A spread coherent sound refers to directional sound that, instead of being a point source, originates coherently from more than one direction. For example, amplitude panned sound wound constitute a "spread coherent" sound. The spread coherence is expressed by a spread coherence parameter ζ ranging from 0 to 1, where $\zeta = 0$ refers to a point-source, $\zeta = 0.5$ refers to three sources at 30 degrees spacing (i.e., spanning 60 degrees in total), and $\zeta = 1$ refers to two sources at 60-degree spanning.

First, the binaural (i.e., not stereo) gain is formulated for the given azimuth θ and elevation φ. The result is the centre gains $\mathbf{g}_{\mathrm{c}}$. Next, direct part gains are also formulated for directions $(\theta + 30°, \varphi)$ and $(\theta - 30°, \varphi)$. The results are the gains $\mathbf{g}_{\leftarrow}$ and $\mathbf{g}_{\rightarrow}$. The procedure of formulating these gains is as described in the foregoing, for the given angles with three different azimuth values.

Next, centre multiplier g_c and side-multiplier g_s are formulated as follows. If the spread coherence value $\zeta < 0.5$, then

$$g_s = \frac{2\zeta}{\sqrt{3}}$$

$$g_c = 1 - (2\zeta) + g_s$$

Otherwise, i.e., if $\zeta \geq 0.5$, then

$$g_{tmp} = 2 - (2\zeta)$$

$$g_s = \frac{1}{\sqrt{g_{tmp} + 2}}$$

$$g_c = g_{tmp} g_s$$

The combined spread-coherent binaural gains, prior to normalization, are formulated by

$$\mathbf{g}_{nonEq} = g_c \mathbf{g}_\mathrm{c} + g_s(\mathbf{g}_\leftarrow + \mathbf{g}_\rightarrow)$$

Then, an energy-correcting factor is formulated by

$$g_{Eq1} = \frac{\boldsymbol{g}_s^2(\mathbf{g}_\leftarrow^H \mathbf{g}_\leftarrow + \mathbf{g}_\rightarrow^H \mathbf{g}_\rightarrow) + \boldsymbol{g}_c^2 \mathbf{g}_c^H \mathbf{g}_c}{\mathbf{g}_{nonEq}^H \mathbf{g}_{nonEq}}$$

Then, a set of weights are determined which, when $\zeta < 0.5$, are

$$w_1 = 1 - 2\zeta$$

$$w_2 = 2\zeta$$

$$w_3 = 0$$

Otherwise, i.e., when $\zeta \geq 0.5$

$$w_1 = 0$$

$$w_2 = 2 - 2\zeta$$

$$w_3 = 2\zeta - 1$$

The second energy correction factor is $g_{Eq2} = g_{Eq1}$ for all other formats than the multi-channel format. With the multi-channel format, it is determined as follows: if the separate centre channel rendering is enabled, then

$$g_{Eq2} = g_{Eq1}\big(w_1 + (w_2 + w_3)\big)E_\zeta$$

and if the separate centre channel rendering is not enabled, then

$$g_{Eq2} = g_{Eq1}\big(w_1 + w_2 E_{\zeta 05} + w_3 E_{\zeta 1}\big)$$

where $E_{\zeta 05}$ and $E_{\zeta 1}$ are frequency-dependent spread coherence energy correcting factors. TODO: ADD REF TO TABLES.

The gains $\mathbf{g}_{nonEq}$ are then equalized with the energy correction factors to determine the binaural panning gains by

$$\mathbf{g} = \mathbf{g}_{nonEq} \min\big(4, \sqrt{g_{Eq2}}\big)$$

These are then the gain values for the azimuth θ and elevation φ, when the output is binaural and the spread coherence $\zeta > 0$.

7.2.2.2.7 Determining regularization factor

The regularization factor f_{reg} is a control parameter that is determined based on obtained input property parameters associated with the input spatial audio signal, namely the IVAS format and the bitrate. For the SBA, OSBA, OMASA, and ISM formats, the regularization factor is $f_{reg} = 1$.

For the MASA format, the regularization factor is determined based on the bitrate BR:

- if $BR \geq 160$ kbps, $f_{reg} = 0.4$
- else if $BR \geq 128$ kbps, $f_{reg} = 0.5$
- else if $BR \geq 96$ kbps, $f_{reg} = 0.6$
- else if $BR \geq 64$ kbps, $f_{reg} = 0.8$
- else, $f_{reg} = 1$

For the multi-channel format, the regularization factor is determined based on the bitrate BR:

- if $BR \geq 96$ kbps, $f_{reg} = 0.4$
- else if $BR \geq 80$ kbps, $f_{reg} = 0.5$
- else if $BR \geq 64$ kbps, $f_{reg} = 0.7$
- else if $BR \geq 48$ kbps, $f_{reg} = 0.8$
- else, $f_{reg} = 1$

7.2.2.2.8 Decorrelation in the parametric binauralizer

The parametric binauralizer uses two different decorrelators: the time-domain and the frequency domain decorrelators. The time-domain decorrelator is used in these cases:

- If the IVAS format is MASA, and the bitrate is smaller than 48 kbps in case of one transport audio signal or the bitrate is smaller than 24.4 kbps in case of two transport audio signals
- If the IVAS format is multi-channel, and the bitrate is smaller than 48 kbps
- If the IVAS format is SBA or OSBA, and the number of transport audio signals is one.

In other cases, the frequency-domain decorrelator is used.

The operation of the time-domain decorrelator is presented in clause XXX. TODO: ADD REF.

The operation of the frequency-domain decorrelator is presented in clause XXX. TODO: ADD REF.

The decorrelation is applied on all frequencies with the time-domain decorrelator (i.e., $K_{maxdec} = 60$). The decorrelation is applied on frequencies below 6 kHz with the frequency-domain decorrelator (i.e., $K_{maxdec} = 15$). No decorrelation is applied when the IVAS format is ISM_FORMAT (i.e., $K_{maxdec} = 0$).

7.3 Room acoustics rendering

7.3.1 Introduction to room acoustics rendering

IVAS rendering supports synthesis of room acoustics for realistic immersive effect. The room acoustics can be synthesized using room impulse response convolution or late reverb, optionally combined with early reflections. The room impulse response data (BRIRs) are discussed in section 7.4.6. The late reverb and early reflection synthesis are driven by the set of parameters that are discussed in details in section 7.4.7.

7.3.2 Room impulse response convolution

7.3.3 Sparse frequency-domain reverberator

7.3.4 Feedback-delay network reverberator

The feedback-delay network reverberator provided in IVAS decoder/renderer is based on the Jot reverberator. Additionally, filters have been added to control interaural correlation and ear-dependent coloration. A schematic depiction of the modified Jot reverberator is shown in Figure 7.3-1.

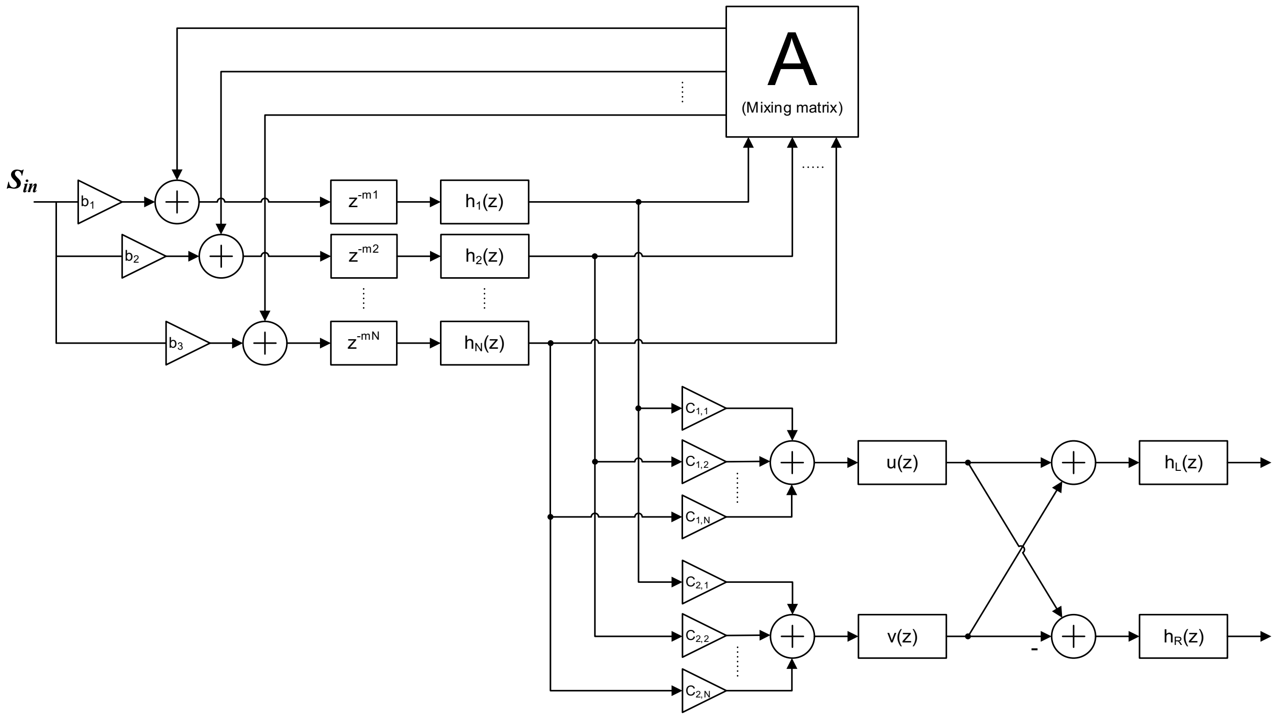


Figure 7.3-1: Feedback-delay network reverberator

The weights b and c control input and output of the feedback-delay network. Interaural coherence is controlled using $u(z)$ and $v(z)$ filter coefficients, while ear-dependent coloration using $h_L(z)$ and $h_R(z)$. To match with the direct path binaural filter characteristics, the coloration filters are pre-computed based on reverb characteristics, and on HRIR used for binauralization.

7.3.5 Early-reflection synthesis

The early reflections part of the room acoustics immersive effect can be enabled when using any multichannel input with binaural output modes. The reflection parameters described in section [xx] define a virtual 3D room with absorption coefficients representing the average broadband acoustic reflection characteristics of each surface in a geometric rectangular room model. The "shoebox" model computes first order early reflections for each source in the multichannel configuration using the image source method, described in [ref?]. This is done by considering the cartesian position of the listener probe and the relative position of the multichannel emitter array within the virtual room. The listener position defaults to the centre of the room (at a standard height) but it can optionally be defined in the renderer parameters to be in a different location than the centre. Once computed, the resulting reflection gains and delay times are used by a process loop to create the reflection signals, totalling to six reflections per source. Each reflection signal is then mixed into the channel buffer that is closest to the direction of arrival, according to the configuration layout. The resulting mix of early and direct sound is thus jointly sent to the orientation rotation processing creating an interactive directional sound effect.

7.3.5.1 Coordinate System

The virtual room is defined by three dimensions, respectively representing the length, width, and height of a rectangular room, in meter units. The resulting room model of dimensions $R(x, y, z)$ is placed at a centre of a 3D coordinate system where the origin is the centre of the room floor. As seen in the figure below, the room can extend in the positive and negative directions of the x and y axis, but only in the positive direction for the z axis. Each wall surface is referenced by an index following the order of the axis from positive to negative (e.g. $W{:}5$ and $W{:}6$ reference the surfaces along the z axis, in this case *ceiling* and *floor*), this permits the definition of independent absorption coefficients for each surface.

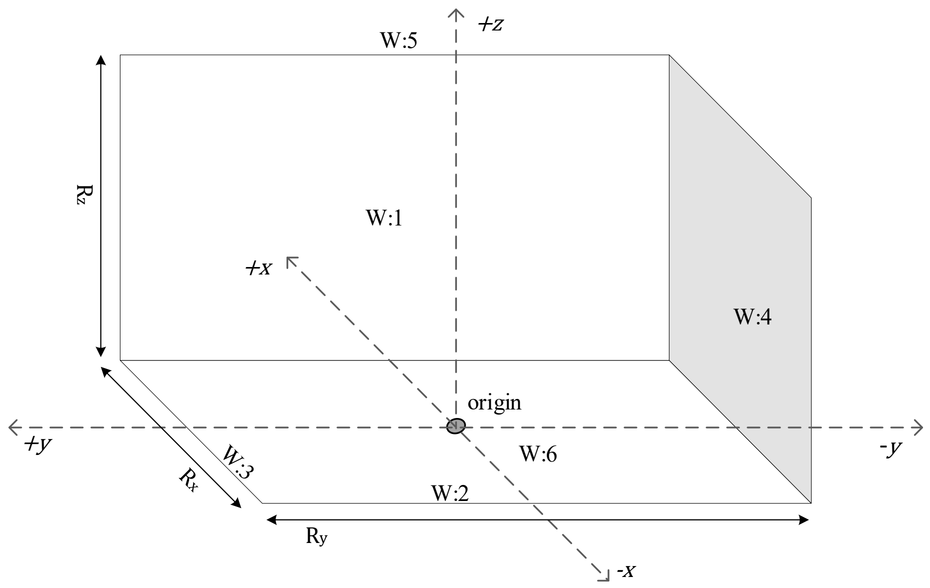


Figure 7.3-2 Coordinate system used by the Early Reflections model.

7.3.5.2 Source-receiver location correction

A boundary check executes at initialization stage to make sure the given source/receiver 3D coordinates at point $P(x, y, z)$ are within the virtual room boundaries. A minimum distance from each wall is defined internally to create enough space for reflections to happen with every surface. The boundary limit reference coordinates $B(x, y, z)$ are calculated using the room dimensions and the minimum distance:

$$\begin{bmatrix} B_x \\ B_y \\ B_z \end{bmatrix} = \begin{bmatrix} R_x/2 \\ R_y/2 \\ R_z \end{bmatrix} - w_d$$

Where R are the room dimensions and w_d is the minimum distance margin allowed for a point from a surface. Each source/receiver point is then checked against the boundary conditions, separately for each axis. Whenever the check fails, the reference coordinate is overwritten to be at the boundary, before feeding the shoebox model:

$$P_x = \begin{cases} \min(P_x, B_x) \text{ if } P_x \geq 0 \\ \max(P_x, -B_x) \text{ if } P_x < 0 \end{cases}, \quad P_y = \begin{cases} \min(P_y, B_y) \text{ if } P_y \geq 0 \\ \max(P_y, -B_y) \text{ if } P_y < 0 \end{cases}, \quad P_z = \begin{cases} \min(P_z, B_z) \text{ if } P_z \geq w_d \\ w_d \text{ if } P_z < w_d \end{cases}$$

The order of operation bounds firstly the listener point, then each source in the given configuration using relative channel positioning.

7.3.5.3 Reflections calculations

Each emitter source $E_{(n)}$ in the multichannel configuration (LFE channel excluded) possesses a set of six image sources representing reflections originating from every surface w in the rectangular model. For each image source I of index (n, w) the model calculates the broadband gain factor using the specified surface absorption rates α_w and the travel path distance, the time of arrival of the reflection to the receiver, and the direction of arrival in spherical coordinate form.

To calculate the reflection gains, the model first calculates the distance vector of each emitter point source E_n and image source $I_{(n,w)}$ to the listener location P in 3D space.

$$d_{(n)} = \begin{bmatrix} E_{(n)_x} \\ E_{(n)_y} \\ E_{(n)_x} \end{bmatrix} - \begin{bmatrix} P_x \\ P_y \\ P_z \end{bmatrix}, \quad v_{(n,w)} = \begin{bmatrix} I_{(n,w)_x} \\ I_{(n,w)_y} \\ I_{(n,w)_x} \end{bmatrix} - \begin{bmatrix} P_x \\ P_y \\ P_z \end{bmatrix}$$

The gain is taken as follows:

$$g_{(n,w)} = (1-\alpha_w)\left(\frac{\|d_{(n)}\|}{\|v_{(n,w)}\|}\right) - \left(\|v_{(n,w)}\| * \alpha_{AIR}\right)$$

Where α_{AIR} is the air absorption coefficient set to $1.37e^{-3}$, $\|v_{(n,w)}\|$ is the norm, or the Euclidean distance, from 3D image source position $I_{n,w}(x,y,z)$ to the receiver probe position $P(x,y,z)$, and $\|d_{(n)}\|$ is the distance from the emitter source $E_{(n)}(x,y,z)$ to the receiver.

The time of arrival for each reflection is calculated by using a standard *speed of sound* coefficient and the total travel path distance from the image source to the receiver probe.

$$TOA_{(n,w)}(s) = \frac{\|v_{(n,w)}\|}{c}$$

Where c is the speed of sound set as 343 m/s. The direction of arrival of each reflection is broken down into azimuth angle θ and vertical angle ϕ.

$$\theta_{(n,w)} = \operatorname{atan2}\left(\frac{v_{(n,w)_y}}{v_{(n,w)_x}}\right) \text{ rad}$$

$$\phi_{(n,w)} = \operatorname{asin}\left(\frac{v_{(n,w)_z}}{\|v_{(n,w)}\|}\right) \text{ rad}$$

7.3.5.4 Process Loop

7.3.5.5 Low-complexity mode

7.4 Rendering control

7.4.1 Rendering control overview

7.4.2 Head tracking

7.4.3 Orientation tracking

7.4.3.1 Orientation tracking introduction

In addition to supporting head rotation processing, the IVAS renderer supports a number of modes for listener orientation tracking. The listener orientation tracking refers to a set of methods used to provide or estimate listener's frontal orientation (being listener's head neutral position or torso position). Such a listener's frontal orientation is further referred to as reference orientation. The following orientation tracking modes are supported:

- External reference orientation,
- External reference vector orientation,
- External reference levelled vector orientation,
- Adaptive long-term average reference orientation.

The externally received or computed average orientation is eventually combined with the head rotation data. Such combined rotation is applied in the rendering to create the final rotated output. The relationship between head rotation and torso orientation is shown in Figure 7.4-1. Note that for the sake of readability, the view from the top is depicted. In actual implementations rotation in all dimensions is considered.

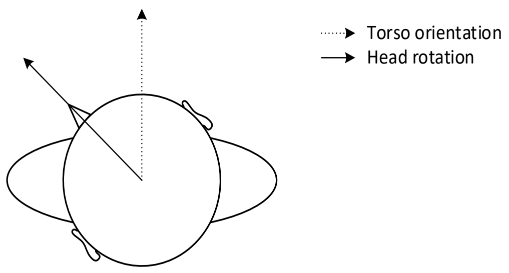


Figure 7.4-1: Head rotation vs torso orientation

7.4.3.2 External reference orientation

7.4.3.3 External reference vector orientation

Editor's Note: The coordinate system of IVAS needs to be properly referenced and aligned, i.e. $p_{ivasforward}$: needs definition elsewhere.

The input parameters to the reference vector orientation (identified as `HEAD_ORIENT_TRK_REF_VEC`) orientation tracking modes are:

- The absolute position of the listeners head in Cartesian 3D coordinates ($p_{listenerabs}$).
- The absolute position of an acoustic reference (p_{refabs}). In case of camera-based head tracking by a UE, where the UE should act as the acoustic reference direction, this would be the position of the phone in Cartesian 3D coordinates.
- The absolute head orientation of the listener ($r_{listenerabs}$).

The position of the listener and the reference must refer to a common coordinate system, e.g., an Earth-fixed coordinate system.

Using the positions of the listeners head and the reference, a vector spanning from the listener's head to the reference position is determined ($p_{acousticfront}$).

$$p_{acousticfront} = p_{listenerabs} - p_{refabs} \, . \qquad (7.4\text{-}31)$$

The absolute head orientation of the user must be transformed such that the resulting head orientation (r_{result}) causes the binaural rendering step, using this processed head orientation, will produce an audio signal, where an audio object with 0-degree azimuth and 0-degree elevation would get rendered in the direction of the reference position. Therefore this vector from the listeners head to the reference acts as the acoustic front of the system in `HEAD_ORIENT_TRK_REF_VEC` mode.

This shall be be implemented by normalizing $p_{acousticfront}$ and $p_{ivasforward}$:

$$p_{acousticfront\_norm} = \frac{p_{acousticfront}}{|p_{acousticfront}|} \qquad (7.4\text{-}31)$$

$$p_{ivasforward\_norm} = \frac{p_{ivasforward}}{|p_{ivasforward}|} \qquad (7.4\text{-}31)$$

And then building the cross product and the dot product of both vectors:

$$cross\_product = p_{acousticfront\_norm} \times p_{ivasforward\_norm} \qquad (7.4\text{-}31)$$

$$dot\_product = p_{acousticfront\_norm} \cdot p_{ivasforward\_norm} \qquad (7.4\text{-}31)$$

When the dot product is close to -1 (i.e., $dot\_product < -0.999999$), the two vectors are almost directly opposite to each other. In such cases, the cross product becomes near-zero, leading to an ambiguous rotational axis. To handle this, a predefined quaternion representing a 180-degree rotation is used:

$$r_{refrot} = \begin{bmatrix} 0.0 \\ 0.0 \\ 0.0 \\ 1.0 \end{bmatrix} \qquad (7.4\text{-}31)$$

Otherwise, the reference rotation quaternion (r_{refrot}) is calculated as:

$$r_{refrot} = \begin{bmatrix} \sqrt{\left(\left|p_{acousticfront\_norm}\right|^2\right)\left(\left|p_{ivasforward\_norm}\right|^2\right)} + \text{dot\_product} \\ cross\_product_x \\ cross\_product_y \\ cross\_product_z \end{bmatrix} \quad (7.4\text{-}31)$$

The r_{refrot}.w part of the quaternion represents the combined magnitudes of the vectors and their dot product. Since the vectors are normalized, the magnitudes are both 1, resulting in:

$$r_{refrot}.w = 1 + \text{dot\_product} \quad (7.4\text{-}31)$$

The resulting Quaternion is normalized:

$$r_{refrot\_norm} = \frac{r_{refrot}}{|r_{refrot}|} \quad (7.4\text{-}31)$$

The inverse of the resulting normalized reference orientation (r_{refrot_norm}) shall then be used to rotate the absolute listener orientation:

$$r_{result} = {r_{refrot\_norm}}^{-1} \otimes r_{listenerabs} \quad (7.4\text{-}31)$$

7.4.3.4 External reference levelled vector orientation

In the reference vector level mode (identified by `HEAD_ORIENT_TRK_REF_VEC_LEV`) orientation tracking mode, the processing is identical to reference vector mode (identified by `HEAD_ORIENT_TRK_REF_VEC`) as specified in clause 7.4.3.3, except that the difference in height between the head of the user and the reference is ignored. This is achieved by aligning the up/down axis input values of listener and reference position to the same value. Therefore, the $p_{acousticfront}$ vector in this orientation tracking mode is defined as:

$$p_{acousticfront} = \begin{bmatrix} p_{listenerabs_x} \\ p_{listenerabs_y} \\ p_{refabs_z} \end{bmatrix} - \begin{bmatrix} p_{refabs_x} \\ p_{refabs_y} \\ p_{refabs_z} \end{bmatrix}. \quad (7.4\text{-}31)$$

7.4.3.5 Adaptive long-term average reference orientation

In many cases no external reference for pose can be provided. Hence, long-term average orientation can be used as a reference. This can be especially efficient to compensate for involuntary rotation, such as might occur while a user is travelling (e.g., taking a turn on a train, while walking, etc.). As a result, a user perceives that the virtual sound sources stay consistent in the scene in relation to the frontal (torso) orientation, also in the case of involuntary rotation.

A general illustration of such adaptation of the reference (torso) orientation is provided in Figure 7.4-2. For the sake of readability, the view from the top is depicted. Also, change of reference orientation in limited to horizontal plane, while in the implementation all dimensions are considered.

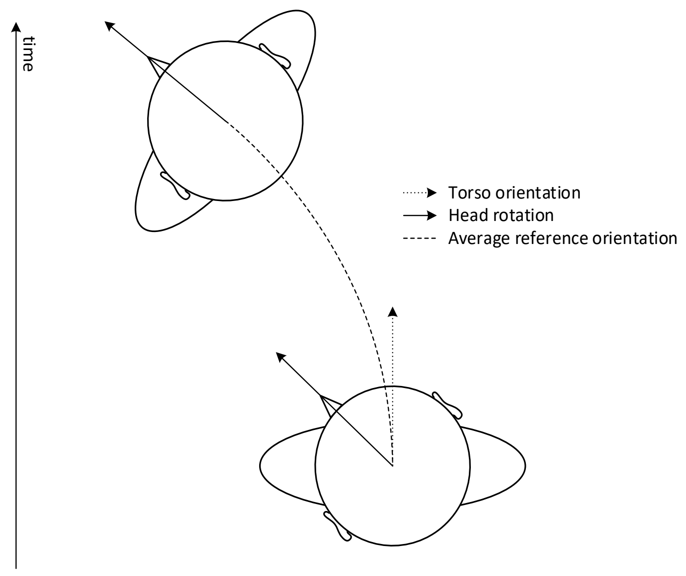


Figure 7.4-2: Adaptive long-term average reference orientation in a function of time

The head rotation relative to the reference $r_{rel}[m]$ for frame m is computed by applying high-pass filter to the input head rotation $r_{in}[m]$. This is achieved by subtracting long term average rotation from the input head rotation:

$$Rot_{rel}[m] = Rot_{in}[m] - Rot_{avg}[m] \tag{7.4-1}$$

where the long-term average orientation is obtained by low pass filtering of the input head rotations $Rot_{in}(t)$. Adaptive averaging is applied to adjust to larger rotation deviations faster than for small ones, making the averaging algorithm less sensitive to small, quick head movements. Such adaptive average rotation is computed by applying an adaptive IIR lowpass filter:

$$Rot_{avg}(m) = \alpha Rot_{in}[m] + (1-\alpha)Rot_{avg}[m-1] \tag{7.4-2}$$

where the adaptive parameter α:

$$\alpha = \sin\left(2\pi \frac{f_c}{f_s}\right) \tag{7.4-3}$$

with the cutoff frequency f_c is linearly interpolated between its minimum and maximum values:

$$f_c = f_{c,min} + RDR\left(f_{c,max} - f_{c,min}\right) \tag{7.4-4}$$

and using relative direction ratio RDR:

$$RDR = \min\left(\frac{|Rot_{rel}[m-1]|}{Rot_{max}}, 1\right) \tag{7.4-5}$$

And f_s equals the audio processing sample rate.

The threshold values of adaptation parameters are set as follows:

$f_{c,min} = \frac{1}{30} Hz,$

$f_{c,max} = \frac{1}{8} Hz,$

$r_{max} = 90°.$

The processing is performed in the quaternion domain. Firstly, the adaptive average rotation quaternion $Rot_{avg}[m]$ is computed using spherical linear interpolation between input rotation and previous average rotation quaternion:

$$Rot_{avg}[m] = \frac{Rot_{in}[m]\sin(\alpha\varphi) + Rot_{avg}[m-1]\sin\big((1-\alpha)\varphi\big)}{sin(\varphi)} \quad (7.4\text{-}6)$$

The adaptive average rotation quaternion Rot_{avg} is then normalized. The resulting relative rotation quaternion Rot_{rel} is computed as:

$$Rot_{rel}[m] = \frac{1}{\widehat{Rot}_{avg[m]}} \times Rot_{in}[m] \quad (7.4\text{-}7)$$

Where the $\times$ operator denotes quaternion product. The adaptive parameter α is eventually updated as shown in (7.4-3). The relative rotation quaternion $Rot_{rel}[m]$ is eventually used as the control input for the binaural renderer.

7.4.4 External orientation input handling

7.4.4.1 Overview

The external orientation input provides to the IVAS renderer any orientation information separate from the head orientation (rotation) data of the listener. The external orientation data, when available, is therefore processed and applied in addition to the head-tracking data, which is described in clause 7.4.2.

For example, a capture device (e.g., sending UE) captures or otherwise obtains one or more audio signals part of a spatial audio scene. In addition, the sending UE may determine or otherwise obtain orientation information that relates to this spatial audio scene. The spatial audio scene, represented by one or more audio signals, is encoded by the IVAS encoder according to input format specific operation, as described in clause 5 (e.g., according to clause 5.4 for SBA operation or clause 5.5 for MASA format operation), and transmitted to the receiving UE (e.g., using IVAS RTP payload, defined in Annex A). The UE receives the transmitted IVAS stream comprising the encoded audio signal(s) part of the spatial audio scene. Additionally, the receiver may obtain the encoded orientation information which is associated with the capture device (e.g., sending UE). The receiver then decodes the encoded audio signal(s), as described in clause 6, and the encoded orientation information. The orientation information may comprise information associated with, e.g., a default scene orientation, orientation of the capture device (e.g., sending UE), and orientation compensation. This decoded orientation information is then provided to the IVAS renderer as external orientation input data, and the renderer uses it to process the audio signal(s) as part of the rendering.

The external orientation information may, e.g., include (intended or default) scene orientations, device orientations, orientation compensations, orientations of a user operating a capture device (e.g., a sending UE), or other such reference orientations that may be used to guide or modify the rendering processing. The (intended or default) scene orientations may comprise the orientation of the scene at the sending UE or the local scene orientation at the receiving UE. A default scene orientation or an intended scene orientation may comprise, e.g., an orientation determined by the sending UE, which is transmitted as a guidance for rendering orientation of the decoded audio scene. The scene orientation data is thus intended to modify the rendering orientation of the spatial audio. The orientation compensation may comprise, e.g., orientation data for removing from, or adding to, rendering the orientation (rotational) information of the capture device (e.g., a sending UE). A reference orientation, related to a sender UE, e.g., may also be used to define a scene orientation. A reference orientation may also identify a global orientation reference. The device orientations may comprise the orientation of the capture device (on send side) or the orientation of the playback device (on receive side).

The external orientation input to the IVAS renderer may be transmitted to the receiving UE or, e.g., determined at the receiving UE. Concerning the rendering, the external orientation input thus provides augmentation control parameters that are configured to control in part the rendering of the spatial audio scene by the renderer obtaining the corresponding spatial audio signals and augmentation control parameters. Typically, the rendering is in part controlled also by the head-tracking data. The same data can also be made available to (e.g., by transmitting, storing, or relaying through the external renderer interface), and applied at, any external renderer in addition to the internal IVAS renderer and the external IVAS renderer. If there exist multiple external orientations that are to be applied at the renderer, the multiple external orientations should be combined into a single external orientation to provide the external orientation data to the IVAS renderer.

7.4.4.2 Processing of the external orientation data

The external orientation data comprises of the external orientation information and the external orientation control information. The external orientation information (w, x, y, z) shall be provided as quaternions. The external orientation control information may comprise of signalling (e.g., a flag, a switch) to enable, disable, or freeze the application of the external orientation information and/or head orientation (rotation). Additionally, the external orientation control information may include activation signalling and time for indicating that the external orientation information shall be applied at a current frame or at a future frame.

The control signalling is indicated with $Flag_{head}(m)$ for head rotations and with $Flag_{ext}(m)$ for external rotations. A flag value of 0 disables the respective rotation in the rendering process, a flag value of 1 enables the rotation. A flag value of 2 freezes the respective rotation to a last rendered rotation. Based on the control flags, a rendering rotation of the head and/or external rotation is accordingly modified and then applied in the rendering of the spatial audio. If a rotation flag is not present in the external orientation control information, the associated rotation shall be enabled by default (i.e., as if the flag would have a value of 1). These are explained in more detail in clause xxx.

For future frame appliance indication, the receiver may receive an orientation change data set as part of the external orientation data. The data set comprises orientation change value specifying a change to an orientation (e.g., a change to an orientation of a capture device with respect of the spatial audio scene comprising the audio signal) and an orientation change delay time for the change to the orientation (e.g., to the device orientation). In the external orientation data, the future appliance of external rotation is signalled with $Flag_{ext,intrp}(m)$, where a value of 1 indicates that the external rotation is changed after a period of time specified by an external orientation change delay time $N_{frames,intrp}(m)$. A value of 0 for $Flag_{ext,intrp}(m)$ indicates that the future appliance of external rotation is disabled, and that the external rotation should be applied immediately. The orientation change delay time is signalled or expressed with $N_{frames,intrp}(m)$ which indicates the number or units of audio frames (20 ms) of the audio signal after the associated external rotation is applied. During the transition period $N_{frames,intrp}(m)$ the external rotation is interpolated from the previously applied external rotation to the target external rotation. If the future appliance flag is not present in the external orientation control information, the future appliance of external rotation shall be disabled by default (i.e., as if the future appliance flag would have a value of 0). If the orientation change delay time is not present in the external orientation control information, the change delay time shall be set to zero (0). The interpolation process is explained in more detail in clauses xxx and xx.

7.4.5 Combined rotations for rendering

7.4.5.1 Combining head and external rotations

A received external orientation $O_{ext}(m)$ is combined with head-tracking data (head rotation $Rot_{head}(m)$) in a rotation combiner to determine the rotation for rendering. Both the external orientation and the head orientation (rotation) are processed as quaternions in the rotation combiner. Before the combination process, the received external orientation $O_{ext}(m)$ shall be inverted to operate as a rotation, $Rot_{ext}(m) = {O_{ext}}^{-1}(m)$.

In case there is no external rotation present, or the external rotation signalling flag disables the external rotation ($Flag_{ext}(m) == 0$), and there is head rotation present, the combined rotation $Rot_{com}(m)$ shall consist of the head rotation only, i.e., $Rot_{com}(m) = Rot_{head}(m)$.

If the external orientation control information indicates freezing of the head rotation ($Flag_{head}(m) == 2$) and the freezing indicator is the first in a series (i.e., $Flag_{head,frozen} == 0$), the head rotation shall be frozen to the received head rotation value $Rot_{head,frozen} = Rot_{head}(m)$, the indicator flag for frozen head rotation shall be activated as $Flag_{head,frozen} = 1$ and the frozen head rotation value shall be used in the processing, $Rot_{head,applied}(m) = Rot_{head,frozen}$. In the subsequent processing subframes $m + d, d \in \mathbb{N}$ with $Flag_{head}(m + d) == 2$ and $Flag_{head,frozen} == 1$, the frozen head rotation value shall be used as the head rotation in the processing, $Rot_{head,applied}(m + d) = Rot_{head,frozen}$. If a head rotation signalling flag of $Flag_{head}(m) \neq 2$ is noticed, the frozen head rotation indicator flag shall be deactivated as $Flag_{head,frozen} = 0$ and the frozen head rotation shall be set to identity as $Rot_{head,frozen} = (1, 0, 0, 0)$.

If the external orientation control information indicates disabling of the head rotation ($Flag_{head}(m) == 0$), the head rotation shall be disabled for the processing. If the external orientation control information indicates enabling of the head rotation ($Flag_{head}(m) == 1$) or the head rotation signalling flag is missing, the received head rotation shall be used in the processing, $Rot_{head,applied}(m) = Rot_{head}(m)$.

If the external orientation control information indicates freezing of the external rotation $(Flag_{ext}(m) == 2)$ and the freezing indicator is the first in a series (i.e., $Flag_{ext,frozen} == 0$), and the future frame application is disabled $\left(Flag_{ext,intrp}(m) == 0\right)$, the external rotation shall be frozen to the received external rotation value $Rot_{ext,frozen} = Rot_{ext}(m)$, the indicator flag for frozen external rotation shall be activated as $Flag_{ext,frozen} = 1$ and the frozen external rotation value shall be used in the processing, $Rot_{ext,applied}(m) = Rot_{ext,frozen}$.. In the subsequent processing subframes $m + d, d \in \mathbb{N}$ with $Flag_{ext}(m + d) == 2$ and $Flag_{ext,frozen} == 1$, the frozen external rotation value shall be used as the external rotation in the processing, $Rot_{ext,applied}(m + d) = Rot_{ext,frozen}$. If an external rotation signalling flag of $Flag_{ext}(m) \neq 2$ is noticed, the frozen external rotation indicator flag shall be deactivated as $Flag_{ext,frozen} = 0$ and the frozen external rotation shall be set to identity as $Rot_{ext,frozen} = (1, 0, 0, 0)$.

If the external orientation control information indicates disabling of the external rotation $(Flag_{ext}(m) == 0)$, the external rotation shall be disabled for the processing. If the external orientation control information indicates enabling of the external rotation $(Flag_{ext}(m) == 1)$ or the external orientation signalling flag is missing, the received external rotation shall be used in the processing, $Rot_{ext,applied}(m) = Rot_{ext}(m)$.

If the external orientation future frame application is enabled $\left(Flag_{ext,intrp}(m) == 1\right)$, the external rotation is processed as explained in clause xxx.

If both the head rotation and the external rotation are present and indicated to be used in the processing, the rotations $Rot_{head,applied}(m)$ and $Rot_{ext,applied}(m)$ shall be combined into a single rotation before applying them in the processing. The combination of the two rotations are given as:

$$Rot_{com}(m) = Rot_{ext,applied}(m) \times Rot_{head,applied}(m) \qquad (7.4\text{-}8)$$

where the $\times$ operator denotes quaternion product.

In case there is no head rotation present or the head rotation signalling flag disables the head rotation $(Flag_{head}(m) == 0)$ and there is external rotation present and the external rotation is enabled $(Flag_{ext}(m) == 1 \; or \; Flag_{ext}(m) == 2)$, the combined rotation shall consist of the external rotation only, $Rot_{com}(m) = Rot_{ext,applied}(m)$.

The combined rotation $Rot_{com}(m)$ in quaternions is transformed into a rotation matrix $R_{com}(m)$ as:

QuatToRotMat() equation HERE (also used in TDREND_Update_listener_orientation())

If there are no head rotation and no external orientation present in a subframe, the combined rotation variables shall be set to their initial values as presented in Table 7.4-1.

$Flag_{com}(m)$ indicates if the combined rotation matrix $R_{com}(m)$ is enabled or disabled for the processing. A value of $Flag_{com}(m) == 1$ indicates that $R_{com}(m)$ shall be used in the processing and a value of $Flag_{com}(m) == 0$ indicates that $R_{com}(m)$ shall not be used in the processing. If there is head rotation present and no external rotation present for a subframe m, the $Flag_{com}(m)$ shall be set to 1. If there is external rotation present and no head rotation present for a subframe m and the external orientation control information indicates enabling the external rotation $(Flag_{ext}(m) == 1$ or $Flag_{ext}(m) == 2)$, the $Flag_{com}(m)$ shall be set to 1. If there is both head rotation and external rotation present for a subframe m and the external orientation control information indicates enabling either one or both of the rotations $((Flag_{ext}(m) == 1$ or $Flag_{ext}(m) == 2)$ and/or $(Flag_{head}(m) == 1$ or $Flag_{head}(m) == 2))$, the $Flag_{com}(m)$ shall be set to 1. Otherwise, the $Flag_{com}(m)$ shall be set to 0. $Flag_{com}(m)$ is used in the various rendering processes to check if the combined rotations shall be applied or not.

The applied listener position for subframe m $\left(Pos_{listener,applied}(m)\right)$ is also controlled in the rotation combiner. If head tracking data is available for subframe m, the applied listener position shall be set as the received listener position, $Pos_{listener,applied}(m) = Pos_{listener}(m)$. If both head tracking and external rotation data is not available for subframe m, the applied listener position shall be set to origo, $Pos_{listener,applied}(m) = (0, 0, 0)$. The listener position is processed as (x, y, z) coordinates.

7.4.5.2 External rotation interpolation

If the external orientation future frame application is enabled $\left(Flag_{ext,intrp}(m) == 1\right)$, the received external rotation $Rot_{ext}(m)$ should be applied in the future. The time of appliance is indicated by $N_{frames,intrp}(m)$ which indicates that after $N_{frames,intrp}(m)$ frames (20 ms) the received external rotation $Rot_{ext}(m)$ is reached. The time of appliance in subframes is $4 * N_{frames,intrp}(m)$. During the transition time $\left[m, m + 4 * N_{frames,intrp}(m)\right]$, the external rotation is interpolated from a start rotation $Rot_{ext,start}$ to a target rotation $Rot_{ext,target}$. The start rotation depends on the state of the rendering process. The target rotation is set as the received external rotation at the beginning of the interpolation process $Rot_{ext,target} = Rot_{ext}(m)$.

The progress of an interpolation process is observed with an interpolation increment $Intrp_{incr}$, an interpolation coefficient $Intrp_{coeff}(m)$ and an interpolation progress flag $Flag_{intrp,ongoing}$. The interpolation increment is a linear increment based on to the ratio of the external orientation change delay time $N_{frames,intrp}(m)$, calculated as:

$$Intrp_{incr} = \frac{1}{4 * N_{frames,intrp}(m)} \tag{7.4-9}$$

At the beginning of a new interpolation process, the interpolation coefficient is set as $Intrp_{coeff}(m) = Intrp_{incr}$ and the interpolation progress flag is activated as $Flag_{intrp,ongoing} = 1$. After each interpolation step, the interpolation coefficient is incremented as $Intrp_{coeff}(m + 1) = Intrp_{coeff}(m) + Intrp_{incr}$ for the next subframe.

At each subframe m with the future frame application enabled $\left(Flag_{ext,intrp}(m) == 1\right)$, a check for a current ongoing external rotation interpolation is performed. If the interpolation progress flag is active ($Flag_{intrp,ongoing} == 1$), the interpolation coefficient $Intrp_{coeff}(m) \leq 1.0$ and the set target rotation is approximately the same as the received external rotation at the current subframe $Rot_{ext,target} \approx Rot_{ext}(m)$, a current interpolation process is continued. The rotations $Rot_{ext,start}$ and $Rot_{ext,target}$ are interpolated (as in Equation (7.4-6)):

$$Rot_{ext,intrp}(m) = \frac{Rot_{ext,start} * \sin\left(\left(1 - Intrp_{coeff}(m)\right) * \varphi\right) + Rot_{ext,target} * \sin\left(Intrp_{coeff}(m) * \varphi\right)}{sin(\varphi)} \tag{7.4-10}$$

Where φ is the angle between the start and target rotations given as:

$$\varphi = \arccos\left(Rot_{ext,start} \cdot Rot_{ext,target}\right) \tag{7.4-11}$$

Where the $(\cdot)$ operator denotes quaternion dot product. The interpolated rotation is normalized before applying the rotation in the processing:

$$Rot_{ext,applied}(m) = \widehat{Rot}_{ext,intrp}(m) \tag{7.4-12}$$

In the interpolation process, an increment of rotation change is applied to the starting rotation by audio subframe basis for the period of time specified by the external orientation change delay time $N_{frames,intrp}(m)$. The increment of rotation change is determined linearly as the rotation of a ratio of the rotation change to the external change orientation delay time. The rotation change is the change in rotation from the starting rotation to the target rotation. I.e., the interpolation coefficient $Intrp_{coeff}(m)$ determines how many increments of rotation change are applied to the starting rotation $Rot_{ext,start}$ to determine the interpolated rotation for a subframe m.

After the interpolation step, the interpolation coefficient is incremented as $Intrp_{coeff}(m + 1) = Intrp_{coeff}(m) + Intrp_{incr}$. Two rotations are approximated to be the same if the following conditions are all fulfilled:

$$Rot_1 \approx Rot_2, \quad if \begin{cases} |w_1 - w_2| \leq 0.05 \\ |x_1 - x_2| \leq 0.05 \\ |y_1 - y_2| \leq 0.05 \\ |z_1 - z_2| \leq 0.05 \end{cases} \tag{7.4-13}$$

where (w_1, x_1, y_1, z_1) represents the first rotation Rot_1 in quaternions and (w_2, x_2, y_2, z_2) represents the second rotation Rot_2 in quaternions.

If a current interpolation is not in progress $\left(Flag_{intrp,ongoing} == 0\right)$ or the current interpolation process is completed $\left(Intrp_{coeff}(m) > 1.0\right)$ or a new external target rotation is set $\left(Rot_{ext,target} \not\approx Rot_{ext}(m)\right)$, the interpolation variables are set as $Flag_{intrp,ongoing} = 0$, $Intrp_{coeff}(m) = 1.0$ and $Intrp_{incr} = 1.0$ and a new interpolation process can be started.

At the beginning of the interpolation process, the time of appliance $N_{frames,intrp}(m)$ is forced to a range of $[0, 500]$ frames (20 ms) (i.e., to a range of $[0, 4 * 500]$ subframes (5 ms)). For $N_{frames,intrp}(m) < 0$ the time of appliance is set to zero (0), and for $N_{frames,intrp}(m) > 500$ the time of appliance is set to 500. If $N_{frames,intrp}(m) == 0$, the target rotation is applied immediately $\left(Rot_{ext,applied} = Rot_{ext,target}\right)$, the interpolation process is skipped and the interpolation variables are set as $Flag_{intrp,ongoing} = 0$, $Intrp_{coeff}(m) = 1.0$ and $Intrp_{incr} = 1.0$.

If $N_{frames,intrp}(m) > 0$, a new interpolation process is started. If the current target rotation and the received external rotation are not approximately the same $\left(Rot_{ext,target} \not\approx Rot_{ext}(m)\right)$, new interpolation values are calculated and set. The target rotation is set as the received external rotation, $Rot_{ext,target} = Rot_{ext}(m)$. The starting rotation depends on the codec state. If the external rotation was disabled in the previous subframe, the starting rotation shall be set to identity, i.e., $Rot_{ext,start} = (1, 0, 0, 0), if\ Flag_{ext,intrp}(m-1) == 0$. If the external rotation was frozen in the previous subframe $\left(Flag_{ext,intrp}(m-1) == 2\right)$ or the current external rotation is frozen $\left(Flag_{ext,intrp}(m) == 2\right)$, the starting rotation shall be set to the frozen external rotation, $Rot_{ext,start} = Rot_{ext,frozen}$. Otherwise, the starting rotation shall be set to the external rotation from the previous subframe, $Rot_{ext,start} = Rot_{ext}(m-1)$. The interpolation increment shall be calculated as in equation (7.4-9) and the interpolation coefficient shall be set as $Intrp_{coeff}(m) = Intrp_{incr}$. The interpolation progress flag shall be set as $Flag_{intrp,ongoing} == 1$. The rotations $Rot_{ext,start}$ and $Rot_{ext,target}$ shall be interpolated as in equations (7.4-10), (7.4-11) and (7.4-12) and the interpolated rotation shall be used as the applied external rotation $Rot_{ext,applied}(m)$ for the current subframe. The interpolation coefficient shall be incremented as $Intrp_{coeff}(m+1) = Intrp_{coeff}(m) + Intrp_{incr}$.

If in the beginning of a new interpolation process the current target rotation and the received external rotation are approximately the same $\left(Rot_{ext,target} \approx Rot_{ext}(m)\right)$, the calculations for the interpolation values are skipped and the old values are used. The interpolation progress flag shall be set as $Flag_{intrp,ongoing} == 1$. The rotations $Rot_{ext,start}$ and $Rot_{ext,target}$ shall be interpolated as in equations (7.4-10), (7.4-11) and (7.4-12) and the interpolated rotation shall be used as the applied external rotation $Rot_{ext,applied}(m)$ for the current subframe. The interpolation coefficient shall be incremented as $Intrp_{coeff}(m+1) = Intrp_{coeff}(m) + Intrp_{incr}$.

7.4.5.3 Initial values for combined rotation variables

The variables in the rotator combiner shall be initialized as given in Table 7.4-1.

Table 7.4-1:

Variable	Initial value
$Intrp_{coeff}$	1.0
$Intrp_{incr}$	1.0
$Flag_{intrp,ongoing}$	0
$Rot_{ext,start}$	(1, 0, 0, 0)
$Rot_{ext,target}$	(1, 0, 0, 0)
Rot_{com}	(1, 0, 0, 0)
R_{com}	Identity
$Flag_{com}$	0
$Pos_{listener,applied}$	(0, 0, 0)

7.4.6 HRTF and BRIR sets

[Begin with text extracted from S4-231233]

7.4.6.1 HRTFs and BRIR conversion methods

7.4.6.1.1 Conversion from spatial domain to spherical harmonics domain

The conversion from a spatial-domain head-related impulse response (SD HRIR) set to a spherical-harmonics-domain head-related impose response (SHD HRIR) set consists of 4 main steps:

1. Determine the bulk delays of the HRIRs in the SD HRIR set and compute a delay-less HRIR set from the SD HRIR set by removing the bulk delay computed previously.
2. Determine a set of all-pass filters that are dependent on the previously calculated bulk-delays.
3. Apply the all-pass filters to the delay-less HRTFs.

4. Form the SHD HRIR set by linearly combining the now delay-less sampled non-sparse HRTF set.

These main steps can be further elaborated into the following intermediate steps.

Step 1 – Determine bulk delay and compute delay-less HRIRs

Interpolate the existing HRIRs onto a dense sphere of known positions.

$$H_{interp}(k) = interp\big(H_{SD}(n)\big) \tag{7.4-10}$$

Where $H_{interp}(k)$ is the k dense positions interpolated HRIRs and $H_{SD}(n)$ is the n original HRIRs.

Compute the frequency domain representation of the interpolated HRIRs using an MDFT.

$$F = MDFT(H_{interp}) \tag{7.4-11}$$

Where F is the complex frequency responses of the HRIRs.

Find the equivalent minimum phase versions of the frequency responses (FR), using the Hilbert transform.

$$F_{minphase} = mag2minphase(F) \tag{7.4-12}$$

Calculate the excess phase by subtracting the min phase FR from the original FR.

$$ExcessPhase = angle(F) - angle(F_{minphase}) \tag{7.4-13}$$

Where $ExcessPhase$ is a matrix of $k \times nBins$ dimensions.

Find the interaural time delay (ITD) by looking at the excess phase at 1.2 kHz.

$$ITD = ExcessPhase(1\,..\,k,\ i) \tag{7.4-14}$$

Where ITD is the k number of delays and i is the index of the 1.2 kHz bin.

Find the max delay from all of the ITDs.

$$MaxDel = \max\,(ITD) \tag{7.4-15}$$

Create delays for both ears using the ITDs and max delay to ensure causality.

Step 2 – Determine a set of all-pass filters

Create the desired phase response by using the delays calculated in the previous step and pre-defined base polynomials. Then map the phase response polynomials to filter poles in the S domain and take the bilinear transform of the poles to convert into z-domain poles.

$$P_Z = bilinear(P_S) \tag{7.4-16}$$

Map the filter poles to IRs.

$$AP_{IR} = filter(\delta(x),\ P_Z) \tag{7.4-17}$$

Where AP_{IR} is the impulse response all-pass filters, $\delta(x)$ is an impulse and P_Z is the z-domain poles.

Step 3 – Apply the all-pass filters

Compute the average magnitude resonse using the mean of the L ear and R ear FRs

$$M_{ave} = mean(abs(F_{minphase})) \tag{7.4-18}$$

Where M_{ave} is the average magnitude response.

Convolve the all-pass filters with the averaged magnitude response.

$$F_{phasecomp} = M_{ave} \times AP \qquad (7.4\text{-}19)$$

Find the equivalent min phase version of the convolved FRs.

$$F_{mpc} = mag2minphase(F_{phasecomp}) \qquad (7.4\text{-}20)$$

Step 4 – Form the SHD HRIR set

Weight the FRs computed in Step 3 to ensure stability when solving for the final FRs.

Compute the pseudo-inverse matrix of the basis function panning gains that map the dense sampled sphere to SHD and solve for the SHD FRs. The panning gains can either be computed online or be stored as a set of coefficients, due to the gains depending on the SHD format, but not the HRIRs themselves.

Where:

$$F_{mpc} = G \times B \qquad (7.4\text{-}15)$$

F defines a set of SD left-ear HRTF filter responses (previously phase compensated in the steps above), G defines the basis function panning gains and B(t) defines a set of SHD left-ear HRTF filter responses. B(t) can then be solved for using:

$$B = G^{+} \times F_{mpc} \qquad (7.4\text{-}15)$$

Which is a rearrangement of the previous equations where G^{+} is the pseudo-inverse of matrix G.

Convert the FRs back to IRs, by taking the inverse MDFT of the FRs.

$$H_{SHD} = IMDFT(B) \qquad (7.4\text{-}15)$$

Create the symmetric right ear HRIRs by multiplying the left ear HRIRs by either [-1, 1] depending on the SHD channel.

7.4.6.1.2 Conversion from Time domain to CLDFB domain

7.4.6.1.3 Conversion from time domain to HRIR/ITD model (Time Domain binaural renderer)

7.4.7 Room acoustics parameters

7.4.7.1 Overview

The late reverb is driven by the set of parameters comprising of:

- RT60 – indicating the time that it takes for the reflections to drop 60 dB in energy level,
- DSR – diffuse to source signal energy ratio,
- Pre-delay – delay at which the computation of DSR values was made. Can be interpreted as the threshold between early reflections and late reverberation phase.

Spatialized, rotation-responsive, first-order early reflections can be added when using multichannel input (any configuration accepted). The early reflections rendering is determined by several parameters that drive a shoebox model using the image-source method. The set of parameters consists of:

- 3D rectangular virtual room dimensions,

- Broadband energy absorption coefficient per wall,
- Listener origin coordinates within room (optional),
- Low-complexity mode (optional) – favours efficient early reflection rendering over spatial accuracy.

Figure 7.4-3 illustrates the main reverberation properties with relevant reverberation synthesis control parameters indicated.

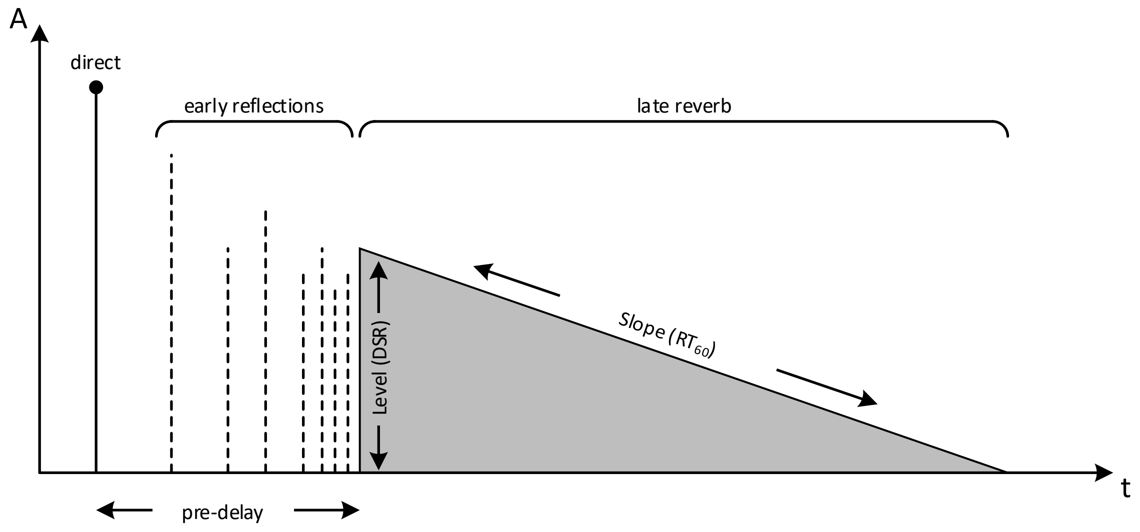


Figure 7.4-3: Simple representation of the main reverberation properties

Room acoustics parameters are provided to the renderer as metadata. Two metadata formats are supported in the IVAS decoder/renderer implementation: binary renderer config metadata format, and text renderer config metadata format (see [11], clause 5.14.1 and 5.14.2 respectively). Regardless of the metadata format, the general metadata processing is shared, as discussed in 7.4.7.2 for late reverb, and in 7.4.7.3 for early reflections. Both metadata formats support multiple acoustic environment datasets, allowing for selecting between such acoustic environments.

7.4.7.2 Late reverb parameters

The reverberation characteristics of a typical room are strongly frequency dependent. Therefore, RT_{60} and DSR parameters are specified per frequency band. For different room characteristics different frequency grids to specify reverb parameters might be suitable. For instance, in some cases octave or even coarser girds could provide sufficient frequency resolution, while in the other much finer resolution is required (e.g., in order to model strong local resonances in a room). Hence, reverb metadata precedes with data describing such frequency grid. This allows for reusing frequency grids across multiple room acoustics environments. The frequency grid metadata can be represented as:

- default grid identifier – indicating a predefined frequency grid to be used,
- set of individual frequencies – forming a frequency grid,
- start-hop-amount data – indicating start frequency, frequency hop factor, and the number of frequency bands in the grid.

The default frequency grids are provided in Table 7.4-2. It is possible to select a subsection of a default frequency grid by specifying an offset and number of frequency bands to be used.

Table 7.4-2: Default frequency grids for late reverb parameters

ID	Centre frequencies [Hz]	# bands	Description
0	31.5, 63, 125, 250, 500, 1000, 2000, 4000, 8000, 16000	10	Octave – ISO
1	25, 50, 100, 200, 400, 800, 1600, 3150, 6300, 12500	10	Octave - alternative
2	20, 25, 31.5, 40, 50, 63, 80, 100, 125, 160, 200, 250, 315, 400, 500, 630, 800, 1000, 1250, 1600, 2000, 2500, 3150, 4000, 5000, 6300, 8000, 10000, 12500, 16000, 20000	31	1/3 octave – ISO
3	25, 100, 400, 1600, 6400	5	2 octave – ISO
4	125, 250, 500, 1000, 2000, 4000	6	Octave subset
5	250, 250, 2500	3	
6	27, 56, 89, 126, 168, 214, 265, 323, 387, 459, 539, 628, 727, 839, 963, 1101,1256, 1429, 1621, 1836, 2077, 2345, 2644, 2978, 3351, 3767, 4232, 4750, 5329, 5975, 6697, 7502, 8401, 9405, 10525, 11775, 13171, 14729, 16468, 18410, 20577	41	1 ERB scale
7	27, 89, 168, 265, 387, 539, 727, 963, 1256, 1621, 2077, 2644, 3351, 4232, 5329, 6697, 8401, 10525, 13171, 16468, 20577	21	2 ERB scale
8	50, 150, 250, 350, 450, 570, 700, 840, 1000, 1170, 1370, 1600, 1850, 2150, 2150, 2500, 2900, 3400, 4000, 4800, 5800, 7000, 8500, 10500, 13500	25	Bark scale

Each room acoustic environment contains a frequency grid identifier, a set of RT_{60} and DSR parameters per frequency band, and a single pre-delay parameter. The pre-delay parameter indicates the delay at which the DSR values were calculated.

The pre-delay parameter provided for DSR values does not necessarily correspond to the actual late reverb render delay, regardless of the synthesis domain and algorithm used (see clauses 7.3.3 and 7.3.4, respectively). Therefore, the reverb output energy needs to be adjusted to maintain the room acoustics characteristics. The correction factor is applied to the DSR values corresponding to the relevant RT_{60} as derived from exponential decay model:

$$\mathrm{DSR}_{rend}(f) = DSR_{in}(f)e^{-2\alpha(f)(t_{rend}-t_{in})} \qquad (7.4\text{-}14)$$

where

$$\alpha(f) = \frac{\ln\left(10^{-\frac{60}{20}}\right)}{RT_{60}(f)} \qquad (7.4\text{-}15)$$

t_{rend} denotes late reverb render delay, and t_{in} pre-delay parameter used for computing DSR values.

7.4.7.3 Early reflections parameters

7.4.8 Use of custom loudspeaker layouts

The multi-channel loudspeaker rendering supports using custom loudspeaker layouts. The custom loudspeaker layouts are described by azimuth $\theta_{LS}(i)$ and elevation $\phi_{LS}(i)$ angles, where i is the loudspeaker channel.

For MASA, OMASA, and multi-channel (in McMASA mode) formats, the multi-channel loudspeaker rendering is performed using the DirAC renderer. For the direct sound part, the DirAC renderer determines the panning gains using VBAP (see clause 7.2.1.2), which supports custom loudspeaker layouts. For the non-directional sound part, the DirAC renderer determines the rendering gains using the diffuse response gains (see clause TODO: ADD REF TO DIFFUSE RESPONSE DETERMINATION), which also supports custom loudspeaker layouts.

TODO: WRITE THE CUSTOM LOUDSPEAKER RENDERING ASPECTS FOR OTHER FORMATS

8 Description of the transmitted parameter indices

Editor's note: Assumption here is that subclauses, e.g., 7.2.1, etc. will be based on bit rate ranges / technology selections at each rate, from lowest bit rate to highest bit rate.

8.1 Bit allocation overview

8.2 Bit allocation for stereo audio

8.3 Bit allocation for scene-based audio (SBA)

8.3.1 Bit allocation for SBA in non-DTX frames

Table 8.3-1: Bit allocation at 13.2, 16.4, 24.4 and 32 kbps

Description	Ordering of bits	13.2 kbps	16.4 kbps	24.4 kbps	32 kbps
total bits	Forward ordering of bits	264	324	488	640
IVAS format header		3	3	3	3
SBA header – planar mode		1	1	1	1
SBA header – ambisonics order		2	2	2	2
Core coder - SCE		variable	variable	variable	variable
AGC gain bits	Reverse ordering of bits	0 or 2	0 or 2	0 or 2	0 or 2
AGC flag		1	1	1	1
SPAR metadata (PR, CP and D coefficients)		variable	variable	variable	variable
SPAR coding strategy		3	3	3	3
SPAR quantization strategy		2	2	2	2
DirAC metadata		TBD	TBD	TBD	TBD
Metadata active/inactive mode flag		1	1	1	1

Table 8.3-2: Bit allocation at 48, 64 and 80 kbps

Description	Ordering of bits	48 kbps	64 kbps	80 kbps
total bits	Forward ordering of bits	960	1280	1600
IVAS format header		3	3	3
SBA header – planar mode		1	1	1
SBA header – ambisonics order		2	2	2
Core coder - CPE		variable	variable	variable
SPAR metadata (PR, CP and D coefficients)	Reverse ordering of bits	variable	variable	variable
SPAR coding strategy		3	3	3
SPAR quantization strategy		2	2	2
active W residual index		0 or 1	0 or 1	0 or 1
Dynamic active W flag		1	1	1
DirAC metadata		TBD	TBD	TBD
Metadata active/inactive mode flag		1	1	1

Table 8.3-3: Bit allocation at 96, 128, 160 and 192 kbps

Description	Ordering of bits	96 kbps	128 kbps	160 kbps	192 kbps
total bits	Forward ordering of bits	1920	2560	3200	3840
IVAS format header		3	3	3	3
SBA header – planar mode		1	1	1	1
SBA header – ambisonics order		2	2	2	2
Core coder - MCT		variable	variable	variable	variable
SPAR metadata (PR, CP and D coefficients)	Reverse ordering of bits	variable	variable	variable	variable
SPAR coding strategy		3	3	3	3
SPAR quantization strategy		2	2	2	2
Dynamic active W flag		1	1	1	1
DirAC metadata		TBD	TBD	TBD	TBD
reserved		1	1	1	1

Table 8.3-4: Bit allocation at 256, 384 and 512 kbps

Description	Ordering of bits	256 kbps		384 kbps	512 kbps
		FOA	**HOA2, HOA3**		
total bits	Forward ordering of bits	5120	5120	7680	10240
IVAS format header		3	3	3	3
SBA header – planar mode		1	1	1	1
SBA header – ambisonics order		2	2	2	2
Core coder - MCT		variable	variable	variable	variable
PCA metadata	Reverse ordering of bits	0 orTBD	0	0	0
PCA flag		1	0	0	0
SPAR metadata (PR, CP and D coefficients)		variable	variable	variable	variable
SPAR coding strategy		3	3	3	3
SPAR quantization strategy		1	1	1	1
reserved		1	1	1	1
DirAC metadata		TBD	TBD	TBD	TBD
reserved		1	1	1	1

8.3.2 To be described per bitrate (13.2 to 512 kbps)

8.4 Bit allocation for metadata-assisted spatial audio (MASA)

8.5 Bit allocation for object-based audio (ISM)

8.6 Bit allocation for multi-channel audio

8.7 Bit allocation for combined Object-based audio and SBA (OSBA)

8.8 Bit allocation for combined Object-based audio and MASA (OMASA)

8.9 Bit allocation for EVS-compatible mono audio

Annex A (normative): RTP Payload Format and SDP Parameters

A.1 Introduction

This annex specifies a generic RTP payload format and SDP parameters for the Immersive Voice and Audio Services (IVAS) codec for mobile communication, as outlined in [5] and described in detail in the present document. The IVAS RTP packets consists of the RTP header, potentially including RTP header extensions, the IVAS payload header, one or more IVAS frame data and optional extension frame data. The IVAS SDP parameters allow describing parameters of IVAS RTP sessions and offer-answer based negotiation to configure IVAS RTP sessions according to the requirements of the endpoints.

Editor's Note: A compact format and a corresponding SDP parameter (hf-only) as in EVS are not expected for the IVAS RTP Payload Format and SDP parameters.

As IVAS is the immersive voice and audio extension of the Enhanced Voice Services (EVS) codec [3], this specification is incorporating the EVS codec's RTP payload format and SDP parameters specified in [3].

Editor's Note: Details on the RTP Payload Format and SDP Parameters and incorporation of EVS RTP payload format aspects are expected for the next version of this document.

Annex B (informative): Change history

Change history							
Date	Meeting	TDoc	CR	Rev	Cat	Subject/Comment	New version
11-2023	SA4#126	S4-231842				Presented to 3GPP SA4 Audio SWG	0.0.1