TSG-RAN Working Group 1 meeting No. 11 TSGR1#11(00)0255
Februar 29 – March 3, San Diego, USA

Agenda Item: **14. Contributions on issues were CRs are still needed for Release–99 specifications**

Source: **Siemens,**

Title: CR 25.212-052 and CR 25.222-026: Padding function for Turbo coding of small blocks

Document for: **Decision**

Introduction

At the WG1 meeting #10 in Beijing it was decided to lower the minimum block size for turbo coding to 40 bits. The definition of a function to adjust the block size to 40 bits for blocks smaller than 40 bits was left open. Padding bits of value zero provides a good solution with least added complexity. However there are different implications for the receiver depending on the position of the padded 'zeros'. Depending on whether the padding is performed in the beginning or in the end, the receiver can optionally make use of different a-priori information.

The following two cases are described for discussion:

A) Padding at the end of the input block ('end padding')

B) Padding at the beginning of the input block ('front padding')

Case B) has a performance advantage over case A) , if an optional optimisation possibility is utilised in the decoder while the coder shows no complexity increase.

Because the performance will probably not be critical for release 99, no unnecessarily complex handling should be mandated for release 99. However, as there is no guarantee that this will always be the case, there should be the possibility to extend the performance later on, of course accepting a higher complexity then. Furthermore this extended solution should be compatible on the air interface with the solution for release 99 in order to avoid compatibility problems.

Taking all this in mind we propose to implement case B) for release 99.

Description

A) Padding at the end of the input block ('end padding')

The short input block is followed by a number of padding bits of value zero to achieve a total block size of 40 bits. The turbo coder remains unchanged. This was proposed by NEC recently on the reflector and is used as a starting point for the discussion.

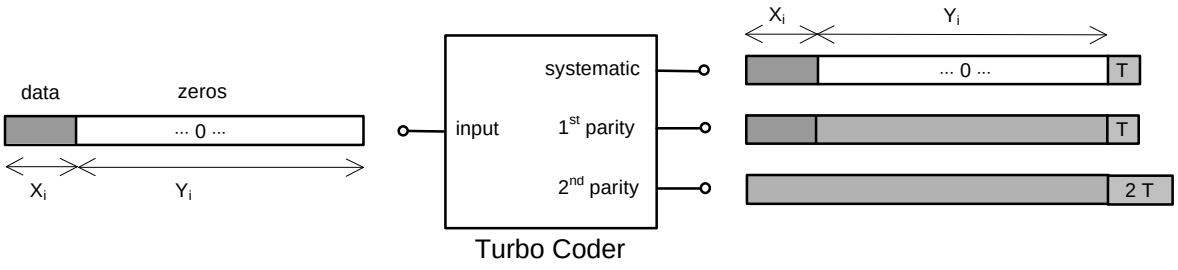

Figure 1: end padding

As described in Figure 1, at the output of the coder the systematic bits will start with the original data block of length X_i followed by $Y_i = 40 – X_i$ bits of value zero and then the tail bits for termination of the coder (T = 3 bits). Since the convolutional coder is recursive, the 1st parity bit sequence will not consist of pre-determined bits of value zero after

the block of size X_i was encoded. The contents of the registers of the coder is fed back until termination is initiated. To exploit this fed back information the receiver structure needs to be changed which increases complexity considerably.

The 2nd parity bits are not predictable in a straight-forward way because of the internal interleaver. They are not subject to this proposal and are therefore not treated by this discussion.

B) Padding at the beginning of the input block ('front padding')

Bits of value zero are padded at the beginning (rather than the end) of the input data block. For the systematic and the 1st parity output sequence the coded data bits are now transmitted right before the tail bits. Since the coder starts in the 'all-zero-state' the output of the 1st parity path remains zero until the actual data block starts (Figure 2). This way more pre-defined knowledge is available at the receiver, which can be exploited by ignoring the first $Y_i = 40 - X_i$ bits of the systematic and the 1st parity path and setting them firmly to zero before entering them to the soft-decoder. The structure of the Turbo decoder itself remains unchanged. In this way an improved performance can be realised for little extra complexity.

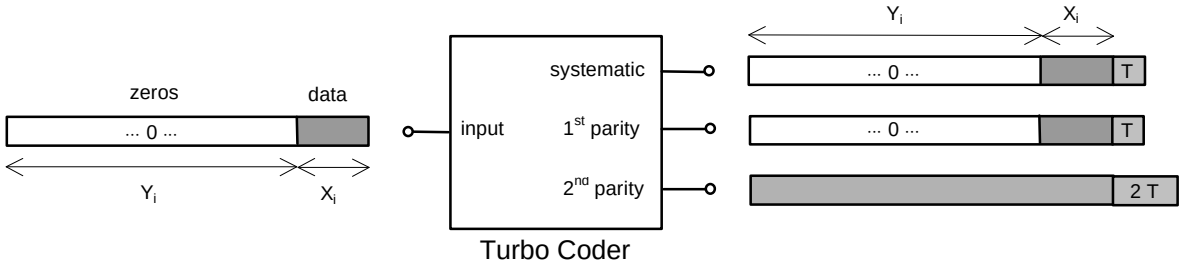

Figure 2: front padding

Performance evaluation

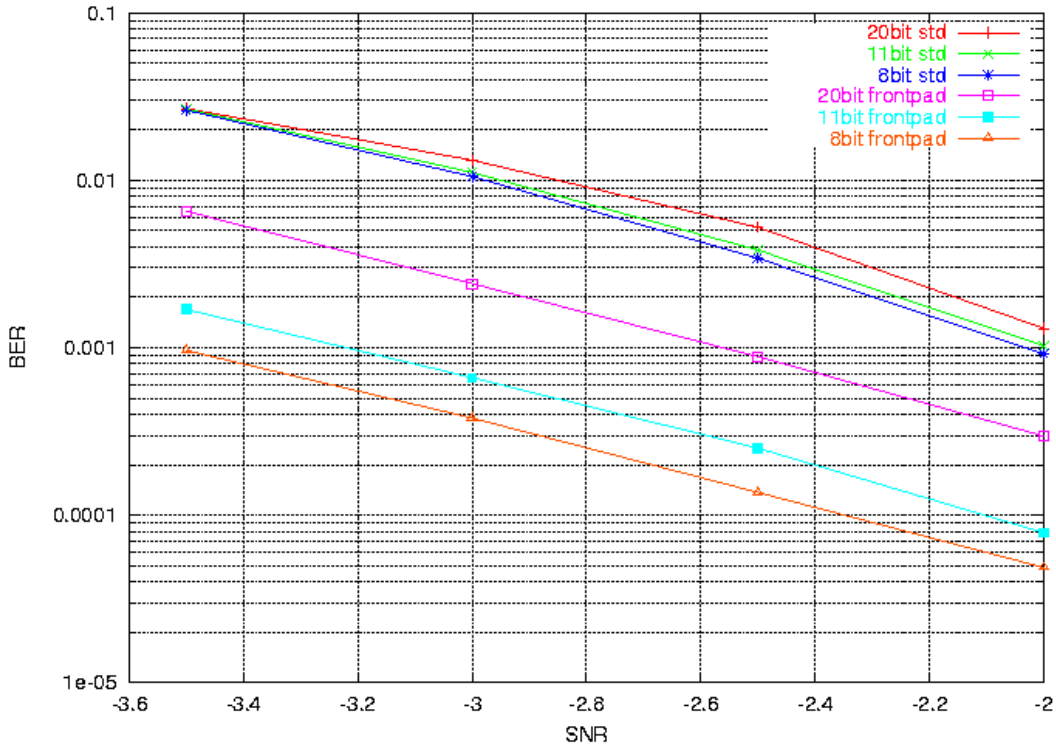

Figure 3: Performance advantage of using the a priori information available in method B

The performance advantage of the scheme stems from the fact, that the receiver can utilise the a priori information that some of the received raw bits have a predetermined value of 0. Instead of using the value which is actually received via the air-interface, the receiver can instead substitute the a-priori knowledge. Because the latter does not contain any noise or other degradation, the data can be detected more reliable. When this a priori information is not taken into account, both case A and B will have the same performance.

Figure 3 shows this performance comparison. All simulations were conducted using AWGN conditions. The three upper curves show the performance without making use of the a priori information assuming 20, 11 or 8 information bits. When the receiver does make use of the a-priori information, then a performance advantage can be realised, as is shown in the lower three curves, again for blocksizes of 20, 11 and 8 bits. The performance advantage increases with decreasing blocksize (increasing number of padding bits).

Implementation in conjunction with code block segmentation.

There is already a padding function integrated in the code block segmentation. It is used to make sure that all blocks after segmentation have the same size. It is therefore proposed, that the padding function for turbo codes is also integrated into this section.

There are two options to implement the front padding option in the code block segmentation section:

1) Front padding is implemented individually for turbo codes with blocksizes smaller than 40 bits.

2) Front padding is implemented both for padding of small turbo code blocks and for padding for segmentation of large blocks

There is probably only a marginal difference as far as the implementational complexity of this scheme is concerned. We use option 2 for the appended CR, because it allows a more elegant formulation and provides a slight performance advantage for convolutionally coded data. The reason for this advantage is as follows: Because e_{ini} has recently been changed to be 1, always the first bit of a coded block will be punctured (assuming that at least one bit is punctured of course). If the first bit corresponds to a padding bit, then of course this minimises the impact of this first punctured bit on the data bits.

Proposal

We propose to implement the ‘front padding’ function to adjust the block size of small blocks to 40 bits for release 99 .

Using front padding the receiver can (as a receiver implementation option) ignore the first $Y_i = 40 – X_i$ bits of the systematic *and* the 1st parity path and set them to zero without loosing any information since these bits would be zero anyway. This will actually improve the performance of the receiver, because the predetermined bits do not carry noise and therefore do not cause misdetections, as would be the case for received bits.

While it is generally expected that the performance of turbo codes for small block sizes will not be crucial in the system because this case will seldom occur, the discussion about the AMR mode signalling clearly shows that such generally agreed assumptions can easily turn out to be incorrect sooner or later. Therefore it would be highly desirable to build some potential for performance optimisations into the system if they do not cause extra complexity if the performance is not needed.

The attached CR implements the front padding procedure i.e. case B) is implemented according to option 2) for release 99.

3GPP/SMG Meeting #11
San Diego, USA, 29 April-03 March 2000

Document R1-00-0255
e.g. for 3GPP use the format TP-99xxx
or for SMG, use the format P-99-xxx

CHANGE REQUEST

Please see embedded help file at the bottom of this page for instructions on how to fill in this form correctly.

25.212	CR	052	Current Version:	3.1.1

GSM (AA.BB) or 3G (AA.BBB) specification number ↑ *↑ CR number as allocated by MCC support team*

For submission to:	WG1 #11	for approval	X	strategic		(for SMG use only)
list expected approval meeting # here ↑		for information		non-strategic		

Form: CR cover sheet, version 2 for 3GPP and SMG *The latest version of this form is available from: ftp://ftp.3gpp.org/Information/CR-Form-v2.doc*

Proposed change affects: *(at least one should be marked with an X)*	(U)SIM		ME	X	UTRAN / Radio	X	Core Network	

Source: Siemens AG **Date:** 29.02.2000

Subject: Padding Function for Turbo coding of small blocks

Work item: TS 25.212

Category: *(only one category shall be marked with an X)*

F	Correction		**Release:**	Phase 2	
A	Corresponds to a correction in an earlier release			Release 96	
B	Addition of feature	X		Release 97	
C	Functional modification of feature			Release 98	
D	Editorial modification			Release 99	X
				Release 00	

Reason for change: At the last meeting #10 the smallest block size for Turbo coding was set to 40. It was left over to include a functionality to handle blocks of smaller size than 40 bits. This is added in the specification by this CR.

Clauses affected: 4.2.2.2 Code block segmentation

Other specs affected:

Other 3G core specifications		→ List of CRs:
Other GSM core specifications		→ List of CRs:
MS test specifications		→ List of CRs:
BSS test specifications		→ List of CRs:
O&M specifications		→ List of CRs:

Other comments: The corresponding CR for TDD mode is 25.222-026.

4.2.2 Transport block concatenation and code block segmentation

All transport blocks in a TTI are serially concatenated. If the number of bits in a TTI is larger than *Z*, the maximum size of a code block in question, then code block segmentation is performed after the concatenation of the transport blocks. The maximum size of the code blocks depends on whether convolutional coding, turbo coding or no coding is used for the TrCH.

4.2.2.1 Concatenation of transport blocks

The bits input to the transport block concatenation are denoted by $b_{im1}, b_{im2}, b_{im3}, \ldots, b_{imB_i}$ where *i* is the TrCH number, *m* is the transport block number, and B_i is the number of bits in each block (including CRC). The number of transport blocks on TrCH *i* is denoted by M_i. The bits after concatenation are denoted by $x_{i1}, x_{i2}, x_{i3}, \ldots, x_{iX_i}$, where *i* is the TrCH number and $X_i = M_i B_i$. They are defined by the following relations:

$x_{ik} = b_{i1k}$ $k = 1, 2, \ldots, B_i$

$x_{ik} = b_{i,2,(k-B_i)}$ $k = B_i + 1, B_i + 2, \ldots, 2B_i$

$x_{ik} = b_{i,3,(k-2B_i)}$ $k = 2B_i + 1, 2B_i + 2, \ldots, 3B_i$

…

$x_{ik} = b_{i,M_i,(k-(M_i-1)B_i)}$ $k = (M_i - 1)B_i + 1, (M_i - 1)B_i + 2, \ldots, M_i B_i$

4.2.2.2 Code block segmentation

Segmentation of the bit sequence from transport block concatenation is performed if $X_i > Z$. The code blocks after segmentation are of the same size. The number of code blocks on TrCH *i* is denoted by C_i. If the number of bits input to the segmentation, X_i, is not a multiple of C_i, filler bits are added to the beginning of the first ~~last~~ block. The filler bits are transmitted and they are always set to 0. The maximum code block sizes are:

convolutional coding: $Z = 504$

turbo coding: $Z = 5114$

no channel coding: $Z =$ *unlimited*

The bits output from code block segmentation are denoted by $o_{ir1}, o_{ir2}, o_{ir3}, \ldots, o_{irK_i}$, where *i* is the TrCH number, *r* is the code block number, and K_i is the number of bits.

Number of code blocks: $C_i = \acute{e}X_i / Z\grave{u}$

Number of bits in each code block:

if $X_i < 40$ and Turbo coding is used, then
$K_i = 40$
else
$K_i = \acute{e}X_i / C_i\grave{u}$
end if

Number of filler bits: $Y_i = C_i K_i - X_i$

If $X_i \le Z$, then

$o_{i1k} = 0$ $k = 1, 2, \ldots, Y_i$

$o_{i1k} = x_{i,(k-Y_i)}$ ~~$o_{i1k} = x_{ik}$~~ $k = Y_i+1, Y_i+2, \ldots, K_i$

end if

~~, and $K_i = X_i$.~~

If $X_i \geq Z$, then

$o_{i1k} = 0$ $k = 1, 2, \ldots, Y_i$

$o_{i1k} = x_{i,(k-Y_i)}$ ~~$o_i = x_{ik}$~~ $k = \_i+$ Y_i 2, …, i

$o_{i2} = x_{i\ (k\ \ \ _i-Y_i)}$ ~~$o_{i2k} = x_{i,(k+K_i)}$~~ $k = 1, 2, \ldots, K_i$

$o_{i3k} = x_{i,(k+2K_i-Y_i)}$ ~~$o_{i3k} = x_{i,(k+2K_i)}$~~ $k = 1, 2, \ldots, K_i$

…

$o_{iC_ik} = x_{i,(k+(C_i-1)K_i-Y_i)}$ ~~$o_{iC_ik} = x_{i(k+(C_i-1)K_i)}$~~ $k = 1, 2, \ldots, K_i$ ~~$-Y_i$~~

~~$o_{iC_ik} = 0$ $k = (K_i - Y_i) + 1, (K_i - Y_i) + 2, \ldots, K_i$~~

end if

3GPP/SMG Meeting #11
San Diego, USA, 29 April-03 March 2000

Document R1-00-0255
e.g. for 3GPP use the format TP-99xxx
or for SMG, use the format P-99-xxx

CHANGE REQUEST

Please see embedded help file at the bottom of this page for instructions on how to fill in this form correctly.

25.222	CR	026	Current Version:	3.1.1

GSM (AA.BB) or 3G (AA.BBB) specification number ↑ *↑ CR number as allocated by MCC support team*

For submission to:	WG1 #11	for approval	X	strategic		(for SMG use only)
list expected approval meeting # here ↑		for information		non-strategic		

Form: CR cover sheet, version 2 for 3GPP and SMG *The latest version of this form is available from: ftp://ftp.3gpp.org/Information/CR-Form-v2.doc*

Proposed change affects: *(at least one should be marked with an X)*

(U)SIM		ME	X	UTRAN / Radio	X	Core Network	

Source: Siemens AG **Date:** 29.02.2000

Subject: Padding Function for Turbo coding of small blocks

Work item: TS 25.222

Category: *(only one category shall be marked with an X)*

F	Correction		**Release:**	Phase 2	
A	Corresponds to a correction in an earlier release			Release 96	
B	Addition of feature	X		Release 97	
C	Functional modification of feature			Release 98	
D	Editorial modification			Release 99	X
				Release 00	

Reason for change: At the last meeting #10 the smallest block size for Turbo coding was set to 40. It was left over to include a functionality to handle blocks of smaller size than 40 bits. This is added in the specification by this CR.

Clauses affected: 4.2.2.2 Code block segmentation

Other specs affected:

Other 3G core specifications		→ List of CRs:
Other GSM core specifications		→ List of CRs:
MS test specifications		→ List of CRs:
BSS test specifications		→ List of CRs:
O&M specifications		→ List of CRs:

Other comments: The corresponding CR for FDD mode is 25.212-052.

4.2.2 Transport block concatenation and code block segmentation

All transport blocks in a TTI are serially concatenated. If the number of bits in a TTI is larger than *Z*, then code block segmentation is performed after the concatenation of the transport blocks. The maximum size of the code blocks depend on if convolutional or turbo coding is used for the TrCH.

4.2.2.1 Concatenation of transport blocks

The bits input to the transport block concatenation are denoted by $b_{im1}, b_{im2}, b_{im3}, \ldots, b_{imB_i}$ where *i* is the TrCH number, *m* is the transport block number, and B_i is the number of bits in each block (including CRC). The number of transport blocks on TrCH *i* is denoted by M_i. The bits after concatenation are denoted by $x_{i1}, x_{i2}, x_{i3}, \ldots, x_{iX_i}$, where *i* is the TrCH number and $X_i = M_i B_i$. They are defined by the following relations:

$x_{ik} = b_{i1k}$ $k = 1, 2, \ldots, B_i$

$x_{ik} = b_{i,2,(k-B_i)}$ $k = B_i + 1, B_i + 2, \ldots, 2B_i$

$x_{ik} = b_{i,3,(k-2B_i)}$ $k = 2B_i + 1, 2B_i + 2, \ldots, 3B_i$

…

$x_{ik} = b_{i,M_i,(k-(M_i-1)B_i)}$ $k = (M_i - 1)B_i + 1, (M_i - 1)B_i + 2, \ldots, M_i B_i$

4.2.2.2 Code block segmentation

~~NOTE: It is assumed that filler bits are set to 0.~~

Segmentation of the bit sequence from transport block concatenation is performed if $X_i > Z$. The code blocks after segmentation are of the same size. The number of code blocks on TrCH *i* is denoted by C_i. If the number of bits input to the segmentation, X_i, is not a multiple of C_i, filler bits are added to the beginning of the first ~~last~~ block. The filler bits are transmitted and they are always set to 0. The maximum code block sizes are:

convolutional coding: $Z = 504$

turbo coding: $Z = 5114$

no channel coding: $Z = unlimited$

The bits output from code block segmentation are denoted by $o_{ir1}, o_{ir2}, o_{ir3}, \ldots, o_{irK_i}$, where *i* is the TrCH number, *r* is the code block number, and K_i is the number of bits.

Number of code blocks: $C_i = \acute{e}X_i / Z\grave{u}$

Number of bits in each code block:

if $X_i < 40$ and Turbo coding is used, then
$K_i = 40$
else
$K_i = \acute{e}X_i / C_i\grave{u}$
end if

Number of filler bits: $Y_i = C_i K_i - X_i$

If $X_i \pounds Z$, then

$o_{i1k} = 0$ $k = 1, 2, \ldots, Y_i$

$o_{i1k} = x_{i,(k-Y_i)}$ ~~$o_{i1k} = x_{ik}$~~ $k = Y_i+1, Y_i+2, \ldots, K_i$

end if

~~, and $K_i = X_i$.~~

If X_i ~~>~~ Z, then

$o_{i1k} = 0$ $k = 1, 2, \ldots, Y_i$

$o_{i1k} = x_{i,(k-Y_i)}$ ~~$o_{i1k} = x_{ik}$~~ $k = Y_i+1, Y_i+2, \ldots, K_i$

$o_{i2k} = x_{i,(k+K_i-Y_i)}$ ~~$o_{i2k} = x_{i,(k+K_i)}$~~ $k = 1, 2, \ldots, K_i$

$o_{i3k} = x_{i,(k+2K_i-Y_i)}$ ~~$o_{i3k} = x_{i,(k+2K_i)}$~~ $k = 1, 2, \ldots, K_i$

…

$o_{iC_ik} = x_{i,(k+(C_i-1)K_i-Y_i)}$ ~~$o_{iC_ik} = x_{i(k+(C_i-1)K_i)}$~~ $k = 1, 2, \ldots, K_i$ ~~$- Y_i$~~

~~$o_{iC_ik} = 0$ $k = (K_i - Y_i) + 1, (K_i - Y_i) + 2, \ldots, K_i$~~

end if