ANNEX 34

(to the Report of SG16)

| | |
|---|---|
| QUESTIONS: | Q.22/16; ETSI SMG11 |
| SOURCE: | ITU-T SG 16 (Geneva, 7-18 February 2000) |
| TITLE: | Cooperation on basic operators |

---

## COMMUNICATION

| | |
|---|---|
| FROM: | Q.22/16 |
| TO: | ETSI SMG11 |
| APPROVAL: | Approved by SG 16 (Geneva, 7-18 February 2000) |
| FOR: | Action |
| DEADLINE: | November 10, 2000 |
| CONTACT: | **Q.22/16:** Redwan Salami |

University of Sherbrooke
Dept. Electrical & Computer Eng.
Sherbrooke, Quebec, J1K 2R1
CANADA
Tel: +1-819-821-8000 x3893
Fax: +1-819-821-7937
E-mail: salami@gel.usherb.ca

**ETSI/SMG11:** Kari Jarvinen
Nokia Research Center
P.O.Box 100 (Sinitaival 6)
FIN-33721 Tampere
FINLAND
Tel: +358-3-272-5854
Fax: +358-3-272-5888
Email:kari.jarvinen@research.nokia.com

ITU-T Study Group 16 Question 22 (Software tools) is the question within ITU-T to maintain and improve the ITU-T Software Tool Library. One of our recent work items was the inclusion of the G.729 set of basic operators, which are also used in ETSI EFR and AMR, into the library.

Recently, it has been proposed that a set of “alternative” libraries of basic operators be included in addition to the existing one, which is based on 32-bit accumulators (with left shift). Two specific proposals are an alternative set for 32-bit accumulators with right shift, and another with 40-bit accumulators. Consideration will also be given to 32-bit accumulators without shift. During the discussions, it was clarified that it would be up to a given standardization effort to choose among the different libraries which one should be used. Coupled with this discussion is the revision of the existing weights associated with each basic operator. It is our understanding that these weights, which are used in deriving the WMOPS figures, were defined some years ago, and may justify revisiting.

We would like to take this opportunity to invite ETSI SMG11 to discuss the issue and possibly provide some feedback on the existing proposals, since substantial expertise on the subject resides in ETSI SMG11. Attached is the summary and rationale of the proposals for these alternative basic operator libraries.

**Joint TSG-S4#10 - SMG11#15 Meeting** **S4/SMG11 Tdoc 118/00**
**Helsinki, Finland**
**February 28th - March 3rd 2000**
**Attach:** (a) Pages 1 through 6 of COM16-D.418

**Joint TSG-S4#10 - SMG11#15 Meeting** **S4/SMG11 Tdoc 118/00**
**Helsinki, Finland**
**February 28th - March 3rd 2000**

**UIT - Secteur de la normalisation des télécommunications**
**ITU - Telecommunication Standardization Sector**
**UIT - Sector de Normalización de las Telecomunicaciones**

**Study Period 1997-2000**

| | | | |
|---|---|---|---|
| Commission d'études<br>Study Group<br>Comisión de Estudio | **16** | Contribution tardive<br>Delayed Contribution<br>Contribución tardía | **D.418 (WP 3/16)** |

Geneva, 7-18 February 2000

| | |
|---|---|
| Texte disponible seulement en<br>Text available only in<br>Texto disponible solamente en | **E** |

Question: 22/16

SOURCE*: CANADA

TITLE: PROPOSAL FOR AN UPDATE OF BASIC OPERATORS

---

ABSTRACT: In this contribution, we propose extending the library of basic operators to be used in defining ITU-T speech coding standards. We propose adding two sets of operators: one set for performing 32-bit accumulation without shift, and another set for performing 40-bit accumulation or 32-bit accumulation with right shift.

Details about the proposed set of operators are given in this contribution.

## Introduction:

In the May 1999 meeting of SG16, the issue of basic operators was discussed in Question 22/16. Two documents were presented: D.275 and D.306. D.275 proposed that an updated version of the basic operators used in G.729 should be added to the ITU-T STL (Software Tools Library). In addition, D.275 proposes that the weights associated with each basic operator should be reviewed, and that new operators might need to be added. D.306 proposed that some 40-bit basic operators should be added to the STL, as well as three basic operators used in G.723.1 which are not present in the set of basic operators used by G.729.

During the discussions, it was clarified that the main purpose of the basic operators is to emulate instructions available in current digital signal processors, and the basic operator definition will benefit future standardization efforts.

After some discussion, the meeting agreed to incorporate the updated G.729 operators into the STL, together with the three G.723.1 basic operators not present in G.729. The group felt that more discussion is necessary on the definition of new basic operators. Interested experts were invited to continue the discussion by correspondence using the Q19-22 e-mail reflector. However, this issue has not been discussed since then (no contributions were presented in the September 1999 meeting).

This contribution is an attempt to revive the discussion in this important issue, since it is envisaged that these operators will be used in the definition of future speech coding standards (e.g. 4 kbit/s narrowband and 16 kbit/s wideband future standards). We propose extending the current basic operators by adding two sets of operators: one set for performing 32-bit accumulation without shift, and another set for performing 40-bit accumulation or 32-bit accumulation with right shift.

## 40-bit accumulation and 32-bit accumulation with right shift :

In the current basic operators, the function *L_mac(L_var3,var1,var2)* performs the following:

> *Multiply var1 by var2 (both 16 bit) and shift the result left by 1; add the 32 bit result to L_var3 with saturation; return a 32 bit result:*
>
> *L_mac(L_var3,var1,var2) = L_add(L_var3,(L_mult(var1,var2)).*

So when a multiply-and-add procedure is performed (filtering, correlation, etc.), there is no mechanism to guarantee that no saturation would occur during this procedure. In some implementations, the overflow flag is tested at the end of the procedure and if it is on, the input is scaled down and the procedure is repeated. Otherwise the input should be overscaled down which results in loss of precision.

If a 40-bit accumulator is used, it is guaranteed that no saturation will occur. In case of 32-bit accumulators, DSPs already possess a mechanism to prevent the saturation within a multiply-and-add procedure by allowing a right shift by a certain number of bits within the accumulation procedure. Neither this right shift nor 40-bit accumulation is addressed within the current basic operators, while current DSPs can do either one or the other. Therefore we feel the need to extend the current basic operators to allow these two operations: 40-bit accumulation and 32-bit accumulation with right shift.

We propose adding a set of basic operators which permit performing either 40-bit accumlation or 32-bit accumulation with right shift. A flag called *PRODUCT_SHIFT* is defined to signify the number of right shifts used in the accumulation. –1 is used for a 40-bit accumulator (with left shift by 1) and n is used for a 32-bit accumulator with n right shifts (n is typically 6).

The code is the same for 40-bit or 32-bit accumulation. It is only necessary to change the value of *PRODUCT_SHIFT* in order to select the target DSP operation. The precision will be different depending on the selected value of *PRODUCT_SHIFT.*

**Accumulation without shift:**

Another important advantage of introducing a mechanism of preventing overflow within an accumulation procedure is not to penalize processors which don't perform overflow control. DSP-based devices are not the only platforms where speech coding algorithm are implemented. With the increasing use of speech coders in mulimedia applications, speech coders will be increasingly implemented on the host processor in desktop environments. In such environments, genetal purpose processors (GPP) are usually used, which don't have an overflow control mechanism. Examples of GPPs are Intel (MMX) and RISC (MIPS, PowerPC, ARM). Apart from the lack of overflow control, GPPs don't perform accumulate and left shift in one cycle, as done by L_mac. Therefore, these processors are penalized if a bit-exact implementation is required. As an example, a bit-exact implementation of G.729 using MMX is significantly penalized in complexity.

If GPPs are target platforms for the standard, it is a good practice not to use loops based on the overflow flag. In general, the use of the overflow flag should be recommended only for debugging purposes.

A solution that works for both DSPs and GPPs is to introduce accumulation without left shift. The advantage for GPPs is that eliminating the left shift saves one cycle/sample, and the overflow is avoided in most operations due to the extra precision which enables bit-exact implementations on both DSPs and GPPs (which lack overflow control).

Accumulation without shift is not present in the current basic operators, and if introduced it offers the following advantages:

- Accumulation without shift (L_mac0) can be performed by both DSPs (one cycle/sample) and GPPs (½ a cycle/sample in case of MMX).
- It gives one more bit of precision compared to L_mac.
- It is usually sufficient for avoiding overflow problems in cases of moderate accumulation.

Note that in the initial basic operators, there was no function for multiplying two integers without left sheft, and for this reason the function `i_mult()` was added in G.723.1. However, `i_mult()` returns a 16-bit value, and in order to be able to perform 32-bit accumulation without shift a similar function which returns a 32-bit value is needed. We propose adding a multiplication function `L_mult0()` which returns a 32-bit value and the accumulation functions `L_mac0()` and `L_msu0()` for performing 32-bit accumulation without shift.

It is worth noting that it was accepted by ETSI to add these 3 operators to the library of basic operators in the case of the TETRA speech coding standard.

ADVANTAGES OF NEW OPERATORS:

The advantages of introducing these two new sets of basic operators are the following:

- It is an addition of few operators, and it is 100% compatible with current operators. So there is no need to redefine current standards.
- Resolve problem of saturation during calculation of energy or correlation.
- Increase precision of product. In some processing, we don't need to reduce amplitude of coefficients.
- Compatible with 32 bit and 40 bit DSPs and others platforms without overflow control.
- ▪ for 32 bit DSPs, a right shift if performed before the accumulation to avoid saturation.
- ▪ for 40 bit DSPs, a standard left shift is performed and accumulation is done on 40 bits.
- Allow more flexibility in choices of operators:
- ▪ L_mac(): standard 1 left shift MAC for Q15 multiplication.
- ▪ L_mac0(): avoids saturation in cases of moderate accumulations (most filtering procedures), and it doesn’t penalize GPPs.
- ▪ acc_mac() : new MAC giving extra bits to avoid saturation in energy/correlation calculation or in double-precision filtering.

The only disadvantage of introducing these new basic operators it that there might be a need for more than 1 set of test vectors: one for 40-bit operation and another for 32-bit operation with a given right shift.

**Proposed operators:**

The proposed operators are given below:

For accumulation without shift

```
Word32 L_mult0(Word16 var1, Word16 var2);
Word32 L_mac0(Word32 L_var3, Word16 var1, Word16 var2);
Word32 L_msu0(Word32 L_var3, Word16 var1, Word16 var2);
```

For 40-bit accumulation or 32-bit with right shift accumulation.

```
void acc_ld(Word32 L_var1);
void acc_add(Word32 L_var1);
void acc_sub(Word32 L_var1);
void acc_mult(Word16 var1, Word16 var2);
void acc_mac(Word16 var1, Word16 var2);
void acc_msu(Word16 var1, Word16 var2);
void acc_shl(Word16 var1);
void acc_shr(Word16 var1);
void acc_negate(void);
Word16 norm_acc(void);
Word32 L_extract_acc(Word16 var1);
```

The table below shows how acc_mac(a, b) is done on many platforms (with a right shift of n (typically 6) in case of 32-bit):

| Platform | Accumulator | Operation | Comments |
|---|---|---|---|
| DSP | 40 bit | acc += (a*b)<<1<br>result = acc | 1 cycle / sample |
| DSP | 32 bit | acc += (a*b)>>n<br>result = acc<<(n+1) | 1 cycle / sample |
| MMX | 32 bit | 1) (x=a*b) and (y=c*d)<br>2) x=x>>n and y=y>>n<br>3) acc += x + y<br>result = acc<<(n+1) | 1.5 cycle / sample |

A static variable called *accumulator* is used.

A flag called *PRODUCT_SHIFT* is defined to signify the number of right shifts used in the accumulation. –1 is used for a 40-bit accumulator and n is used for a 32-bit accumulator with n right shifts.

See the comments in the Appendix for more details.

**Example1: correlation of vector x[i] with vector y[i]:**

| New operators | 40-bit accumulator | 32-bit with right shift by n |
|---|---|---|
| acc_ld(0);<br>for (i=0; i<L; i++)<br>   acc_mac(x[i], y[i]);<br>result = L_extract_acc(-8); | Acc = 0;<br>For (i=0; i<L; i++)<br>Acc += (x[i]*y[i])<<1;<br>Result = acc>>8; | acc = 0;<br>for (i=0; i<L; i++)<br>acc += (x[i]*y[i])>>n;<br>result = acc<<(n+1-8); |

In this example, the result is divided by 256.

**Example2: normalized correlation of vector x[i] with vector y[i]:**

| New operators | 40-bit accumulator | 32-bit with right shift by n |
|---|---|---|
| acc_ld(0);<br>for (i=0; i<L; i++)<br>   acc_mac(x[i], y[i]);<br>norm = norm_acc(); | Acc = 0;<br>for (i=0; i<L; i++)<br>acc += (x[i]*y[i])<<1;<br>calculate norm; | acc = 0;<br>for (i=0; i<L; i++)<br>acc += (x[i]*y[i])>>n;<br>calculate norm; |

| | | |
|---|---|---|
| result = L_extract_acc(norm); | result = acc<<norm; | result = acc<<(n+1+norm); |

In the example, the result is a fraction in Q31, and the exponent is *norm*.

More examples can be given for other types of operations such as double precision filtering.

From these examples, it is clear that the code is the same for 40-bit or 32-bit accumulation. It is only necessary to change the value of *PRODUCT_SHIFT* in order to select the target DSP operation. The precision will be different depending on the selected value of *PRODUCT_SHIFT*.

**Conclusion:**

We propose to add few more operators to the set of basic operators recently incorporated in the STL. A set of these operators permits 40-bit accumulation and 32-bit accumulation with right shift. Another set (3 operators) premits 32-bit accumulation without shift.

Accumulation without shift is useful for avoiding overflow problems in case of moderate accumulation. It also works with DSPs as well as general purpose processors (such as RISC and MMX). GPPs don't perform accumulation and shift in one cycle, and don't possess an overflow control mechanism. Using accumulation without shift is useful for developing standards targeted for both DSPs and GPPs.

Accumulation with 40 bits and 32-bit accumulation with right shift are useful for avoiding overflow in procedures needing more precision.

We invite the meeting to form an expert group to further discuss this issue and solicit more contributions from DSP experts.