GSM EFR IMPLEMENTATION FOR TRAU APPLICATION ON DSP16000

Junchen Du, George Warner, Erik Vallow, Penny Breyer and Tom Hollenbach

Wireless and Multimedia
Microelectronics group, Lucent Technologies
Allentown, PA 18103
E-mail: *jdu, warnergt, ev, pbreyer, tbach@lucent.com*

ABSTRACT

An implementation of GSM EFR using traditional single MAC DSPs (DSP1600) takes 24 MIPS to run a single speech channel. Lucent DSP16000 is a dual MAC high performance DSP. Analysis has shown DSP16000 code can run GSM EFR at 10 MIPS per channel with the same program space as DSP1600. The code has been re-structured to minimize delays and RAM usage for multi-channel TRAU applications. For a 100 MIPS DSP16210 running 6 EFR speech channels, DSP16210 code takes 15.7K words of RAM with 3.2ms maximum delay for the encoder and 0.5ms for the decoder. Without re-structuring, the numbers are 27K, 15.4ms and 1.5ms respectively. This makes the DSP16000 very attractive for TRAU applications. The initial implementation runs at 12.8 MIPS per channel with 19.3K words of RAM, 4.7ms maximum encoding delay and 0.5ms maximum decoding delay for 6 speech channels.

1. INTRODUCTION

Transcoder/Rate Adaptor Units (TRAU) are generally positioned remote to the Base Transceiver Station (BTS) [5]. The Channel Codec Units (CCU) are located in the BTS. In general, 16 kbits/s traffic channels can be used for full rate speech between the TRAU and BTS. By putting TRAU remote to the BTS, DSPs for speech coding can be utilized more efficiently to cut system cost. High performance DSPs, such as Lucent DSP16000, can be used to further cut the cost per speech channel.

Processing delay is a serious constraint for speech communication. One-way end-to-end delay of more than 150 ms can severally degrade the quality of real-time conversations.[6] The components of the total system delay includes the speech frame size, the look ahead, other algorithmic delay, multiplexing delay, processing delay for computation, and transmission delay. At the TRAU, the only delay that can be manipulated is processing delay.

The output of the speech decoder in the TRAU is coded according to CCITT Recommendation G.711 PCM format, and 64 kbits/s traffic channels are used [5]. 16 kbits/s traffic channels are used for full rate speech encoder output. The maximum processing delay means after this delay, the 16 kbits/s (or 64 kbits/s) traffic channel can continuously transmit speech data for the corresponding speech frame, or the speech data will be available at the time for transmission.

TRAU applications require each DSP to support as many speech channels as possible. But the resources of the DSP, such as the available MIPS and on chip RAM, are limited. Supporting the maximum number of speech channels under these limitations is the challenge.

This paper presents an implementation of GSM EFR codec on the Lucent Technologies' DSP16000. The original ETSI C code has been re-structured to address the issues of MIPS, RAM usage, and processing delay.

This paper is organized as the follows: Section 2 briefly discusses the architecture of the DSP16000. Section 3 gives the detail on how the ETSI C code is re-structured. The DSP16210 implementation results are presented in Section 4. The conclusion is given in Section 5.

2. DSP16000 ARCHITECTURE

The DSP16000 architecture is a modified Harvard architecture. There are two 16-bit $\times$ 16-bit multipliers, each capable of producing a full 32-bit result. The output of each multiplier can route to one of the three units: a 40-bit ALU (arithmetic logic unit) with ACS (add, compare, and select), a 40-bit three-input adder/subtracter, and a 40-bit BMU (bit manipulation unit). There are eight 40-bit accumulators, two 16-bit counters, eight 20-bit data pointers, two 20-bit coefficient pointers, and one dedicated stack pointer. Fig. 1 show a block diagram of DSP16000 data path. See [1] for more detailed features.

The instruction width is 16 or 32 bits. Up to 31 instructions can be loaded into a do-loop cache. By using cache, conflict wait-states between coefficients and instruction fetching can be eliminated. One of the most powerful instruction formats is

$$Accumulate \quad Product \quad Transfer \tag{1}$$

Although the instructions have 3 parts, the instruction width is 16 (or 32) bits, and takes one cycle. Part one, Accumulate, can do ALU and/or Adder operation, such as

$$aDE = aSE \pm p0 \pm p1 \tag{2}$$

$$aDE = aSE \pm p0 \quad aDP = aSP \pm p1 \tag{3}$$

where aDE, aSE, aDP, aSP is one of the 8 accumulators. p0 and p1 are 32-bit registers used to store the output of the

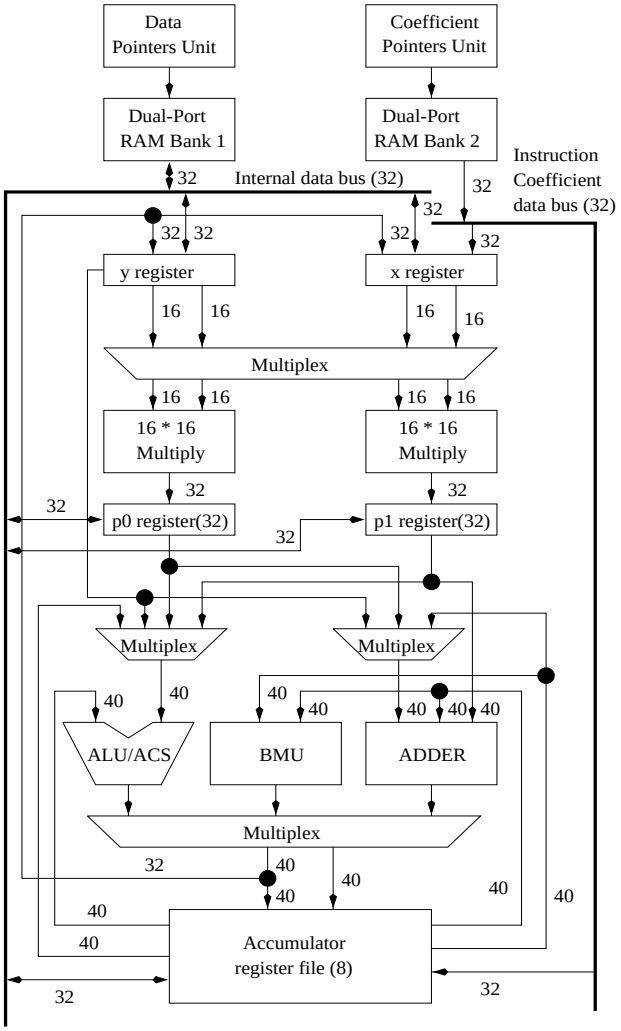

Figure 1: Overview of DSP16000 data path

multipliers. Part two, Product, can do up to two multiplies. For example,

$$p0 = xh * yh \quad p1 = xl * yl \tag{4}$$

where xh and yh are the high 16 bits of 32 bit register x and y, and xl and yl are the lower 16 bits. Part three, Transfer, can read in two 32 bits data or read in one 32 bit data and write out one 32 bit data, such as

$$y = YE \quad x = XE \tag{5}$$

$$YE = a6\_7h \quad x = XE \tag{6}$$

Where YE is one of the eight data pointers, and XE is one of the two coefficient pointers. a6_7h represents the 32-bit value resulting from the concatenation of the high halves of accumulators a6 and a7.

Some DSP algorithms require higher computation precision. To support this, DSP16000 provides instructions for mixed precision multiplication (16-bit $\times$ 32-bit) and double precision multiplication (32-bit $\times$ 32-bits). These instructions take one cycle and 2 cycles respectively. They have the same 3 part instruction format as expression (1). For example, the Accumulate and Product parts of a mixed precision multiplication instruction is

$$aDE = p0 + (p1 >> 15) \quad p0 = xh * yh \quad p1 = xh * (yl >>> 1) \tag{7}$$

where >>> is a logic right shift.

With all these features, Lucent DSP16000 architecture is ideal for vocoder applications.

3. RE-STRUCTURE OF SPEECH CODEC

ANSI-C code for GSM EFR is provided by the European Telecommunications Standards Institute (ETSI) [4]. On the encoder side, voice activity detection (VAD) and noise parameter encoding are combined with regular speech encoding. There are no logical break-points in the ETSI C code. No speech parameter will be ready for transmission until the encoder finishes the whole speech frame. On the decoder side, substitution and muting of lost frames, comfort noise generation are combined with regular speech decoding. No speech samples will be available until the last speech sample of that speech frame is decoded. This packing of speech encoder and decoder, if used in TRAU application with multiple speech channels, will cause large processing delay and require large RAM. Analysis shows that DSP16000 code can run at 10 MIPS per speech channel with 4.5k RAM without C code re-structure. For TRAU application with six speech channels, in the worst case, a 100 MIPS DSP16210 will require $4.5 \times 6 = 27k$ RAM with 15.4 ms encoding delay and 1.5ms decoding delay.

Both the encoder and decoder need to be split into smaller logic blocks in order to reduce the processing delay and RAM usage. In the worst case, our analysis shows that with codec re-structure, a 100 MIPS DSP16210 running 6 speech channels will only require 15.7K words of RAM with 3.2ms encoding delay and 0.5ms decoding delay.

Subsection 3.1 gives a brief description of GSM EFR. Then, Subsection 3.2 describes how we split EFR into these small blocks, which we call entry level modules.

3.1. Description of GSM EFR

GSM EFR is based on a code-excited linear prediction coding model (CELP). Analysis-by-synthesis search procedure is used to choose the optimum excitation sequence from the codebook. The frame size is 20 ms. A 10th order LP analysis is performed twice per frame with no look-ahead. The two sets of LP parameters are converted to line spectrum pairs (LSP) and quantized using split matrix quantization (SMQ) with 5 subvectors. The speech frame is divided into 4 subframes of 5 ms each. The two sets of quantized and unquantized LP filters are used for the 2nd and 4th subframes while interpolated LP filters are used for the 1st and 3nd subframes. An open-loop pitch lag is estimated twice per frame based on perceptually weighted speech. For each subframe, closed-loop pitch analysis is used to find the best pitch lag, followed by pitch gain search and quantization. The target signal updated by removing the adaptive (pitch)

codebook contribution is used in the fixed algebraic codebook search. Each algebraic codevector contains 10 pulses. The gain of the algebraic codevector is scalar quantized. Finally, the filter memories are updated for next subframe or frames. When there is no active voice, the encoder will transmit comfort noise at a regular lower rate. There would not be any subframe-based coding. Random number generators will produce random numbers for excitation parameters to update filter memories. A Silence Descriptor (SID) codeword together with LP code words will be transmitted.

The decoder first obtains speech parameters from the received bitstream. From the 2 LSP vectors, LP filter coefficients and interpolated LP filter coefficients are computed for each subframe. The excitation is constructed by adding scaled adaptive and algebraic codevectors. The speech is synthesized by passing the excitation through the LP filters. Finally, the restructed speech is passed through an adaptive filter.

3.2. Codec re-structure

To reduce the processing delay and RAM usage, GSM EFR is split into smaller logic blocks, which are called entry modules. Three memory sections, permanent memory, temporary memory, and stack memory, are used. The same stack memory can be shared for different speech entry modules (the same speech channel or different speech channels), and other application software. The temporary memory is maintained within each speech frame. Between speech frames, this temporary memory can be reused for other purposes. The permanent memory shall be maintained during the whole conversation period. This memory structure will not only reduce the total amount of RAM usage, it also reduces the conflict wait-states between coefficient and instruction fetching .

Ten entry modules are identified for EFR encoder:

EF_Pre_Proc_Entry: perform LP analysis and VAD detection. Convert 2 set LP parameters into LSPs.

EF_Vq_Sub_Entry: VQ quantization of LSP submatrices. The first, second, fourth, and fifth LSP submatrices are quantized by this modules.

EF_Vq_Subs_Entry: VQ quantization of the third LSP submatrix. A signed VQ codebook is used.

EF_Frame_Entry: perform open-loop pitch search, LSP interpolation and LSP to LPC conversion. If it is a SID frame, there will be no open-loop pitch search, and subframe-based parameters will be assigned values from SID codeword.

EF_Pitch_Entry: perform closed-loop pitch search. The adaptive codebook index (or relative index) will be returned. If it is a SID frame, no closed-loop pitch search will be performed.

EF_Pitch_G_Entry: perform pitch gain search. Pitch gain is scalar quantized with 4 bits If it is a SID frame, the gain is set to zero.

EF_Code_Entry: perform algebraic codebook search and return 10 pulse indices. If it is a SID frame, 10 pulses are generated from random number generator.

EF_Code_G_Entry: perform codebook gain search. The gain is scalar quantized with 5 bits. If it is a SID frame, comfort noise excitation gain is computed from LP residual energy.

EF_Subframe_Entry: update filter memories for next subframe.

EF_Pst_Proc_Entry: update memories for next speech frame.

The following is the calling procedure for the EFR encoder. The available parameters after the entry call are also shown:

1. EF_Pre_Proc_Entry
2. EF_Vq_Sub_Entry: index of 1st LSP submatrix
3. EF_Vq_Sub_Entry: index of 2nd LSP submatrix
4. EF_Vq_Subs_Entry: index of 3rd LSP submatrix
5. EF_Vq_Sub_Entry: index of 4th LSP submatrix
6. EF_Vq_Sub_Entry: index of 5th LSP submatrix
7. EF_Frame_Entry
8. Set subframe counter $i = 0$
9. EF_Pitch_Entry: pitch lag of i-th subframe
10. EF_Pitch_G_Entry: pitch gain of i-th subframe
11. EF_Code_Entry: 10 pulse indices of i-th subframe
12. EF_Code_G_Entry: Codebook gain of i-th subframe
13. EF_Subframe_Entry
14. $i = i + 1$, if $i < 4$ goto 9
15. EF_Pst_Proc_Entry

Three entry modules are identified for EFR decoder:

EF_D_Preprc_Entry: obtain speech parameters from received bitstream. If it is a SID frame, comfort noise parameters will be produced. If it is a bad frame, substitution and muting will be done at speech parameter level. It then performs LSP interpolation and LSP to LPC conversion.

EF_Decoder_12k2_Entry: perform subframe-based speech decoding. Speech is synthesized by passing scaled adaptive and algebraic codevectors through synthesis filter. A formant postfilter and a tilt compensation filter are used to enhance speech quality.

EF_D_Pst_Proc_Entry: update memories for next decoder frame.

The following is the calling procedure. The available speech samples after the entry call are also shown:

1. EF_D_Preprc_Entry
2. Set subframe counter $i = 0$
3. EF_Decoder_12k2_Entry: 40 speech samples
4. $i = i + 1$, if $i < 4$ goto 3
5. EF_D_Pst_Proc_Entry

4. DSP16210 IMPLEMENTATION RESULTS

The re-structured GSM EFR C code is implemented on DSP16210. The results of the initial GSM EFR implementation on DSP16210 are shown in Table 1. These number are the maximum cycles and average cycles per call for each entry. Entry EF_Vq_Sub_Entry is used for different LSP submatrices with different codebook sizes. The number for each case is listed in the table.

Entry name	Maximum	Average
EF_Pre_Proc_Entry	34225	31465
EF_Vq_Sub_Entry (1st)	1046	1046
EF_Vq_Sub_Entry (2nd)	1940	1940
EF_Vq_Subs_Entry (3rd)	3737	3734
EF_Vq_Sub_Entry (4th)	1942	1942
EF_Vq_Sub_Entry (5th)	597	597
EF_Frame_Entry	27258	26879
EF_Pitch_Entry	7906	6256
EF_Pitch_G_Entry	2104	1705
EF_Code_Entry	27821	24091
EF_Code_G_Entry	920	811
EF_Subframe_Entry	1171	1098
EF_Pst_Proc_Entry	1360	313
EF_D_Preprc_Entry	2609	2361
EF_Decoder_12k2_Entry	5222	4693
EF_D_Pst_Proc_Entry	1177	99

Table 1: Cycles per call for each entry module

If we add up all the maximum numbers, GSM EFR runs at 12.8 MIPS (peak) per speech channel. It is assumed that the decoder has a higher priority, which means whenever a decoder channel is ready to decode, any running encoder shall be interrupted. In general, when an entry module is finished, the scheduler (or real-time operation system) will decide which entry module to run next. For a 100 MIPS DSP16210 running 6 speech channels, in the worst case, the maximum processing delay for one decoder channel occurs while waiting for each of the other 5 decoder channels to decode one subframe of speech, plus the delay to decode the waiting subframe of speech. The maximum delay for the decoder is 0.5ms. For the encoder, assuming 16 kbits/s traffic channels are used, the bottle neck for maximum delay is a EF_Code_Entry call in the first subframe. The maximum delay for the encoder is 4.7ms.

In this initial implementation, the permanent memory size is 926 words for encoder and 416 words for the decoder. The temporary memory size is 1395 words for the encoder and 207 words for the the decoder. The stack memory size is 1886 words for encoder and 242 words for the decoder. The total RAM usage for 6 speech channels is $(926 + 416 + 207 + 1395) \times 6 + 1886 + 242 = 19.3k$ words.

To give a comparison between DSP1600 and DSP16000, Table 2 shows the maximum cycles for some key EFR modules. Search_10i40 performs an algebraic codebook search for 10 pulses. Vq_subvec does VQ quantization for LSP submatrices. Vq_subvec_s performs a signed VQ codebook search. Convolve performs convolution between two 40 sample vectors. Syn_filt performs a 10-th order IIR filtering and Residu performs a 10-th FIR filtering. From Table 2, it can be seen that the speed improvement of DSP16000 over DSP1600 is between 2.1x and 3.9x for computation intense DSP algorithms.

Subroutine name	DSP1600	DSP16000
Search_10i40	37359	13059
Vq_subvec	7635	1940
Vq_subvec_s	13869	3737
Convolve	1337	604
Syn_filt	843	373
Residu	622	290

Table 2: Comparison between DSP1600 and DSP16000

5. CONCLUSION

The DSP16000 shows very promising results for TRAU application. For a 100 MIPS DSP16210, the initial implementation runs at 12.8 MIPS per channel with 19.3K words of RAM, 4.7ms maximum encoding delay and 0.5ms maximum decoding delay for 6 speech channels. Our analysis shows that with further optimization, the RAM usage can be reduced to 15.7K, and the maximum delays for the encoder can be reduced to 3.2ms.

6. ACKNOWLEDGEMENTS

The authors thank David Cooley, David Bishop, and Mark Warner from Lucent Technologies for providing insight suggestions for the codec re-structure. They also thank Berkerley Design Technology Inc. (BDTi) for coding some EFR subroutines in DSP16000 assembly language.

7. REFERENCES

[1] "DSP16000 Digital Signal Processor Core Information Manual", Microelectronics group, Lucent Technologies, Feb. 1997.

[2] GSM 06.60, "Digital Cellular telecommunications system: Enhanced Full Rate (EFR) speech transcoding"

[3] GSM 06.82, "Digital Cellular telecommunications system: Voice Activity Detection (VAD) for Enhanced Full Rate (EFR) speech traffic channels".

[4] GSM 06.53, "Digital Cellular telecommunications system: ANSI-C code for the GSM Enhanced Full Rate (EFR) speech codec"

[5] GSM 08.60, "European digital cellular telecommunications system: Inband control of remote transcoders and rate adaptors for full rate traffic channels"

[6] R. V. Cox, B. G. Haskell, Y. Lecun, B. Shahraray and L. Rabiner, "On the Applications of Multimedia Processing to Communications", Proc. IEEE, Vol. 86, No. 5, pp. 755-824, May 1998.