Design of A Generic CELP Architecture

Madhusudhan.S*, Monga.S*, Ramakrishna.S.T.G*, Prof.Jamadagni.H.S*, Dr.AshokRao*

Abstract

Modern mobile communications require optimum bandwidth utilization with minimum loss, delay and good quality of speech transmission. This triggered the usage of low bit rate voice Codecs among which CELP Codecs inherit the merits of both Waveform and Source codecs, like toll quality, low bit rate etc.

In this paper an attempt has been made to form some special generic hardware blocks for the ITU G.729 standard (CS-ACELP, Conjugate Structure Algebraic Code Excited Linear Prediction) of 8Kbps bit rate CELP algorithm. This is aimed at overcoming the limitation of computational burden and also scaling this application for enhanced speed and extracting more number of channels.

Keywords: CS-ACELP, DSP, Parallelism, and Pipelining.

1. Introduction

Up to now we have seen most of the speech codecs are implemented in general purpose programmable DSP, which supports only few numbers of channels. Among the present day codecs.,[1][2] CELP codecs can achieve high-speech quality at extremely low bit rates. But the formidable computational complexity limits it from supporting more numbers of channels with the programmable DSP.

This motivates the IC designers in developing the hardware accelerators or HW-SW co-design methods In this paper, G.729 (CS-ACELP) ITU-T standard[3][4] is taken as reference. The G.729 CS-ACELP algorithm is used today in applications requiring high quality speech compression such as Video Conferencing, VON (including VoIP, VoDSL, VToA), Digital Satellite Systems, Digital Mobile Radios (specially for Military use), PSTN, and ISDN. Yet, as is exemplified by G.729, as compression ratios, algorithmic delay and the subjective quality of speech compression algorithms improve, the complexity of real-time implementation generally increases. It is only with the advent of high performance, low cost DSPs that these implementation can be realized in practical systems. Furthermore, optimized assembly language programming today reminds the only viable route to a cost effective single chip realization of a full-duplex G.729 CS-ACELP voice Codec. Generic DSP cores are needed to allow customized development of function and application specific integrated circuits (FASICs) with high degree of integration to enable CELP Codecs to be brought to market.

The whole algorithm was first implemented as per the ITU G.729 Standard, its Source code is decoded and the data flow diagrams were extracted. Later these data flow diagrams where individually mapped to hardware. Most of the hardware blocks are weird structures. Optimization was done twice over, first individually for each block and then overall as a system. This work is aimed to get some experience and a good starting point for the VLSI implementation of Speech Codecs.

This paper is organized as follows: In section 2 we explained generation of hardware blocks for both encoder and decoder of G.729 .Design optimization with specialized arithmetic units is explained in section 3.Estimation of clocks with serial ,data parallelism, data and algorithmic parallelism described in section 4.Design offsets are described in section 5 and scope for the future work in section 6.

* CEDT, I I Sc, Bangalore.
{ smadhu, smonga, sramki, hsjam, ashokrao }@cedt.iisc.ernet.in

2. Hardware blocks generation

2.1 Algorithm to Hardware blocks mapping

The whole algorithm was first interpreted as per the ITU G.729 standard ,its C-code is decoded and the data -flow diagrams[5] were extracted. Optimization was done twice over, first individually for each block [6]and then overall as a system. Design of the hardware blocks for each stage of the algorithm was implemented by mapping of the algorithm to hardware in the following two ways.

2.1.1Equation to Hardware blocks mapping

The ITU standard for G.729 was used for extraction of the equations for various stages of the algorithm. These equations were then written in expanded form for each iteration. This enabled us to extract data flow graphs, movement of data in each of the subsequent step, data parallelism, Hardware redundancies ,availability of intermediate data which could be utilized later .One of the examples is given below.

$$w_1 = 10(\omega_2 - (0.04*\pi + 1))^2 + 1, \quad \omega_2 - (0.04*\pi + 1) < 0$$
$$w_2 = 10(\omega_3 - (\omega_1 + 1))^2 + 1, \quad \omega_3 - (\omega_1 + 1) < 0$$
$$\vdots$$
$$w_9 = 10(\omega_{10} - (\omega_8 + 1))^2 + 1, \quad \omega_{10} - (\omega_8 + 1) < 0$$
$$w_{10} = 10(0.92*\pi - (\omega_9 + 1))^2 + 1, \quad 0.92*\pi - (\omega_9 + 1) < 0$$

NOTE :- Weights w_5 and w_6 are each multiplied by 1.2.

The above are the equations of weights calculation written from the general equation given by ITU which are used in selection of the indices of the codebook 2 and codebook 3.After going through these equation s, the following hardware block Fig 1 is generated which takes less number of clock cycles (60) when compared to software implementation.

2.1.2 C-code to Hardware blocks mapping

At various stages of the data flow ,the ITU mentioned the nature of manipulation of the input to get the desired output .The mapping of the C-code ,at such stages, became unavoidable . In Fig 2 the highlighted part reflects the code given below. However the C-code has been mapped after optimizing it as per hardware requirements .

Hardware block for finding whether taming is require or not.

C-code:

```
check  if taming of gain is required or not
        if (frac > 0)
        then T0 = T + 1 else T0 = T
        if (T0 < 50), i = 0 else i = T0 - 50
        zone 1 = tab_zone[i]  (lower limit of search)
        zone2 = tab_zone[i]   (upper limit of search)
        if (l_maxloc > L_THRESH_ERR)
    flag = 1  taming required
        if(flag = 1) && (gp > 0.95)
        then gp = 0.95
```

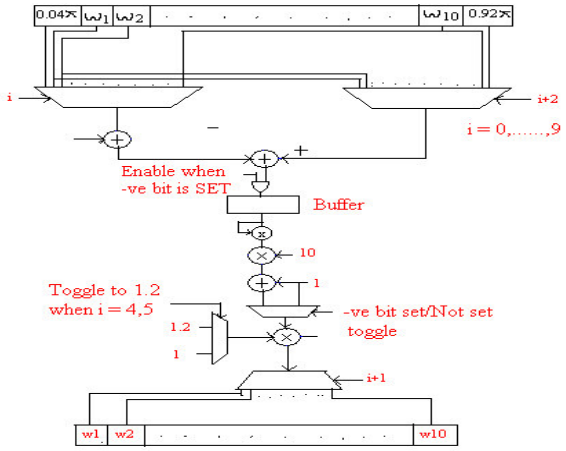


Fig.1 Calculation of weights in Code-book index generation

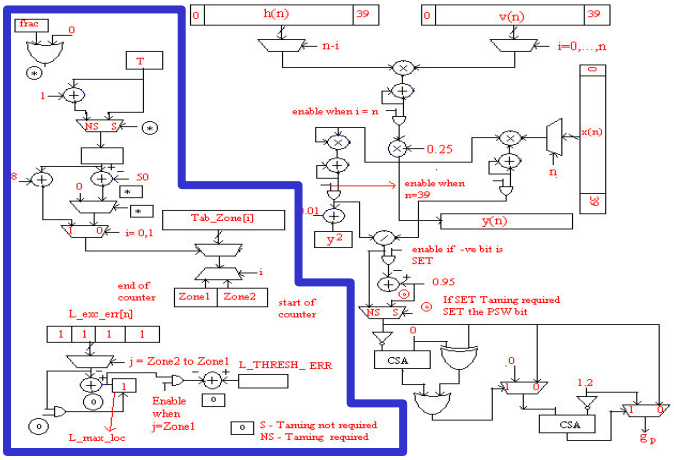


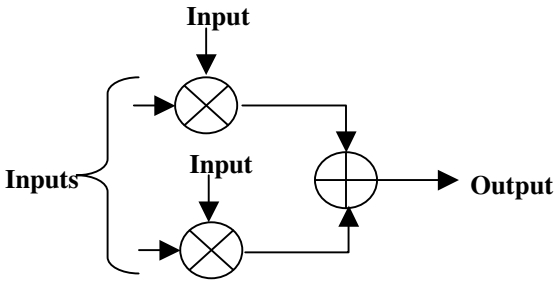
Fig.2 Hardware block for finding whether taming is required are not

3. Design Optimization

The designing of the hardware blocks individually was followed after the exercise of making dataflow and control flow diagram[7] for the entire algorithm. This provided the information for identification of specialized arithmetic units appearing extensively in the algorithm and also helped in tracking data parallelism and the algorithmic parallelism.

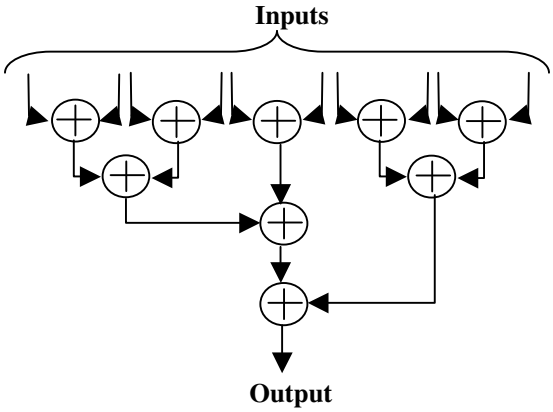
3.1 Specialized Hardware Arithmetic Units

3.1.1 Two Multipliers and Adder Combination



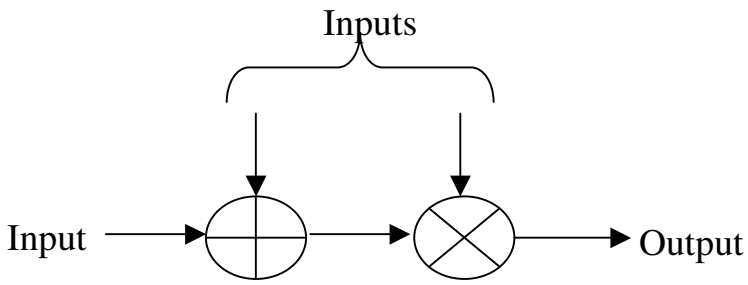
This unit occurs frequently for manipulation of data like in the calculation of elementary LSF, Updating of excitation blocks etc.

3.1.2 Four Level Adder Block



This arithmetic unit occurs frequently for adding five inputs to produce a single output, used extensively in synthesis and residual filters like calculation of impulse response, target signal, updating of memory buffers etc and also used in certain other blocks like selection of codebook 1,2 and 3.

3.1.3 An Adder and Multiplier Combination



This arithmetic block occurs frequently for manipulation of data and used extensively in the entire algorithm.

4. Estimation of Clock cycles

Any hardware design is incomplete without an estimation of the clock cycles recorded from the start of the first input till it ends at the output. CELP algorithm though basically meant for low bit rate speech transmission, it is computationally intensive, implying either the use of very high clock frequency DSP or exploitation of data and algorithm parallelism and pipelining .The total clock cycles for the individual blocks were calculated as per following assumptions.

Unit	No of Clock Cycles
MAC Unit	1
Adder Structure	1
Multiplier Structure	1
Four level Adder Block	4
Adder	1
Subtraction	1
Multiplier	1
Division	4
Square root	3
Logarithm	5
Power	5

Table1:Assumptions for estimation of clock cycles for both encoder and decoder

Explanation of estimation of clock cycles required for selection codebook1 index is given below.

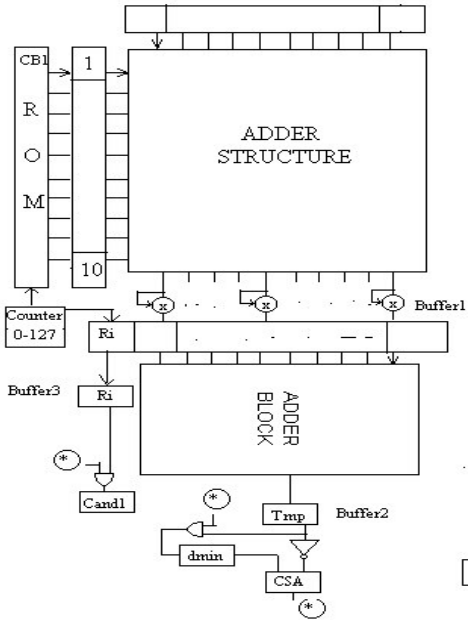


Fig 4: Hardware block for selection of index codebook 1

C-code:

```
    cand = 0
 dmin = MAX_32
 for  row = 0 to (128-1)
 {    for column = 0 to (10-1)
      tmp=(lsp_ele[col]-lspcb1[row][col])^2 + tmp
      if(tmp-dmin)< 0 then  dmin = tmp;
     cand = row;
 }
```

The codebook1 consists of 128 vectors each 10 elements. Index is one which gives minimum mean square error when comparing elementary line spectral pairs and codebook vectors. The number of clock cycles by processing the C-code is **7700**({[2*10+10]*128+10}*2).By using specialized arithmetic units with data parallelism ,the clock cycles reduced to **1794**([7*128+1]*2).The clock cycles still get reduced to **897**(7*128+1) by applying algorithmic and data parallelism[7]. The block which consuming more number clock cycles in encoder is open loop pitch analysis(**serial – 20460** clock cycles ,**Data parallelism – 10540** clock cycles, **Algorithm + data parallelism – 5440** clock cycles). Following is the summary of the total number of clock cycles for encoder and decoder.

Methods	Encoder Clocks	Decoder Clocks
Serial	88664	29395
Data parallelism	37345	14725
Algorithmic + Data parallelism	24576	13979

Table2 : Total number of clock cycles for both encoder and decoder

5. Design offsets

Hardware solutions for complex, computationally intensive algorithms like CELP can be faster than their software counter parts .They perform better and more cost effective. Hardware solutions may be less expensive than they appear at first glance.

The general blocks designed for CELP architecture can also be used for related algorithms implementing filtering of large amounts of data and other codecs like the GSM etc. The clock cycles obtained above brings out the large penalty we have pay ,in terms of silicon area, for greater area, for greater speed by exploiting parallelism .The amount of resources required for implementing the algorithm with only data parallelism is much less as compared to its implementation as a combination of data and algorithmic parallelism .

The total number of clock cycles for the decoder, for data and algorithmic combination implementation can be reduced further by 777 cycles , if pipelining of data is used in the multiplier structure and four level adder block.

Another important issue, which emerges, is that there exist a large variation in the time taken for manipulation at various stages of the algorithm. This would require defining of different clock domains for the architecture.

6. Conclusions:

A large number of filters have been used in this algorithm. Ensuring their stability is very important for good quality operation of the codecs. This can be done by proper choice of word lengths and using clipping circuit to limit the maximum values of the synthesized speech signals. Suggested optimal word length is 16 including the signal bit. The same approach can be applied to any hardware accelerator design.

7. References:

1) A.M. Kondoz (1994) “Digital Speech “,John Wiley & Sons .
2) R.Schroeder and B.S.Atal, (1985) ”CELP: High quality speech at low bit rates”, proceedings of ICASSP,pp.937-940
3) Redwan Salami et. all (March 1998), “ Design and Description of the CS-ACELP : A Toll Quality 8Kb/s Speech Coder” , IEEE Transaction on Speech and Audio Processing Vol.6.No.2 pp.116-129.
4) ITU-T Draft Recommendation G.729, “ Coding of speech at 8Kbps using the Conjugate Structure Algebraic Code Excited Linear – Prediction (CS- ACELP)”
5) Keshab K. Parhi ,(January 1999), “VLSI Digital Signal Processing Systems: Design and Implementation”
6) Sherif Galal and Maged Attia , University of California , Los Angeles, “Project Report : Hardware Implementation of LPC Speech Coder”.
7) Computer Organization and Design : The Hardware/Software Interface by David A. Patterson, John L. Hennessy ,Morgan Kaufmann Publishers; 2nd edition (September 1997)