THE ADSP-21535 BLACKFIN AND SPEECH CODING

Mikael Olausson

Computer Engineering
Department of Electrical Engineering
Linköpings universitet
SE-581 83 Linköping, Sweden
{mikol}@isy.liu.se

Dake Liu

Computer Engineering
Department of Electrical Engineering
Linköpings universitet
SE-581 83 Linköping, Sweden
{dake}@isy.liu.se

ABSTRACT

This paper will evaluate the new processor ADSP-21535 Blackfin from Analog Devices in terms of speech coding. Three speech coding algorithms has been studied, G.723.1, 6.3 kbit/s and 5.3 kbit/s and G.729 8 kbit/s. All these three algorithms are proposed by the H.323 standard together with G.711 64 kbit/s and G.728 16 kbit/s. I will discuss some important issues of these speech coding algoriths and compare them with the capabilities of Blackfin. The work has been done by thoroughly examining the fixed point source code from ITU, International Telecommunication Unions [1], [2] and by the instruction reference manual [3] and the hardware reference guide [4].

1. INTRODUCTION

The market for voice over Internet protocol, also called VoIP, has increased over the years. Voice has been a natural choice of communicating for a long long time and will continue to be so. The H.323 standard contains four different speech coders with different complexity and bit rates. The first one is G.711, which is mandatory and uses A/u-law compression at 64 kbit/s. Another coder is the G.728 Adaptive differential PCM (ADPCM) at 16 kbit/s. The last two are more interesting if we are dealing with bandwidth limited transmission channels. These are G.723.1 and G.729. While the first one have two different bit rates specified, 6.3 and 5.3 kbit/s, the last have three different, 6.4/8.0/11.8 kbit/s. These two both have parts that are common, but also parts that differ a lot. From a market point of view it is of highest interest to make the implementations of these algorithms as efficient as possible. A couple of factors may influence the choice of algorithm. For example some users want to squeeze as many channels as possible on a limited transmission channel. Then their choice is as low bit rate as possible if the speech quality is good enough. Others might use them in battery powered applications and their aim is low power consumption with reduced speech quality as a tradeoff.

2. GENERAL DESCRIPTION OF G.723.1 AND G.729

The first difference between the G.723.1 and the G.729 is the frame size. While the G.723.1 is based on 30 ms(240 samples), the G.729 is based on 10 ms(80 samples) frames. The delay of the algorithms are 37.5 ms and 15 ms respectively. Both algorithms includes a high pass filter to get rid of undesired low frequency components. The short term analysis is based on 10th order linear prediction (LP). For G.729 the LP is calculated for every frame, 10 ms, while for the G.723.1 they are calculated for every subframe 7.5 ms. Both uses the Durbin-Levinson algorithm to calculate the coefficients. The unquantized coefficients are then transferred to Linear Spectral Pairs (LSP). For the G.723.1, are only the LP coefficients from the last subframe transferred. These LSP are then quantized using a two-stage Vector Quantization(VQ) for G.729 and a Predictive Split Vector Quantizer(PSVQ) for G.723.1. The excitation parameters from both the fixed and the adaptive codebook are determined on subframe basis. In G.729 the frames are divided into two subframes of 5 ms each, while in the G.723.1 each frame are divide into 4 subframes of 7.5 ms each. Both codecs uses an open-loop approach to calculate the pitch delay. For the G.729 this delay is estimated once every frame, while it is estimated every 15 ms(every second subframe) in the G.723.1. This pitch value is then refined in the closed-loop pitch analysis. The closed-loop analysis is done for every subframe for both codecs. The gains and the pitch delay are referred to as adaptive codebook parameters. The fixed codebook extraction is the most exhaustive part of the whole algorithm. In these two codecs there exist two different approaches. One which are used in both the lower bit rate of G.723.1 and in G.729 and is Algebraic-Code-Excited-Linear-Prediction, (ACELP). This ACELP places at most 4 non-zero pulses within a subframe. The positions are determined from the codebook. The second approach is to use Multi-Pulse Maximum Likelihood Quantization (MP-MLQ). In this case you have more opportunities to place the pulses more individually and not based on a algebraic

	G.723.1 (5.3/6.3)	G.729 (8.0)
Transmission rate	5.3/6.3 kbit/s	8.0 kbit/s
Algorithmic delay	37.5 ms	15 ms
Frame size	30 ms	10 ms
# subframes	4	2
LP calculation	every 7.5 ms	every 10 ms
Open-loop pitch estimation	every 15 ms	every 10 ms
Closed-loop pitch analysis	every 7.5 ms	every 5 ms
Fixed-codebook excitation	ACELP/MP-MLQ	ACELP

Table 1: General description of G.723.1 and G.729

codebook.

3. THE ADSP-21535 BLACKFIN

Traditionally, DSPs, Digital Signal Processor, has focused on number crunching, while a MCU, Micro Controller Unit, has concentrated on performing control functions. This has lead to two different branches of processor development. It is usually a bad idea to perform control functions on a DSP and vice versa. At the same time one need both control functionality and computing power in a system. In the Blackfin processor one tries to combine them both. Instead of adding DSP functionality to a MCU, they have added control functionality to the DSP. The instruction size is either 16 bits for control functions or 32 bits for DSP functions. This makes the code very flexible and dense. You can also execute instructions in parallel. One DSP instruction and two control functions at the same time. This gives an instruction length of 64 bits. It is not allowed to issue two DSP instructions in parallel neither is more than two control instructions in parallel allowed. The architecture is not superscalar and cannot perform out of order execution. In a MCU, it is preferable to have a short pipeline due to all the changes in the program flow. On the other hand is a deep pipeline almost mandatory in order to get high throughput in heavy computations. Deeper pipeline can give higher clock speed. The Blackfin is running at 300 MHz. This is two factors that cannot be united. In Blackfin they have an eight stage pipeline. To reduce the penalty in branches, they have included static branch prediction. The Blackfin is a 32-bit architecture with a double MAC, two 40-bit ALUs and one 40-bit shifter and four 8-bit video ALUs.

4. STATISTICS OF BASIC OPERATIONS

All the arithmetic and logic functions like add, sub, shift, multiply and so on are implemented with a standard C library. This makes it simply to do statistics over how many times different functions are used. Additional to this, the C code has been thoroughly examined and all the branch, loop and move functions have also been identified and classified. All these statistics over basic operations, branch, loop and move instructions give a good hint on where to find improvements on instruction and architecture level. The statistics are presented in [5]. From the statistics one can see that multiplications or multiply-and-accumulate is the dominant operation, around 1/3 of all operations.

5. SPEECH CODING

In this section I will bring up some aspects of speech coding and the computinal demands. First of all we have make clear that we are dealing with fixed point implementations. The Blackfin does not have any floating point unit. The Blackfin processor is a 32-bit architecture. The data format can be 32, 16 or even 8 bit wide. This makes it suitable for many applications. When performing 16-bit operations, there is support for both dual and quad processing. Data loads and stores are also 32 bit wide. This together with the dual MAC makes for example filter calculation much faster. Windowing is also done in half the time due to the dual processing. 32-bit additions and subtractions are supported directly by the ALU. In an ordinary 16-bit DSP you will have to do upper and lower halves separately. The same is true for the shifter and absolute value calculations. With the 32-bit instructions comes also saturation arithmetic. This is much better than using software solutions. Of course will the multiplier allow both signed, unsigned, fractional and integer operations. The execution times is also reduced by the fact that instructions can be issued in parallel. One 32-bit DSP instruction in parallel with two 16-bit control instructions.

As mentioned before is the Fixed-Codebook excitation the most extensive part of the whole algorithm. The calculations are done by Analysis-by-Synthesis. This means that we try all possible combinations or at least a certain amount. We then choose the one with the smallest error signal. We need instructions with the following functionality:

```
a16 = abs_s(a16);
if ( a16 > b16 )
   b16 = a16;
```

This is covered by the MAX instruction in the Blackfin. The absolute value is not incorporated, but Blackfin can perform two comparisons and two decisions within the MAX

instruction. This is also possible for comparisons lesser than. This instruction is very useful when you are searching an array for the largest value. This is done in speech coding together with scaling to avoid overflow. An even more important instruction is based on 32-bit decisions. It looks like the following:

```
for-loop with index i
basic operations
.
.
.
a32 = L_abs(a32); Optional
if ( a32 > b32 )
    b32 = a32;
    store index i;
end of if-statement
end of for-loop
```

This is common in codebook search. You try one entry, perform all the synthesis calculations and get a synthesized signal. Compare this signal with the original one and calculate an error measurement like Least-Mean-Square(LMS). This value is then compared to the best one so far. Except from the error value itself it is also important to store the entry of the codebook for the best choice. Either the loop counter or the a pointer to the memory. The blackfin processor has incorporated such an instruction, but only for 16-bit values. It is called Vector SEARCH. This instruction can store both the best error value and a pointer to the memory. Actually, it can perform two of these calculation within an instruction. The drawback is that it only works for 16-bit values. With 32-bit values you have to solve it in software with compare instructions and conditional moves. There is a 32-bit MAX instruction, just like for 16 bits. But there is no good way to store the loop counter or the a pointer. With an instruction that merge this into one instruction, one can save up to 15% of the total execution time [5]. A propose of a hardware architecture that will merge these instructions into one is shown in figure 1. The 32-bit full adder (FA) on the left in the figure is used for absolute calculations and register R4 is used as a reference register for the branch instruction. In addition to this a control signal must be sent to the register file if a new loop counter value has to be stored. The path delay from ACR via the 32-bit full adder (FA) and the accumulator to the accumulator register (ACR) is less the path delay through the multiplier. This extra hardware will not add extra delay to the system.

It is not just the calculations themself that consumes clock cycles. The address calculations can be pretty tricky as well. A DSP has good support for addressing the memory. Modulo addressing, bitreverse addressing, autoincrement and decrement and more than one address register. But sometimes can the addressing be more complex than the actual calculation. Look at the following:

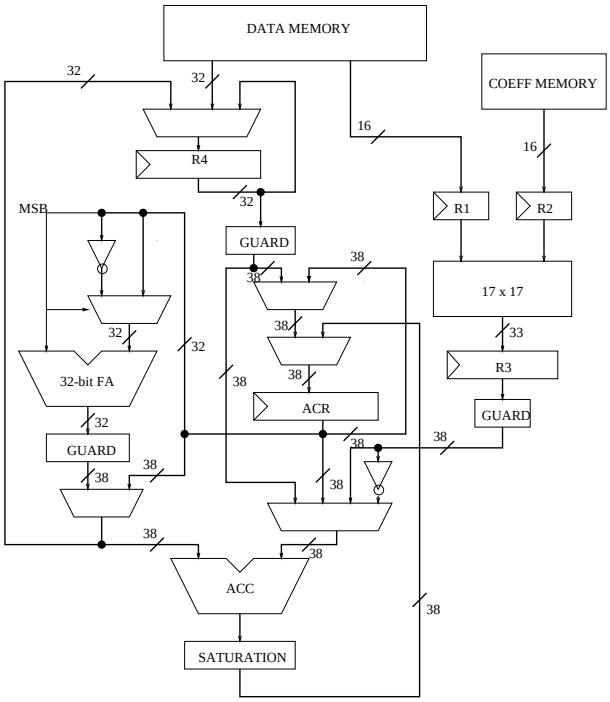

Figure 1: A 32-bit conditional move with absolute value and loop counter move.

```
for ( l = 0 ; l < 60 ; l += 2 ) {

    if ( OccPos[l] != (Word16) 0 ){
        continue ;
    }

    Acc0 = WrkBlk[l] ;
    Acc0 = L_msu( Acc0, Temp.Pamp[j-1],
        ImrCorr[abs_s((Word16)(l-Temp.Ploc[j-1]))] ) ;
    WrkBlk[l] = Acc0 ;
    Acc0 = L_abs( Acc0 ) ;

    if ( Acc0 > Acc1 ) {
        Acc1 = Acc0 ;
        Temp.Ploc[j] = (Word16) l ;
    }
}
```

This loop will in the worst case be entered 288 times. The last part of this loop, from the L_abs instruction, where dis-

cussed in the previous section. We can not use an ordinary pointer and just post increment after data fetch due to the absolute value. The trick in software is to split the loop into two parts. One where l-Temp.Ploc[j-1] is always positive and the other where it is negative. When we have made this separation, it is correct to use the autoincrement or autodecrement functionality of address registers. The hardware solution is instead segmentation addressing with offset [6]. The principle of this addressing is shown in figure 2 and the offset calculation is shown in figure 3. The value stored in Temp.Ploc[j-1] is constant during the whole loop and will be stored in a register, REG in figure 3. To get an efficient calculation of the offset, we have to use a loop counter with variable step size. In this case the step size needs to be two.

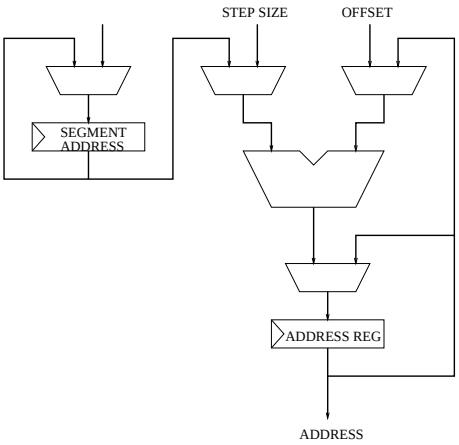

Figure 2: Address generator with offset addressing and segment addressing.

6. CONCLUSION

The Blackfin processor is a big improvement in calculation capacity compared to an ordinary 16-bit DSP. The 32-bit architecture allows dual operations and 32-bit loads and stores from memory and registers, which is a speed up. This together with special instructions like MAX makes it suitable for speech coding. The drawback is the lack of more complex addressing schemes and to migrate some of the special instructions for 16-bit operands to 32-bit operands. Especially the Vector SEARCH instruction. Searching for the best entry in a table and at the same time store the lowest error measurement value is a common function in speech coding. We have also looked into some hardware structures that will solve these problems. These are of course very specialized and would not fit into a general processor like Blackfin. Blackfin is not just a DSP processor it is also a MCU at the same time. The idea is to integrate a DSP and MCU into one processor.

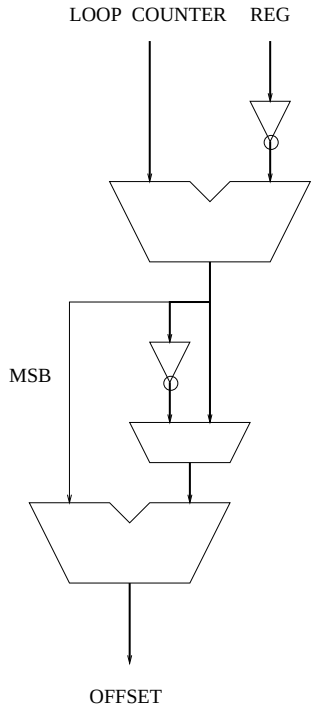

Figure 3: Offset calculation with loop counter and absolute value operation.

7. REFERENCES

[1] Itu-t recommendation g.723.1, dual rate speech coder for multimedia communications transmitting at 5.3 and 6.3 kbit/s, 1996.

[2] Itu-t recommendation g.729, coding of speech at 8 kbit/s using conjugate-structure algebraic-code-excited-linear-prediction (cs-acelp), 1996.

[3] *Blackfin DSP Instruction Set Reference*. Part number 82-000410-14 edition, 2002.

[4] *ADSP-21535 Blackfin DSP Hardware Reference*. Part number 82-000410-13 edition, 2002.

[5] M. Olausson and D. Liu. Instruction and hardware accelerations in G.723.1(6.3/5.3) and G.729. In *The 1st IEEE International Symposium on Signal Processing and Information Technology*, pages 34–39, 2001.

[6] M. Olausson and D. Liu. Instruction and hardware accelerations for MP_MLQ in G.723.1. In *2002 IEEE Workshop on Signal Processing Systems Design and Implementation*, pages 235–239, 2002.