SOFTWARE IMPLEMENTATION OF ADSL APPLICATION WITH A CONVOLUTION CO-PROCESSOR

E. Dujardin and O. Gay-Bellile

Laboratoires d'Électronique Philips S.A.S.
22, av. Descartes
94453 Limeil Brévannes- France
e-mail: dujardin@lep-philips.fr, gaybel@lep-philips.fr

Abstract

More and more applications have software implementations in order to cope with the cohabitation of several emerging standards and the fast evolution of consumer products. We show in this paper how to implement efficiently digital communications applications into DSPs with the help of the Convolution Co-Processors, which is described in this paper, and how co-processors are useful to empower DSP performances at a small cost. ADSL application is taken as an example for broadband communications.

Introduction

With the emergence of interactive television and other new digital services, the broadband communications become necessary. Nowadays, different standards (IEEE, ADSL...) emerge and they may cohabit on the market place for several years. Therefore the programmability aspect of broadband communications systems is important in order to provide one multistandard platform. There is a great interest of the market into broadband communications systems like ADSL (Asymmetric Digital Subscriber Lines) as well as a few others. ADSL is considered as a mean to provide the bandwidth required for digital interactive video and entertainment services via the existing access networks [2] [4]. In quite a different field, this technology is mentioned as a solution to bring a fast access to Internet services at speeds which are 100 to 1000 times as fast as telephone line V34 modems.

This paper is organized as follows: the first section describes the ADSL application. The second section describes the performances of commercial DSPs on filter implementation. The third section describes the co-processor architecture and its performances. The fourth section presents the VLSI complexity of this co-processor. The fifth section compares the performances with and without co-processor. And in the last section, we conclude.

1. THE ADSL APPLICATION

The Asymmetric Digital Subscriber Line (ADSL) system provides a 1.536 to 6.144 Mbit/s downstream channel toward the customer installation and a low bit rate (from 176 up to 640 kbit/s) upstream channel toward the central office. The transport medium is the conventional telephone network. The highest frequencies are occupied by the downstream channel. The lowest frequencies are used for the return channel. The signal is sampled at 2.208 MHz. Both channels do not interfere with the voice band and allow the use of ADSL transmission simultaneously with phone transmission.

The ADSL technique begins by studying the phone line quality between the server and the user. The results are that each carrier is affected with a number of bits to transmit and an amplification. The figure 1 shows the different functional units of an ADSL modem. Only the subscriber part is studied in this paper since it is the most demanding computation because of its sampling frequency. The demodulator is composed of a Time Equalizer (TEQ), which includes two filters. The first one, $H(z)$, is a standard filter with 128 taps. The second one, $B(z)$, is an adaptive filter with 32 taps; its updating is made every 69 blocks where a block corresponds to 256 samples. On each block, a FFT is computed and on each carrier a gain control is computed. The carriers are the representation in frequential domain of samples. The demapping retries the information bits frmo the carrier. A Forward Error Correction, which is not shown in this figure, retrieves the original data.

In the modulator part, data are encoded to provide a protection against errors. The mapper puts information bit onto the carriers and an amplification is calculated on each carrier. Both operations are done according to the results of the phone line evaluation. Next an I-FFT is computed on 32 data and the result is sent through the phone line.

A study on the computation power shows that all the ADSL modem functionalities require 800 Mops while 85% are taken by the TEQ.

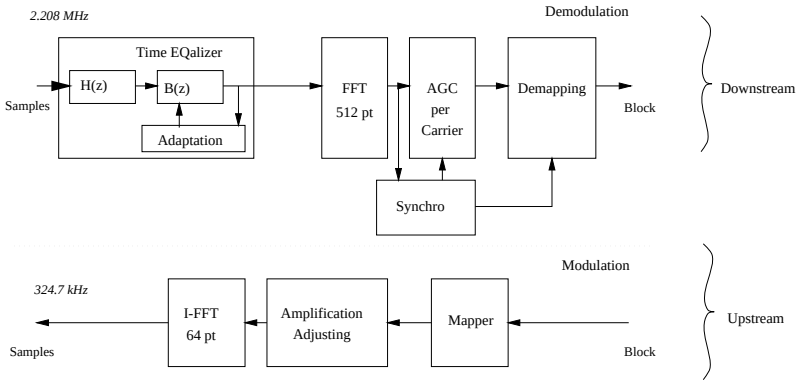

Figure 1: Global diagram of subscriber's ADSL modem

Now we study the requirements of filter computation on commercial DSPs.

2. DSP PERFORMANCES

Both Philips' TriMedia [5] and Texas Instruments' C6x are VLIW processors. However, they have significant differences. TriMedia has a register file with 128 words of 32 bit and can execute 5 operations concurrently. Moreover, it can use a multiplier like a FIR filter of two taps on 16-bit data accuracy. Since it has two multipliers it can compute 4-tap filter in one cycle.

The second DSP, the C6x, is able to compute two taps per cycle because it has only two multipliers and does not have special operations. But it runs twice as fast as TriMedia, then their peak of performances for tap calculation is the same. It has two separate 16 32-bit words register file and two accesses in the on-chip memory per cycle.

For the sake of the clearness, we estimate the occupation of both DSPs by considering the number of cycles taken by the multiplications, and the computation of both filters as one of 160 taps. Since the update function is done every 69 blocks, it is not a critical function and it is not taken into account.

The implementation will be explained later in order to use multipliers at the maximum. Then for the TriMedia running at 100 MHz, it needs 40 cycles to compute one result, because it can computes 4 taps per cycle and it compute a total filter of 160 taps. To insure the real-time it has to do these computations in 45 cycles which is its running frequency divided by the incoming data rate. Then 90% of TriMedia is used to compute the filters in the ADSL application. The C6x running at 200 MHz needs 80 cycles and it has 90 cycles, then identically 90% of C6x is taken to compute filters in the ADSL application.

Because both are constrained about register file and bandwidth memory, the filter computation is decomposed into several small l-tap filters. In the first phase, the taps and the data of the first l-tap filter are in the register file. Then, a first intermediate result is computed. Without changing taps and data, a new data is read and stored in place of the oldest one. Then a second intermediate result is computed. This process is repeated l times until all the data in the register file are replaced. Then, l intermediate results have been computed. During this time, the l taps of the next filter are loaded. Hence a first intermediate result on the second l-tap filter is computed and added to the first intermediate result of the previous l-tap filter, which has been read from the memory.

Therefore, on a l-tap filter, there were $2l$ read of taps and data and $2l$ read and write for computing l intermediate results. If the total filter has L taps, the bandwidth used becomes: $\frac{4L}{l}$. Since a word has 2 taps and 2 data, the data need to be shifted of half word in order to compute all results (a word $< x(0), x(1) >$ and the shifted word $< x(1), x(2) >$). Then, the value of l is $\frac{1}{4}$ of the register file ($\frac{1}{2}$ for data and the shifted data, $\frac{1}{4}$ for the taps and the same size to load the next taps).

According to the characteristic of TriMedia, the bandwidth used to compute filter in ADSL is: $\frac{4\times160}{32\times2} \times 2.208 =$ 22 Mword/s. A word is a 32-bit word. The bandwidth is divided per 2 because each 32-bit word contains two 16-bit data.

For the C6x the bandwidth is: $\frac{4\times160}{4\times2} \times 2.208 =$ 176 Mword/s. The bandwidth is more important because the size of the register file is smaller.

Now, we describe the CCP and its performances.

3. THE CONVOLUTION CO-PROCESSOR ARCHITECTURE

The CCP (Convolution Co-Processor) is able to compute different FIR (Finite Impulse Response) filters: symmetrical, decimation or adaptive filters with real and complex

data and taps. It manages subword parallelism too [3]. This parallelism is useful because different applications require different data accuracies.

3.1. description

The CCP is composed of several FPEs (Filter Processor Element) organized in a systolic way (fig. 2). The CCP is connected to a host processor through a 32-bit bidirectional way, which provides initialization and data while it reads the results. x is the new data input for the filter computation, y is the result and $error$ is the correction value to update the taps.

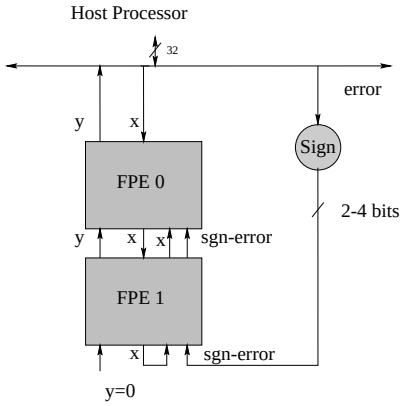

Figure 2: Convolution Co-Processor with 2 FPEs.

The behavior of the Filter Processor Element is shown with the help of a dependency graph (figure 3). It illustrates the calculation progress in the space-time. In this figure, a point represents the computation of one tap. A bold number at the right of each point notifies the instant of the computation. A solid vertical arrow shows the y result propagation. And a dashed diagonal arrow shows the x data progression. We have selected the following parameters for the figure: a filter with $L = 4$ taps, $nc = 2$ FPEs in the convolution co-processor and $z = 2$ data in each word. When a x word arrives, it is used in the computation with the w_0 tap, next with $w1$ and so on. The dashed box shows all the points which have to be computed at the same time. If we want to provide a y word every cycle (i.e the data rate is one x word every cycle), the dashed box has to contained at least z^2 points (if a word of z data is considered). Otherwise, this would mean that the x data progression comes back in the time! Therefore, the FPE should include z^2 multipliers to provide a result every cycle. If a FPE has only $z' \leq z^2$ multipliers then the maximal data rate will be: 1 word every $\left\lceil \frac{z^2}{z'} \right\rceil$ cycle. The operator $\lceil\ \rceil$ ($\lfloor\ \rfloor$) means the rounding function upwards (downwards) to the nearest integer. Let's consider a FPE constituted of z' multipliers. At each cycle,

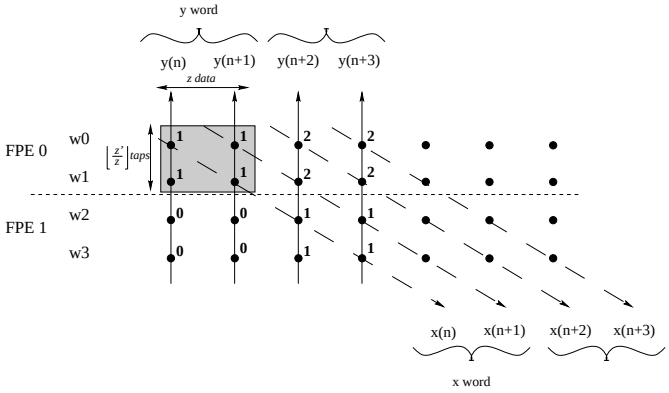

Figure 3: Dependency graph of a 4-tap filter on 2 FPEs with 2 data by word.

it has to provide $\left\lfloor \frac{z'}{z} \right\rfloor$ successive taps and $z + \left\lfloor \frac{z'}{z} \right\rfloor - 1$ successive data. These numbers can be found out by taking the sides of the dashed box in figure 3.

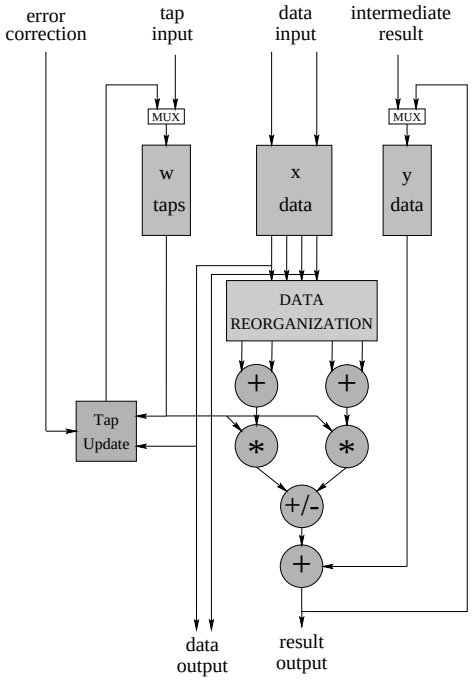

Figure 4: Description of convolution co-processor FPE

A more detailed description of a FPE is pictured on figure 4. Three different register files store taps, data and results. Each one has respectively N, N and K words, where N is the maximum number of taps and K the maximum number of filters that the co-processor can realize. The four read and write accesses are needed by the symmetric computation filters: two for the usual progression data (usual

filters), and two for the reverse data. The reorganization box reorders data read from the x data register file before using them in the operative part, according to the accuracy used. An additional logic block is required for the adaptive filters in order to update the taps. It computes the following adaptive function:

$$w_k = w_k - \Delta \times sgn(e(n)) \times sgn(x(n-k)^*)$$

where w_k is the k$^{\text{th}}$ tap of the filter, $(x(n-k)^*)$ is the conjugate of $x(n-k)$. Δ is the constant step of the tap correction. The function $sgn()$ gives the sign of the data. In digital communications field, it is a current optimization to take the sign in updating function. $e(n)$ is the error computed on the result $y(n)$ at time n. This computation is made by the DSP, because it depends on the application.

3.2. Performances

Now, we calculate the CCP performances in latency and bandwidth aspects. Let's consider z as the number of data in a word, z' as the number of multipliers in data accuracy in a FPE and nc as the number of FPEs in the CCP. The latency is formed of:

- $nc\frac{z'}{z}$ taps are computed each cycle and there are L taps.
- The operator latency op_{lat}.

Therefore, the latency to compute a filter with L taps is:

$$latency \quad = \quad op_{lat} + \left\lceil \frac{L \times z}{nc \times z'} \right\rceil \quad \text{cycles}$$

The bandwidth used by the CCP is limited to one result write and one input read for each filter computation. In addition this read and write can contain several data and result according to the data accuracy.

4. VLSI COMPLEXITY

An entire convolution co-processor including two FPEs has been synthesized. For a first estimation we use one 16-bit data accuracy and include limiting and rounding logic in the operators. Each FPE is composed of four 16-bit multipliers and have 64 32-bit registers for data and taps and 4 32-bit registers for intermediate results. We used the COMPASS library with a 0.8 μm technology. After synthesis, the following results are obtained: the total number of gates for the operative part and the sequencer is 33500 equivalent gates. The four register files, x and w, with each 64 32-bit words takes 8 mm^2 in 0.8 μm. Therefore, we estimate at 65,000 equivalent gates this CCP configuration.

Now we compare the results with and without the use of the CCP for the filter computation.

5. COMPARATIVE RESULTS

With the CCP, the bandwidth is reduced to the write and the read of a 32-bit word which contains 2 results and 2 data, then the bandwidth is: 2.208 Mword/s. It is 8 times as small as TriMedia and 32 times as small as C6x ones. The CCP takes care of all filter computations and only managing operations are settled in the DSP.

In area aspect, the C6x DSP requires 270,000 equivalent gates for the CPU [1] and for filter computation it is used at 90%, then 243,000 equivalent gates are used. The CCP uses 65,000 equivalent gates and it is used at 45% then 29,250 equivalent gates are used for filter computation. Therefore the area requires for the computation of filters in CCP is 8 times as small as in the C6x DSP.

Conclusion

We have studied the ADSL application and we have shown how the implementation in DSPs of the most critical part can be improved by a factor of 8 in bandwidth and area required if the CCP is used. All computation operations are insured by the CCP, then DSPs are freed from these operations. Other functions, needed by ADSL application, require 150 Mops. Both DSPs presented in the paper can manage these functions without problems.

Therefore, an additional filter co-processor frees the DSPs from cumbersome filter computations without loss of programmability. Now, the processor can compute the specific functions which give the originality to the application and integrate higher layer communications.

Furthermore, the co-processor could also improve any application requiring filtering, such as video or 3D graphic processing and can be associated with any processor or DSP.

6. REFERENCES

[1] http://www.ti.com/sc/dsps/products/c6x/faq.htm.

[2] S. Kempainen. The end of the wait for home internet? *EDN*, 10:53–70, october 1996.

[3] R. Lee. Subword parallelism with MAX-2. *IEEE Micro*, pages 51–59, August 1996.

[4] D.W. Lin, C-T. Chen, and T.R. Hsing. Video on phone lines: Technology and applications. *Proceedings of the IEEE*, 83(2):175–193, february 1995.

[5] G.A. Slavenburg, S. Rathman, and H. Dijkstra. The trimedia TM-1 PCI VLIW media processor. In *1996 hot chips symposium*, July 1996.