MEMORY ASPECTS IN SIGNAL PROCESSING AND HLS TOOL : SOME RESULTS

J.L.Philippe - D.Chillet - O.Sentieys - J.P.Diguet
LASTI-ENSSAT
6, rue de kérampont
22300 Lannion FRANCE
Tel : (+33) 96-46-66-41
eMail : *name*@enssat.fr
http://www.enssat.fr/RECHERCHE/ARCHI

ABSTRACT

The digital signal processing system design consists in four synthesis phases which concern the processing, the control, the memory and the communication units. Today, many tools enables us to produce the processing unit. However, in many applications, the hardware solution may be challenged by the number and complexity of memories. This paper proposes a design methodology of the memory units for algorithms restricted by a real time constraint. The original nature of our approach, is due to the fact that it proposes a global memory solution for a transfer sequence computed by the synthesis tools, like GAUT.

KEY WORDS

CAD Tools, High Level Synthesis, ASIC Design, Digital Signal Processing, Memory Synthesis

1 Introduction

Until now, the architecture synthesis has been concentrated on the conception of the processing unit (PU) of the application [1] [2] [3] [4]. Consequently, with regards to a behavioural description, the tools for architecture synthesis provide a structural description of the algorithm (RTL level). This description is optimised under some criteria (area, time of calculation, consumption, etc). A control unit (CU) is then described in the form of a finite state machine (FSM), in order to drive the PU control signals together. Finally, information relative to the data movements will be created and will be used to generate the memory (MU) and communication (UCom) units (see figure 1).

This paper presents a methodology and a tool which produce the PU, CU, MU for DSP applications ; the memory aspect will be pointed out.

The automatic synthesis of the storage units have been hardly touched upon in literature. Nevertheless, it is possible to distinguish several significant research topics which are the following : The distribution and the placement of data [5] [6] [7], the generation of addresses [8], the selection of memory [9], the high level transformations [10] [11]. This last point aims to estimate and minimise the storage as early as possible in the design by completing the area estimation, the power estimation etc, before the architectural synthesis.

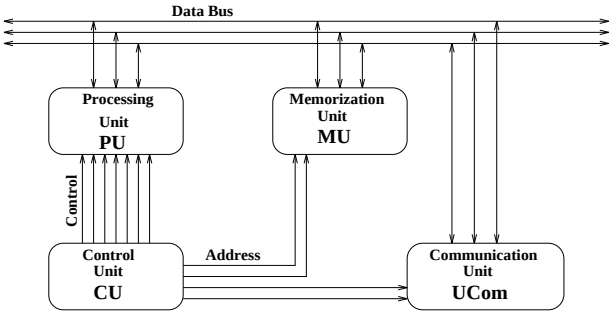

Figure 1: System decomposition

In our approach, GAUT, the real time aspect implies that the processing unit must be synthesised first [12] (see figure 2). This is done in respect with two memory parameters selected by the designer : the memory access time and the memory threshold. This last parameter is used during the PU synthesis in order to decide if a data has to be recorded in memory between two successive uses, or may stay in register located inside the PU. After the PU synthesis, the requirements of the memory transfer sequence are generated. From this graph and from designer constraints, such as memory access time, the memory synthesis will take place.

The paper will be focused around two main points. In the second section we will briefly describe the MU design flow. In the third section we will present some results in ASIC design by using high level synthesis CAD tools, for acoustic echo cancellation. Finally we will conclude by presenting the works currently in progress.

2 Design methodology

The PU synthesis produce a transfer sequence between the processing and the memory units. This sequence is, in the first step, describe in VHDL format. Note that the transfer sequence is transformed in view to adapt

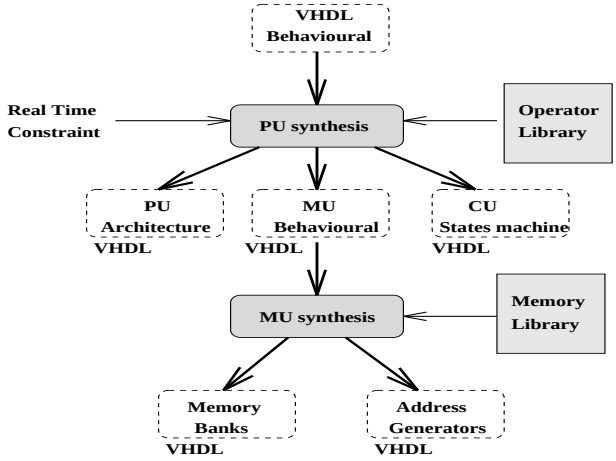

Figure 2: Design methodology

the rate between the PU requirement and the memory capabilities. It is done by smoothing the sequence : some transfers are bring forward and some others are delayed. It makes easier the memory unit synthesis by producing a less constraining memory transfer graph. This unit is organised around register file.

The methodology that we have developed is based on MU model represented in figure 3. This model is composed of N_B memory banks associated with N_G address generators. The memories can be made up of varying access time memory components.

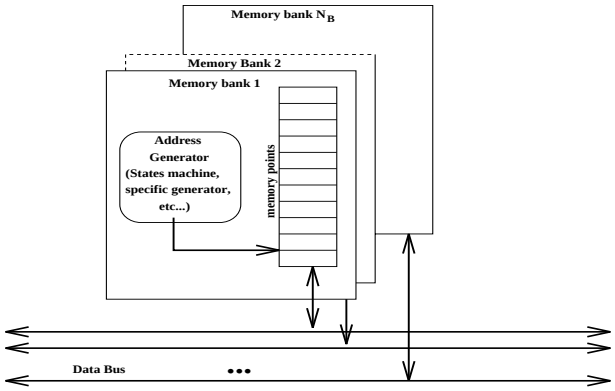

Figure 3: Memory unit model

Our methodology consists in three phases, which are the following :

• **Memory module selection :** For the memory unit implementation, our tool examines a library to select all the components. This selection is based on an estimation of the ASIC cost. The library contains internal (dedicated) and external (commercial) memories. The selection problem is expressed by an ILP formulation, and classical techniques are used (simplex, branch&bound or linear). The cost function is valued by the number of components to produce, i.e. the selection takes into account the number of Integrated Circuits per year.

• **Formulation of the data distribution problem :** This firstly, concerns computing the number of memory banks to be placed in order to solve the data coherence[1] problems, and the simultaneity of transfers. Then secondly, it concerns the distribution of each data into a particular memory bank. We have used a graph $\mathcal{G}\{N, A\}$ to represent the conflicts between data. The research of the number of memory banks consist in finding the minimum number of colours to colour the vertices of the graph $\mathcal{G}$, under the restriction that two vertices connected by an edge cannot carry the same colour. After this coloration, all data which have the same colour are assigned in the same memory bank. A specific method is used for the distribution of the shifting signal vectors which are often used in DSP application. An ILP formulation has been realized, and some techniques have been developed : simplex, exhaustive and linear.

• **Formulation of the problem of the data placing in memory banks :** After the distribution phase, each data from the sequence is assigned in a memory bank. The organization in the bank is not however defined. In fact, the placement of each data in a precise address is yet to be decided. The placing of data involves attributing an exact physical address to each of the data. What is important during the placement is the control of the cost of the address generators involved. Three address generators can be used, by our tool : up/down counter, states machine or ALU with index registers. The resolution of this phase is based on a distance evaluation between all the data to be placed in each memory bank. First, an up/down counter is targeted, if the addressing is not possible, a states machine is selected. For the placement of all elements of each shifting vector, an up/down counter is targeted and the placement consists to assign the vector elements consecutively in the memory.

The methodology which is implemented in our tool [13] enables us to obtain results in few seconds. For example, a LMS filter with 128 taps is computed in about 2 seconds on Sparc 5 workstation.

3 Results

We have validated our first results on an acoustic echo cancellation application for hands-free telephony and teleconference calls. A collaboration with the France Telecom (CNET) center of Lannion have been done and consists in prototyping some adaptive filtering algorithms. Echo appears by the fact that a speaker hears

[1] The data coherence is defined using three points :
• temporal coherence : two data simultaneously transferred must not be found in the same memory bank ;
• Spatial coherence : the reading and writing of the same data must be carried out in the same memory bank ;
• Functional coherence : in the case of a pipeline structure, the writings of a pipeline stage must not erase data which are still of use for other pipeline stages.

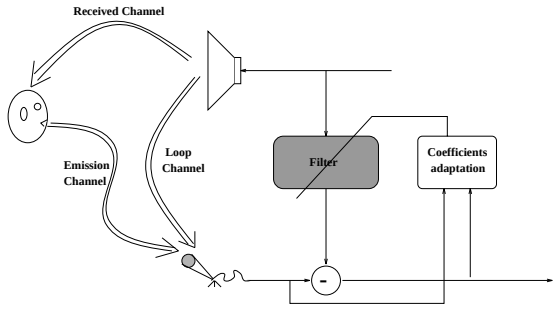

Figure 4: Principle of the acoustic echo cancellation

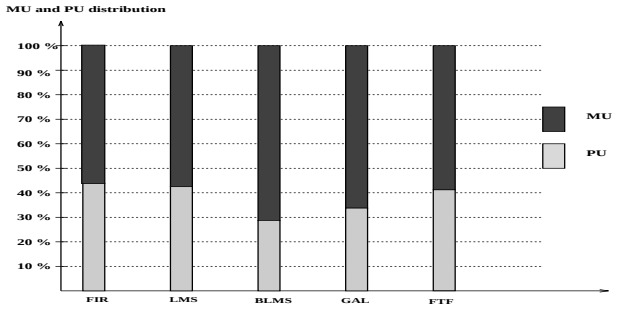

Figure 5: Distribution of total area between the processing unit and the memorisation unit

what he transmits with a certain delay due to propagation time. This could lead to a perceptible decrease in the quality of communication. The cancellation is achieved by estimating the echo and subtracting it from the return signal (see figure 4). This estimation is carried out by an adaptive filtering.

We consider here several adaptive filter algorithms (FIR, LMS, BLMS, GMDF, FTF, etc), that are used in acoustic echo cancellation.

	FIR	LMS	BLMS	GAL †	FTF
Nb of * operations	N	2N	2N	3N+5C	8N+13
Nb of + operations	N	2N	2N	2N+4C	8N+5
Nb of * operators	1	1	1	1	1
Nb of + operators	1	2	2	2	1
Nb of registers	4	4	4	4	4
Nb of data	2 112	3 328	5 760	5 184	15 744
Nb of buses	2	2	2	2	3
Nb of banks	2	2	4	3	4
$\frac{S_{MU}}{Total\ Area} * 100$	57	58	71	67	58

† : C if the number of cell, which is comprised between 1 and N

Table 1: Complexity of some adaptive filters for acoustic echo cancellation (1024 taps, 8 Khz, 16 bit data), with S_{MU} = MU silicon area

The table 1 shows the complexity results of these algorithms. During a prototyping phase, the mathematical complexity analysis consists in enumerating the operations (see the two first lines of table 1). The high level synthesis tool GAUT [14] completes this complexity analysis with an accurate processing unit cost (see lines 3, 4 and 5 of table 1) .

The complexity analysis, at an mathematical level, as well as at architectural level, allows us to compare these filters. In this way, the first two lines of table three shows that the FIR filter is the least complex, (but it is also the least interesting due to its non adaptive characteristics). Let us note in addition that the first five lines give rise to an equivalence between LMS and BLMS filters.

Our memory synthesis tool enables us to complete these results by providing : the number of memory banks, the number of points per bank and the address generator cost. All these informations make a cost evaluation, in terms of memory area. The last three lines of the table 1, and figure 5 show the importance that the memory has on the total cost of the ASIC. For each of the filters, the memory represents over half the ASIC area. Let us note that the LMS and the BLMS filters are not equivalent

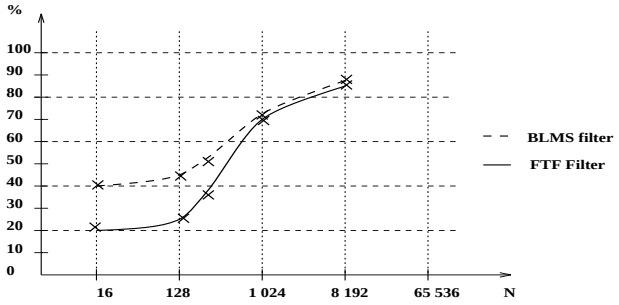

Figure 6: Evolution of $\frac{S_{UM}}{Surface\ Total} * 100$ ratio in relation to the number of taps for FTF and BLMS filters

when the memory is synthesized (the LMS filter area is equal to 57 % of the BLMS filter area).

The curves of figure 6 show that the more the filter size increases, the more dominating the memory becomes in relation to the processing unit. The processing frequency constrains the processing part very little, which explains why its cost remains constant, (except for the 8192 taps filter size).

The figures 7 and 8 show the floorplans for the LMS filter with two different sizes and implemented with the ES2 1μ CMOS technology. These floorplans have been synthesized by the Compass tool from the VHDL descriptions produced by the PU synthesis tool, GAUT, and the MU design tool. The selected memories are internal to the ASIC. In the first case (fig 7), we can see that the MU represents 25 % of the ASIC core (MU silicon area = $2.34mm^2$) while the PU represents 47 % of the ASIC core. In the second case (fig 8), the MU represents 56 % of the ASIC core (MU silicon area = $12.05mm^2$) while the PU represents 22 %. We can note also that the generators are, in the two cases, very little (they are the two little standard cells in the floorplans near the Ram's, which areas are less than $0.2mm^2$). Likewise, the UC silicon areas are very little in the two cases (less than 5 % of the ASIC core). Note that the multiplication operators have the same size ($1.22mm^2$) in the two floorplans because the sizes of data to compute are the same.

The high level synthesis produces results which are not take account of the routing. Nevertheless, we can note

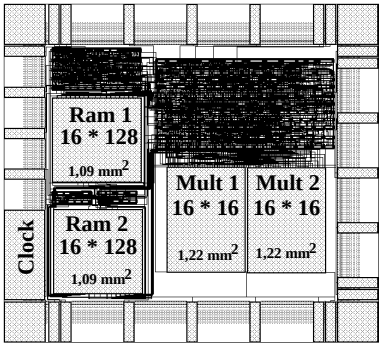

Figure 7: Floorplan of the LMS with 128 taps ($4.56 * 4.15mm^2 = 18.92mm^2$)

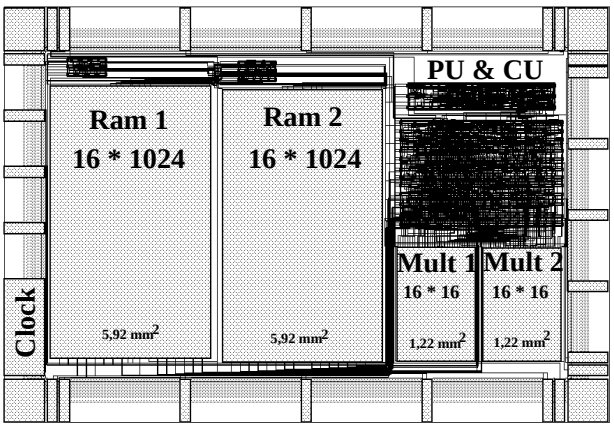

Figure 8: Floorplan of the LMS with 1024 taps ($7.34 * 4.70mm^2 = 34.50mm^2$)

that the ratios between the processing and the memory units are similar before and after the logical synthesis.

4 Conclusion

The quality criterion linked to DSP applications, is obtained either by suggesting new algorithms, or by extending the size of known filter responses. In the first case, it is necessary to evaluate the implementation, in an ASIC, for these algorithms by a prototyping stage. In the second case, the processing unit of the application evolves little, however, the number of data to memorize rises considerably. So, it becomes of increasing importance to take into account the memorization of data in order to supply an accurate evaluation of the cost of each solution.

For some applications, the memory cost make the ASIC not feasible or too much expensive. So, it seems attractive to develop a design flow in which the MU is synthesized first. Consequently, the PU synthesis will be realized under constraints given by the memory unit.

In addition, the development of system (e.g. codesign) requires to control all the functional units from the first steps of the design. This requires various estimators (cost, consumption, performances, ...), that are computed all along the design steps, in order to guide the following steps or to realize some transformations (such as reordering loops). This last point form the subject of the studies in our laboratory.

References

[1] M.Potkonjak and J.M.Rabaey. Exploring the algorithmic design space using high level synthesis. *VLSI Design Methodologies for Digital Signal Processing Architectures*, Tome 257:131–167, 1994.

[2] D.C.Ku and G.De Micheli. *High Level Synthesis of ASICs Under Timing and Synchronization Constraints.* Kluwer Academic Publishers, 1992.

[3] R.A.Walker and R.Camposano. A survey of high level synthesis systems. *Kluwer Academic Publishers*, 1991.

[4] M.A.Bayoumi. *VLSI Design Metholodogies for Digital Signal Processing Architectures.* Kluwer Academic Publishers, 1994.

[5] T.Kim and C.L.Liu. A new approach to the multiport memory allocation problem in data path synthesis. *Integration, the VLSI Journal*, pages 133–160, 1995.

[6] I.Verbauwhede, F.Catthoor, J.Vandewalle, and H.De Man. In-place memory mangement of algebrical algorithms on application-specific ic's. *Journal of VLSI Signal Processing*, 1991.

[7] D T Harper III and D.A Linebarger. A dynamic storage scheme for conflict free access. In *16th Annual International Symposium on Computer Architecture*, volume 17, pages 72–77, June 1989.

[8] F Grant, J.V Meerbergen, and P Lippens. Optimization of address generator hardware. In *Proc 5th ACM/IEEE European Design and Test Conference*, pages 325–329, February 1994.

[9] S Bakshi and D.D Gajski. A memory selection algorithm for high-performance pipelines. In *EURO-DAC Brighton*, September 1995.

[10] F Balasa, F Catthoor, and H De Man. Exact evaluation of memory size for multi-dimensional signal processing systems. In *IEEE International Conference on Computer Aided Design*, 1993.

[11] I.M Verbauwhede, C.J Scheers, and J.M Rabaey. Memory estimation for high level synthesis. In *ACM/IEEE Design Automation Conference*, number 31, 1994.

[12] E Martin, O Sentieys, and J.L Philippe. Synth se architecturale de coeur de processeurs de traitement du signal. *Techniques et Sciences Informatiques*, 1(2), 1994.

[13] D.Chillet, J.P.Diguet, J.L.Philippe, and O.Sentieys. Memory unit design for real time dsp applications. *Submited to Special Issue on Integrated Design of Embedded Computer Systems*, 1996.

[14] O Sentieys. *Analyse et synthese d'architectures en traitement du signal et d'images : vers la conception d'architectures heterogenes.* PhD thesis, LASTI-ENSSAT-Universite de Rennes, February 1993.