MEMORY MODULE SELECTION FOR HIGH LEVEL SYNTHESIS

O.Sentieys - D.Chillet - J.P.Diguet - J.L.Philippe
LASTI-ENSSAT
6, rue de kérampont
22 300 Lannion, France
Tél : (+33) 96-46-66-41
eMail : sentieys@enssat.fr
http://www.enssat.fr/RECHERCHE/ARCHI

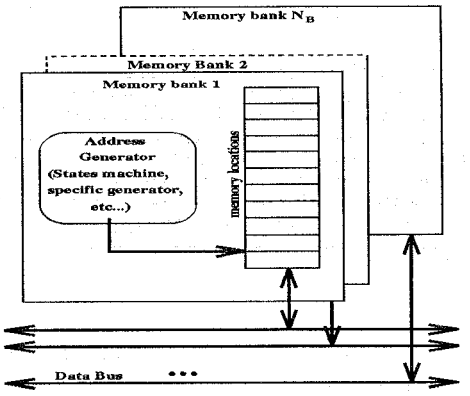
Abstract - High level synthesis studies have produced many tools which enable us to design the processing unit of applications. The emergence of new communication services has lead to significant growth in the amount of data to be processed in VLSI chips. It involves to synthesis of memory architecture which enables us to satisfy all the application constraints. To obtain this organization, the first step is to select memory from a component library. This paper suggests a formulation of this problem through a minimization of function under constraints. Our approach takes place after the processing unit synthesis and our methodology can be applied for FPGA chips.

1 INTRODUCTION

Over the past years, new business and home applications using advanced communication systems and also multi-media are emerging. They are becoming the most prominent growth sectors of the electronics industry. They need to implement complex real time algorithms under constraints of time, cost and power chiefly using VLSI circuits. At the same time, these systems need to be designed in a very short "time-to-market". Moreover, all these digital and signal processing applications handle a large amount of data [1] [2]. The chips will therefore mostly be composed of memory location and address generation. To take all these problems into consideration, we need new high level tools that can help the designer to select a promising path in the huge decision tree from high level specifications to hardware. At first, the studies of high level synthesis (HLS) put forward tools which helped the design of the processing unit (PU) of the application [3] [4] [5]. A short time ago, many memory-unit-design studies appeared. There can be classified according to three criteria : 1. Consider the memorization problem as soon as possible in the design cycle and suggest synthesis of the memory unit (MU) after the processing unit ; 2. Look at the storage during the synthesis of the PU through the maximization of register utilization ; 3. Our approach suggests a global solution of storage (with processing registers and memory) which ensures that the needs of the PU are satisfied. Our choice is made because of the application domain. Indeed, the digital signal processing (DSP) is characterized by a time constraint that HLS must respect through the synthesis of PU at first. In addition to the PU structural description and the control unit (CU) behavioural description, our HLS tool, GAUT [6], gives the production and consumption dates of each

0-7803-3134-6/96 $5.00 © 1996 IEEE

transfer between the PU and the MU. This transfer sequence is written in VHDL in order to be simulated (together with the PU), or to be synthesized by the tool that we present in this paper. Our MU-design methodology build the solution in four phases which are : the memory selection in a library, the distribution of data into the memory banks, the placement of data in each bank, and the address generator (AG) synthesis.
This paper deals with the memory selection problem. We will, however, see in the paragraph 5 that our methodology enables us to optimize the architectural structure given by HLS tools.



The model of MU we have adopted is illustrated in figure 1.
This model is composed of N_B memory banks associated with N_G address generators (with $N_G \leq N_B$). The memory unit can be made up of varying time access memory.

Figure 1: Memory unit model

The problem of the memory selection can be explained in the following way :
• ***Let TS be a transfer sequence of an algorithm, what are the types, the number and the position of the MU to ensure the total storage of the data ?***

In the following paragraphs, we will analyse the publications concerning the MU synthesis domain. Then, we will present a formulation of the memory selection problem answering the question above. We will explain how our methodology could be applied for FPGA implementation. A paragraph will be internal to the results and finally, we will conclude and put forward a few viewpoints.

2 STATE OF THE ART

Very few articles tackle the problem of the MU design in a global way. Most articles address a particular problem and suggest solutions which strongly depend on the types of data manipulated or type of memory used. These studies can be classified according to three main points.
• **Memory selection :** In [7], a specific selection of MU organization is suggested. A horizontal selection which associates several MU, ensures that the

width of the data corresponds to the width of the memory selected. A specific size of memory can be achieved through vertical selection. Furthermore, to increase the transfer frequency, a selection of interleaved memory can be used. Finally, another selection can virtually increase the number of memory ports. In [8], the suggested method enables them to provide the number, the type (RAM or ROM), the word width, and the number of ports of each memory. The selection rests on the evaluation of a silicon-area-cost function.

• **Allocation of storage units :** These approaches are principally the scalar and the vectorial approaches. In the scalar, the goal is to minimize the number of registers needed to store data [9] [10]. It concerns the analysis of production and consumption data dates, in order to try to maximize the sharing of memory locations. The ILP resolution techniques (Integer Linear Programming), graph coloration, or clique partitioning are widely used. The design cycle of HLS tools, e.g. the GAUT cycle, integrates this approach while proposing algorithms to minimize the number of PU register. These techniques revealed major flaws when we computed algorithms handling multi-dimensional data [11]. In fact, the size of the problem makes its resolution difficult. The vectorial approaches use the regularity of data structures and the regularity of the processing, to carry out an optimal arrangement in different memory banks [12]. This involves making sure that the particular accesses (line, column, or matrix type) can be executed without a time delay.

• **Generation of Addresses :** Several technics are proposed to bring about this function. Some of them are made up of a PU with a set of index registers [13], or a specific PU using the DSP properties [14], or a state machine, generating the necessary addresses for accessing the data from or to the memory [16] at each step of the algorithm. These solutions have different costs and allow us to produce adress sequences more or less complex.

3 FORMULATION OF THE SELECTION PROBLEM

In a first approach, we suggest a selection of memory components for a fixed model of organization (figure 1). This selection, in turn, consists of searching through a library the set of memory best suited to the data storage, all according to the requirements expressed by the PU synthesis. We can notice that to reach the best selection, the higher number of memory type is, the higher number of explored solutions is. The MU and the AG can be inside or outside the ASIC. If the AG is inside, the cost considerably increases because of the number of necessary I/O pads for the address bus. On the other hand if the generator is outside, it will be built in an FPGA circuit.

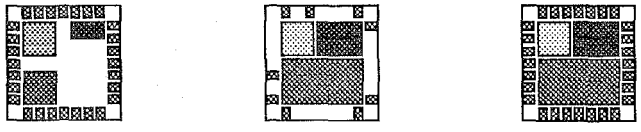
3.1 Definitions

We suggest a formulation for the selection problem with the following notations and variables classified in alphabetical order.

1. A_b is the available area if the number of pads is known ;
2. A_D is the area of a binary counter per bit ;
3. $APts_i$ is the location area of the i^{th} internal memory ($APts_i = f(NBits_i, NP_i)$) ;
4. A_u and A_{PU} are respectively the silicon core area (PU, CU and MU) and the PU area
5. C_i is the location cost of the i^{th} memory ;
6. CG_i is the cost of the i^{th} AG ;
7. DI_i is an integer variable, such that $DI_i = 1$ if the i^{th} data is store inside the ASIC, else $DI_i = 0$;
8. G_i is an integer variable, such that $G_i = 1$ if the i^{th} AG is inside the ASIC, else $G_i = 0$;
9. Max_i is the max size of the i^{th} memory when it is internal ;
10. N_B is the necessary number of memory banks to ensure the coherence of data[1] ;
11. $NBits_i$ is the width of data buses of the i^{th} memory ;
12. N_{BR} is the number of pads of the ASIC ;
13. N_D is the number of data to store ;
14. N_{MI} and N_{ME} are respectively the internal (internal) and the external MU available in the library ;
15. N_{MTS} is the max number of simultaneous transfers, it involves the number of buses to place on the ASIC : $N_{MTS} = Max\left(\sum_{k=1}^{N_D} T_{kj} \cdot (1 - DI_k)\right) \quad \forall\, j = 1, \cdots, N_{TT}$.
16. NP_i is the number of locations of the i^{th} memory ;
17. N_{TT} is the total number of transfers to realize in the [0 ; T_L] interval ;
18. P_i is the price of the i^{th} external memory ;
19. P_{mm^2} is the price of one silicon mm^2 for the usual technology ;
20. PP is the distance between two consecutive pads of the ASIC ;
21. $\{SE_i\}$ is an integer variable which represents one selection of external memory component, such that : $SE_i = n$ if the memory i is taken n time $\quad \forall\ i = 1, \cdots, N_{ME}$.
22. $\{SI_i\}$ is an integer variable which represent one selection of internal memory component, such that : $SI_i = n$ if the memory i is taken n time $\quad \forall\ i = 1, \cdots, N_{MI}$.
23. TA_i is the access time of the i^{th} memory ;
24. TB_i is the size of the i^{th} data bus outgoing of the ASIC ;
25. T_{ij} is an integer variable, such that $T_{ij} = 1$ if the i^{th} data is transferred at the time j, else $T_{ij} = 0$;
26. T_L is the latency time of the algorithm ;

3.2 Minimization of a cost function

The feasibility depends greatly on its total area (technology constraint). This area depends on the silicon area attributed to the core and the necessary area to ensure exchanges with the outside. In this paragraph, we will discuss the cost of the in/out pads of the ASIC. It thus involves finding a compromise between core or pad limited ASIC.



a) b) c)

Figure 2: a) Pad limited ; b) Core limited ; c) $A_u/A_b = 1$

[1]The coherence of data is defined using three criteria :

- temporal coherence : two data simultaneously transferred can not be 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 2.c represents the situation for which the necessary area of the ASIC, A_u is equal to the available area A_b. In figure 2.b, it is possible to add pads without increasing the ASIC cost.
The ideal choice is explained mathematically by a relationship between A_u, A_b and the cost of external memory components. In the general case, the relationship A_u/A_b tends towards to 1 and the loss area is minimal (see figure 2.c). To obtain an optimal selection, we suggest to minimize its cost, it can be explain by :

$$Max\,(A_u, A_b) \cdot P_{mm^2} + \sum_{i=1}^{N_{ME}} SE_i \cdot P_i + \mathcal{F}_{fpga}\left(\sum_{i=1}^{N_{ME}} SE_i \cdot (1 - G_i)\log_2(NP_i)\right)$$

The first term of the sum expresses the ASIC cost by multiplying the mm^2 cost by the ASIC area. The second term expresses the cost of external selected MU. Finally, the third term evaluates the external AG cost which will be implemented in one or several FPGA (the $\mathcal{F}_{fpga}$ function return the FPGA price).
The A_u and A_b area are given by :

$$A_u = A_{UT} + \sum_{i=1}^{N_{MI}} SI_i \cdot C_i + \sum_{i=1}^{N_{MI}} SI_i \cdot G_i \cdot CG_i \qquad A_b = \left(\frac{N_{BR} \cdot PP}{4}\right)^2$$

The necessary area A_u is computed from the sum of the PU area (including the CU area), internal MU and internal AG. The A_b area is computed from the square of the ASIC side which is fixed by the number of pads to implement. The MU and AG costs are defined by :

$$C_i = NP_i \cdot NBits_i \cdot APts_i \qquad CG_i = \lceil \log_2 NP_i \rceil \cdot A_D$$

The AG cost is estimated by a preloaded binary counter area which is equivalent to 12 logic gates (eg. for an $1\mu m$ technology, $A_D = 12 \cdot Density = 12 \cdot 437\mu m^2/gate$).
The number of pads N_{BR} is estimated by the following relationship :

$$N_{BR} = C + \sum_{i=1}^{N_{MTS}} TB_i + \log_2\left(\frac{\sum_{k=1}^{N_D}(1 - DI_k)}{\sum_{j=1}^{N_{ME}} SE_j}\right) \cdot \sum_{k=1}^{N_{ME}} SE_k \cdot G_k$$

The C term takes into account the pads of power supply, clock, $\cdots$, necessary for the ASIC. The second term expresses the number of data bus pads to place on the ASIC. The third term estimates the number of address bus pads necessary to address the external memory, that depends on the average number of accessing data in the external memory by internal generator.

3.3 The constraints of the memory selection problem

We will put forward four constraints. Three of them will attempt to satisfy the data storage and the transfer sequence and the last will try to ensure the feasibility of the internal memory. The constraints that we proposed are minimal and allow us to have a good estimation of application needs. Note that during the MU synthesis, it is possible to increase the number of selected MU.

• **Number of memory banks :** it must ensure parallelism of transfers and coherence of data. The first is obtained by a simple analysis of the transfer sequence, while the second calls for the research of a number of memory banks (N_B) which are far more complex. The number of MU chosen must comply with the constraint 1.

• **Total number of memory locations of the selection :** it must ensure the storage of all data, see the constraint 2. In the first step, each data needs a memory location. This can be optimized if one envisages to share out memory locations by several data which have disjointed life times.

• **Number of transfers :** this must be ensured by the selected MU, and is explained by the constraint 3.

The number of available access time in a memory bank during latency time is represented by T_L/TA_i. The total amount of possible accesses is therefore the sum of all possible accesses in the selection.

• **Feasibility of internal memory :** For a given technology and foundry, the internal memory must not be larger than a specific size. This maximum size is illustrated by the product $NumberOfBits \cdot NumberOfWords$. Thus, we can explain a constraint for internal memory, see constraint 4.

$$\sum_{i=1}^{N_{MI}} SI_i + \sum_{j=1}^{N_{ME}} SE_j \geq N_B \quad (1)$$

$$\sum_{i=1}^{N_{MI}} SI_i \cdot NP_i + \sum_{j=1}^{N_{ME}} SE_j \cdot NP_j \geq N_D \quad (2)$$

$$\sum_{i=1}^{N_{MI}} SI_i \cdot \frac{T_L}{TA_i} + \sum_{j=1}^{N_{ME}} SE_j \cdot \frac{T_L}{TA_j} \geq N_{TT} \quad (3)$$

$$\forall\, i = 1, \cdots, N_{MI} \Longrightarrow NP_i \cdot NBits_i \leq Max_i \quad (4)$$

3.4 Implementation of the application in a FPGA

During the prototyping step, the designer can hope to estimate the feasibility of the solution for a particular FPGA implementation. A given family of FPGA is characterized by a set of circuits which are pad and core limited. In this case, the circuit imposes some new constraints which are the number

of pads, the available area and the size of internal memory. These first two constraints can be explained by :

$$A_u \leq S_{fpga} \cdot \tau_u \qquad N_{BR} \leq N_{BRfpga} \qquad \sum_{k=1}^{N_D} DI_K \leq NP_{fpga}$$

With S_{fpga} the available area in the FPGA (from the number of gates it is possible to know the area). This area is multiplied by a constant τ_u which provide to specify that FPGA circuit is never used at 100% (with $\tau_u \simeq 80\%$).
With N_{BRfpga} the number of pads of the targeted FPGA.
And with NP_{fpga} the number of memory locations available in the FPGA.
The library can contain all the available circuits for one family of FPGA, and for each FPGA circuits the library must describe its characteristics.

Note : We can not use the simplex resolution technics to solve this problem because it is not an ILP formulation. However the weak number of MU in library allow us to use exhautive technics.

4 MEMORY LIBRARY DESCRIPTION

Memories	Type †	Size	Access Time ns	Cost mm^2 $
RAM 1	D	64*8	15	0.414
RAM 2	D	128*8	15	0.458
RAM 3	D	256*8	20	0.548
RAM 4	D	512*8	20	0.713
RAM 5	D	1024*8	30	1.06
RAM 6	D	2048*8	30	1.73
RAM 7	E	32*8	25	2.20
RAM 8	E	256*4	30	2.60
RAM 9	E	512*8	45	6.40
RAM 10	E	1024*8	50	11.20
RAM 11	E	2048*8	50	12.80

† : D : internal (Dedicated), E : external

NOTE : The price of an ASIC is composed first by the NRE (Non Recurring Engineering) that corresponds to the mask fabrication and to prototyping, and a silicon square millimeter price that strongly depends on the number of Integrated Circuits per year that will be produced. It is well known that it is very difficult to obtain an accurate price estimation of an IC, but we can have an idea through an example of a $1\mu m$ CMOS technology. Let NRE be equal to 64,000 $, and let 45 $ the price for 2,000 ASIC of 20 mm^2. The price of the mm^2 is equal to $(64,000/2,000 + 45) \cdot \frac{1}{20} = 3.85$ $. On the other hand, if we have exclusively to prototype an application by using MPW (Multi Project Wafer) the mm^2 cost depends only of the technology and is about 200 $ (CMP/Europractice source). All these parameters can simply be controled by the designer in his library.

Table 1: Memory library used for the selection

In the table 1 the RAM 1 to 6 are internal and the price is given by the ASIC area. On the other hand RAM 7 to 11 are external (standard RAM components). All this prices can be given by the designer and strongly depend on the application. The operator library must define the group of parameters necessary for the synthesis of the MU. An external view of each memory component is described in VHDL language and shows the number and type of each input/output. The use of generics allows us further precision concerning

the area, the access time, the global size of the memory and thus, the cost of each memory component. In table 1 we show the library that we have used to produce the results of paragraph 5. It is interesting to note that the cost of internal RAMs is not shown. This is independent of memory characteristics and depends solely on the price of silicon and the technology targeted. It is also interesting to notice that the external memory area is not mentioned. In fact, this area is not characteristic of the cost as it includes the whole packaging. It cannot be compared to the area of an equivalent internal memory for which only the silicon area of the memory is taken into account.

5 MEMORY SELECTION RESULTS

Filter size	16	64	128
Selection	7	7	7
RAM	7	(2 *) 8	9
Nb of Data	52	196	388
Nb of Pts	64	288	544
MU Cost ($ or mm^2)	22.00	7.40	8.60
Total Cost ($ or mm^2)	554.40	557.40	558.60

a) prototype

16	32	64	128
1	1	2	3
1	1	2	3
52	100	196	388
128	128	256	512
0.828	0.828	0.916	1.096
3.36	3.36	3.45	3.63

b) high volume fabrication

Filter size	16	32	64	128
Internal RAM		2	3	4
External RAM	(2 *) 7	7	7	7
Nb of Data	52	100	196	388
Nb of Pts	64	160	288	544
MU Price external	00.00	2.20	2.20	2.20
Total Price ($)	19.40	19.80	20.40	21.40
MU area internal	0	0.458	0.548	0.713
Total area (mm^2)	2.50	2.96	3.05	3.21

c) Memory selection for a low volume fabrication of LMS filters (using both internal and external MU)

The table 2.b presents the results of these selections when we have to design a high volume fabrication of integrated circuits. In this case, the price of the internal memory location decreases, and is almost the external memory one. This explains why the selection module leads to implement all the memory in the ASIC. The memory area corresponds to about more than 25 % of the total ASIC area.

Table 2: Memory selection for LMS filters (8 Khz)

We have carried out three series of tests on the LMS filter (which is an adaptive filter used for an acoustic echo cancellation application). Each serie of tests corresponds to a different number of ASIC to be produced. In this way, the first serie (table 2.a) explores the memory solutions when a prototype is produced, (meaning the production of a single ASIC). In this case we can see how the selection module chooses external MU with a view to reducing as much as possible the cost of the ASIC. In actual fact, the cost of the internal memory location is much greater than its equivalent in external memory.
Figure 3 represents the rate ($\tau = NbData/NbPts \cdot 100$) of the memory location utilization. This is about 70 %, except for the number of taps equal to

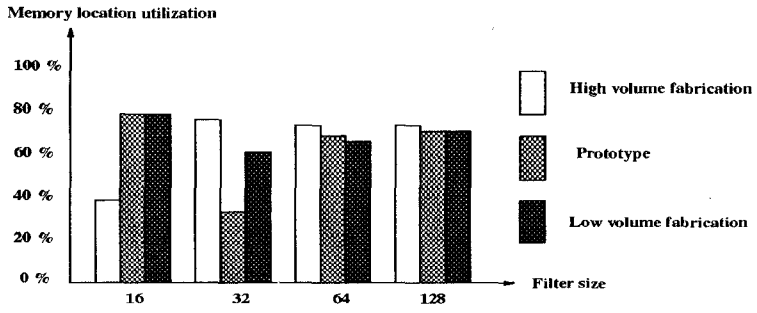


Figure 3: Memory location utilization

16. This weak utilization rate is due to the small number of MU present in the library. The rate value is a criterion to evaluate the quality of the solution. Indeed, if this rate is great, this corresponds to have few unused memory locations. It is clear that the quality of the solution depends on the number of MU described in the library.
Table 2.c shows the results of the selection when it is necessary to produce around one hundred ASICs. When this is the case, the choice of memory types is mixed, meaning that the selection module offers to store some data inside the ASIC and the others in external memory. The cost of the external memorization comes to about 10 % of the total cost of the operation.
As with the two previous cases, the rate of use of memory locations is in the region of 70 %.

6 CONCLUSION

We have shown an original method of memory selection based on the minimization of MU cost. Let us bear in mind that our approach follow the PU synthesis and is, therefore bound by its constraints.
The formulation of this selection problem that we are putting forward allows us to consider, in a realistic fashion, the cost of all the memory components which the ASIC production will use. Thus, the number of ASIC that we hope to produce becomes a critical parameter while selecting MU.
One can find in [16] the results of MU synthesis for the application presented here.
We have developed a synthesis tool dedicated to MU [16] integrating the selection module presented in this paper. Together with HLS tool, this allows either the refining of the prototype algorithm (rapidly suggesting an evaluation of memory organization), or the achievment of complete memory solution. The results of architecture synthesis, in terms of type, number of operators, and registers are thus completed by an exact evaluation of cost and organization of the MU. Together, these form a framework for development

or fast prototyping phases.
We are currently working, in our laboratory, on the problem of the selection of MU hierarchy. It will allow us to suggest, when necessary, a memory comprised of several levels, e.g. a cache memory.

References

[1] F Balasa, F Catthoor, H De Man, "Exact Evaluation Of Memory Size For Multi-Dimensional Signal Processing Systems", *Proc. IEEE Int. Conf. on Computer Aided Design*, , 1993.

[2] T.Kim, C.L Liu, "A new approach to the multiport memory allocation problem in data path synthesis", *Integration, the VLS Journal 19*, pp 133-160, 1995.

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

[4] R.A.Walker, R.Camposano, "A Survey of High Level Synthesis Systems", *Kluwer Academic-Publisher*, , 1991.

[5] C.Y.Wang, K.K.Parhi, "Resource-Constrained Loop List Scheduler for DSP Algorithms", *Journal of VLSI Signal Processing*, Kluwer Academic-Publisher, Boston, pp 75-96, 1995.

[6] J.L.Philippe, O.Sentieys, J.P.Diguet, E.Martin, "Synthesis : From Digital Signal Processing Specifications to Layout", *In Logic and Architecture Synthesis : State of the art and novel approaches*, IFIP, Chapman & Hall, 307-313, 1994.

[7] S Bakshi, D.D Gajski, "A Memory Selection Algorithm for High-Performance Pipelines", *EURO-DAC Brighton*, Great Britain, September 18-22, 1995.

[8] F Balasa, F Catthoor, H De Man, "Dataflow-Driven Memory Allocation for Multi-dimensional Signal Processing Systems", *Proceeding IEEE Int. Conf. On Computer Aided Design*, Santa Jose, pp 31-34, April 1994.

[9] I Ahmad, C Chen, "Post-Processor for Data Path Synthesis Using Multiport Memories", *IEEE International Conference on Computer Aided Design*, ICCAD, 91.

[10] M Balakrishnan, A.K Majumdar, D k Banerji, J.G Linders, J.C Majithia, "Allocation Of Multiport Memories In Data Path Synthesis", *IEEE Transactions on Computer Aided Design*, Vol 7, No 4, April 1988.

[11] I.M Verbauwhede, C.J Scheers, J.M Rabaey, "Memory Estimation for High Level Synthesis", 31^{st} *ACM/IEEE Design Automation Conference*, , 1994.

[12] D T Harper III, D.A Linebarger, "A Dynamic Storage Scheme For Conflict Free Access", 16^{th} *Annual Int. Symp. on Computer Architecture*, Vol 17, No 3, June 1989.

[13] I Bazon, "Programmation logique avec contraintes appliquée à la synthèse de l'unité de mémorisation de GAUT", *Internal report, LASTI - ENSSAT*, , 1993.

[14] F Grant, P.B Denyer, I Finlay, "Synthesis of Address Generators", *Proc IEEE Int Conference Computer Aided Design*, pp 116-119, November 1989.

[15] O.Sentieys, E.Martin, J.L.Philippe, "VLSI Architecture Synthesis for Acoustic Echo Cancellation Applications", *VLSI Signal Processing VI*, IEEE Workshop on VLSI Signal Processing, Veldhoven, Holland, 20-22 October 1993.

[16] D.Chillet, J.P.Diguet, J.L.Philippe, O.Sentieys, "Memory Unit Design for Real Time Applications", *Submited to Special Issue on Integrated Design of Embedded Computer Systems*, , .