How to Transform an Architectural Synthesis Tool for Low Power VLSI Designs

S. Gailhard, N. Julien, J.-Ph. Diguet, E. Martin
LESTER-UBS Lab, France
{gailhard,julien,diguet,emartin}@iuplo.univ-ubs.fr

Abstract

High Level Synthesis (HLS) for Low Power VLSI design is a complex optimization problem due to the Area/Time/Power interdependence. As few low power design tools are available, a new approach providing a modular low power synthesis method is proposed. Although based for the moment on a generic architectural synthesis tool Gaut, the use of different "commercial" tools is possible. The Gaut_w HLS tool is constituted of low power modules : High level power dissipation estimation, Assignment, Module selection (operators and supply voltage), Optimization criteria and Operators library. As illustration, power saving factors on DWT algorithms are presented.

1. Introduction

Most VLSI researches have been focused on optimizing circuit speed and area to perform complex signal processing. Indeed, the rapid advance in VLSI technology and the increasing computation required for recent real time DSP applications infer large chips density and high clock frequency. Then, the power dissipation minimization in modern circuits, such as mobile systems, is a crucial problem. To minimize the power consumption of a system, VLSI researches work on low power HLS tools. Power optimization can be processed at several levels and the expecting power saving factor at each level can be expressed as follows : behavioral (10-100%), architectural (10-100%), logical (10-50%) or physical (<20%) [3]. Therefore, at behavioral and architectural levels, the expected saving power is more significant : however these two domains have been less explored in the literature. The method integrated in our HLS *Gaut_w* proposes techniques in order to optimize the design at these two levels.

In this paper, a new approach to HLS tool for low power and time constrained DSP systems is proposed. The next section presents previous works on power estimation and optimization. Section 3 explores briefly our *Gaut_w* HLS tool and its different modules *(High Level Power Estimation, Module Selection, Optimization Criteria, Assignment*). In section 4, results on DWT algorithm [2] illustrating our method are given before a conclusion.

2. Previous Works

2.1 Power estimation

There are three major sources of power dissipation in digital CMOS circuits summarized in the following equation (1). The first term represents the power switching component, the second term is due to direct-path short circuit current and the last term is the power dissipation due to leakage current.

$$P = P_{switching} + P_{shortcircuit} + P_{leakage} = (C.p).V_{dd}^2.f \quad (1)$$

where C is the physical capacitance of the CMOS circuit, p is the power consuming transition ($0 \rightarrow 1$) probability and V_{dd} the supply voltage.
For CMOS circuits is assumed that both short-circuit and leakage power dissipation are negligible. Therefore, the power dissipation is given by (2) [4] :

$$P = C_{comp}.V_{dd}^2.f \quad (2)$$

where C_{comp} is the average capacitance switched to perform a computation and f the sampling frequency.
It is well known that the signal statistic influences the power dissipation [5]. Nevertheless, at a high level, it can be supposed that signal values are uniformly distributed. Then, the capacitance C_{comp} is described by the following expression (3) [6]:

$$C_{comp} = \sum_i N_i.C_i + N_{reg}.C_{reg} + C_{interconnect} + C_{control} \quad (3)$$

where N_i is the number of i operations, C_i the i operator capacitance (intrablock routing and gate capacitance), $N_{reg}C_{reg}$ the effective register capacitance, $C_{interconnect}$ the physical interconnected capacitance estimation and $C_{control}$ the controller capacitance estimation.

At the architectural level, the complete architecture is known and signal statistics in the circuit can be calculated. Therefore, power dissipation estimation in a typical execution can be computed more precisely [5].

2.2 Power optimization

Low power optimization is achieved by reducing either C_i with technology considerations or C_{comp} by the operators selection and the architectural synthesis. Moreover, voltage scaling has a great impact on the power dissipation because of the quadratic dependence of V_{dd}. Unfortunately, reducing V_{dd} increases operator latency T significantly when the supply voltage approaches the device threshold voltage V_t (4) [4] :

$$T \propto \frac{V_{dd}}{(V_{dd} - V_t)^2} \quad (4)$$

Therefore it is important to find a good trade-off between power dissipation and performance.

2.3 Low power HLS tools

Few HLS optimization tools have been proposed in the literature. Hyper_LP [6] has integrated high level transformations and an ATP space exploration that shows the supply voltage scaling impact on the design. Concerning the module selection, most of previous works only goals design area optimization [7]. Moreover, a module selection (different operator set at different supply voltages) has been proposed which gives the lower bounds on Area, Performance and Power dissipation [8]. Scheduling methods have been presented to schedule Multi-Voltage Datapaths [9].
Nevertheless, no HLS synthesis tool using a modular approach and a classical architectural synthesis tool has been proposed yet.

3. HLS tool *Gaut_w*

The three entries are the algorithmic description of the application (behavioral VHDL), the operator library and the optimization criteria. At present, this tool is based on the *Gaut* architectural synthesis tool [1,14], but such a global method could be extended to any other tool. *Gaut_w* contains two low power modules : *Module Selection* that selects the best operator set and the best supply voltage for a given time constraint, and *Assignment* that assigns operations of the Data Flow Graph (DFG) on physical operators in order to decrease the transition rate of operator entries. These modules minimize a cost function (depending on both area and power) whose weighted parameters are chosen by the user (*Optimization Criteria).* The design is optimized through a *Fast Power dissipation estimation* based on average power dissipation of resources (*Operators library)* used in the architecture and a probabilistic estimation of accesses on each different resource (registers, memory, operators).

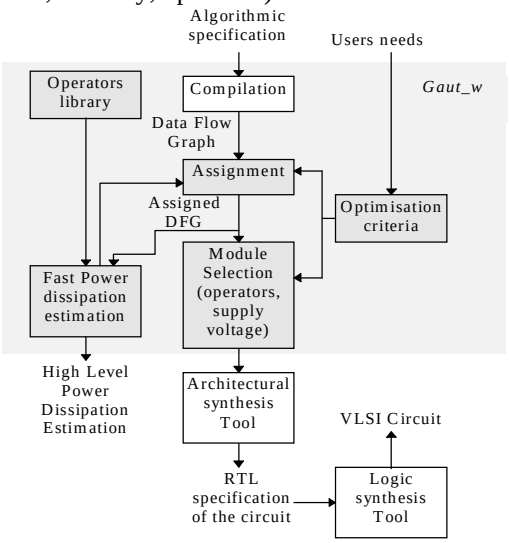

Figure 1 : Low Power DSP HLS Tool

3.1 Formal library

The synchronous architecture synthesized by *Gaut* tool used operators, memories, registers and a clock tree.

Operators :
Each operator has been simulated by the logic simulator *Compass* to estimate the average power dissipation [10]. This estimate is the mean result of 5 simulations with 10.000 values length stimuli [11] implying a 95% confidence interval and a less than 1% error. This method provides a good trade-off between the simulation time and precision of the results.
So, the library contains operators with different ATP characteristics. For example, an addition function can be realized with 4 different operators (Table I).

Memory library :
The great transistors number in a memory implies that the power dissipation due to the leakage current is no more negligible according to the switching component. The power dissipation estimator integrated in the *Compass* CAD tool is used to determine the switching and the leakage component of the power dissipation. These terms depend on the memory size.The memory power dissipation is also based on a static power dissipation $Stat(S).Vdd^2$ and on a dynamic effective

capacitance $C_{dyn}(S)$. No analytic expression but only experimental values for each memory size are available.

clock l :
The C_{clk} clock tree capacitance estimation is based on the $\overline{N}_{Reg}(S)$ estimated register number and on the C_{reg} clock tree capacitance per register: . $C_{Clk}(S) = \overline{N}_{reg}.C_{reg}$.

Library example :
Table I represents a library extract, where *T* is the latency time for a 5.5 Volts supply voltage, *S* the area and *P* the power dissipation estimation for a 5.5 Volts supply voltage. The operator C_i effective capacitance can be calculated with expression (2) (*P*, V_{dd} and *f* are known). The latency time at different supply voltages is estimated using expression (4) (*Vth* is a known device characteristic).

16 bits Operators	T (ns)	S (10^{-3} mm^{2})	P (µW)
Add (DataPath)	14	73	412
Fast add (DataPath)	8.6	122	711
Add (VHDL)	18.9	53	229
Fast add (VHDL)	9.9	117	626
Mult (DataPath)	32.3	1337	18827
Pipeline mult (DataPath)	22	1337	18821
Mult (VHDL)	53.3	1023	11820
Fast mult (VHDL)	33.2	1270	16231
Register	4	70	71
Clock per register	-	-	65
memory 28×16 bits Static : 36 mW	20	1163	961

Table I : Library example (1 µm)

3.2 Power dissipation estimation

The aim of this module is to give a fast power dissipation estimation calculated at a high level of abstraction. Currently, the method targets time constrained DSP applications : operators used in the circuit have an activation rate close to 100%.
The first estimation step is the scheduling probability computation of each operation in the DFG [12]. This allows to estimate the storage probability of each variable (transfer via a register, the memory and a register or transfer via a register) and then the probable activity of the memory (N_{mem}) and the registers (N_{Reg}). The probable register number can also be obtained allowing to estimate the power dissipation of the clock tree. The N_i operator activity is calculated on the DFG. Therefore, this probable activity of resources and their average power dissipation available in the library leads to the total power dissipation *Power(S)* (5) :

$$Power(S) = V_{dd}^2 \left(\frac{1}{T_r}.\left(C_{Op}(S) + C_{Reg}(S) + C_{Dyn\_Mem}(S)\right) + \frac{1}{T_{Clk}}.C_{Clk} + Static_{Mem}(S) \right) \quad (5)$$

with :

$$C_{Op}(S) = \sum_{i=1}^{N_0} N_i.C_i \qquad C_{Reg}(S) = N_{reg}C_{reg}$$

$$C_{Clk}(S) = \overline{N}_{reg}.C_{reg} \qquad C_{Dyn\_Mem}(S) = N_{mem}.C_{dyn}$$

where $C_{Op}(S)$ and $C_{Reg}(S)$ represent respectively the effective capacitance of operators and registers, each of these capacitances switching every T_r *ns* (time constraint). C_{clk} is the clock tree capacitance estimation, $\overline{N}_{Reg}(S)$ the probable register number and C_{reg} the capacitance per register (C_{Clk} switches every T_{Clk} ns : internal clock frequency). $Stat(S).Vdd^2$ is the static power dissipation and $C_{Dyn\_mem}(S)$ the dynamic effective capacitance for the memory.

3.3 Optimization criteria

Each designer has its own optimization problem : circuits for mobile electronic systems require low power dissipation, whereas some designs need a minimum area circuit in order to be cheaper. Therefore, the optimization criteria must depend on both area and power (Figure 2).

A lot of functions can be integrated in our optimization criteria module. Cost functions using linear expressions of both area and power dissipation have been chosen; an α factor ($0 \le \alpha \le 1$) is introduced to select the power and area rate. β is a normalization coefficient (6).

$$Cost_{\alpha}(S) = (1-\alpha).\beta.Area(S) + \alpha.Power(S) \quad (6)$$

where *Area(S)* is the area estimation (operators, registers, interconnections and bus [7]), *Power(S)* is the power dissipation estimation (5).

3.4 Assignment

Operations assignment on operators in the circuit is an important task in the architectural synthesis to decrease the power dissipation. A high switching factor at operator entries implies a high switching rate of operator gates and then a high power dissipation [5]. But, at a high level of abstraction, it is impossible to know exactly the circuit architecture and to calculate the activity rate of all operators entries. Therefore, only two models of operator power dissipation are used :

- both of the operator entries change between two operations with white noise inputs (two variables operation).

• only one of the operator entry changes between two operations, the other one is constant (a variable and a constant operation).

The power dissipation is greater in the first model than in the second one in a 25 to 50% order depending on the operator. Therefore, at a high level of abstraction, it is important to guide the architectural synthesis to use the second model operator as often as possible in order to optimize the design power dissipation. Let's take an example with the following DFG. A, B, C, D and E are variables and their values change every clock cycle, but Cst is a constant that keeps the same value for every clock cycle.

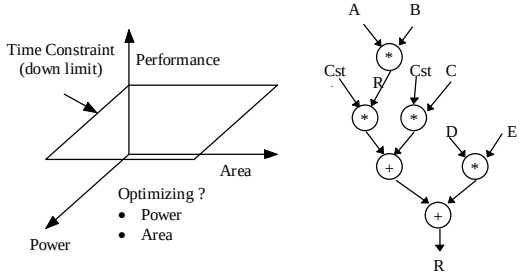

Figure 2 : Optimization criteria / Figure 3 : DFG

Multiplication on the DFG can be scheduled in different ways on one or more multipliers. With the power dissipation model used, it is clear that the minimum power dissipation is obtained when one multiplier is dedicated only for multiplication with the *Cst value* (Constant value) : two multiplications are done with the second model (*Cst*R*, *Cst*C*) and two multiplications with the first model (*A*B* and *D*E*) to compute the DFG. If only one multiplier is used, the operator power dissipation is greater : for example, if *Cst*R* is scheduled after *A*B*, then both of the two operator entries change (*A→ Cst and B→R*) and then the multiplication is done in the first model.

Therefore, an assignment module detecting operations with constant has been developed. Then, the *Assignment Module* assigns or not operators performing operations with constant to optimize the cost function (6) defined by the user (*Optimization Criteria*).

3.5 Module selection

The module selection is the second step in the HLS GAUT design flow (Figure 1). Its utility is to determine the optimal supply voltage and the optimal operator set from a given library according to a DSP application (an algorithm and a time constraint) for design optimization.

The module selection presented here explores all the different solutions dealing with different operators (ATP, mono or multi-function, pipeline operators) and different supply voltages. Our DFG processing is carried in the order defined in Figure 3. To estimate the cost function (6), the operator allocation has to be obtained from the number of pipeline stages. This requires to know the operator latency and therefore the selected operators and the supply voltage (4). To simplify the ATP space exploration and reduce the cost estimation number, the following lemma is proposed [7] : the use of multi-functional units is necessary only if one of these functions is underused, e.g. if there is at least one mono-functional operator in the operator set S with an efficiency less than 100%. So, a first step provides the best selection of mono-functional operators. If some of them are underused, an optimization with multi-functional units is processed for reducing the circuit area.

for *Supply voltage* from *Vdd_min* to *Vdd_max*
  for all *mono-functional* operators set
  Latency time calculation of each selected operator (4)
  Number of pipeline stages calculation
  Allocation estimation in each pipeline stage
    Cost calculation (6)
    End *mono-functional* operators set
    Multifunctional operators resolution
    => Best operators set selection for each supply voltage
  End for *Supply voltage*
  => Best selection (*Supply voltage* and *operators set*)

Figure 4 : Module Selection Algorithm

In conclusion, our module selection algorithm provides the best operator set for each supply voltage (mono and multi-functions) and the best supply voltage selection in a range from *Vdd_min* to *Vdd_max* (Figure 4) [13].

Nevertheless, it is possible to use normalised supply voltages (5, 3.3 ... Volts).

This ATP exploration leads to the minimum of the cost function chosen by the designer.

4. Results

There are a lot of image transformations as Fast Fourier Transform, Discrete Cosinus Transform, and Wavelet transform. The last one has got several interesting properties and is more and more used. The wavelet transform used here is based on a "g" high pass filter and on a "h" low pass filter [2]. Several steps are needed to calculate the results as shown in the Figure 5 and 6. The *H* and *G* filters are first computed on rows and after on columns of the initial image.

To calculate the DWT with one more resolution level, the same computation is realized on a quarter of the image.

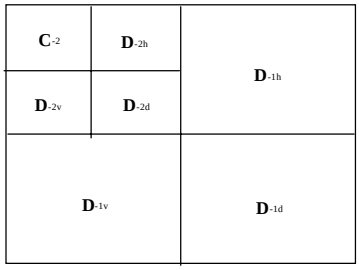

Figure 5 : Result after the DWT algorithm with two resolution levels

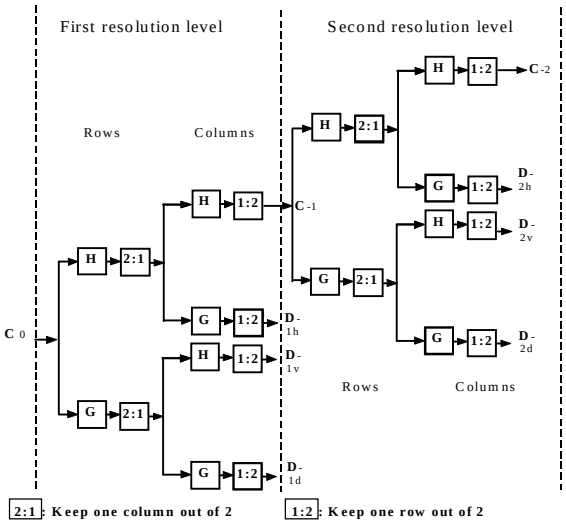

Figure 6 : DWT with three resolution levels

Several h and g filters are possible and two linear phase FIR filters with different sizes have been chosen because of their good results [2] :
9-3 : 9th (h) and 3rd (g) linear phase FIR filter
9-7 : 9th (h) and 7th (g) linear phase FIR filter
The algorithm can be used with different resolution levels. The example of 512×512 pixel images at the frequency of 10 images per second is treated : it implies a time constraint at 1/10=100 ms per image. To satisfy this constraint, one algorithm is specified on a *n*×*n* block (*algo_bloc*) and another one realizes the calculus only on one image point (*algo_point*).

The power dissipation of both algorithms is compared just after their description. Therefore, the *power dissipation estimation* module permits to select at the first step of the system design the more suitable algorithm according to the power dissipation.
Table II gives results of the *power dissipation estimation* and the required CPU time on a *Sparc Station*. It has been proved on a moving detection application [11] that the absolute power dissipation error is less than 20% compared with the value obtained by a logic simulator *Compass* (after architectural and logical synthesis with *Gaut* [1,14])

	algo_bloc	*algo_point*
Power dissipation estimation	1131 mW	436 mW
Relative Power Dissipation	1	0.39
CPU time ona *Sparc Station*	13 minutes	7 seconds

Table II : DWT transforms comparison

These results show that the *algo_point* algorithm is more convenient to a power minimisation and leads to a 61% power saving factor.

5. Module selection results

The *algo_point* algorithm is selected to illustrate results on *Module Selection*. The circuit has to implement the algorithm with three resolution levels on 10 images per second. So, it needs to specify a processor that performs two FIR filters (g, a *7th* and h, a *9th* linear phase FIR filters) (7) every 300 ns :

$$y(n) = h_0.x_n + \sum_{i=0}^{3}\left(h_i.\left(x_{2n-i} + x_{2n-i}\right)\right) \qquad (7)$$

$$y'(n) = g_0.x_n + \sum_{i=0}^{2}\left(g_i.\left(x_{2n-i} + x_{2n-i}\right)\right)$$

Table I in section 3 is the part of the operators library used in our HLS tool for this application. The time constraint is 300 ns. Figure 7 shows the results (*Power(S)* (5), *Area(S)* and *Cost(S)* (6)*)* of the best operator set for each supply voltage, with α equal to 0.5.

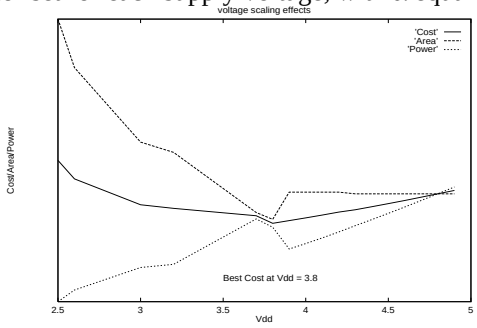

Figure 7 : Cost, a Vdd function (DWT algorithm)

The power dissipation decreases with the supply voltage because of the quadratic dependence of V_{dd} on the power dissipation (1). However, when the supply voltage decreases, operator latency (4) increases, that involves more operators allocation and a more important area.
Table III shows the HLS GAUT tool results for some supply voltages : best operator set (*Module selection*), operator power dissipation estimation, total area and operator number in the circuit after an architectural synthesis.

Vdd *V*	Number of selected Operator (latency time (ns))	Power *mW*	Area mm^2
4.9	1 Adder datapath (20) ; 3 Mult Vhdl (70)	360	5.16
3.9	2 Adder Vhdl (40) ; 3 Mult Vhdl (90)	222	5.76
3.8	2 Add Vhdl (40) ; 2 Fast Mult Vhdl (60)	277	4.82
3.2	2 Add Datapath (40) ; 3 Fast Mult Vhdl (80)	199	6.14

Table III : Module selection results(DWT)

Breaks in the Figure 7 are explained by the table III. For a 3.9 Volts supply voltage, three multipliers and two adders are allocated. However, at 3.8 Volts, two adders and only two multipliers are allocated. This infers that the area cost is smaller at 4.5 Volts whereas the power dissipation estimation is greater.

Cost functions (6) depend on the α term which represents the relative importance between the area and the power dissipation. Figure 8 shows the impact of this term variation on the cost function. The optimal supply voltage when $0.2<\alpha<1$ is around 3.8 Volts. It means that the supply voltage can decrease from 5 Volts to 3.8 Volts without a significant area penalty (only few per cents) and with a 40% power dissipation optimization. Therefore, the α term allows the user to optimize the design according to his application problem (area or power dissipation).

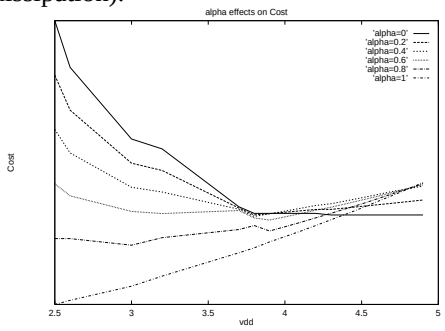

Figure 8 : Apha impact on the cost function

Module Selection requires less than one CPU minute on a *Sparc station*. The method can also be applied on a large operator library and on complex applications without computation time problems.

6. Conclusion

A generic method for low power VLSI design, integrated in the *Gaut_w* HLS tool (*Estimation* and *Optimization)* is presented here. This new approach is modular and is realized before the architectural synthesis. Therefore, it can be applied to any other architectural synthesis tool. A fast power dissipation estimation at a high level of abstraction has also been developed which compares different time constrained DSP algorithms at an early stage of the design.

Results presented on DWT algorithms show that the *High level power dissipation estimation* permits to obtain quickly power savings equal to 3 by the selection of the best algorithm. Moreover, a 40% power reduction can be reasonably envisioned by using *Module selection* without a significant area penality.

References :

[1]E. Martin, O. Sentieys, H. Dubois, J.-L. Philippe, "GAUT, an architectural synthesis tool for dedicated signal processors", *Proceedings EURO-DAC 93*

[2]M. Antonini, M. Barlaud, P. Mathieu, I. Daubechies, "Image coding using wavelet transform", *IEEE trans. on Image Processing*, Vol. 1, No 2, Avril 92, pp. 205-220

[3]P. E. Landman, "Low-power Architectural design methodologies", PhD Thesis, Berkeley, August 94

[4]A. Chandrakasan, S. Sheng, R. W. Brodersen, "Low Power CMOS digital Design", IEEE journ. of solid state circuits, Vol. 22, N. 4, pp. 473-484

[5]P. E. Landman, J. M. Rabaey, "Architectural power analysis: the dual bit method", IEEE trans. on VLSI systems, Vol. 3, No 2, Junev 1995, pp. 173-187

[6]A. Chandrakasan, M. Potkonjak, R. Mehra, J. Rabaey, R. Brodersen, "Optimising Power Using Transformations", IEEE Trans. on CAD and Systems, Vol. 14, No 1, pp. 12-30, Jan. 95

[7]O. Sentieys, J.-Ph. Diguet, J. L. Philippe, E. Martin, "Hardware Module Selection for Real Time Pipeline Architectures using Probabilistic Estimation"; 9th Annual IEEE Inter. ASIC Conf. and Exhibit, 1996, pp.147-150

[8]Z. S. Shen, C. C. Jong, "Exploring module selection space for architectural synthesis of low power designs", IEEE Int. Symp. on circuits & systems, June 97, pp.1532-1535

[9]M. C. Johnson, K. roy, "Scheduling and optimal voltage selection for Low-Power multi-volatge DSP Datapaths", IEEE Int. Symp. on circuits and systems, Hong Kong, June 97, pp.2152-2155

[10]R. Burch, F. Najm, P. Yang, T. "Trick, McPOWER a monte carlo approach to power estimation", IEEE CAD 1992, pp. 90-97

[11]S. Gailhard, O. Ingremeau, N. Julien, J.-Ph. Diguet, E. Martin, "Une méthode probabiliste pour estimer la consommation à un niveau algorithmique", Colloque CAO, Villars de Lans, Jan. 97, pp. 124-127

[12]J-Ph. Diguet, O. Sentieys, J. L. Philippe, E. Martin; "Probabilistic resource Estimation for Pipeline Architecture", IEEE Workshop on VLSI S.P.,Sakai, Japan, Oct. 95

[13]S. Gailhard, O. Sentieys, N. Julien, E. Martin, "Area/Time /Power Space Exploration in Module Selection for DSP High Level Synthesis", Int. Workshop, PATMOS'97, Louvain la Neuve, Belgium, Sept. 97, pp.35-44

[14]A. Baganne, J.-L Philippe, E. Martin , "Hardware Interface Design For Real Time Embedded Systems", 7-th IEEE GLS-VLSI 97, Urbana Champain , Illinois, Mars 1997