Power Estimation of a C algorithm on a VLIW Processor

Nathalie Julien, Eric Senn, Johann Laurent, Eric Martin
LESTER, University of South Brittany, France
Nathalie.Julien@univ-ubs.fr

Abstract

A complete methodology to estimate power consumption directly at the C-level for on-the-shelf processors is proposed. It relies on a power model of the processor that describes the consumption variations relatively to algorithmic and configuration parameters. The algorithmic parameters represent the power and quality metrics of the code and can be predicted directly from the C-algorithm with simple assumptions on the compilation. To check the algorithm performances with the application constraints without compiling, direct estimation results on the C code can be summarized on a consumption map. This method strongly reduces the design complexity in terms of number of lines to be studied and allows to spot the 'hot parts' of the code in order to target the writing effort. Applied to a VLIW processor, the TI TMSC6201, the estimation method provides an accurate power consumption estimation together with the maximum and minimum bounds; a maximum error of 8% against measurements for only 1.3% of the code studied is obtained for a MPEG decoder; other classical DSP applications are also presented.

1. Introduction

Algorithm designers have mainly focused on improving performances. But software can have a substantial impact on the power dissipation of a system [1]. Moreover, two codes can have the same performances but different energy dissipation [2]. As power consumption is currently a decisive design criteria, the programmer needs to easily characterize his algorithm at the system level.

The power consumption estimation of a C algorithm has several interests. As the feedback to the designer is very fast, the programmer is efficiently guided in his choices. For a given algorithm, power consumption can be estimated on different processors without compiling. The best target can then be selected, without specific development tools. For a given processor, power consumption of different scripts of the same algorithm can be easily checked with the application constraints.

For on-the-shelf processors, details about the processor micro-architecture are often unavailable. This assumption prohibits methods based on cycle-level simulation like in Wattch or SimplePower [3-4]. In this case, a classical approach is to evaluate the power consumption of an algorithm by the instruction-level power analysis (ILPA) [5]. This method relies on current measurements for each instruction and instruction pair. Its main limitation is the unrealistic number of measurements for complex architectures. Some approaches have proposed to group the instructions [6] or to work on a reduced instruction set [7]; but still, parallelism possibilities are not taken into account. Finally, recent studies have added a functional approach [8, 9]. All these methods perform power estimation only at the assembly-level with an accuracy from 2-4% for simple models to 10% when parallelism and pipeline stalls are effectively considered.

This paper demonstrates that, differing from the instruction level approach, a functional approach makes power estimation at the C-level possible. A first power estimation method has been initially developed and validated for the assembly-level with a maximum error of 3.5% against measurements [10]. This method is composed of two steps: the model definition and the estimation process. The model definition provides a complete power model of the processor with algorithmic and configuration parameters as inputs. The estimation process analyzes a reduced part of the code and extracts the required parameters. From the same power model of the processor, we propose here to predict some algorithmic parameters directly from the C-algorithm, assuming different ways of compiling the code. The maximum and minimum bounds are obtained along with an accurate estimate, with an average error of 4.4% against physical measurements. A *'consumption map'* is also provided to describe to the designer the power variations of the algorithm.

The estimation methodology and the model definition are presented in section 2. The Functional Level Power Analysis is explained through a case study: the VLIW processor TMS320C6201. Then, the C-level estimation process is detailed in section 3 together with the different prediction models, defined to evaluate algorithmic parameter values. In section 4, estimation results for several DSP applications are provided. First, the accuracy of the estimation method is validated. Then, we exhibit how to use these estimates to guide the designer. Finally, current and future works are presented in conclusion.

2. Model Definition

2.1 Estimation Methodology

The two steps of the estimation methodology are represented in Figure 1. The *Model Definition* is done once

and before any estimation to begin. It is based on a Functional Level Power Analysis (FLPA) of the processor, that determines the relevant parameters and provides the complete power model of the processor. This model is a set of consumption rules that describes how the average supply current of the processor core evolves with some algorithmic and configuration parameters. These rules were elaborated from a reduced set of physical measurements for elementary assembly programs.

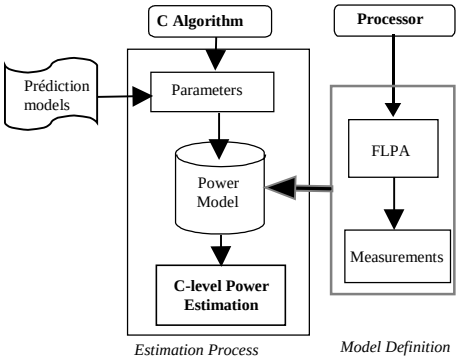

Fig. 1. The estimation methodology

The *Estimation Process* is done every time the consumption of an algorithm has to be evaluated. At the assembly-level, algorithmic parameters are directly computed from the compiled code through a simple profiling. These parameters are the inputs of the power model of the processor [10]. At the C-level, algorithmic parameters are not known exactly; they must be predicted from simple assumptions about the capability of the compiler to efficiently target the processor architecture. These assumptions are defined in the prediction models.

2.2. Case study of the TMS320C6201

The FLPA has been applied on the C6201 from Texas Instruments for which a complete power model has been developed. This processor has been chosen for its complex architecture: a deep pipeline (up to 11 stages), VLIW instructions set, and parallelism capabilities (up to 8 instructions in parallel). It also contains an External Memory Interface (EMIF), used to load data and program from the external memory. Its clock frequency F can reach 200 MHz. Its internal program memory can be used in four different memory modes (MM). In the mapped mode (MM_M), all the instructions are in internal memory. In the bypass mode (MM_B), all the instructions are in external memory. In the cache mode (MM_C), the internal program memory is used as a direct mapped cache and the freeze mode (MM_F) is similar to the cache mode with no writing allowed [11].

From our experiments on the C6201, several preliminary remarks can be done on the power dissipation in this VLIW processor. First, there is no significant power consumption variations between different operations: an addition or a multiplication nearly dissipates the same amount of power. The same conclusion occurs for a read or a write instruction in the internal memory. Moreover, the effect of data correlation on the global power consumption is less than 2%. It seems that the architecture complexity hides many power variations, relatively to consumption cost of cache misses or pipeline stalls.

The FLPA actually consists in a functional analysis of the architecture from the power point-of-view. The aim is to determine which parameters are significant for the global power consumption. The FLPA results for this processor are summarized in Figure 2. The architecture is divided into four blocks: the Instructions Management Unit (IMU), the Processing Unit (PU), the Memory Management Unit (MMU) and the Control Unit (CU). The CU contains every configuration device in the DSP (PLL, Direct Memory Access - DMA control registers, EMIF control registers, etc). As its power consumption is relatively negligible in signal processing applications, it is not represented here although both pipeline control and sequencer are actually taken into account.

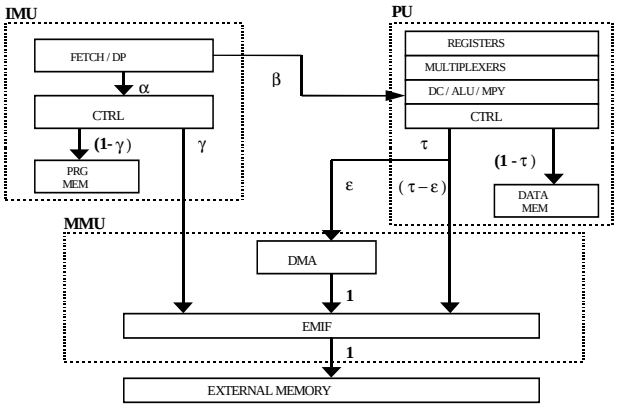

Fig. 2. FLPA on the C6201

Each link on this functional diagram is associated to an algorithmic parameter, that represents its activity rate, directly impacting on the global power consumption. The *parallelism rate* α assesses the flow between the fetch stages and the internal program memory controller inside the IMU. The *processing rate* β between IMU and PU represents the average utilization rate of the processing units (ALU, MPY). The activity rate between IMU and MMU is expressed by the *program cache miss rate* γ. The parameter τ corresponds to the *external data memory access rate*. The parameter ε stands for the activity rate between the data memory controller and the DMA. All these parameters also are representative metrics of the code.

2.3. Power Model of the processor

Once the functional analysis achieved, consumption rules have to be precisely determined to get the complete power model. These rules are mathematical functions of both algorithmic and configuration parameters. To determine these functions and their coefficients, the average supply current of

the processor core I_{TOTAL} was measured in relation with the variations of each parameter. These variations were achieved by the mean of small programs, called scenarios, which are unbounded loops written in assembly language. The consumption rules were finally obtained by curve-fitting the measurements. Current measurements are done on the core supply pad (with the supply voltage V_{DD} = 2.5 V) and do not include external memory. Though the choice of the external memory fully relies on the designer, the addition of a generic memory model based on works in [12, 13] will be an important part of future developments.

Algorithmic parameters defined in Figure 2 are α, β, γ, τ and ε. Although the DMA is modeled, for the sake of simplicity, the ε parameter will be set here to 0. In fact, the four other parameters are not fully independent. Indeed, γ and τ directly impact on the number of pipeline stalls, and then on the pipeline stall rate (PSR) modifying the average parallelism rate and the average number of processing units. As a result, only 4 algorithmic parameters α, β, PSR and γ are the inputs of the final power model.

The consumption rules obtained for the TMS320C6201 are given in Table 1. These rules express the average supply current I_{TOTAL} by linear functions of both algorithmic and configuration parameters. The configuration parameters are the clock frequency (F) and the memory mode (MM). Values of the constant coefficients a_i, bi, c_i, d_i, e_i and f_i with $\{i = 0 \text{ to } 4\}$ can be found in [10] together with details on their determination. The dependence between parameters implies that our expressions are more complex than those derived from a linear regression analysis. The static contribution, actually known as a non-negligible part in the power dissipation, appears explicitly in the consumption rules.

Table 1. Consumption Rules for the C6201

MM	CONSUMPTION RULES I_{TOTAL} =
MM_M	$a_0\beta(1 - PSR)F + (a_1\alpha(1 - PSR) + b_1)(c_1F + d_1)$
MM_B	$(a_0\beta(1 - PSR) + b_2)F + c_2$
MM_C	$a_0\beta(1 - PSR)F + (a_3\alpha(1 - PSR) + b_3)(c_3\gamma + d_3)(e_3F + f_3)$
MM_F	$a_0\beta(1 - PSR)F + (a_4\alpha(1 - PSR) + b_4)(c_4\gamma + d_4)(e_4F + f_4)$

Finally, the global power consumption P for the application is computed as follows:

$$P = V_{DD} * I_{TOTAL} \quad (2)$$

This processor power model, first settled and validated for the assembly level estimation, can also be used for the C-level power estimation, as presented now in the next section.

3. Estimation Process

The inputs of the power model of the processor are both configuration and algorithmic parameters. The configuration parameters are part of the application and therefore are known at the C-level. Among the algorithmic parameters, the pipeline stall rate PSR and the cache miss rate γ are strongly depending on the data mapping, the processor architecture and the writing of the code. In several cases, they can be defined (in the mapped memory mode, γ = 0) or approximated; else, a dynamic profiling of the code would be necessary to obtain these parameters. The section 4 will present how a *consumption map* of the algorithm is provided when γ and the PSR are still undetermined at the early step of the design process.

The two remaining algorithmic parameters are α and β. In the C6201, 8 instructions are fetched at the same time. They form a *fetch packet* (FP). In this fetch packet, operations are gathered in *execution packets* (EP) depending on the available resources and the parallelism capabilities [11]. The parallelism rate α and the processing rate β are computed as follows:

$$\alpha = \frac{NFP}{NEP} \leq 1 \; ; \beta = \frac{1}{NPU_{MAX}} \frac{NPU}{NEP} \leq 1 \quad (3)$$

NFP and *NEP* stands for the average number of respectively FP and EP. *NPU* is the average number of processing units (every instruction except the NOP) and NPU_{MAX} is the maximum number of processing units; here, NPU_{max} = 8.

Then, the determination of the α and β parameters relies on the knowledge of *NFP*, *NEP* and *NPU* that directly depend of the compiled code. The prediction of these parameters must anticipate the way the code is compiled. According to the processor architecture, four prediction have been defined for DSP applications, where loops are dominant:

- the *sequential model* (*SEQ*) is the simplest since it assumes that all the operations are executed sequentially. This model is only realistic for non-parallel processors.

- the *maximum model* (*MAX)* corresponds to the case where the compiler fully exploits all the architecture possibilities. In the C6201, 8 operations (with 2 load instructions maximum) can be done in parallel. This model gives a maximum bound of the application power consumption.

- the *minimum model* (*MIN*) assumes that load and store instructions are never executed at the same time - indeed, it was noticed on the compiled code that all parallelism capabilities were not always fully exploited for these instructions. That will give a lower bound for the algorithm's power consumption.

- at last, the *data model* (*DATA*) expresses more acutely the parallelism of load and store instructions. It supposes that one load and one store can be executed in the same cycle only if they involve two different data.

By our experience on the TI compiler, the performance compiler optimization is a more efficient way to optimize the power consumption of the application than optimizing the code size. Considering that the user will always try to compile with the best results, we consider the highest level for the performance. Of course, in another case, as example a specific low power compiler, it could be possible, if necessary, to develop a more appropriate prediction model. But the prediction only relies on the quality of the results of the compilation in terms of using properly the architecture.

As illustration, a simple example is presented here.

For (i = 0; i < 512; i++)

*Y = X[i] * (H[i] + H[i+1] + H[i-1]) + Y ;*

In the loop nest are needed 4 loads (LD), and 4 other operations (OP): 1 multiplication, and 3 additions. Operations at the beginning or at the end of the loop body are neglected. As example, the final store for Y, only done once at the end of the loop, is not considered. Here, our 8 operations will always be gathered in one single FP so *NFP* = 1. Because no NOP operation is involved, *NPU* = 8 and α and β parameters have the same value.

In the *SEQ* model, all instructions are assumed to be executed sequentially. Then *NEP* = 8, and $\alpha = \beta = 0.125$. Results for the other models are summarized in Table 2.

Table 2. Prediction models for the example

MODEL	EP1	EP2	EP3	EP4	α, β
MAX	2 LD	2 LD, 4 OP	-	-	0.5
MIN	1 LD	1 LD	1 LD	1 LD, 4 OP	0.25
DATA	2 LD	1 LD	1 LD, 4 OP	-	0.33

Of course, realistic cases are more elaborated: the prediction has to be done for each part of the program (loop, subroutine…) for which local values are obtained. The global parameter values, for the complete C source, are computed by averaging all the local values. Such an approach permits to easily spot 'hot points' in the program.

4. Applications

First, the estimation method at the C-level is validated by a direct comparison with measurements. Next, an application of this estimation method to explore the power consumption of an algorithm is proposed.

4.1 Estimation validation

Our prediction models are applied on classical digital signal processing algorithms: a FIR filter, a FFT, a LMS filter, a Discrete Wavelet Transform (DWT) with two different image sizes (64*64 and 512*512), an Enhanced Full Rate (EFR) vocoder for GSM and a MPEG1 decoder. In Table 3 are reported the size of the different C and assembly codes. Obviously, studying directly the C code instead of the assembly code strongly reduces the complexity of the estimation and then improves its rapidity. Moreover, for the most complex application (MPEG with 11 different functions), only 1.3% of the C code has to be studied.

Table 3: Reduction of the complexity in code line number

Application	**Line code number**		**C Lines studied**	
	C	**ASM**	**Number**	**%C**
FFT	77	408	10	13
LMS	30	408	4	13.3
DWT 64*64	46	714	17	37
EFR	118	1323	37	31.2
MPEG	2267	8488	30	1.3

The purpose here is to validate the C-level estimation method by evaluating its accuracy. The α and β algorithmic parameters are predicted as presented above. The parameter γ is set to 0 because the power model of the processor has already been validated at the assembly level for a variable cache miss rate [10]. The global power consumption is computed with the PSR obtained after compilation. Indeed, our aim is to provide the designer with estimates about all the possible consumption variations, including the real case. Results are presented in Table 4, for a nominal clock frequency F = 200MHz, different memory modes (MM) and data placement (INT/EXT). The relative error between power estimation and measurement is given for the DATA model.

The SEQ model provides unsatisfying results since it does not take account of the architecture possibilities. In fact, this model has been developed to explore the estimation possibilities without any knowledge about the architecture of the targeted processor.

It could be noticed that, for the LMS in bypass mode, all the prediction models overestimate the power consumption with close results. In fact, in this marginal memory mode, every instruction is loaded from the external memory and thus pipeline stalls are dominant. As the SEQ model assumes sequential operations, it is the most accurate prediction model in this mode.

Eventually, the estimation possibilities at the C-level are summarized:

- to determine precisely the power consumption without any knowledge about the targeted processor is not possible (SEQ model).

Table 4. Comparison between measurements and power estimation

Algorithm			Measurements			Power estimation (W)				
Application	MM	INT/EXT*	T_{EXE}	P(W)	Energy	SEQ	MAX	MIN	DATA	
FIR	MM_M	INT	6.885µs	4.5	30.98µJ	2.745	4.725	3.015	4.725	+5%
FFT	MM_M	INT	1.389ms	2.65	3.68mJ	2.36	2.97	2.57	2.58	-2.6%
LMS	MM_B	INT	1.847s	4.97	9.18J	5.02	5.12	5.07	5.12	+3%
LMS	MM_C	INT	165.75ms	5.665	939mJ	2.55	6	4.76	6	+5.9%
DWT 64*64	MM_M	INT	2.32ms	3.755	8.71mJ	2.82	4.24	3.27	3.53	-6%
DWT 64*64	MM_M	EXT	9.19ms	2.55	23.46mJ	2.295	2.63	2.4	2.46	-3.5%
DWT 512*512	MM_M	EXT	577.77ms	2.55	1.473J	2.27	2.61	2.37	2.45	-3.9%
EFR vocoder	MM_M	INT	39µs	5.0775	198µJ	2.54	5.636	3.86	5.13	+1%
MPEG decoder	MM_M	INT	40.37µs	5.823	235.08µJ	2.665	6.380	3.927	5.353	-8%
average error						28.2%	7.3%	15.2%	4.4%	

** INT/EXT: data in internal/external memory*

- a coarse grain prediction model, including only the architecture possibilities in terms of parallelism, number of processing units,... provides the maximum and minimum bounds of the algorithm power consumption with an average error of 7.3% and 15.2% respectively.

- the fine grain prediction model, with both elementary information on the architecture and data placement, offers a very accurate estimation with a maximum error of 8% against measurements.

4.2 Algorithm Power Consumption Exploration

If the *cache miss rate* (γ) and/or the *pipeline stall rate* (PSR) are unknown at the C-level, a *'consumption map'* is provided to the programmer. This map represents the power consumption variations of the algorithm according to these parameters. Thus, by evaluating sensible variations for these two parameters, it is possible to locate, on the *consumption map*, the probable power consumption limits. Furthermore, the major part of current embedded applications have a program size (after compilation) easily contained in the internal memory of the C6201 (64 Kbytes) which also gives $\gamma = 0$.

Let us reconsider the application of the EFR vocoder. The Figure 3 represents the power consumption exploration through all the prediction models for the mapped memory mode (γ = 0%). Of course, the PSR cannot be equal to 100% since no operation would be executed. Obviously, the average power consumption decreases when the PSR gets higher. In the same time, the minimum and maximum bounds of the estimation become closer because the PSR dominates the global power consumption by lowering the parallelism rate. The measurement value, very close of the DATA model, is also represented.

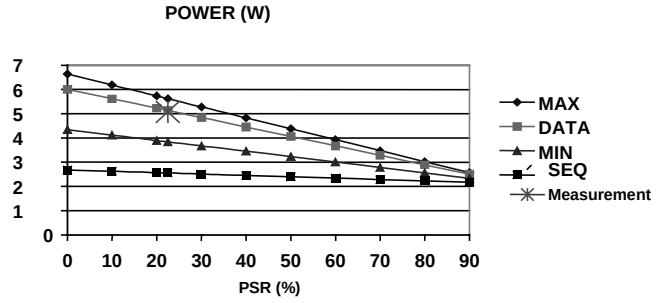

Fig. 3. Power Consumption Exploration for the EFR vocoder in mapped mode.

For the cache mode and the DATA prediction model, results are presented in Figure 4. Here, the cache miss rate γ also varies. The minimum power consumption value is obtained for γ = 0% and the maximum PSR. Indeed for these values, α and β are minimum. The maximum power consumption is obtained when γ = 100% and PSR = 0%; actually, this case is unrealistic since each cache miss provokes pipeline stalls through external memory access.

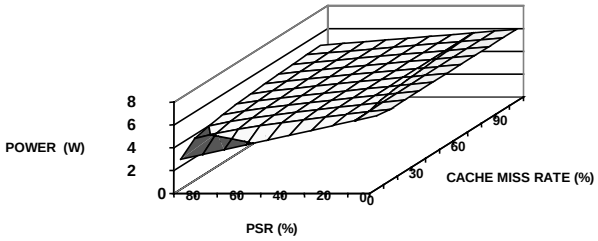

Fig. 4. Power consumption Exploration for the EFR vocoder in cache mode

The power consumption exploration points out if the algorithm does not respect the application consumption constraints (in terms of energy and/or power). Since, at the C-level, the execution time is unknown, the energy could be evaluated from the execution time constraint (given by the programmer). If the algorithm consumption estimation is always under the constraint, then the C code is suitable. Else, the programmer can focus on the more dissipating parts of the algorithm, spotted through the local α and β parameters; the program can be modified being aware on the data mapping that strongly affects pipeline stalls and cache misses. The comparison of several codes or parts of codes can be conducted with the algorithmic parameters as code quality metrics; actually, the higher α and β are and the lower PSR and γ are, the more the code is efficient, both for the performance and the energy consumption.

5. Conclusion

This paper has demonstrated the possibility of performing an accurate power estimation of a C-algorithm, reducing the complexity by focusing only on the loops in the code. A complete power model of the VLIW processor has been elaborated, taken account of important phenomena like pipeline stalls and cache misses. The conditions for this estimation have also been settled. For DSP applications, and with elementary information on both architecture and data placement, our C-level power estimation method provides accurate results together with the maximum and minimum bounds of the algorithm power consumption. . For a MPEG decoder where only 1.3% lines of the C-code have been studied, a maximum error of 8% is reported; such an accuracy is similar to the estimation methods at the assembly-level. When cache miss rate and/or pipeline stall rate are undefined at the C-level, a *consumption map* allows to verify if the application constraints are respected.

Current works are the development of an automatic tool, and the implementation of the FLPA method on other processors. Future works will concern the addition of a generic memory model to include the external memory in our power estimation.

References

1. K. Roy, M. C. Johnson "Software Design for Low Power," in *NATO Advanced Study Institute on Low Power Design in Deep Submicron Electronics*, Aug. 1996, NATO ASI Series, chap. 6.3.
2. M. Valluri, L. John "Is Compiling for Performance == Compiling for Power?," presented at the 5th Annual Workshop on Interaction between Compilers and Computer Architectures INTERACT-5, Monterey, Mexico, Jan. 2001.
3. W. Ye, N. Vijaykrishnan, M. Kandemir, M.J. Irwin "The Design and Use of SimplePower: A Cycle Accurate Energy Estimation Tool," in *Proc. Design Automation Conf.*, June 2000, pp. 340-345.
4. D. Brooks, V. Tiwari, M. Martonosi "Wattch: A Framework for Architectural-Level Power Analysis and Optimizations," in *Proc. Int. Symp. on Computer Architecture*, June 2000, pp. 83-94.
5. V. Tiwari, S. Malik, A. Wolfe "Power analysis of embedded software: a first step towards software power minimization," *IEEE Trans. VLSI Systems*, vol.2, n°4, Dec. 1994, pp. 437-445.
6. M. T.-C. Lee, V. Tiwari, S. Malik, M. Fujita "Power Analysis and Minimization Techniques for Embedded DSP Software," *IEEE Trans. VLSI Systems*, vol. 5, n°1, March 1997, pp. 123-135.
7. C. Brandolese, W. Fornaciari, F. Salice, D. Sciuto "An Instruction-Level Functionality-Based Energy Estimation Model for 32-bits Microprocessors," in *Proc. Design Automation Conf.*, June 2000, pp. 346-351.
8. L. Benini, D. Bruni, M. Chinosi, C. Silvano, V. Zaccaria, R. Zafalon "A Power Modeling and Estimation Framework for VLIW-based Embedded Systems," in *Proc. Int. Workshop on Power And Timing Modeling, Optimization and Simulation PATMOS*, Sept. 2001, pp. 2.3.1-2.3.10.
9. G. Qu, N. Kawabe, K. Usami, M. Potkonjak "Function-Level Power Estimation Methodology for Microprocessors," in *Proc. Design Automation Conf*, June 2000, pp. 810-813.
10. J. Laurent, E. Senn, N. Julien, E. Martin "High Level Energy Estimation for DSP Systems," in *Proc. Int. Workshop on Power And Timing Modeling, Optimization and Simulation PATMOS*, Sept. 2001, pp. 311-316.
11. TMS320C6x User's Guide, Texas Instruments Inc., 1999
12. S. Steinke, M. Knauer, L. Wehmeyer, P. Marwedel "An accurate and Fine Grain Instruction-Level Energy Model Supporting Software Optimizations," in *Proc. Int. Workshop on Power And Timing Modeling, Optimization and Simulation PATMOS*, Sept. 2001, pp. 3.2.1-3.2.10.
13. S. L. Coumeri, D. E. Thomas "Memory Modeling for System Synthesis," *IEEE Trans. VLSI Systems*, vol. 8, n°3, June 2000, pp. 327-334.