Power Estimation of a C algorithm based on the Functional-Level Power Analysis of a Digital Signal Processor

Nathalie Julien, Johann Laurent, Eric

LESTER, University of South Brittany - Lorient, FRANCE

Abstract. A complete methodology to estimate power consumption at the C-level for on-the-shelf processors is introduced. It relies on the Functional-Level Power Analysis, which results in a power model of the processor that describes the consumption variations relatively to algorithmic and configuration parameters. Some parameters can be predicted directly from the C-algorithm with simple assumptions on the compilation. Maximum and minimum bounds for power consumption are obtained, together with a very accurate estimation; for the TI C6x, a maximum error of 6% against measurements is obtained for classical digital signal processing algorithms. Estimation results are summarized on a *consumption map*; the designer can compare the algorithm consumption, and its variations, with the application constraints.

1 Introduction

Power consumption has become a critical design constraint in many embedded applications. Amount all the causes of dissipation, the software can have a substantial impact on the global consumption [1]. As two codes can have the same performances but different energy consumption [2], only a reliable power consumption estimation at the algorithmic level can efficiently guide the designer. Therefore, a C-level estimation can be worthwhile if it proposes a fast and easy estimation of the consumption characteristics of the algorithm. As there is no need of compiling, direct comparisons on different processors can be done before choosing the target and then without purchasing the development tools. It also allows to compare the consumption of different versions of the same algorithm to check if the application constraints are respected. More precisely, it allows to locate the 'hot points' of the program on which the writing efforts have to be focused.

When the applications are running on commercial processors, details on the architecture implementation are unavailable, and methods based on a cycle-level simulation [3][4] are then inappropriate. In this case, the classical approach to evaluate the power consumption of an algorithm is the instruction-level power analysis (ILPA) at the assembly-level [5]. This method relies on the measurements of the power consumption for each instruction and inter-instruction (a pair of successive instructions); but for complex

processor architectures, the number of required measures becomes unrealistic. Many studies are currently focusing on a functional analysis of the power consumption in the processor [6][7]. But all of them are also made at the assembly-level. Only one attempt of algorithmic estimation has already been made, concluding that a satisfying model could not be provided [8].

This paper presents an accurate estimation of the power consumption of an algorithm directly at the C-level. A simple power model for a complex processor is proposed, including all the important phenomena like pipeline stalls, cache misses, parallelism possibilities… This model expresses the software power dissipation from parameters, determined through a functional analysis of the processor power consumption. It has already been validated with an estimation method at the assembly-level. In this case, the parameters are computed from the code by profiling [9]. To estimate at the C-level, some parameters are predicted assuming different ways of compiling the code. For classical digital signal processing algorithms, C-level estimates are obtained with an average error of 4% against measurements, which is an accuracy similar to the other methods at the assembly-level. We also provide the user with a *consumption map*, representing the consumption behavior of the C algorithm.

The estimation methodology is sketched in Section 2 with the general framework, and the case study of the functional analysis on the TMS320C6201, resulting in the power model of the processor. Section 3 presents the assumptions to predict the parameters required as inputs of the power model of the processor. The application results for digital signal processing algorithms are discussed in Section 4. The conclusion summarizes the conditions and limits of this algorithmic power estimation method and indicates the future works.

2 Functional-Level Power Analysis and Model Definition

2.a The estimation framework

The complete estimation methodology, represented in Figure 1, is composed of two steps: the *Model definition* and the *Estimation process*.

The *Model definition* is done once and before any estimation to begin. It is first based on a Functional Level Power Analysis (FLPA) of the processor, that allows discerning which parameter has a significant impact on the global power consumption. This step conducts to define the power model of the processor: this is a set of consumption rules describing the evolution of the processor core power related to algorithmic and configuration parameters. These consumption rules are computed from a reduced set of measurements performed on the processor for various elementary programs.

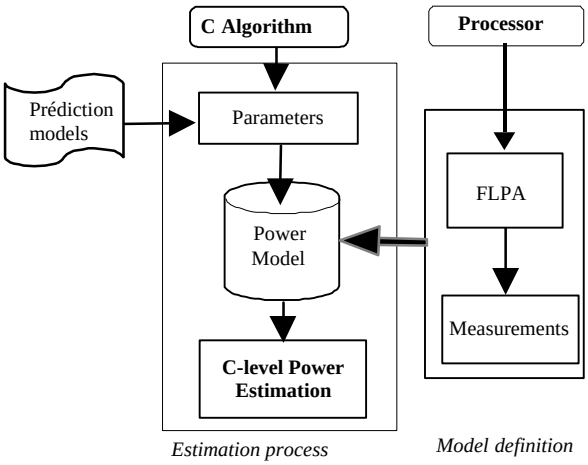

Fig. 1. The Estimation Method Flow

The *Estimation Process* is done every time the power consumption of an algorithm has to be evaluated. This step allows to determine the parameter values to apply in input of the power model of the processor. Some algorithmic parameters are estimated through simple assumptions about compilation: the prediction models.

2.b The model definition for the TMS320C6201

As a case study, the power model of the C6x from Texas Instruments has been developed. This processor has a complex architecture with a deep pipeline (up to 11 stages), VLIW instructions set, the possibility of up to 8 parallel instructions, an internal program memory and an External Memory Interface (EMIF), dedicated to load data and program from the external memory. Four memory modes are available: mapped, bypass, cache and freeze [10]. On this processor, there is no significant difference in power consumption between an addition or a multiplication, or a read or a write instruction in the internal memory. Moreover, the data correlation have no more effect than 2% on the global energy consumption. It seems that the architecture complexity of the C6x hides many power variations.

The first part in the *Model Definition* is the FLPA: it consists in a functional analysis of the processor architecture to determine which parameter is relevant from a consumption viewpoint. The complete analysis has already been presented in previous works [9]. The result of this step for the C6x is summarized in Figure 2, with two class of inputs: the algorithmic parameters and the configuration parameters. The configuration parameters, known with the application, are the clock frequency F and the memory mode MM. The algorithmic parameters represent activity rates between functional blocks: the

parallelism rate α, the processing rate β, the program cache miss rate γ, and the Pipeline Stall Rate PSR.

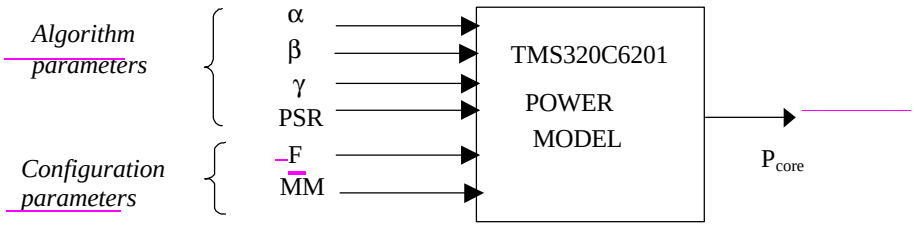

Fig. 2. Power Model for the C6x

The second step in the *Model Definition* is to establish the consumption rules, expressing the core power consumption P_{CORE} from the input parameters. Measurements have been performed for different values of these parameters by elementary assembly programs. Final consumption rules have been obtained by curve fitting. The present processor model does not include external memory yet; to add it will be part of future works.

Here is given by Equation 1 the consumption rule for the mapped mode:

$$P_{CORE} = V_{DD} * ([a\beta(1\text{-}PSR) + b_m] F + \alpha(1\text{-}PSR) [a_m F + c_m] + d_m) \quad (1)$$

where V_{DD} is the supply voltage and $a=0.64$, $a_m=5.21$, $b_m=4.19$, $c_m=42.401$, and $d_m=7.6$. Details on the other cases can be found in [9]. The expressions obtained are more complex than those derived from a linear regression analysis; that explains why, in [8], the model conducts to very important errors.

3. Estimation Process

To compute the power consumption from the power model, the algorithmic parameters must be determined. Both α and β can be estimated directly from the C code by a prediction model that anticipates the compilation. In some particular cases, γ and/or PSR can be defined, like in the mapped memory mode where γ = 0. If not, these parameters will be considered as variables.

In the C6x, 8 instructions, forming a *fetch packet* (FP), are fetched at the same time. In this fetch packet, operations are gathered in *execution packets* (EP) depending on the available resources and the parallelism capabilities [10]. The parallelism rate α and the processing rate β are obtained from Equation 2:

$$a = \frac{NFP}{NEP} \leq 1 \; ; b = \frac{1}{8}\frac{NPU}{NEP} \leq 1 \qquad (2)$$

NFP and *NEP* stands for the average number of FP and EP. *NPU* is the average number of processing units (every instruction except the NOP). To estimate α and β, it is then necessary to predict *NFP*, *NPU* and *NEP*. Four prediction models were developed according to the processor architecture and requiring a slight knowledge of the compiler:

- SEQ model: all the operations are executed sequentially.
- MAX model: all the architecture possibilities are fully exploited.
- MIN model: the load/store instructions can never be executed in parallel.
- DATA model: the load/store instructions are executed in parallel only if they involve different data.

```
For (i = 0; i < 512; i++)
Y = X[i] * (H[i] + H[i+1] + H[i-1]) + Y[i] ;
```

Fig. 3. Program example for the prediction models

As illustration, a trivial example is presented in Figure 3. The operations out of the loop body are neglected. In the loop nest, 4 loads (LD), and 4 other operations (OP) are needed. Then, we predict 8 instructions, gathered in one single FP, so *NFP* = 1. Because no NOP operation is involved, *NPU* = 8 and $\alpha = \beta$. 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 1.

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

Table 1. Prediction models for the example

For realistic cases, the prediction has to be done for each part of the program (loop, subroutine…) to obtain local values. The global parameter values, for the complete C source, are computed by averaging all the local values. Such an approach also permits to spot 'hot points' in the program.

4 Applications

4.1 Estimation validation

The power estimation method is applied on classical digital signal processing algorithms: a FIR filter, a LMS filter, a Discrete Wavelet Transform with two image sizes: 64*64 (DWT1) or 512*512 (DWT2) and an Enhanced Full Rate vocoder for GSM (EFR

v.). The results for these applications are presented in Table 2 for different memory modes (MM) : mapped (M), bypass (B) and cache (C) and different data placement (DP): in internal or external memory (Int/Ext). The nominal clock frequency F is 200MHz. The estimates at the C-level for the different predictions models are presented and compared against measurements. The estimates at the assembly level (Asm) are also provided to confirm the previous validation of the power model of the processor.

Applications			Measurements			Power estimation (W)						
Algo	MM	DP	T_{EXE}	P(W)	Energy	Asm	%	Seq	Max	Min	Data	%
FIR	M	Int	6.885µs	4.5	30.98µJ	4.6	2.3	2.745	4.725	3.015	4.725	5
FFT	M	Int	1.389ms	2.65	3.68mJ	2.585	2.5	2.36	2.97	2.57	2.58	-2.6
LMS	B	Int	1.847s	4.97	9.18J	5.145	3.5	5.02	5.12	5.07	5.12	3
LMS	C	Int	165.75ms	5.665	939mJ	5.56	-1.8	2.55	6	4.76	6	5.9
DWT1	M	Int	2.32ms	3.755	8.71mJ	3.83	1.9	2.82	4.24	3.27	3.53	-6
DWT1	M	Ext	9.19ms	2.55	23.46mJ	2.548	-0.2	2.295	2.63	2.4	2.46	-3.5
DWT2	M	Ext	577.77ms	2.55	1.473J	2.53	-1	2.27	2.61	2.37	2.45	-3.9
EFR v.	M	Int	39µs	5.078	198µJ	4.935	-2.8	2.54	5.636	3.86	5.13	1
error							2%	25%	7%	13%		3.9%

Table 2. Comparison between measurements and power estimation

The aim is to provide the designer with accurate estimates about all the possible consumption variations, including the particular point representing the real case. Then we have set $\gamma = 0$ and the global power consumption is computed with the PSR obtained after compilation. Validations of the power model at the assembly level for various values of the cache miss rate can be found in [9].

The results of the SEQ model proves that it is not possible to provide an estimation without any knowledge about the targeted architecture. Except in one particular case, the MAX and the MIN models always overestimates and underestimates respectively the power consumption of the application. For the LMS in bypass mode, all the models overestimate the power consumption with close results; in this marginal memory mode, pipeline stalls are so dominant that all the instructions become sequential. The DATA model is shown as a very accurate estimation with a maximum error of 6% and an average error of 4%. Although this fine grain model implies to also consider the data placement, it remains very easy to determine.

4.2 Algorithm Power Consumption Exploration

If the *cache miss rate* γ and/or the *pipeline stall rate* (PSR) are unknown, it is possible to give to the programmer a *'consumption map'*. This map represents the power consumption variations of the algorithm. Moreover, in many applications, the designer can evaluate the realistic domain of variation for PSR and γ. It is thus possible to locate, on the *consumption map*, the more probable power consumption limits. In particular, the

major part of the embedded applications have a program size (after compilation) that can easily be contained in the internal memory of the C6x (64 kbytes).

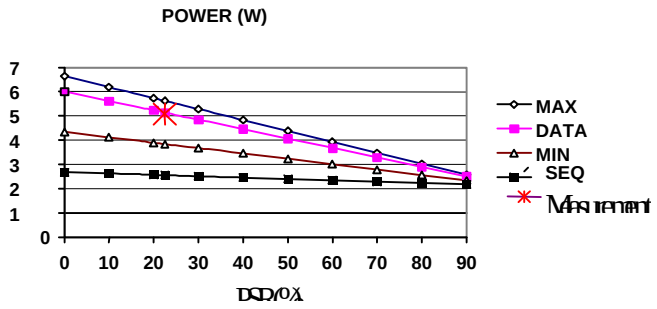

Fig. 4. Power Consumption Exploration for the DWT1 in mapped mode.

Let us reconsider the application of the EFR vocoder, in the mapped mode ($\gamma = 0$). In Figure 4, its *consumption map* is represented together with the measurement. Once again, the DATA prediction model is close to the reality.

The power consumption exploration tells the programmer if the algorithm respects the application consumption constraints (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 constraints then the C code is suitable. On the contrary, the programmer can focus on the more dissipating parts of the algorithm (selected with local values of α and β parameters), being aware on pipeline stalls and cache misses. At last, several versions of the same algorithm could be efficiently compared through their *consumption maps*.

5 Conclusion

In this paper, an accurate power consumption estimation of a C-algorithm is proposed for DSP applications. The accuracy of this estimation directly depends on the considered information about the processor: (i) it is not possible to determine precisely the power consumption without any knowledge about the targeted processor. (ii) an estimation with a coarse grain model taking into account only the architecture possibilities provides the maximum and minimum bounds of the power consumption for the algorithm. (iii) the fine grain model including both elementary information on the architecture and data placement offers a very accurate estimation with a maximum error of 6% against measurements.

Current works include first the development of an on-line power consumption estimation tool. FLPA will also be applied on other processors in order to provide other

estimation examples at both C-level and assembly-level. Finally, the development of a power model for the external memory will be an important part of future works.

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. 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.
7. 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.
8. C. H. Gebotys, R. J. Gebotys "An Empirical Comparison of Algorithmic, Instruction, and Architectural Power Prediction Models for High Performance Embedded DSP Processors," in *Proc. ACM Int. Symp. on Low Power Electronics Design,* Aug. 1998, pp. 121-123.
9. 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.
10. TMS320C6x User's Guide, Texas Instruments Inc., 1999.