SoftExplorer: estimation, characterization and optimization of the power and energy consumption at the algorithmic level

Eric Senn, Johann Laurent, Nathalie Julien, Eric Martin

▶ **To cite this version:**

Eric Senn, Johann Laurent, Nathalie Julien, Eric Martin. SoftExplorer: estimation, characterization and optimization of the power and energy consumption at the algorithmic level. 2004, pp.10. hal-00013977

HAL Id: hal-00013977
https://hal.science/hal-00013977v1

Submitted on 16 Nov 2005

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

SoftExplorer: estimation, characterization and optimization of the power and energy consumption at the algorithmic level

Eric Senn, Johann Laurent, Nathalie Julien, and Eric Martin

L.E.S.T.E.R., University of South-Brittany, BP92116
56321 Lorient cedex, France
eric.senn@univ-ubs.fr

Abstract. We present SoftExplorer, a tool to estimate and analyze the power and energy consumption of an algorithm from the C program. The consumption of every loop is analyzed, and the influence of the data mapping is characterized. Several models of processor are available, from the simple RISC ARM7 to the very complex VLIW DSP TI-C67. Cache misses, pipeline stalls, and internal / external memory accesses are taken into account. We show how to analyze and optimize the power and energy consumption, and how to choose a processor and its operating frequency, for a MPEG-1 decoder. We also explain how to find the best data mapping for a DSP application.

1 Introduction

Traditional methods to estimate a processor's power consumption work mainly at the cycle-level or at the instruction-level. The tools Wattch and SimplePower [1, 2] are based on cycle-level simulations. These simulations necessitate a low-level model of the processor's architecture; they are very time consuming for large programs. The Instruction-Level Power Analysis (ILPA) [3] relies on current measurements for each instruction and couple of successive instructions of the processor. The time to obtain a model is prohibitive for very complex architectures [4]. These methods perform power estimation from the assembly program with an accuracy from 4% for simple cases to 10% when both parallelism and pipeline stalls are effectively considered. Recent studies have introduced a functional approach [5, 6], but few works are considering VLIW processors and the possibility for pipeline stalls [7]. As far as we know, only one unsuccessful attempt of algorithmic estimation has already been made [8].

A lot of optimizations can be conducted at the algorithmic level with a very strong impact on the system's power consumption [9]. To evaluate the impact of high-level transformations of the algorithm, we propose to estimate the application's consumption at the early stages in the design flow. We demonstrate that an accurate estimation of an algorithm's power consumption can be conducted directly from the C program without execution. Our estimation method relies on a *power model* of the targeted processor, elaborated during the *model*

definition step. This model definition is based on the *Functional Level Power Analysis* of the processor's architecture [10]. During this analysis, functional blocks are identified, and the consumption of each block is characterized by physical measurements. Once elaborated the model for the processor, a model of the algorithm is needed. This model is obtained during the *estimation process*, which consists in extracting the values of a few parameters from the code; these values are eventually injected in the power model to compute the power consumption. The estimation process is very fast since it relies on a static profiling of the code (3s for a FIR1024 or a FFT1024, 8s for the MPEG-1 decoder, 10s for the DWT512x512 [11]). Several targets can be evaluated as long as several power models are available in the library.

To perform estimation from the assembly code, only the two former models are needed. The model for the processor represents the way the processor's consumption varies with its activity. The model for the algorithm links the algorithm with the activity it induces in the processor. At the algorithmic level (that we also call "C-level"), a model for the compiler is also necessary: the *prediction model*. Indeed, depending on the compiler behavior, or merely on the options settled by the programmer during compilation, the assembly code will differ, and so will the processor's activity.

2 SoftExplorer

SoftExplorer can perform power and energy estimation both at the assembly and at the C level. Basically, SoftExplorer automatically performs the estimation process - it extracts parameters from the code - and computes the power consumption and execution time for the targeted power model. Table 1 presents the parameters for the power models included in SoftExplorer. Four power models have been developed so far, for the Texas Instruments' C62, C55, C67, and the ARM7.

Table 1. Power models' parameters

Parameters	C62	C67	C55	ARM7
α	X	X		
β	X	X	X	
γ	X	X	X	
τ	X	X		
ε	X	X	X	
PSR	X	X		
W	X	X	X	
F	X	X	X	X
MM	X	X	X	X
DM	X	X		
PM			X	

The parallelism rate α assesses the activity between the fetch stages and the internal program memory controller of the processor. The processing rate β represents the utilization rate of the processing units (ALU, MPY). γ is the program cache miss rate. τ is the external data memory access rate. The parameter ε stands for the activity rate between the data memory controller and the Direct Memory Access (DMA). PSR is the pipeline stall rate. These parameters actually depend on the algorithm itself, they are called *algorithmic parameters*.

The processor's consumption also depends on *architectural parameters*, which are rather related to the configuration of the processor, or more generally of the application. They are the clock frequency (F), the memory mode for instructions (MM), the data mapping (DM), the DMA data width (W), and the power management parameter PM, which indicates the units in sleep mode for low-power processors.

In SoftExplorer, once the target selected, the memory mode (mapped, cache, freeze, bypass) and the processor's frequency must be provided. At the C-level, the prediction model is also required for the C62, C55, or C67. We have defined four prediction models. The DATA (TIME_optimal) model corresponds to the case where the compiler is settled to optimize the code for performance. The MIN (SIZE_optimal) model corresponds to an optimization for size. The MAX (FULL_parallel) model gives a maximum bound for the power consumption, and represents a situation where all the processing resources would be used restlessly. The SEQ (SEQuential) model stands for a situation where operations would be executed one after the other in the processor. That gives an absolute minimum bound for the power consumption (the MIN model gives a more realistic lower bound however). In the case of the two first prediction models, the data mapping is taken into account. Indeed, we have demonstrated that the number of external accesses, and the number of memory conflicts, are directly linked to the processor's processing and parallelism rates (β and α), and to the pipeline stall rate (PSR), which have a great impact on the final power consumption and execution time [12]. α, β, and PSR are some of the power model's parameters which are automatically computed by the tool. For the ARM7, only the memory mode and operating frequency, are needed.

Then, the estimation can begin once the prediction type is chosen; indeed, SoftExplorer can perform three types of prediction:

Coarse prediction: There is no need for any further information from the user. The C-code is parsed and the power consumption is computed with different values for the instruction cache miss rate γ and the PSR from 0 to 100 %. The result is a curve, or an area, displayed on the "C curves results" or the "C area results" page. Indeed, if the memory mode is *bypass* or *mapped*, then $\gamma = 0$, only the PSR varies, and a curve is computed. If the memory mode is *cache* or *freeze*, both the PSR and γ vary, and an area is drawn. We call these curves and areas *consumption maps*. The interest of the coarse prediction is to allow an estimation of an algorithm's power consumption very early in the design process. Indeed, even if the data mapping is not settled (the data mapping is not considered in this mode), the programmer

can have an idea of the power consumption, choose the processor, compare different algorithms, and be guided in the optimizations to be conducted on the code.

Fine prediction: In this mode, the data mapping file is used to determine the data access conflicts (internal and external) for each loop in the code. If $\gamma = 0$, which is the case for a large number of DSP applications, these conflicts permit to determine, for every loop, the precise number of pipeline stalls (i.e. the PSR) and the execution time. This local knowledge of every parameter in the power model of the target permits to compute, for each loop, the power consumption, the execution time, and the energy consumption, as well as the parallelism and processing rates. This fine analysis of the algorithm's consumption is very helpful to the designer. An accurate optimization of each portion of the code can be conducted that way.

Exact prediction: The execution time, the cache miss rate and the pipeline stall rate are no more predicted by SoftExplorer, but are instead provided by the user. They can be obtained with the help of the targeted processor's development tools (e.g. TI's CodeComposer for the C6x). C-level profilers can also be used for an estimation of the execution time.

The precision of SoftExplorer was evaluated by comparing power estimations with measures for several representative DSP applications. The precision varies slightly from a processor to the other. For the C62, the maximal / average errors are 4% / 2.5% at the assembly level, and 8% / 4.2% at the C-level [10]. Energy estimation is less precise (average error is 11%, max error is 21%). This is due to the difficulty in estimating the execution time at the C-level, for very dynamic or control oriented algorithms. Indeed, in the case of dynamic loops, the number of iterations is not known by advance. Whenever a dynamic loop is encountered, the user must provide SoftExplorer with a value for the number of times the loop is iterated. A dynamic profiling is necessary; specific tools are used for this purpose. The inaccuracy of these tools is directly reported on SoftExplorer's results. Another approach for the user could be to provide the maximum limit for the number of iterations to get a maximum for the execution time and energy. Secondly, SoftExplorer's parser discards every control structure in the code. This is not a problem for data intensive applications - indeed power estimations are always very good - but this increases the error for the execution time when the algorithm includes a lot of control. Again, a dynamic profiling could be used, but with the same drawbacks than before.

3 Applications

3.1 Optimizing the algorithm

We show, on an example, the results that are obtained with the three prediction types of SoftExplorer, and how optimizations can be conducted on the algorithm. The application is the MPEG1 decoder from *MediaBench*. Estimation is performed first in coarse mode for the C62. If the memory mode is mapped, a

curve that shows the evolution of the power consumption P with the PSR is plotted. The maximal value for P is 4400mW when PSR=0%, and its minimal value is 2200mW for PSR=100%. Figure 1 shows the evolution of the power consumption with both the PSR and γ (memory mode is cache). This time, the max/min values for P are 5592/2353mW. It can be observed that for the same PSR, the power consumption is always lower in mapped mode. Indeed, the max/min power consumption in freeze and bypass mode are respectively 5690/2442 and 5882/5073mW.

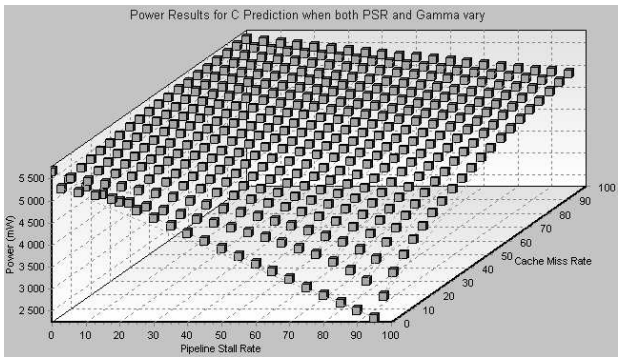

Fig. 1. C area: the power is a function of PSR and γ

A fine prediction takes into account the data mapping. In mapped or bypass mode, the global power and energy consumptions are presented on the "results" page, with the execution time, and the global values of parameters α and β. The power repartition in every functional part of the processor is also given (Figure 2). The DMA unit consumes no power for it is not used in this application. It is remarkable that a great part (47.4%) of the power consumption is due to the clock. The "loops" page displays the power consumption, execution time, and α and β values, for each loop (Figure 3). These results are also presented in the form of a chronogram on the "graphics" page (Figure 4) with the power consumption per functional part.

As stated before, the designer gets a direct analysis of the power and energy consumption of every part of its algorithm, together with their intrinsic parallelism and processing rates. In this example, there are three small loops (loops 1, 4 & 7) that consume a lot of power, and two long loops (loops 2 & 5) that consume a lot of energy. In the small loops, the parallelism rate is 1 and the processing rate only 0.125. In order to lower the algorithm's maximal power consumption, the programmer can investigate these loops and decrease the parallelism rate by rewriting the code. In the two long loops, the parallelism and processing rates are 0.5 and 0.375. Again, the designer can modify the code to increase this rates and to decrease the loops' execution time, especially if some

efforts are made to reduce the number of memory conflicts. The algorithm's energy consumption can be reduced this way.

In cache or freeze mode, the variations of the power consumption, execution time, and energy consumption with γ are given on the "C curve" page.

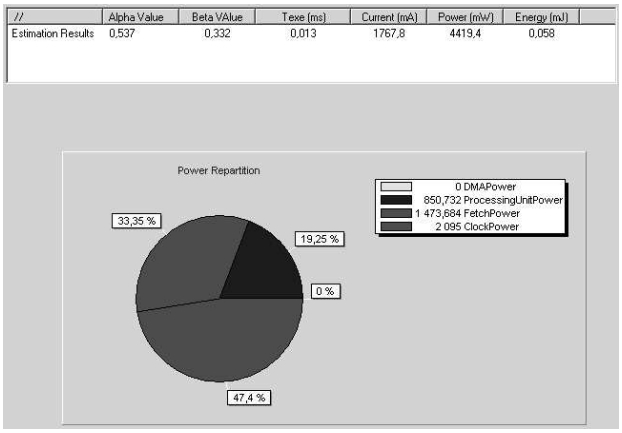

Fig. 2. The "Results" page also gives the power repartition

//	Loop Number	PLoop (mW)	TexeLoop (CPU Cycles)	NbItLoop	NbEPLoop	AlphaLoop	BetaLoop
Loop	1	5145,0	64	64	1	1,000	0,125
Loop	2	4429,5	1024	512	2	0,500	0,375
Loop	3	3949,5	128	64	2	0,500	0,188
Loop	4	5145,0	64	64	1	1,000	0,125
Loop	5	4429,5	1024	512	2	0,500	0,375
Loop	6	3949,5	128	64	2	0,500	0,188
Loop	7	5145,0	64	64	1	1,000	0,125
Loop	8	4109,5	128	64	2	0,500	0,250

Fig. 3. The "Loops" page presents the results for each loop

An exact prediction is possible when the exact values for γ, PSR and Texe are known. In our example, the TI's development tool gives, after compilation, $\gamma = 0$, PSR=0.2 and Texe=40μs. The power and energy estimations are displayed on the "results" page. The "loops" and "graphics" pages display local values for α, β, and the power consumption, assuming that pipeline stalls and cache misses are equally scattered in the algorithm.

3.2 Influence of the data mapping

We demonstrate on a simple example the influence of the data mapping on the power consumption and execution time. The algorithm that we use as a test bench manipulates at the same time 3 images of size 100x100 (a,b,c) and 2 vectors of size 10 (e,f), in 3 successively and differently imbricated loop nests.

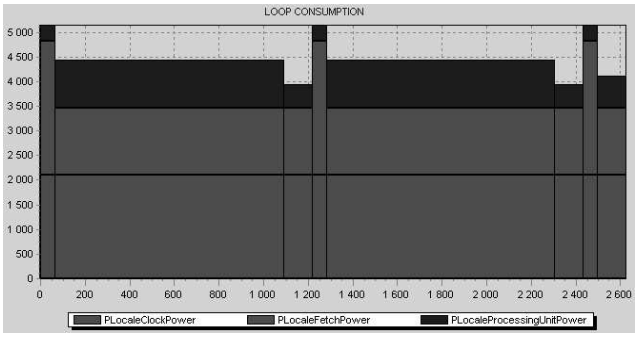

Fig. 4. A graphical display of the consumption per loop

Since the images and vectors are manipulated at the same time, their placement in memory has a strong influence on the number of access conflicts, and thus on the number of pipeline stalls. We show here that it is very quick and easy to try different placements, and to reach an optimal data mapping with the help of SoftExplorer. The results are presented on table 2.

Table 2. Power and energy estimation with different mappings

Mapping	α	β	Texe (ms)	Current (mA)	Power (mW)	Energy (mJ)	Conflicts	Tconflicts
1	0.02	0.01	6.9	877	2194	15.1	27601	441616
2	0.2	0.1	0.69	1165	2912	2.01	27601	27601
3	0.2	0.1	0.69	1165	2912	2.01	0	0
4	0.2	0.1	0.69	1165	2912	2.01	0	0

In the first mapping, all the data structures are placed in the external memory. As a result, there are as much external accesses than accesses to the memory, and for every access the pipeline is stalled (during 16 cycles for the C6x). The relation between parameters α and β and the power consumption is obvious. When the pipeline is stalled, the number of instructions that the processor executes in parallel (α) and the processing rate (γ) decrease. As a result, the power consumption of the processor is reduced, but the execution time is lengthened and the energy consumption increases.

In the second mapping, all the structures are placed in the same bank in the internal memory (B0). There will be as much conflicts as before, but this time, the conflicts are internal. The C6x's pipeline is stalled during one cycle in case of an internal conflict. As a result, the time necessary to resolve all the conflicts, expressed in number of cycles, equals the number of conflicts itself.

The interest of the last mapping (4) is to give a minimum bound in term of number of conflicts since every structure in the algorithm is in a different

bank. This solution will also give the higher power consumption and the smaller execution time. Indeed, since the pipeline is never stalled, the processor is used at its maximal capacity. The third mapping, achievable with a C6x, with (a,c,f) in bank B0 and (b,e) in bank B1, is as good as mapping 4 since it does not yield any conflict.

3.3 Finding the right processor / frequency couple

Even if it is easy to obtain the power consumption and the execution time of an algorithm with SoftExplorer, to actually find the right processor and its operating frequency is not straightforward. Indeed, the global energy consumed by the application depends on the energy consumed when the algorithm is executed, but also on the energy consumed when the processor is idle (equation 1).

$$E_{global} = P_{exe} \times T_{exe} + P_{idle} \times (T_{constraint} - T_{exe}) \quad (1)$$

The timing constraint $T_{constraint}$ is the maximum bound for the execution time. Over this limit, the application's data rate is not respected. Basically, if the frequency is high, the execution time is small and the active power (Pexe) increases. The idle time also increases with the frequency. On the other hand, as long as the execution time is lower than the timing constraint, it seems possible to slow down the processor to decrease Pexe. So, is it better to operate with a high or a low frequency ? In fact, it actually depends on the application.

We pursue a little farther the analysis with our preceding example, the MPEG-1 decoder. This algorithm treats 4 macro-blocs of a QCIF image in one iteration. A QCIF image (88x72) contains 396 macro-blocs. Suppose a data-rate of 10 images/s, the timing constraint is $T_{constraint} = 1.01ms$. Then we use SoftExplorer to compute, at different frequencies, the execution time, power, and energy consumed by one iteration of the algorithm. Finally, we calculate with equation 1 the global energy consumed by the application at these frequencies. The results are presented on Figure 5 for the C55, C62 and the C67.

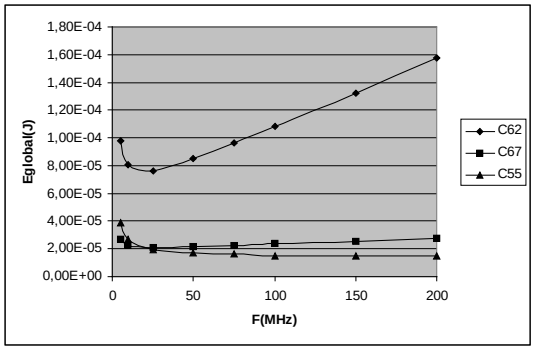

Fig. 5. Energy vs Frequency for the MPEG-1 decoder

The two curves for the C62 and C67 present a minimum that gives the optimal operating frequency for this application: about 20MHz for the C62 and 40MHz for the C67. This minimum is 0.076mJ for the C62 and 0.021mJ for the C67, hence, the best couple processor/frequency among those two would be the C67 at 40MHz.

Whenever P_{idle} does not vary with the frequency, the resulting curve does not present a minimum anymore [11]; this is the case for the C55. For this processor, the frequency that gives the lower global energy consumption is the higher possible (200MHz). However, since the energy consumption is almost the same at 100MHz, this last frequency will be preferred for it implies a lower power consumption: cooling devices will be lighter in this case. In fact, the global energy consumption for the C55 at 100MHz is 0.015mJ. As a result, the couple C55 / 100MHz is definitely better than the two preceding.

We did not consider the wake-up time in equation 1. In fact, we measured that this wake-up time is very small before the execution time of the algorithm, and that the energy consumption involved is negligible. Moreover, to avoid waking-up at each iteration, it is preferable to process a whole image with no interruption, and then to idle the processor until the next image. This decreases again the energy contribution of the waking-up. Of course, this can only be done if the application can bare a latency of one image. In a situation where the wake-up time T_{wu} could not be neglected, it is still possible to evaluate the global energy. Assuming that T_{wu} is counted in processor's cycles, and that the wake-up power P_{wu} is proportional to the frequency F, the wake-up energy E_{wu} is obviously constant. As a result, the curves that give the global energy in function of the frequency, are shifted of the value of E_{wu}. The method to find the best processor and frequency remains the same.

4 Conclusion

SoftExplorer is a tool that performs power and energy estimation both at the assembly level (from the assembly code) and at the algorithmic level (from the C-code). It is based on the Functional Level Power Analysis, which proves very fast and accurate for the estimation, as well as for the modelling of processors. Four power models are included. 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 power consumption. SoftExplorer appears very helpful for analyzing the power and energy consumption of every part of the algorithm. Every loop is characterized, its intrinsic parallelism and processing rates are exhibited, memory access conflicts are determined. Hot spots are detected; the designer is guided in the writing of the code, and in the mapping of data. We show how to use SoftExplorer to choose a processor and its operating frequency in order to minimize the application's overall energy consumption.

Future works include the development of new power models for the C64, the ARM9, the PowerPC and the OMAP. A generic memory model will be added to

include the external memory consumption in our power estimation. Estimation of the execution time will be improved for control oriented applications, to increase SoftExplorer's accuracy for energy estimation at the C-level. Power and energy estimation will be investigated at the system level.

References

1. D. Brooks, V. Tiwari, and M. Martonosi, "Wattch: A framework for architectural-level power analysis and optimizations," in *Proc. International Symposium on Computer Architecture ISCA'00*, 2000, pp. 83–94.
2. W. Ye, N. Vijaykrishnan, M. Kandemir, and M. Irwin, "The design and use of SimplePower: A cycle accurate energy estimation tool," in *Proc. Design Automation Conf. DAC'00*, 2000, pp. 340–345.
3. V. Tiwari, S. Malik, and A. Wolfe, "Power analysis of embedded software: a first step towards software power minimization," *IEEE Trans. VLSI Systems*, vol. 2, pp. 437–445, 1994.
4. B. Klass, D. Thomas, H. Schmit, and D. Nagle, "Modeling inter-instruction energy effects in a digital signal processor," in *Proc. of the Power Driven Microarchitecture Workshop in ISCA'98*, 1998.
5. S. Steinke, M. Knauer, L. Wehmeyer, and 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'01*, 2001, pp. 3.2.1–3.2.10.
6. G. Qu, N. Kawabe, K. Usami, and M. Potkonjak, "Function-level power estimation methodology for microprocessors," in *Proc. Design Automation Conf. DAC'00*, 2000, pp. 810–813.
7. L. Benini, D. Bruni, M. Chinosi, C. Silvano, V. Zaccaria, and 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'01*, 2001, pp. 2.3.1–2.3.10.
8. C. H. Gebotys and 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 ISLPED'98*, 1998, pp. 121–123.
9. M. Valluri and L. John, "Is compiling for performance == compiling for power?" in *Proc. of the 5th Annual Workshop on Interaction between Compilers and Computer Architectures INTERACT-5*, 2001, mexico.
10. N. Julien, J. Laurent, E. Senn, and E. Martin, "Power consumption modeling of the TI C6201 and characterization of its architectural complexity," *IEEE Micro, Special Issue on Power- and Complexity-Aware Design*, Sept./Oct. 2003.
11. J. Laurent, N. Julien, E. Senn, and E. Martin, "Functional level power analysis: An efficient approach for modeling the power consumption of complex processors," in *Proc. Design Automation and Test in Europe DATE'04*, Paris, France, Mar. 2004.
12. E. Senn, N. Julien, J. Laurent, and E. Martin, "Power estimation of a c algorithm on a VLIW processor," in *Proceedings of WCED'02, Workshop on Complexity-Effective Design (in conjunction with the 29th annual International Symposium on Computer Architecture, ISCA'02)*, Anchorage, Alaska, USA, May 2002.