

System-Level Power Evaluation Metrics

W.Fornaciari (†), P.Gubian (‡), D.Sciuto (†), C.Silvano(‡)

(†) Politecnico di Milano, Dipartimento di Elettronica e Informazione, 32 P.zza L.da Vinci, I-20133 Milano, Italy. E-mail: {fornacia, sciuto}@elet.polimi.it, Fax:+39-2-2399 3411.

(‡) Università di Brescia, Dipartimento di Elettronica per l'Automazione, 38 Via Branze, I-25123 Brescia - Italy. E-mail: {gubian, silvano}@bsing.ing.unibs.it, Fax: +39-30-380014.

Abstract

High-level power estimation is a key issue for IC designers and system engineers. The goal is to widely explore the architectural design space and to compare alternative solutions, while maintaining an acceptable accuracy and a competitive design time. In this paper, an approach is proposed for evaluating the system-level power consumption of embedded systems implemented by using VLSI circuits. Accurate and efficient early power evaluation metrics have been defined to guide the system-level partitioning phase of a more general HW/SW co-design approach for control dominated embedded systems. The hardware and software contributions to the power consumption at the system-level have been considered as well as the contribution of the HW/SW communication.

1. Introduction

Embedded systems have become widely used in the most recent past for a lot of computing and telecommunication applications. As opposed to general purpose computer systems, embedded systems are computing and control systems conceived for dedicated applications, to respond to specific requirements [1]. In general, the embedded system architecture is composed of both an hardware and a software part: one or more dedicated devices such as ASICs or FPGAs implement the hardware part, while a set of routines running on a dedicated processor or ASIP implement the software part. Using the most advanced CMOS technologies, the whole embedded system can be implemented into a single ASIC, including the embedded core processor, the on-chip memory, the I/O interface and the custom hardware part.

Innovative co-design techniques have recently emerged as a new CAD discipline to support embedded systems design. Co-design techniques aim at meeting the system-level requirements by adopting a concurrent design and validation methodology, thus exploiting the synergism of the HW and SW parts [1]. An ideal co-design flow includes several design activities, such as the system-level modeling and co-specification, analysis and validation of the co-specification, system-level partitioning, architectural design space exploration, co-synthesis and co-simulation. The availability of a co-design methodology, covering all design phases, is a primary issue during embedded systems design [1] [2].

More specifically, the cost/performance figure of merit at the system-level is greatly impacted by the effects of the partitioning phase, aiming at the assignment of the co-specification operations to the HW or SW parts. To guide the partitioning algorithm, estimation metrics should be provided to compare the results of alternative partitionings and to evaluate their conformance to system-level design goals. These goals typically include

system-throughput, operating frequency, area, power dissipation, global cost, etc. [1]. Since an optimal partitioning solution satisfying all those constraints is difficult to obtain in a reasonable time, a sub-optimal solution is usually accepted, derived from a partial exploration of the architectural design space.

The relevance of the power issues among the other design constraints is steadily increasing [3] [4]. Power evaluation and optimization techniques should be included in the co-design flow at several abstraction levels to cope with system-level power requirements in an acceptable design time. However, the availability of a high-level power estimation tool is mandatory to obtain early estimation results, while maintaining an acceptable accuracy level and a competitive design time. In general, the relative accuracy during early power estimation is much more important than the absolute accuracy, the main goal being the comparison of different design alternatives and the exploration of the architectural design space [5].

The aim of this paper is to define system-level power evaluation metrics to guide the partitioning phase of a co-design methodology suitable for control dominated embedded applications. The proposed approach is going to be integrated in the *TOSCA* (Tools for System Co-design Automation) co-design environment [6] [7], shown in Figure 1.

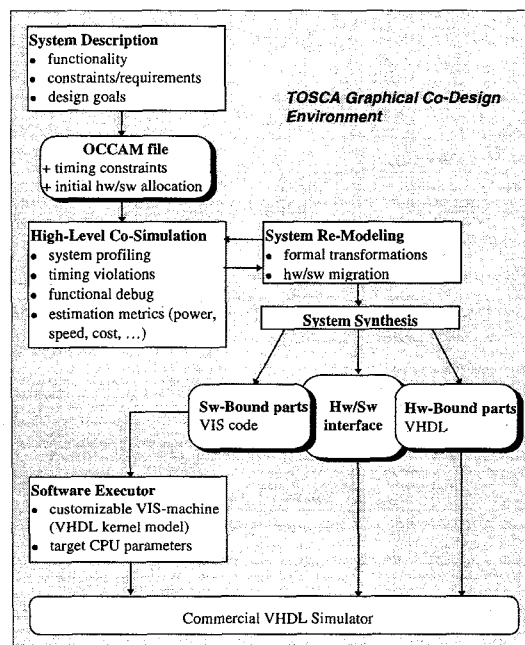


Figure 1. The TOSCA Co-design Environment

Metrics suitable for the early power evaluation of both the HW and SW contributions have been defined. Globally, the proposed approach can be considered as one of the first attempts to cover power estimation issues from a comprehensive HW/SW perspective. As a matter of fact, both the HW and SW contributions to the power budget at the system-level have been considered. The HW power model has been extensively described in [8] [9], while the SW power model as well as the contribution of the HW/SW communication part to the global power are described in this paper.

The rest of the paper is organized as follows. In Section 2, the problem of the system-level HW/SW partitioning is stated and a set of metrics to capture the relevant system constraints is defined. Section 3 shows the proposed power estimation approach for the SW-bound part, while Section 4 is devoted to the future works and to some concluding remarks.

2. Metric Driven System-Level Partitioning

System-level partitioning consists of the assignment of operations to the processor (SW part) or to one or more dedicated circuits (HW part), while minimizing the HW/SW communication overheads and meeting the system-level requirements. The partitioning is thus one of the most relevant tasks of existing co-design flows, since shifting the project goals fulfillment towards early design phases can provide a significant improvement on the global turn-around time. Several partitioning algorithms have been presented in literature [6] [2] [10] [11] [12] [13] [14] [15] [16]. Those schemes are usually based on an iterative procedure starting from an initial solution, that can span from a fully HW solution to a fully SW solution. During the successive iterations, the operations can be moved from the HW to the SW side to reduce the system costs, while the operations can be moved from SW to HW to improve performances. Furthermore, the partitioning algorithms can differ in the granularity levels on which they operate.

The task of system-level partitioning in the *TOSCA* co-design environment [6] aims at considering alternative solutions in terms of cost/performance to trade-off the speed of the analysis with the accuracy and the implementation costs. Common characteristic of the partitioning algorithms is the definition of a cost function to be optimized, based on a set of *metrics* to capture the relevant system constraints, such as speed, area, cost, power, etc.. The cost function in *TOSCA* is a weighted sum:

$$f = \sum_{m \in N} w_m m = W \times M$$

where m is the generic metric, w_m is the weight factor expressing the relevance of the corresponding metric m according to the design requirements and N is the number of metrics. All data derived from the metrics evaluation are included in the matrix M , while the matrix W expresses the metric relevance. All those metrics should consider the contribution of both the HW and SW sections.

The current version of *TOSCA* evaluates a set of metrics, based on the analysis of an object oriented representation of the system-level description, which is based on the *OCCAM2* formalism [6]. The preliminary, and iterated, phase is the metric-based analysis of the system-level description to evaluate the quality of a partitioning solution in terms of fulfilment of several design optimization criteria to define the initial HW vs SW allocation.

Metrics to evaluate area and performance are described in [7] [17], while metrics to measure and to compare the power consumption of several design alternatives are the subject of this paper. In general, it is quite difficult to devise a single metric suitable for accurate and efficient power assessment for all the embedded applications. We can classify the embedded systems as timing or area constrained systems if the speed or area is the most important design constraint. In turn, several computational modes characterize the timing-constrained systems, depending on the system throughput, T , defined as the number of operations performed in a given time [18]. Since the power budget strictly depends on the computational mode for which the system is dedicated, a specific power metric can be defined depending on the operating mode [18].

For *fixed throughput systems*, that is those systems featuring a fixed number of operations

per second, such as DSP applications, a suitable metric is represented by the power/throughput ratio or equivalently the energy per operation:

$$M_F = \text{Power} / T = \text{Energy} / \text{Operation}.$$

Since the throughput is fixed, if a partitioning solution leads to a reduction of M_F with respect to an initial partitioning, the corresponding power dissipation is reduced.

For *maximum throughput systems*, characterized by an operating mode continuously providing useful computation at the maximum speed, such as microprocessor-based systems, the most appropriate metric should account for both the low power and high performance needs. A suitable metric is thus the *Energy to Throughput Ratio (ETR)* defined as in [18]:

$$ETR = E_{MAX} / T_{MAX}$$

where E_{MAX} is the maximum energy per operation or equivalently the power per throughput and T_{MAX} is the maximum throughput. Hence, the *ETR* metric can also be expressed as:

$$ETR = \text{Power} / T^2$$

The *ETR* metric expresses the concept of optimization of both throughput and power. A partitioning corresponding to a lower value of *ETR* represents a solution with lower energy per operation for equal throughput as well as a solution with greater throughput for the same amount of energy per operation.

For systems operating in the *burst throughput mode*, characterized by a fraction of time performing useful computations, while idling during the rest of the time, such as systems conceived for user interactive mode, the power metric should account for power reduction, during both idle and computing time, and throughput optimization when computing. For those systems applying power shut-down techniques during idle cycles, an efficient metric is just *ETR*, since the power dissipation has been completely eliminated when idling. For those systems not supporting power saving modes or for those systems where just the main blocks of the processor are shut-down, a most effective metric is [18]:

$$M_{BURST} = (E_{MAX} + E_{IDLE}) / T_{MAX}$$

where E_{MAX}/E_{IDLE} is the total energy dissipated when computing/idling per total operations and T_{MAX} is the maximum throughput.

For those area-constrained systems for which the target area is fixed, a valid metric is represented by the power by area product (or equivalently the power/area ratio):

$$M_A = \text{Power} * \text{Area}.$$

Since the area is fixed, a reduction in the value of M_A corresponds to a minimization in the power consumption. In general, for those area-constrained systems aiming at both power and area reduction, a good metric is given by the product of the energy per operation times the area:

$$EAP = E_{MAX} * A$$

where E_{MAX} is the maximum energy per operation or equivalently the power per throughput and A is the area. Hence, the *EAP* metric can also be expressed as:

$$EAP = (\text{Power} * A) / T$$

The *EAP* metric expresses the concept of optimization of both area and power dissipation. A partitioning with a lower value of *EAP* represents a solution with lower energy per operation for the same area as well as a solution with lower area for the same energy per operation.

The methodology proposed in [7] can be used to estimate the area and throughput terms in the above metrics. Models to estimate the power terms contained in the above metrics are proposed for both the SW and HW parts. The power assessment for the SW side is described in the next section. The power model for the HW-bound part, described in [8] [9], is based on the VHDL description at the behavioral/RT levels, produced by the *TOSCA* toolset, and the

probabilistic estimation of the switching activity. The HW model is basically an analytical model, that attempts to relate the average power dissipation of the VHDL descriptions to the physical capacitance and the switching activity of the design nets. The estimation approach is hierarchical: at the highest hierarchical level, ad-hoc analytical power models for each part of the target system architecture are proposed; these models are in turn based on a macro-module library, at the lowest hierarchical levels. Furthermore, to avoid a large amount of input patterns to be simulated, our approach is weakly pattern-dependent [19]. User-supplied input probabilities are required, reflecting the typical input behavior, that are derived from the system-level specification.

A significant contribution of the overall power budget is represented by the power due to the HW/SW communication part. This power is related to the activity on system-level busses existing among the embedded processor and the other HW parts, mainly the address, data and control busses.

The increasing importance of this contribution to the global power is essentially due to two factors. First, the width of data and address busses has recently increased, to meet the growing performance requirements. Second, the intrinsic capacitances to be driven by the system-level busses are usually very large. That is especially true for the off-chip busses, rather than for on-chip busses. In fact, the capacitances driven by the I/O pads is up to three orders of magnitude with respect to the internal nodes capacitances. Thus, considerable power is dissipated at the I/O interface of processor when data or addresses are transmitted over the system-level busses.

Power estimation at the processor interface is based on the estimation of the switching activity of the system-level busses. The relevant parameters to estimate the bus switching activity are the bus width, the required bandwidth, the address and data encoding, the data representation (2's complement vs sign and magnitude). As for the SW part, the analysis is performed at an intermediate assembly-like level, called *VIS* (Virtual Instruction Set).

3. Power Model for the Software Part

Aim of this work is the definition and validation of a software-level power estimation model at the instruction-level. The main feature of this approach is to be suitable for different core processors. The availability of such a methodology to evaluate the power of the embedded software, along with the methodology defined in [8] [9] for the hardware part, allows to early verify if the embedded system design meets the global power budget.

The power assessment of the embedded software must be performed by following a bottom-up approach. The *TOSCA* system-level specification is based on a customization of the *OCCAM2* formalism, detailed in [6]. Each software-bound part of the specification is considered in terms of basic blocks and it is compiled into a quasi assembly intermediate representation, called *Virtual Instruction Set (VIS)* [20] [21]. Suitable *VIS* templates have been defined for each basic *OCCAM2* construct, so that the characteristics of the original specification can be modeled with a good accuracy.

The choice to work at the *VIS*-level is motivated by the goal to make our analysis processor-independent, being the *VIS* easily retargetable towards different processors [20]. Thus, the power analysis will be performed at the *VIS*-level, by computing the average power consumption of each *VIS* instruction during the execution of a given program. The methodology should include the mapping between the *VIS* and the instruction set of the selected processors. Once the power analysis is completed for all the basic *VIS*-level instructions, the approach is extended to the upper-level software modules, by weighting the power of each basic block according to the execution frequencies.

An in-house VIS simulator has been developed [21]. The simulation engine is based on the definition of a suitable VHDL model, working at the VIS instruction level, to be executed together with the HW-part of the system (described in VHDL) in order to perform an overall hardware-software co-simulation. The software simulator is able to *profile* the VIS code execution to find out performances in terms of timing, resources accesses and the average power consumption, by taking into account the min/max power of each VIS instruction which is influenced by the target processor and the related template-based mapping strategy.

As depicted in the following report, at the end of the simulation the tool produces a set of information including the average power consumption. As recalled above, such values can be employed to characterize the basic blocks of the system description as well as to analyze the power requirement of a given program.

```

-----STATUS OF PROCESSOR REGISTERS-----
Result of simulation : Simulation was successful
Simulation time from reset event to end simulation : 505205 ns

Minimum number of target CPU clock cycles per VIS instruction: 12306
Maximum number of target CPU clock cycles per VIS instruction: 16408
Extern number of target CPU clock cycles per VIS I/O instruction: 1500

Total Minimum number of target CPU clock cycles: 13806
Total Maximum number of target CPU clock cycles: 17908

Minimum mean power for all instructions of target CPU: 1.200000e+01
Maximum mean power for all instructions of target CPU: 1.300000e+01
Mean power for all instructions of target CPU: 1.250000e+01

Total number of instructions executed: 4.302000e+03

```

Figure 2. A short report produced by the VIS profiler considering both timing and power performances.

The primary task is the definition of a power model for the selected core processors. To compute the average current drawn during the execution of each instruction in the instruction set, it is possible to operate by following two different approaches.

In the first approach, we perform some measurements on the energy cost of each instruction by using an empirical methodology to determine the current drawn by the processor during the repeated execution of loops composed of the same instruction or short instruction sequences [22] [23].

In the second approach, we get detailed power information by the processor supplier, in terms of the energy dissipated by each type of instruction in the instruction set. These latter power information can be derived by simulating the execution of instruction sequences on a lower level (circuit or layout) or gate level model of the processor, to obtain an estimate of the current drawn. By using either the first or the second approach, a power table can be derived for each selected processor, reporting the energy base cost for each instruction in the instruction set and for all the possible addressing modes associated with each instruction type.

Given the information about the power characterization of the selected instruction sets, the next step consists of the definition of a methodology to map the energy cost of each instruction with respect to the VIS. In this way, a full power characterization of each VIS instruction for different addressing modes can be achieved. A technology table will report the power data stating the correspondence between each VIS instruction mapped to each target processor.

Globally, to evaluate the power to execute a given embedded software program, an analysis at the instruction-level should be carried out. The high-level estimates should consider the

relevant parameters influencing the power consumption at the software-level, mainly the execution time, the cache hit-rate, the number of accesses to the main memory and to the I/O, etc.

As a matter of fact, the average power dissipated by a processor while running a program is:

$$P_{SW} = I_{AVE} * V_{DD}$$

where I_{AVE} is the average current and V_{DD} is the supply voltage. The associated energy is:

$$E_{SW} = P_{SW} * t_{SW}$$

where t_{SW} is the execution time of the software program, that can be expressed as:

$$t_{SW} = N_{CLK} * \tau_{CLK}$$

being N_{CLK} the number of clock cycles to execute the program and τ_{CLK} the clock period.

The embedded software execution results in power consumption due to the power corresponding to both the core processor and the memory sub-system. Thus, to compute the average current I_{AVE} , we consider two terms: the first term is the processor related power, while the second term is the memory related power. For the processor related power, we can use the energy base costs derived for each instruction and for different addressing modes of the processor. The power behaviour of the memory sub-system at the system-level should take into account different aspects, such as the memory hierarchy architecture, the physical memory organization and other advanced features, such as burst mode memory operations.

Additional contributions to the global power derive from inter-instruction effects, not considered by computing the base cost of each instruction. These effects are mainly due to the previous state of the processor, the limited number of resources leading to pipeline and write buffer stalls and the rate of cache misses [22] [23]. The condition of the processor in the previous clock cycle may cause an energy overhead due to the different switching activities on data and address busses and the different processors internal behavior. In general, the previous state of the circuit is different during program execution, since there is a switching from an instruction to another, with respect to the execution of the program used for the measurements of the base energy, where the same instruction was executed many times. The circuit state overhead has been measured in [22] by considering all the possible instruction pairs and it results approximately less than 5% of the base energy per instruction. Therefore, this overhead has been considered in [22] as an average constant value to be added to the base cost, without a significant loss of precision. However, as in our case, the effects of resources constraints and cache misses can be neglected in most of embedded software applications, based on simple microcontrollers (e.g. M68000, Intel 8051, ...) where such advanced features are not available.

4. Concluding Remarks

In this paper, a system-level approach to achieve early power estimation for control dominated embedded systems have been proposed. Accurate and efficient power-based metrics have been defined to drive the system-level partitioning to be integrated into a more general HW/SW co-design environment (*TOSCA*). More specifically, during the design space exploration, the possibility of evaluating alternative HW/SW architectures without moving down to the final implementation plays a primary role. Thus, an effective set of metrics for system-level partitioning tailored to consider the power consumption of both the HW and SW parts has been defined.

Future evolution of the proposed approach is in the direction to extend the analysis towards more complex systems, such as microprocessor and DSP based systems.

5. References

- [1] G. De Micheli, "Hardware/Software Co-Design: Application Domains and Technologies," *Hardware/Software Co-design, NATO ASI Series, Series E: Applied Sciences*, Vol.310, pp.1-28, Kluwer Academic Publisher, 1996.
- [2] D. Gajsky, F. Vahid, S. Narayan, J. Gong, *Specification And Design of Embedded Systems*, Prentice Hall, New Jersey, 1994.
- [3] S. Devadas, S. Malik, "A Survey of Optimization Techniques Targeting Low Power VLSI Circuits," *DAC-32: ACM/IEEE Design Automation Conference*, San Francisco, CA, June 1995.
- [4] A. P. Chandrakasan, R. W. Brodersen, "Minimizing Power Consumption in Digital CMOS Circuits," *Proceedings of the IEEE*, 83 (4), pp. 498-523, 1995.
- [5] P. Landman, "High-Level Power Estimation," *ISLPED-96: ACM/IEEE International Symposium on Low Power Electronics and Design*, pp. 29-35, Monterey, CA, August 1996.
- [6] A. Balboni, W. Fornaciari, D. Sciuto, "TOSCA: a Pragmatic Approach to Co-Design Automation of Control Dominated Systems," *Hardware/Software Co-design, NATO ASI Series, Series E: Applied Sciences*, Vol.310, pp.265-294, Kluwer Academic Publisher, 1996.
- [7] A. Balboni, W. Fornaciari, D. Sciuto, "Partitioning of Hw-Sw Embedded Systems: a Metrics-Based Approach," to appear on *Integrated Computer-Aided Engineering Journal*, John Wiley, 1997.
- [8] W. Fornaciari, P. Gubian, D. Sciuto, C. Silvano, "A Conceptual Analysis Framework for Low Power Design of Embedded Systems," *ISIS-96: IEEE International Conference on Innovative System In Silicon*, Austin, TX, October 1996.
- [9] W. Fornaciari, P. Gubian, D. Sciuto, C. Silvano, "A VHDL-based Approach for Power Estimation of Embedded Systems," *Journal of System Architecture, Special Issue on VHDL*, 1997.
- [10] R. Camposano, J. Wilberg, "Embedded System Design," *Design Automation for Embedded Systems*, Kluwer Academic Publishers, 1(1-2), pp.5-50, 1996.
- [11] M. Chiodo, D. Engels, P. Giusto, H. Hsieh, A. Jurecska, L. Lavagno, K. Suzuki, A. Sangiovanni Vincentelli, "A Case Study in Computer-Aided Co-design of Embedded Controllers," *International Journal of Design Automation for Embedded Systems*, Kluwer Academic Publishers, 1(1-2), pp. 51-67, 1996.
- [12] M. Theibinger, P. Stravers, H. Veit, "Castle: an Interactive Environment for HW-SW Co-design", *CODES-94: Third IEEE International Workshop on Hardware-Software Co-design*, pp. 203-209, 1994
- [13] F. Vahid, J. Gong, D.D. Gajski, "A Binary-Constraint Search Algorithm for Minimizing Hardware/Software Partitioning," *EDTC-94: IEEE European Design and Test Conference*, pp.214-219, Paris, France, 1994.
- [14] F. Vahid, D.D. Gajsky, "Clustering for Improved System-Level Partitioning," *ISSS-95: IEEE International Symposium on System Synthesis*, pp.28-33, 1995.
- [15] J. Hou, W. Wolf, "Process Partitioning for Distributed Embedded Systems," *CODES/CASHE-96: 4th IEEE International Workshop on HW/SW Co-design*, pp. 70-76, Pittsburgh, Pennsylvania, 1996.
- [16] P. Eles, Z. Peng, K. Kuchcinski, A. Doboli, "Hardware/Software Partitioning with Iterative Improvement Heuristics," *ISSS-96: 9th IEEE International Symposium on System Synthesis*, pp.71-82, 1996.
- [17] S. Narayan, D.D. Gajski, "Area and Performance Estimation from System-Level Specifications," *Technical Report ICS-92-16*, Dept. of Information and Computer Science, University of California at Irvine, December 20, 1992.
- [18] T.D. Burd, R.W. Brodersen, "Energy Efficient CMOS Microprocessor Design," *28th Hawaii International Conference on System Sciences*, Hawaii, 1995.
- [19] F.N. Najm, "A Survey of Power Estimation Techniques in VLSI Circuits," *IEEE Transactions on VLSI Systems*, Vol.2, No. 4, pp. 446-455, December 1994.
- [20] A. Balboni, W. Fornaciari, D. Sciuto, M. Vincenzi, "The Use of a Virtual Instruction Set for the Software Synthesis of Hw/Sw Embedded Systems," *ISSS-96: 9th IEEE International Symposium on System Synthesis*, La Jolla, CA, November 1996.
- [21] A. Balboni, W. Fornaciari, D. Sciuto, "Co-synthesis and Co-simulation of Control Dominated Embedded Systems," *International Journal of Design Automation for Embedded Systems*, Vol.1, No.3, Kluwer Academic Publisher, Norwell, MA, July 1996.
- [22] V. Tiwari, S. Malik, A. Wolfe, "Power Analysis of Embedded Software: a First Step towards Software Power Minimization," *IEEE Transactions on VLSI Systems*, Vol. 2, No. 4, pp. 437-445, December 1994.
- [23] V. Tiwari, S. Malik, A. Wolfe, M.T.-C. Lee, "Instruction Level Power Analysis and Optimization of Software," *Journal of VLSI Signal Processing*, 1-18, Kluwer Academic Publishers, Boston, 1996.