MPSoC Power Estimation Framework at Transaction Level Modeling

Rabie Ben Atitallah, Smail Niar and Jean-Luc Dekeyser
DaRT Project, INRIA-FUTURS, France
Email: {benatita, niar, dekeyser}@lifl.fr

***Abstract*— Early power estimation is increasingly important in MultiProcessor System-On-Chip (MPSoC) architectures for a reliable Design Space Exploration (DSE). In this paper, we present an MPSoC power modeling framework at the Timed Programmer View (PVT) level that offers a good performance/power tradeoff to be found early in the design flow. Using a hybrid power modeling methodology, we developed several power models derived from both physical measurements and analytical expressions. Plugging these power models into the PVT architectural simulator makes it easy to estimate the application's performance and power consumption with high simulation speedup. The effectiveness of our method is illustrated through a DSE for a parallelized version of H.263 encoder application.**

I. Introduction

Designing next generation MultiProcessor Systems-on-Chip (MPSoC) dedicated to high-performance embedded applications will be increasingly complex. An efficient and fast design space exploration (DSE) of such systems needs a set of tools capable of estimating performance and power at higher abstraction level in the design flow. Nowadays, power consumption has emerged as a primary design metric when developing MPSoC circuit taking into account silicon integration, IP multiplicity and clock frequency rise.

Power estimation at the RTL level cannot adequately support the complex design of future MPSoC since RTL tools require increased simulation time to explore the huge architectural solution space. Among the existing tools which operate at the RTL level we can mention, SPICE [1] and PETROL [2]. These tools are fairly accurate, but require significant amount of simulation time. In an attempt to reduce simulation time, significant research efforts have been expended to evaluate MPSoC power consumption at the Cycle Accurate Bit Accurate (CABA) level [3]. Usually, to move from the RTL to the CABA level, hardware implementation details are hidden from the processing part of the system, while preserving system behavior at the clock cycle level. Though using the CABA level has allowed accurate power estimation, MPSoC DSE at this level is not yet sufficiently rapid compared to RTL [4].

In this paper, we focus on higher abstraction levels especially at the TLM (Transaction Level Modeling) for evaluating power consumption on MPSoC design. In TLM, a set of abstraction levels simplifying the description of inter-module communication is defined. Consequently simulation time is reduced by augmenting communication operation granularity. Signal handshaking, as used in CABA level, is replaced in TLM by transactions using communication channels [5]. Consequently, modeling MPSoC architectures becomes easier and faster than at the CABA level. TLM [5] does not represent one level of abstraction but rather, it refers to a taxonomy of several abstraction levels. Each level differs from the others in the degree of functional or temporal details it expresses. To maintain a good accuracy/simulation speedup tradeoff, our framework is designed at the Timed Programmer View (PVT) level [5]. At this level, the hardware architecture is specified for both processing and communication parts and some arbitration of the communication infrastructure is applied. For performance estimation, this level is annotated with timing specifications.

Recently, we start thinking about the efficiency of power estimation at the transactional levels. In [6] and [7], authors present a characterization methodology for generating power models within TLM adopted to peripheral components. The pertinent activities are identified at several levels and granularities. The characterization phase of the activities is performed at the gate level and helps to deduce power of coarse-grain activities at higher level. However, this methodology cannot be applied to different kinds of components such as processors or interconnect networks. Thus in our approach, characterization from low level will be used as well as analytical modeling methodology. This hybrid approach yields acceptable levels of precision and speed. Another key point in this paper is that we propose a methodology for generating power models with accuracy level control. Thus, we have developed several power models for components that make up the MPSoC architecture: processors, caches, interconnects and memories. Within this framework, a reliable DSE for MPSoC is conceived allowing a good performance/power tradeoff to be found with high simulation speedup. This paper is organized as follows. An overview of our MPSoC platform described at the PVT level is given in Section 2. Section 3 describes the major power modeling methodology at the PVT level. Section 4 includes the developed power consumption models for the MPSoC components. Section 5 presents experimental results for a parallel version of the H.263 encoder application.

II. The MPSoC Platform Overview

In our previous work, we have developed an MPSoC platform described at the PVT level [8]. This platform includes various kinds of component models that have been designed: processors, caches, interconnection network, RAM, DMA controller and a DCT hardware accelerator. Fig 1 shows the general architecture of the MPSoC used in the experiments. At the PVT level, details

related to the computation resources are omitted. Details related to communication are also hidden. To do so, transactions are performed through channels which implement one or several interfaces. Each interface has a set of read and write methods. To load or store data, read() or write() function calls are instantiated by masters and sent through the port to the channel interface. At the level of slaves, the transaction will be recovered to execute the corresponding methods and to send the response. At this level, a timing model is defined and plugged in the architectural simulator to approximate the execution time.

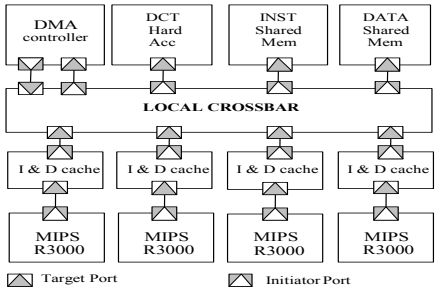

Fig. 1. Exemple of MPSoC structure

III. Power Modeling at the PVT Level

Compared to lower levels, estimations done at the PVT level are sufficiently rapid to allow the entire multiprocessor system to be analyzed in a reasonable time. The total power consumption of a given system is obtained by adding the consumption of each system component together. For this study, we developed power models for the main components in the MPSoC architecture. We integrated these models into the PVT simulator, taking the architectural and technological parameters into account. Our power estimation strategy is based on identifying each component's pertinent activities. For this, a counter is allotted to each kind of activity, and each counter is incremented during the simulation. Thus, the number of activity occurrences is obtained for each component. A power consumption cost is also evaluated for each activity. At the end of the simulation, the values on the activity counters are transmitted to the power consumption models to calculate the total power dissipation. The overall methodology for component power modeling at the PVT level can be described as follows (see Fig 2):

- Identify the pertinent activities that consume power at two granularity levels; fine grain and coarse grain which are adopted respectively for CABA and PVT power models.
- Characterize power consumption of fine grain activities.
  - Using low level simulation with CAD tools for preexisting components like the processor or the memory. Estimation is done with real inputs data.
  - With an accurate analytical model for non preexisting component.
- Deduce coarse grain activities power characterization from its definition and the fine grain activities power.
- Developing the corresponding CABA and PVT power models of the selected component.
- Plugging the developed power models into the architectural simulator and performing simulation.
- Evaluate the power estimation error between CABA and PVT levels. If the error is over the fixed threshold, the coarse grain granularity definition step is then restarted.

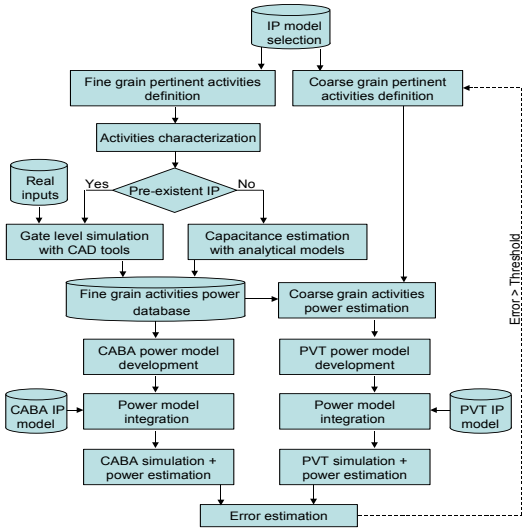

Fig. 2. Power modeling methodology

In our approach, a 90nm technology process is adopted. To predict physical parameters, we used the Berkeley Predictive Technology Model (BPTM) [9].

IV. Power Models development for the PVT platform

A. SRAM Memory Power Model

For a SRAM, three main activities consume power: *Read*, *Write* and *Idle*. These activities correspond respectively to the read, write access modes and the waiting state. In fact for the SRAM model, the defined granularity of pertinent activities at the PVT level is the same as proposed at the CABA level. This is because it is possible to recover these activities at the transactional level which is not usually possible for other components. Nevertheless, this recovery is not obviously found. At the CABA level, activity counters are injected in the FSM which control the component. At the PVT level, the FSM is not implemented. The number of *Read* and *Write* occurrences are deduced from the read() and write() methods described into the component. The waiting time, which corresponds to the idle state power consumption, is calculated as the difference between the total application execution time and the total read and write access time of the memory. To estimate an activity's power cost, different SRAM sizes are simulated at the physical level using the ELDO analog simulator [10]. The objective is to produce a parameterized power model for estimating the cost of SRAM activities, according to the number of words (M) and number of bits per word (N). In our platform, this power model is used to deduce the total consumption of the data memory, instruction memory and FIFO buffer components.

B. Cache Memory Power Model

In the cache description at the CABA level, different states have been defined to guarantee that the component behaves correctly. For instance, the *CACHE_IDLE* state corresponds to the data loading from the cache. The cache power consumption depends on the state of the FSM that controls the component. Each state corresponds to one or several fine grain read or write operations from the tag array, the data array and the FIFO buffer (Fig 3). Therefore, estimating the power consumption of an FSM state is equivalent to evaluating the power cost of a write or read access to the SRAM memory (tag array or data array) and evaluating the power cost of a write access to the FIFO buffer. At the PVT level, details related to the cache FSM are hidden,

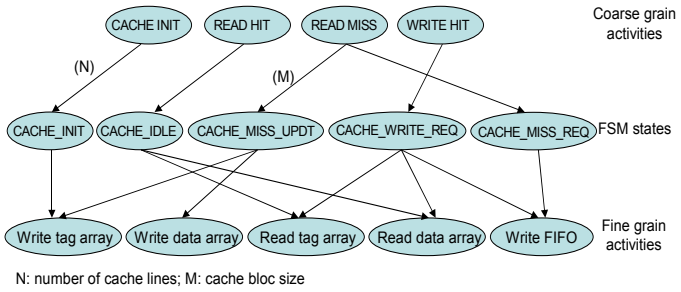

Fig. 3. cache activities definition

thus the defined fine grain cannot be recovered. We have defined a set of coarse grain activities which are *Read* hit or miss, *Write* access modes and the *Idle* state. Each coarse grain activity refers to one or several cache FSM states and thus to several fine grain activities such as read or write operations from the tag and the data array (Fig 3). The power cost of coarse grain activities is then deduced from the preceding SRAM experiments. A set of counters, corresponding to the coarse grain activities, are declared in the component description. At the end of the simulation, the values of these counters are read, and then they are multiplied by the activity power costs to find the overall consumption of the cache component.

C. Processor Power Model

In our study, we used the MIPS R3000 processor. This scalar processor has a 5-stage pipeline. At the CABA level, to estimate the processor power contribution, the most accurate solution is to evaluate the internal unit activities such as the fetch and decoder stages. Nevertheless, the implementation details of these units are omitted at the PVT level. In this level, the processor is described using an Instruction Set Simulator (ISS) on which instructions are executed sequentially. The processor's power consumption in the active state depends on the instruction to be executed. The set of all the instructions' power will constitute the processor power model, and the cost of each instruction can be determined from low level measurements or from micro-architectural power simulator. In this work, the SimPanalyzer simulator [11] has been used to measure the power cost of the MIPS R3000 instruction set. Simulation results show that the maximum current variation between instructions is only 8%. Consequently, for a processor with a relatively simple architecture, a consumption model that considers only an average power value per instruction is sufficient. In our power consumption model for the Mips R3000, we considered two states: *Running* (execution of an instruction) and *Waiting* for data or instruction. The power consumptions for these two states are different.

D. Crossbar Power Model

The main activity of the crossbar is to transfer data between two ports, and its most significant consumption is at the wire level. The power dissipation of these wire connections depends on their length and the used process technology. Thus, to estimate the crossbar consumption accurately, the connection lengths between components must first be estimated, and these lengths depend on the final organization of the components on the chip. For our study, we supposed a particular component structure in order to obtain approximate wire lengths, using a layout editor to measure the size of each component. The power consumption during a data transfer from the initiator i to the target j depends on the number of transmitted words in the packet and the power transfer cost of a word from i to j. A word transfer from an initiator i to a target j (request word) or vice versa (response word) corresponds to several wire activation. At the CABA level, the Virtual Component Interface (VCI) protocol is adopted [12]. Data transfer along the VCI request interface (93 bits) or the VCI response interface (46 bits) needs several cycles and power estimation is analyzed cycle by cycle. At the PVT level, this transfer is considered undivided and a power cost is attributed to the entire request or response packet.

V. Simulation Results

Several experiments were conducted to measure the simulation speedup, the accuracy of the performance and power predictions. These metrics have been reported in comparison to the CABA level. The simulation speedup factor is defined as the ratio between the PVT simulation time and the CABA simulation time. All experiments were conducted using the H.263 coder [13] which is parallelized to be executed using an MPSoC with 4 up to 16 processors and instruction and data cache size varied from 1KB to 32KB (Kilo Bytes). Fig 4 demonstrates PVT level makes it possible to accelerate the simulation by a factor of up to 18. Second, adding processors increased this speedup factor due to the amplification of the communication between the processors and the shared memory modules. In the same way, we can deduce that reducing the data and instruction cache size improves the simulation speedup factor. This is because reduced cache sizes increase cache misses and consequently the traffic in the interconnection network. Despite its performance in terms of speedup, the PVT level suffers from a reasonable performance estimation error. As shown in Fig 4, when the number of processors increases or when the cache size decreases, communication becomes more significant and estimation error increases. This error can be as much as 8% for 16 processors and 1KB cache size; this is because the MPSoC behaves differently with PVT level

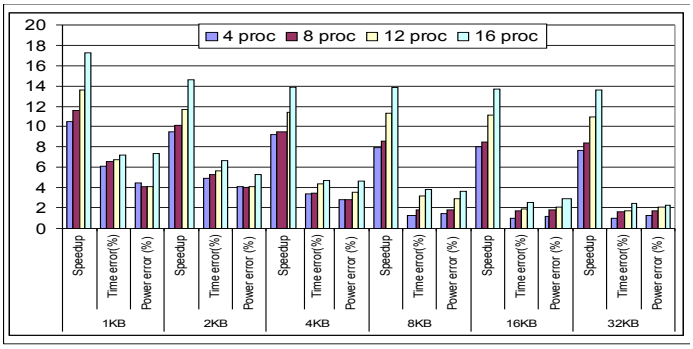

Fig. 4. Simulation speedup, Time error and Power error

in terms of contentions in the interconnection network. The previously developed power models were integrated into the PVT level architectural simulator in order to benefit from a fast architectural exploration environment for MPSoC design. Fig 4 illustrates that the accuracy of the power prediction is more than 93%. The error in power estimation is due to the inaccuracy in counting waiting cycles of the contentions in the interconnection network. During these waiting cycles, static power dissipation of inactive components should be taken into account. Thus the power estimation error increases by adding more processors or using less cache sizes which adds new conflicts in the crossbar.

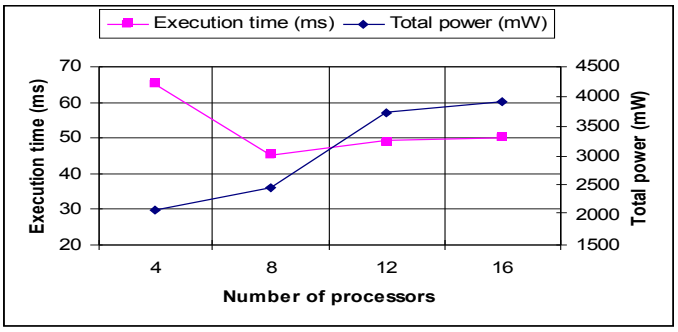

Fig. 5. Performance and power variation in terms of the number of processors

To evaluate the impact of the number of processors on the performance and the total power dissipation of the system, we executed the H.263 coder application using systems with 4 up to 16 processors. The size of the instruction and data caches was set to 8 KB. Fig 5 reports the execution time in ms and the total power in mW. Given these results, it seems that adding processors to the system decreases execution time, which improves system performance. This variation is not linear because the processors share resources, and sometimes they cannot reach the same target simultaneously, which necessitates waiting cycles. Over certain limit, increasing the number of processors tends to be ineffective, as it just adds new conflicts. In terms of power dissipation, although execution time is reduced by 30% while moving from 4 to 8 processors system, the total power dissipation increases slightly. By adding more than 8 processors, the system execution time tends to be stabilized, however this alter dramatically the total power consumption which is raised by 50% while moving from 8 to 12 processors system. Next, we used a 8-processor configuration to examine the impact of varying instruction and data cache size on the performance and power consumption of the whole system. The increase in the cache size significantly increases overall system performance in term of execution time and power consumption. In general, larger caches improve system performance; however, this depends on the size of the task or data to be handled. In our example, the move from 1KB to 2 KB, for instance, reduced execution time by 24% and power consumption by 37% (Fig 6). However, Using more than 8 KB cache size did not improve the execution time which increases the total power consumption.

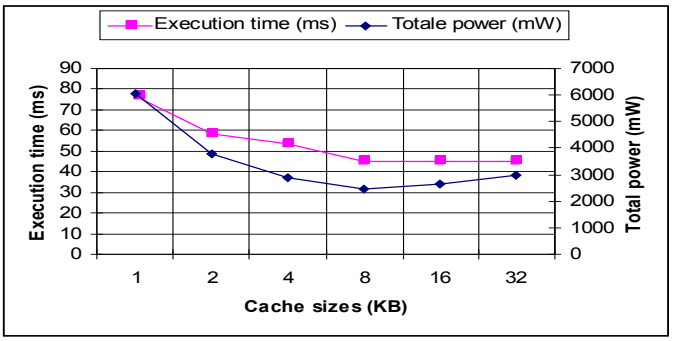

Fig. 6. Performance and power variation in terms of cache sizes

VI. Conclusion

A reliable DSE for MPSoC requires rapid and accurate tools for estimating performance and power consumption. In this study, we enhanced an MPSoC architectural simulator at PVT level with a power consumption estimator. We have designed a flexible MPSoC environment at the PVT level providing a high simulation speedup factor with a negligible performance and power estimation error margin. For future research, we plan to apply the same methodology to more complex architectures, including other types of processors and interconnection networks.

References

[1] "SPICE manual," University of Berkeley (USA), URL: http://bwrc.eecs.berkeley.edu/Classes/IcBook/SPICE/.
[2] R. Peset et al., "The petrol approach to high-level power estimation," in *ISLPED'98*, California, USA.
[3] SoCLib project, 2003, `http://soclib.lip6.fr/`.
[4] R. Ben Atitallah et al., "Estimating energy consumption for an MPSoC architectural exploration," in *ARCS'06*, Frankfurt, Germany.
[5] A. Donlin, "Transaction level: flows and use models," in *CODES+ISSS '04*, Stockholm, Sweden.
[6] I. Lee et al., "PowerViP: SoC power estimation framework at transaction level," in *ASP-DAC'06*, Yokohama, Japan.
[7] Nagu Dhanwada et al., "A power estimation methodology for systemc transaction level models." in *CODES+ISSS'05*, New Jersey, USA.
[8] R. Ben Atitallah et al., "An MPSoC performance estimation framework using transaction level modeling," in *IEEE RTCSA'07*, Daegu, Korea.
[9] Y. Cao et al., "New paradigm of predictive MOSFET and interconnect modeling for early circuit simulation," *IEEE CICC,*, May 2000.
[10] "Mentor home page," http://www.mentor.com.
[11] Sim-Panalyzer home page, `www.eecs.umich.edu/panalyzer/`.
[12] Virtual Component Interface Standard, `http://www.vsi.org/`.
[13] G. Cote et al., "H.263+: Video Coding at Low Bit Rates," *IEEE Trans. On Circuits And Systems For Video Technology*, November 1998.