System-Level Power Analysis Methodology Applied to the AMBA AHB Bus

M. Caldari[*], *M. Conti*[*], *M. Coppola*[**], *P. Crippa*[*], *S. Orcioni*[*], *L. Pieralisi*[*], *C. Turchetti*[*]

[*] University of Ancona, via Brecce Bianche, I-60131, Ancona, Italy

[**] STMicroelectronics, Grenoble, France

Abstract

The specification on power consumption of a digital system is extremely important due to the growing relevance of the market of portable devices and must be taken into account since the early phases of a complex System-on-Chip design.

In this paper some guidelines are provided for the integration of the information on power consumption in the executable model of parameterized cores, with particular attention to the AMBA AHB bus. This will give important information for the analysis and choice between different design architectures driven by functional, timing and power constraints of the System-on-Chip.

1. Introduction

The increasing demand of complex mobile systems, which has been observed during the last years in the worldwide market, led the designers to take into account a new objective in the design of complex digital systems: the minimization of power consumption.

The high diffusion of systems like laptop and palmtop computers, cellular telephones, wireless modems and portable videogames is one of the most important reasons that feed the need of a low-power design. These systems take the energy to work from a battery whose duration is a quality design parameter and so it can have a deep impact on the market success. Furthermore the cost and the weight of batteries directly contribute to the cost and weight of the final product. Therefore a company should consider low-power design as an investment to increase its market segment.

The need for a minimization of power consumption of a system is enforced also by some thermal considerations: a big portion of the energy requested by a device from the power supply is converted into heat. In this way heat dissipation systems and cooling techniques become necessary for the correct and safe operation of the device and for its reliability. It has been proven that an increase of 10 °C in the working temperature of an electronic device causes a 100% increase in the failure rate. If it is possible to decrease the power dissipation it is also possible to reduce costs for expensive cooling and complex packages.

Emerging policies that work for a low environmental impact of electronics systems also encourage low-power design. In the last years some companies activated interesting programs, like the Energy Star by the Environmental Protection Agency (EPA), to establish rules, in terms of energy and dissipated power, to determine if a product is energy-efficient.

The first necessary step to make towards low-power design is the dissipated power estimation of the system under development. This kind of analysis should be performed since the first phases of the design when some good ideas on power dissipation can drive the choice between different architectures, with the requested functional and timing specifications, on the base of the maximum achievable power optimization. This is also evidenced in the MEDEA+ Design Automation Roadmap that outlines the future requirements for the European industry [1].

This paper addresses the question of power analysis for a design modeled at system-level and some guidelines are given to introduce power consumption information in the executable model of parameterized cores. As an example this methodology was applied to the C++ description of the Advanced Microcontroller Bus Architecture (AMBA) Advanced High-performance Bus (AHB) by ARM.

2. Power analysis and SoCs

The high diffusion of complex portable systems is possible thanks to the steady growing integration capability reachable on a silicon chip. In this way the demand, and the possibility, to build embedded systems with increasing performances imply a proportional growth in the system complexity, area occupation, working clock frequencies and power consumption.

System-level design and intellectual property (IP) modeling is the key to fast SoC innovation with the

capability to quickly try out different design alternatives, to confirm the best possible architecture, HW/SW partition and performance parameters, including power consumption, early in the design process. To innovate quickly, system-level design provides a high level of abstraction, very fast simulation speed and allows a high degree of IP reuse. A possibility to implement an efficient system-level design is the use of object-oriented programming languages like C++, which is the base of SystemC for example. One goal of the EDA community is the integration of power analysis and optimization techniques into IP modeling methodologies. This is an important design reuse aspect that is getting increasing relevance in the IP qualification process, whose aim is to establish objective and standardized criteria to check the quality of an intellectual property not only in terms of its functionality.

There are many tools for power estimation and optimization that work very well at lower levels of abstraction and give accurate results from RTL to circuit-level, but at higher levels there is still a lot of work and research to do [2-3]. The process of system-level power estimation does not have to lead necessarily to a high degree of absolute accuracy, difficult to reach at this level of abstraction; the goal is to be able to obtain an early, "cheap" indication of the "hot-spots" of the design, the most critical circuit blocks under the energetic point of view. Optimization efforts will concentrate on these parts during the following development stages to obtain the best results in terms of power consumption.

3. IP characterization

System-level power analysis implies a first stage for IP characterization to obtain an executable model that, during a simulation, should provide, beside functional, timing and performance data, information about the complete system power consumption.

At system-level a core can be seen as a functional unit executing a sequence of instructions or processes without any information on their real successive hardware or software implementation. In this work the word "instruction" is used to indicate an action that, together with others, covers the entire set of core behaviors. It is important that all these instructions are functionally non-overlapping.

The purpose of the presented methodology is to be able to assign to every instruction a relative or absolute value, more or less accurate, of the energy requested for its execution as expressed in [4-5]. Given the functional specification of the core, it should be possible to identify different instruction sets. For example several granularity levels can be considered:

- a low granularity implies a lot of instructions: the time required for their characterization will be reasonably high but it will be for sure possible to obtain accurate results;
- a high granularity implies few instructions: the time required for their characterization will be reasonably low but in general it will be difficult to obtain accurate results.

For these reasons the instruction set choice can be done at different levels, also depending on the degree of knowledge of the final implementation of the system, trying to reach the best trade-off between:

- generality and flexibility of the methodology: it is important to develop an analysis approach that could be reused for different IP typologies, in order to avoid to write each time a new power model from scratch;
- results accuracy: it strongly depends on the way the model is written (SystemC [6], VHDL, RTL or gate-level description) and on the way the system will be implemented;
- characterization time and simulation speed: the addition of power analysis in the simulation of a digital block implies some human and machine time necessary to determine and use the parameters and the model that characterize the power dissipation of the system to be developed.

Once the instruction set has been identified, it is necessary to characterize each instruction in terms of dissipated power, or better energy requested for its execution For this purpose it is necessary to create some macromodels starting from the knowledge of a possible implementation or however trying to complete the model with the maximum number of low-level details. It could be necessary to run lower-level simulations and in this sense it is very important to provide a complete set of testbenches to be able to observe all the different activity states of the system.

The input parameters of the macromodels can belong to two different typologies:

- IP typical parameters like data and address bus width, the dimensions of memory buffers or the frequency of the different clock domains;
- characteristics of the input/output data involved in the instruction execution like the switching-activity, the probability of a signal or the Hamming distance between two successive data.

For the instructions characterization it is required an instrumentation phase for the system model, consisting in the addition of some code to allow:

- probing and storage of the chosen variables and data observed during the simulation;
- the possible processing of these data to obtain the required statistical or probabilistic quantities;
- the use of IP parameters and particular data, dynamically extracted from the simulation, to get power dissipation with the execution of the previously built power macromodels.

4. System-level power analysis

When every instruction has been characterized with its own energy model, it is necessary to develop a further model for the computation of the whole energy requested during a system-level simulation.

The idea is to create, in a first stage, the functional model of the core and to add in a second moment the code dedicated to power analysis; of course this code does not have to modify the system behavior and it should be simply bypassed when it is requested to verify only the functionality at the maximum simulation speed. Furthermore the code that performs power analysis does not have to be included in the successive synthesis process of the model, unless it is necessary to develop a dynamic power management for a run-time energy optimization of the system.

The added code will use some internal variables of the model to recognize the present state of the behavior during a functional simulation and to identify the correct mode for power analysis. The simulation of a complete SoC, that uses system-level IP models, can be several hundreds times faster than an RTL simulation, so in a small time it is possible to evaluate hundreds of different configurations and architectures in order to reach the desired trade-offs in terms of different parameters like speed, throughput and power consumption.

One way to create the executable specification of a SoC is to use SystemC library and simulation kernel; different approaches can be used to add information about power dissipation of a digital system using this particular tool. According to its underlying methodology, a digital system can be represented as a set of communicating modules and the basic computation element of a module is the process. This means that core power dissipation is strictly related to process execution.

It could be possible to characterize each process in terms of energy so that a process is considered as a single, atomic instruction. In this way a private model of power analysis is implemented. This kind of approach can produce very accurate results especially if the written code is easily synthesizable or the macromodel of a possible implementation is already known. The price to pay for accuracy is the low level of the intrumenting code that turns out to be little reusable, highly intrusive and with a deep impact on simulation speed.

If this cost is too high, it is possible to gain speed and generality of the method, by adding a particular process to those already present in the module; this process, that can be also a more complicated FSM, works as a system activity monitor with the purpose to generate the requested energy information when particular events occur. A local dissipation model is implemented, less intrusive, whose accuracy and CPU load depends on the complexity of the model represented in the added FSM.

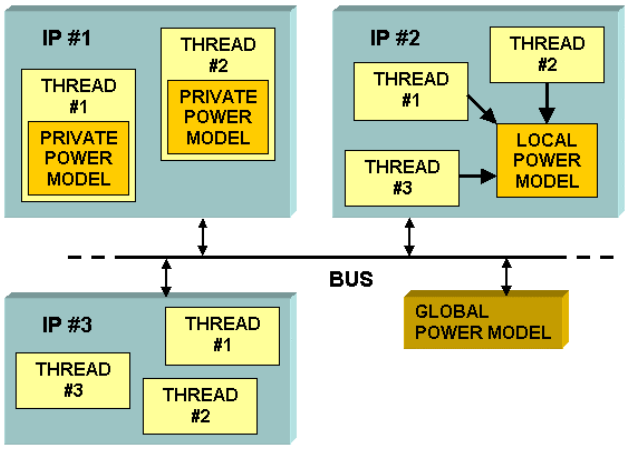

Figure 1 – Different possible power models

The highest degree of generality is reachable if it is possible to implement the power analysis in a further specific module: communicating properly with the other modules it can characterize the energetic behavior of the entire system. In this case specific communication protocols and interfaces should be defined to guarantee the highest level of reuse, the smallest reduction of the simulation speed and the desired accuracy in terms of power dissipation. In this way a global, very general model of power analysis is implemented that gives the possibility to reach the best trade-off using all the different timing, performance and energy parameters that characterize the core. These different power models are briefly represented in Fig. 1.

5. AMBA AHB bus power analysis

As stated above, SystemC library and simulation kernel was used for the development of the system-level bus model together with a library, IPsim [7-8], that provides most SoC modeling concepts; it is developed by STMicroelectronics and one important feature is to allow a simple and efficient use of complex busses like the AMBA bus by ARM [9].

The AMBA protocol defines a standard for on-chip communication and it is an efficient design tool for the development of high-performance embedded systems. AMBA specification aim at satisfying four important requirements:

- to allow the right-first-time development of embedded controllers with different CPU or DSP cores (multiple masters);
- to be technology-independent and to allow a high reusability of different blocks;
- to encourage a modular system design to preserve the best possible CPU's independence;
- to facilitate the testing phase.

AMBA specification defines three different bus topologies: the AHB, the Advanced System Bus (ASB) and the Advanced Peripheral Bus (APB). An AMBA-based architecture typically consists of a high-performance system bus (AHB or ASB), to sustain the external memory bandwidth, on which the CPU, on-chip memory and other DMA devices reside. Also located on the high-performance bus is a bridge to the lower bandwidth APB, where most of the system peripheral devices are located.

In this work the AHB was used, the last generation of AMBA bus, targeted for high-level performances. A particular testbench was created to verify the functionality and the behavior of the bus model: it is made of two master modules, a simple default master and three slave modules connected through the AMBA AHB bus. The two master modules execute *WRITE-READ* non-interruptible sequences and *IDLE* commands, for a random number of times; only in this period a bus handover can occur. This kind of testbench allows the exploration of only a part of the AHB protocol, but in this way it is possible to simplify the bus architecture and to shorten the design stage dedicated to power consumption macromodeling. This limitation does not preclude the effectiveness of the presented methodology; in fact more complete and complex bus models simply require a longer period for the characterization. The approach used for the system-level analysis of power consumption in the AMBA AHB bus is described in detail in the next paragraphs.

5.1. AHB bus structural decomposition

The AMBA AHB bus can be decomposed in the following main blocks represented in Fig. 2: one arbiter, a decoder and some multiplexing logic for read and write operations. Each block has been characterized to obtain a dynamic energy requirement model, using a "low-level" description.

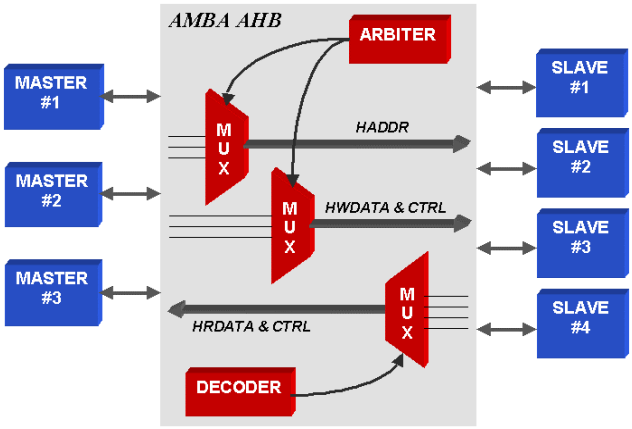

Figure 2 - AMBA AHB bus structure

For example to characterize the decoder a simple one-hot decoding behavior has been chosen and it was synthesized only with NOT and AND gates. The desired macromodel is the following, supposing that the decoder has $n_O \geq 2$ outputs (i.e. the number of slave modules connected to the bus) and defining HD_{IN} the Hamming distance between two consecutive inputs:

$$E_{DEC} = \frac{V_{DD}^2}{4}\left(n_I \cdot n_O \cdot C_{PD} \cdot HD_{IN} + 2 \cdot HD_{OUT} \cdot C_O\right) = f(n_O, HD_{IN})$$

where V_{DD} is the voltage swing between the logic levels '0' and '1', C_{PD} is the equivalent capacitance of one node, n_I is the first integer number greater than $\log_2(n_O\text{-}1)$, HD_{OUT} is '1' if $HD_{IN} \geq 1$ and C_O is the capacitance associated to each output node. This macromodel briefly expresses the dynamic energy requirement of a parametric decoder as a function of the number of slaves and the input data activity. It has been supposed to have a gate-level description of the system or at least an easy synthesizable version.

A similar macromodel was derived for a generic multiplexer:

$$E_{MUX} = f(w, n, HD_{IN}, HD_{SEL})$$

where w is the width, expressed in bit, of the data to be multiplexed, n is the number of inputs, HD_{IN} and HD_{SEL} are respectively the Hamming distance between two consecutive data and selection inputs. A simple FSM was created to model the energy requirement of a simplified version of the arbiter.

All these models were validated using the software SIS, an interactive tool for synthesis and optimization of sequential circuits developed by the University of California at Berkeley [10].

5.2. AHB bus behavioral decomposition

An instruction set has been identified to cover the functionality of the testbench. Each instruction has been characterized to extract an energy model on the base of the specific usage of the previously determined sub-blocks. In particular four main activity modes were identified: *IDLE*, *READ*, *WRITE* and *IDLE* with bus handover; the instruction set is made of all the permissible transitions between one of these states and the others.

5.3. Preliminary instrumentation

In the preliminary model instrumentation phase, a specialized object class was added for the dynamic monitoring and the storage of the activity of the I/O signals of the different blocks:

```
class Activity
{
 ...
 N_uint bit_change_count(...)
 N_uint store_activity(...)

 // Masters signals activity storage:
 ...
 // Slaves signals activity storage:
 ...
}
```

At every bus event a particular bus function, *get_activity*, monitors the value of every bus signal and updates the structures of one object of the class *Activity* using the methods *bit_change_count* and *store_activity*.

5.4. Power finite-state machine

Then a p*ower_FSM* was created with all main states and the related transition functions or instructions. This is the FSM:

```
void power_FSM()
{
 switch (state)
 {
  case IDLE: //instructions & energy computation
    ... // IDLE_IDLE
    ... // IDLE_IDLE_HO
    ... // IDLE_WRITE
  case IDLE_HO:
    ... // IDLE_HO_IDLE_HO
    ... // IDLE_HO_IDLE
    ... // IDLE_HO_WRITE
  case READ:
    ... // READ_WRITE
    ... // READ_IDLE
    ... // READ_IDLE_HO
  case WRITE:
    ... // WRITE_READ
  default:
 }
 // energy value output in a data file
 ...
}
```

During the compilation of the model, it is sufficient to define a variable *POWERTEST* to include the power analysis in the executable specification; otherwise the added code is ignored by the compiler to avoid loss of efficiency in terms of speed of the system-level functional simulation.

6. Simulations and results

The entire system was simulated using the SystemC 2.0 simulator. An analysis was conducted over the selected instruction set to carry out some energy information for a possible future power optimization of the system. In Tab. 1 the average and the total energy for the executed instructions are reported during a 50 µs simulation while the system was working at a clock frequency of 100 MHz.

From this table it is possible to see that there is not a relevant difference in the average energy requirement of each instruction; but the important point to figure out is that the 87.31 % of the entire energy required for the execution of this particular testbench is due to data transfer instructions with no bus handover and only the 12.69 % of the energy is due to bus arbitration. This means that possible optimization efforts, regarding this particular system configuration and application, should better concentrate on the AHB data-path rather than on the arbitration logic; this will give the best results in terms of power consumption reduction.

Instruction	Average energy	Total energy per instruction	
IDLE_HO_IDLE_HO	14.7 pJ	96.5 µJ	11.49 %
IDLE_HO_WRITE	16.7 pJ	0.5 µJ	0.06 %
READ_WRITE	19.8 pJ	417.7 µJ	49.75 %
READ_IDLE_HO	22.4 pJ	9.6 µJ	1.14 %
WRITE_READ	14.7 pJ	315.3 µJ	37.56 %
Total simulation energy		839.6µJ	100 %

Table 1 - Instructions energy analysis

These considerations could be verified with an other simulation where it was possible to analyze the power consumption of the different AHB sub-blocks that were previously characterized in terms of energy requirements.

The next figures show the power dissipation of the AMBA AHB bus analyzed during the first 4 µs of this simulation. In particular in Fig. 3 is reported the total AHB power consumption, in Fig. 4 is reported the arbiter power consumption and in Fig. 5 is reported the power dissipated by the multiplexer that sends data and control signals from the masters side to the slaves side.

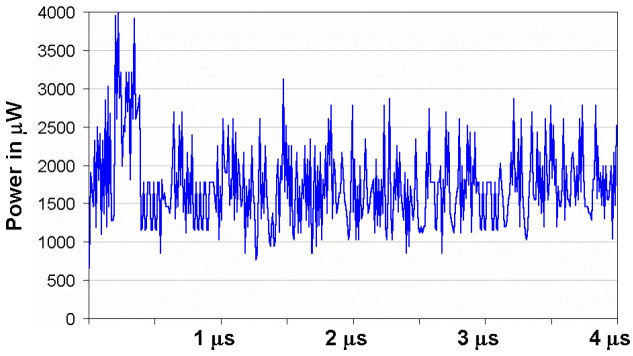

Figure 3 – Total AHB power consumption

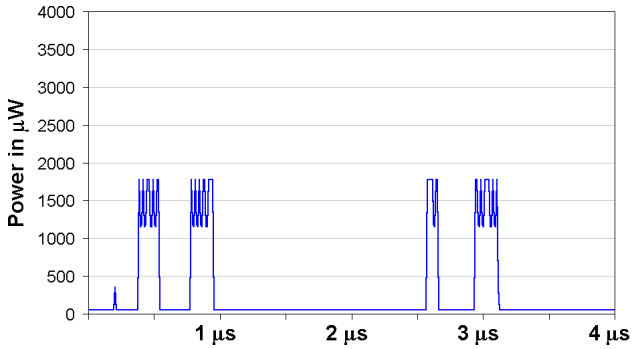

Figure 4 – Arbiter power consumption

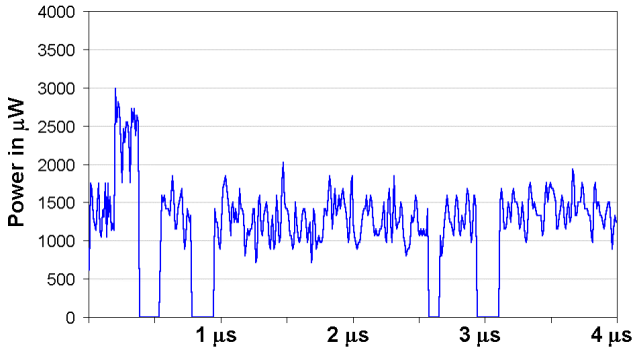

Figure 5 – M2S multiplexer power consumption

It is evident from these two last figures the different amount of power dissipated in two of the principal AHB sub-blocks; this is also graphically represented in the following Fig. 6:

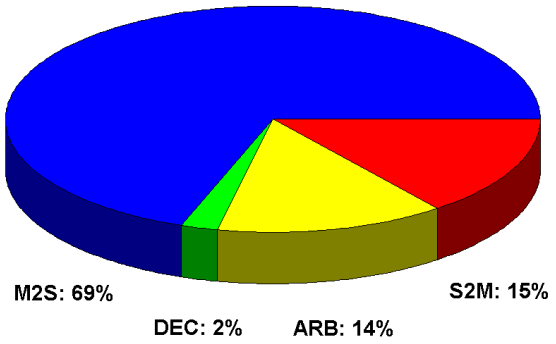

Figure 6 – AHB sub-blocks power contribution

In this figure M2S, DEC, ARB and S2M represents respectively the power contribution of the masters-to-slaves data/control multiplexer, the address decoder, the arbiter and the slaves-to-masters data/control multiplexer. At the moment the price to pay for the application of this analysis methodology to a system-level executable specification is a doubling in the simulation time.

7. Conclusion

In this paper a methodology was presented for the analysis of the power dissipated in digital blocks described at system-level; in particular SystemC and IPsim were used to create the executable specification of the AMBA AHB bus. This kind of analysis was applied to get information about the power dissipated by this bus typology during a system-level simulation. The presented methodology is still object of study and research with the purpose to gain more efficiency in terms of better result accuracy, generality of application, smaller impact on the existing code of the models and on the simulation speed. This methodology allows the designer to easy explore different system architectures taking into account the power dimension, a key aspect in modern embedded system design. In this way it is possible to take important decisions in the early phases of the design promoting a good result for the final physical implementation in terms of cost, performances and reliability.

Acknowledgments

This research has been sponsored in part by EU MEDEA+.

References

[1] MEDEA+ Design Automation Roadmap, March 2002, www.medeaplus.org/webpublic/edaroadmap_merci.htm.

[2] M. Nemani, and F. N. Najm, "Towards a high-level power estimation capability," *IEEE Trans. CAD*, vol. 15, no. 6, pp. 588–598, June 1996.

[3] A. Stammermann, L. Kruse, W. Nebel, A. Pratsch, E. Schmidt, M. Sculte, and A. Schulz, "System level optimization and design space exploration for low power," in *Proc. International Symposium on System Synthesis*, Montreal, Quebec, Canada, pp. 142–146, Sept. 30 – Oct. 3 2001.

[4] T. Givargis, F. Vahid, and J. Henkel, "Instruction based system level power evaluation of system-on-a-chip peripheral cores," in *Proc. International Symposium on System Synthesis*, Madrid, Spain, pp. 163–171, Sept. 2000.

[5] M. Caldari, M. Conti, P. Crippa, G. Nuzzo, S. Orcioni, and C. Turchetti, "Instruction based power consumption estimation methodology," in *Proc. International Conference on Electronics, Circuits and Systems*, Dubrovnik, Croatia, pp. 721–724, Sept. 2002.

[6] SystemC: SystemC documentation at www.systemc.org.

[7] M. Coppola, S. Curaba, M. Grammatikakis, and G. Maruccia, "IPSIM reference manual," ver. 2.8, Oct. 2001, STMicroelectronics internal report.

[8] M. Caldari, M. Conti, M. Coppola, M. Giuliodori, and C. Turchetti, "C++ based System-On-Chip Design," *IEEE Canadian Journal of Electrical and Computer Engineering*, vol. 26, no. 3/4, July/Oct. 2001, pp. 115–123.

[9] ARM: "AMBA specification," rev. 2, May 1999.

[10] E. M. Sentovich, et al., "SIS: a system for sequential analysis," University of California, Berkeley, May 1992.