Doctoral School on Engineering Sciences
Università Politecnica delle Marche

Extended summary

System level power estimation

Curriculum: Electronics, Circuit Theory and Telecommunication Engineering

Author

Marco Giammarini

Tutor

Orcioni Simone

Date: 30-01-2012

Abstract. In recent years, the increase in functionality of Systems-on-Chip (SoC) have favored the explosion of the market of electronic appliances, especially of small mobile and wireless devices. The continuous development in the field of silicon technology led to an increase both in integration and clock frequency of electronics systems, raising both power density and energy dissipation of systems. As a result, power dissipation has become an important constraint in the design of complex or portable systems. Some design and technology issues related to power efficiency are becoming crucial, in particular for power optimized cell-libraries, clock-gating and clock-trees optimization, and dynamic power management. The first necessary step towards low-power design is the estimation of dissipated power/energy of the system under development. Recently, emphasis is moving towards estimation at system level when some good ideas on optimizing power dissipation can drive the choice between different architectures. Power analysis at system level is less accurate than at lower levels since the details of the real implementation of the functionality is not defined yet, but conversely the simulation time is much faster and the power saving opportunity with an optimization is much higher. This work proposes a new framework, called Powersim, that consists of a C++ library added to SystemC, in order to estimate the energy consumption of a system described at system level. Powersim can estimate the energy of a system, interacting with arithmetic operations, logical functions of the various modules constituting the system. Also, thanks to the fact that Powersim monitors operators, it is also able to provide a computational complexity estimate of the system.

Keywords. Complexity estimate, energy estimate, power estimate, powersim, system level

1 Introduction

In recent years, the increase in functionality of Systems-on-Chip (SoC) have favored the explosion of the market of electronic appliances, especially of small mobile and wireless devices. The continuous development in the field of silicon technology led to an increase both in integration and clock frequency of electronics systems, raising both power density and energy dissipation of systems. As a result, power dissipation has become an important constraint in the design of complex or portable system [1].

The first necessary step towards low-power design is the estimation of dissipated power/energy of the system under development. Recently, emphasis is moving towards estimation at system level when some good ideas on optimizing power dissipation can drive the choice between different architectures. Power analysis at system level is less accurate than at lower levels since the details of the real implementation of the functionality is not defined yet, but conversely the simulation time is much faster and the power saving opportunity with an optimization is much higher.

Wattch [2] is an architectural level framework for power analysis, based on parametrized power models of common structures present in modern superscalar microprocessors. Recently the Wattch power simulator has been integrated in a complete simulation framework called SimWattch [3].

Many tools and methodologies have been developed for power estimation at register-transfer level (RTL), for example [4] propose a paradigm to speedup power estimation at RTL. SimplePower [5] is an execution-driven, cycle-accurate, RTL power estimation tool. The framework evaluates the effect of high level algorithmic, architectural, and compilation trade-offs on energy. SimplePower provides cycle-by-cycle energy estimates for processor datapath, memory and on-chip buses. Recently [6] presented a series of tools to enable the power analysis from layout, gate level, RTL, to system level. A power modeling framework that uses a hybrid power modeling methodology, from both physical measurements and analytical expressions is presented in [7]. SystemC-based power estimation frameworks have been proposed in [8] and [9] to perform the power estimation at system level.

Nevertheless power estimation can be carried out only if time details of simulation are known; this can not happen at system level, for example in Transaction Level Modeling, so speaking of energy estimation is more general and will be used in the following.

The next Section will present the tool, while Section 2 will show a simulation of energy in Powersim, compared with the measures of its implementation on a micro-controller.

2 Powersim

Powersim is a C++ class library developed to be used inside a SystemC implementation. Its main purpose is the simulation of energy consumption of digital systems, described at system-level. Its way of operation is based on monitoring the C++ operators, when called on SystemC data type. This allows the user not to change the application source code to obtain the energy consumption during a SystemC simulation. The SystemC modules to be monitored and the energy models to be used must be entered in a configuration file.

The development of Powersim required some changes and addition to some classes of the SystemC library. So when the user want to simulate the power consumption, he must

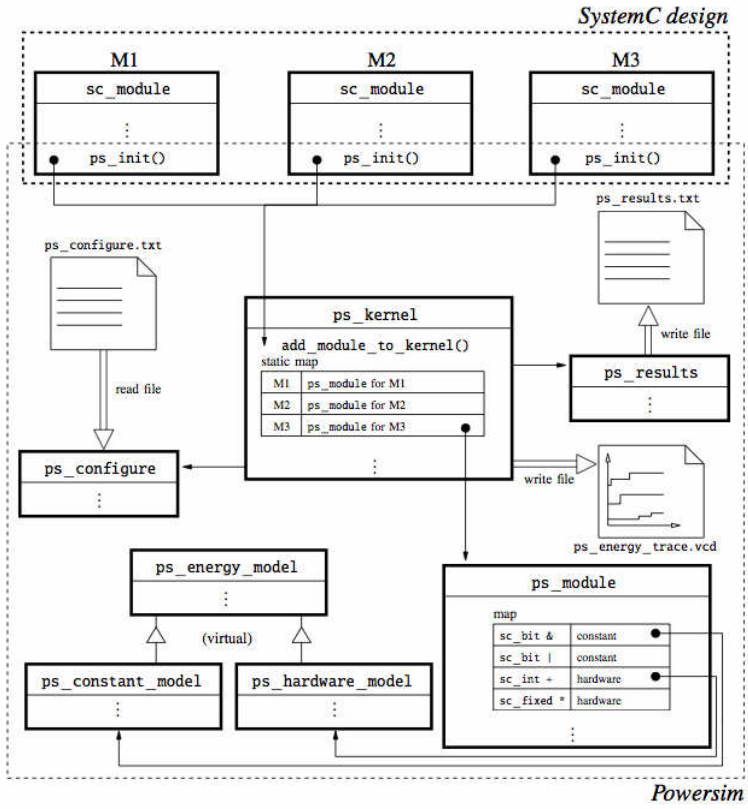

Figure 1. Scheme of the interaction of Powersim with SystemC design.

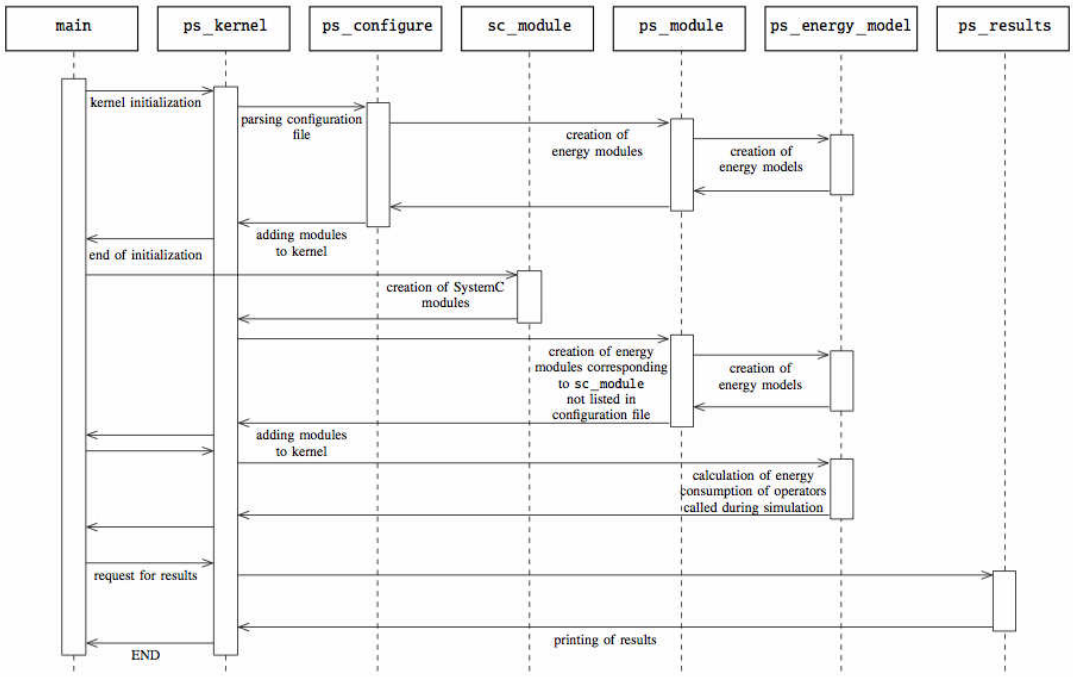

Figure 2. Sequence diagram of Powersim.

compile his source code with a modified SystemC library that include Powersim [10].

2.1 Powersim classes

Powersim library consists basically of five classes, in addition to energy models and error-report classes. Figure 1 shows the structure of Powersim and how it interacts with SystemC.

The main class is `ps_kernel`: an object of this class is created inside the `sc_main` function before the SystemC modules are created. This class, using the static methods of `ps_configure`, parses the configuration file and stores the read data in a static map. The map contains an object of `ps_module` class for each SystemC module used in the source code. Each `ps_module` object takes care to maintain and update data on computational cost and energy dissipation estimation of the module to which it is associated. This is done using a map that contains `ps_energy_model` type pointers that point to classes derived from `ps_energy_model`.

`ps_energy_model` class is an interface which generalizes all functions necessary to create a new energy model. The user can implement its own energy model by extending `ps_energy_model` and implementing all its methods.

The sequence diagram of Powersim classes is shown in Figure 2. Here the interaction between classes is embedded in the flow of the simulation. The comment near the arrows explain the action that the pointed class is called to fulfill.

2.2 SystemC modifications

Regarding the changes to SystemC, we have acted on two types of classes: (i) `sc_module` and (ii) data-type classes.

The changes to the `sc_module` class, which lets to create a new SystemC module, consist of providing a new method called `ps_init()`. This method, during the simulation, calls a static function of `ps_kernel` class. The static method adds the current module under control of the Powersim kernel. Furthermore, the definition of each operator function has been modified to allow the call to `ps_call()`. This function, during simulation, tells to the Powersim kernel that a new operation was performed.

2.3 Energy models

The energy models are the instrument through which Powersim estimate the energy associated with the execution of an operator on a SystemC data type. They are implemented by user-defined functions that can compute energy in function of input, parameters and variables like supply voltage. They can also take into account of static power consumption. As mentioned, the energy models are represented by classes that extend `ps_energy_model` class and implement also the methods necessary to interact with Powersim kernel. These methods have different purposes: (i) to define the model of the energy dissipated by arithmetic or logic operations, (ii) to set the parameters of the energy models, using the values that the user provide in the configuration file, (iii) to return the total energy consumption, (iv) to manage the variables that will be printed in the result tracing file.

2.4 Configuration file

In Powersim, it is possible to assign a different energy model for each operation performed on each data type and it is possible to add, in a simple way, new energy models.
If these energy models are taken as unitary, the simulation gives the computational cost of the running [11,12]. Through the configuration file, the user can choose which modules and which operators to monitor. Default values can be used for modules and operators not listed in the configuration file. In order to write this file, the user must follow a grammar defined by Extended Backus-Naur Form, a syntactic meta-language used to describe formal languages syntax. The rules imposed by the grammar are used by `ps_configure` class to parse the input file.

3 Case Study

In order to verify Powersim, we chose to test it with a real hardware, in particular with a DEMOJM board of Freescale Semiconductor Inc. This demo-board has MC9S08JM60 micro-controller on board with various other device such as LEDs, buttons, buzzer, 3-axis accelerometer, and potentiometer. Furthermore the board allows, using an appropriate jumper, the measurement of the current drawn by the micro.

3.1 Energy models

Before testing Powersim, it was necessary to develop the energy models to be associated with operations to be performed by the micro[13]. In order to derive the energy models, we designed a small board to insert in the proper jumper of DEMOJM. The schematic of this board is shown in Figure 3. We used the ZXCT1010 current monitor of Zetex Semiconductors. The current monitor convert the supplied current in voltage according to the following expression

$$V_{out} = G_m \cdot V_{sense} \cdot R_{out} \tag{1}$$

where G_m=10mAV^{-1} and V_{sense}=R_{sense} I_{Vdd}. The values of used resistances are: R_{sense}= 10Ω and R_{out}=2.7kΩ, so that

$$V_{out} = G_m \cdot R_{sense} \cdot I_{Vdd} \cdot R_{out} = 270\Omega \cdot I_{Vdd} \tag{2}$$

V_{out} was acquired by National Instruments USB-6210 multifunction data acquisition module. The measurement setup is shown in Figure 4.

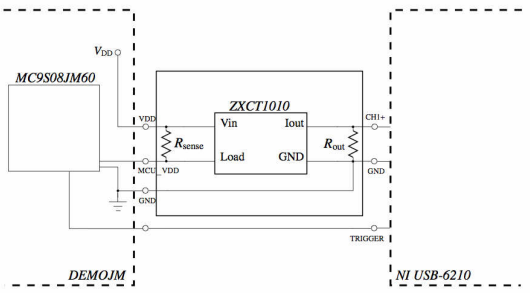

Figure 3. Testing circuits.

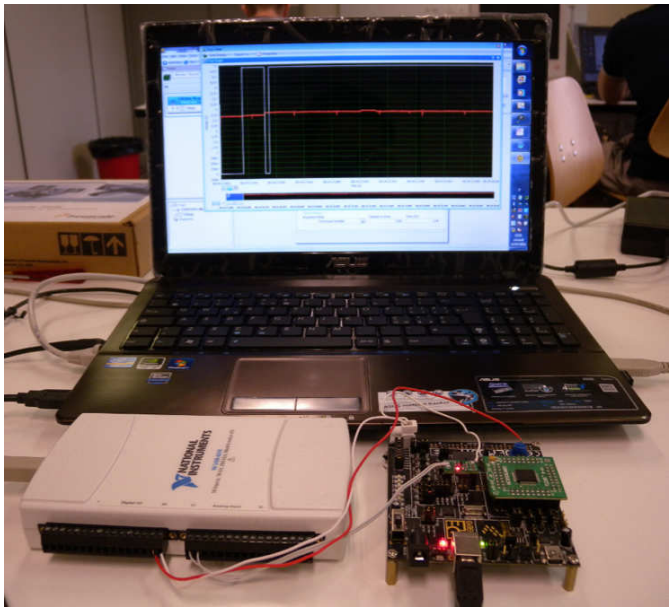

Figure 4. Picture of measurement set-up.

To calculate the energy used for all arithmetic and logic operations we have written a firmware which repeatedly execute the same operation, N_{op} times. The current drawn by the core of the micro during arithmetic or logic calculation is almost constant, but different arithmetic operation requires different clock cycles. The energy model of each operator can be simply calculated by the following expression

$$E_{op} = \frac{T_S \cdot V_{DD} \cdot \sum_n I_{Vdd}[n]}{N_{op}} \tag{3}$$

where V_S is the sampling time. The energy model of different operators, when implemented in the micro MC9S08JM60, are shown in Table 1.

3.2 FIR filter implementation

To test Powersim, we used a SystemC implementation of Finite Impulse Response (FIR) filter contained in the examples of the Open SystemC Initiative implementation. This filter implements the direct form of FIR by the following equation

$$y[n] = \sum_{m=0}^{M} b[m] \cdot x[n-m]. \tag{4}$$

The computational cost of this implementation is of *(N-1)·M* addition and *N·M* multiplication, where *N* is the length of input signal and *M* is the number of filter taps. In our test, a 16-tap FIR filter, with an 24 samples input signal, was used. In this case, the theoretical number of operations is: 384 multiplication and 360 additions. This calculation does not take into account other operations that are performed to implement the algorithm in (Eq. 1), such as subtraction and addition to shift the array and assignments to hold the value of each variables.

Table 1. Energy consumption by each operations in nWs.

C Type **SystemC Type**	**`uint8_t`** **`sc_uint<8>`**	**`uint8_t`** **`sc_uint<8>`**	**`uint8_t`** **`sc_uint<8>`**	**`uint8_t`** **`sc_uint<8>`**
Addition	30.45	30.45	121.63	121.63
Subtraction	30.49	30.49	123.70	123.70
Multiplication	62.63	62.63	514.99	514.99
Division	56.08	1006.53	1664.50	1819.32
Modulo	97.83	1022.22	1819.32	2058.95
Assignment	30.40	30.40	50.57	50.57
Bitwise And	30.75	30.75	119.89	119.89
Bitwise Or	30.67	30.67	121.45	121.45
Bitwise Xor	30.80	30.80	121.74	121.74
Bitwise Not	54.00	54.00	153.42	153.42
And	67.38	67.38	108.37	108.37
Or	44.39	44.39	93.32	93.32
Equal	45.05	45.05	69.16	69.16
Not Equal	45.05	45.05	68.65	68.65
Less than	44.30	44.30	68.52	68.52
Less eq. than	44.30	44.30	68.52	68.52
Greater than	45.17	45.17	67.94	67.94
Greater eq. than	45.17	45.17	67.94	67.94
Right shift	22.82	22.82	479.87	479.87
Left shift	28.59	28.59	469.45	469.45

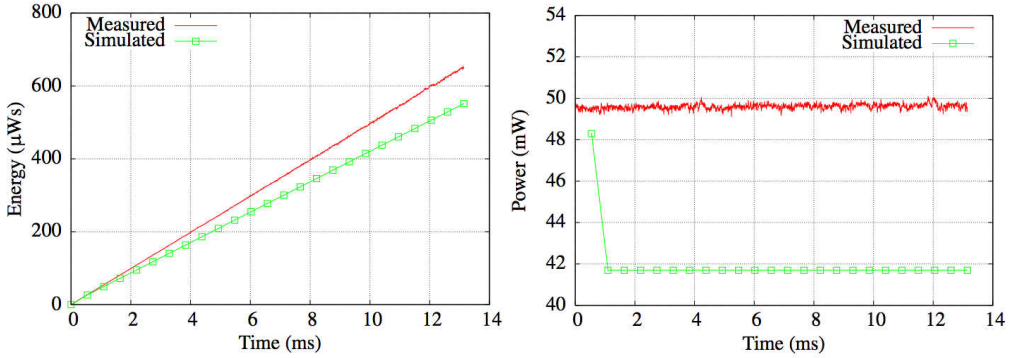

Figure 5. FIR filter energy and power consumption. Comparison between measure and simulated results.

By using the energy models reported in Table 1, Powersim can compute the total energy consumption of the algorithm. In order to compare the simulation with reality, the same code was ported in the MC9S08JM60. The same measurement set-up, shown in Figure 3, was used to measure the energy consumption of the FIR filter implementation. The start and the end of the simulation was triggered by the micro. Because the timing of hardware implementation is now known, also the power of Powersim simulation can be calculated and compared with measurements. The comparison of measurements and simulation are reported in Figure 5 showing a mean relative error in power estimation of 15.8%.

4 Conclusions

A new tool for energy estimation at system level, called Powersim is presented. Powersim is based on C++ operator monitoring and it is characterized by a great flexibility and an extreme easiness of use. Indeed no modification in application source code is needed for energy simulation.

Energy models of operators implemented in MC9S08JM60 were obtained and tested with a FIR filter simulation and implementation. A good agreement between measurement and simulation was obtained.

References

[1] M. Conti, S. Orcioni, G. Vece, and S. Gigli. *Power optimization in multi-core system-on-chip.* In Multi-core Embedded Systems. CRC Press, ch. 3, pp. 71-109.

[2] D. Brooks, V. Tiwari and M. Martonosi. *Wattch: a framework for architectural-level power analysis and optimizations.* In Proceedings of the 27th Int. Synp. On Computer architecture, 2000, pp. 83-94.

[3] J.Chen, M. Dubois, and P. Stenstrom. *Simwattch and learn.* Potentials IEEE, vol. 28, no. 1, pp 17-23, jan.-feb. 2009.

[4] J. Coburn, S. Ravi, and A. Raghunathan. *Power emulation: a new paradigm for power estimation.* In Proc. of 42nd Design Automation Conference 2005, june, pp. 700-705.

[5] W. Ye, N. Vijaykrishnan, M. Kandemir, and M. J. Irwin. *The design and useof simplepower: A cicle accurate energy estimation tool.* In Proc. of ACM/IEEE Design Automation Conference 2000, pp. 95-106.

[6] S. C. Fang, C. C. Weng, C. K. Tseng, C. W. Hsu, J. L. Liao, S. Y. Huang, C. L. Lung, and D. M. Kwai. *SoC power analysis framework and its application to power-thermal co-simulation.* In Proc. of International Symp. on VLSI Design Automation and Test 2011, April, pp. 1-4.

[7] R. Ben Atitallah, S. Niar, and J. L. Dekeyser. *MPSoC power estimation framework at transaction level modeling.* In Proc. on International Conference on Microelectronics 2007, ICM07, December, pp. 245-248.

[8] G. Vece and M. Conti. *Power estimation in embedded systems within a SystemC-based design context: the PKtool environment.* In Proc. on Seventh Workshop in Intelligent Solutions in Embedded Systems 2009, june, pp. 179-184.

[9] F. Klein, G. Araujo, R. Azevedo, R. Leao, and L. dos Santos. *An efficent framework for high-level power exploration.* In Proc. on 50th Midwest Symp. on Circuits and Systems 2007, aug., pp. 1046-1049.

[10] Powersim 0.1.2. Web site, documentation and source code. http://sourceforge.net/projects/powersim

[11] M. Giammarini, S. Orcioni, and M. Conti. *Computational complexity estimate of a DSR front-end compliant to ETSI standard ES 202 212.* In Proc. on Seventh Workshop in Intelligent Solutions in Embedded Systems 2009, june, pp. 171-177.

[12] M. Giammarini, S. Orcioni, and M. Conti. *Powersim: power estimation with SystemC.* In Solutions on Embedded Systems, ser. Lecture Notes in Electrical Engeneering, Springer 2011, vol. 81, ch. 20.

[13] M. Giammarini, M. Conti, and S. Orcioni. *System-level energy estimation with Powersim.* In Proc. on 18th IEEE International Conference on Electronics Circuits and Systems 2011, ICECS11, December.