ISSN 1392 – 124X INFORMATION TECHNOLOGY AND CONTROL, 2006, Vol. 35, No. 2

ESTIMATION OF DESIGN CHARACTERISTICS AT RTL MODELING LEVEL USING SYSTEMC

Robertas Damaševičius

Software Engineering Department, Kaunas University of Technology
Studentų 50-415, LT-51368, Kaunas, Lithuania

Abstract. A successful SoC and embedded system design requires the thorough domain analysis and design space exploration. The early evaluation of design characteristics allows to take advantage of many architectural options available and to modify the system architecture, if needed. Currently, SystemC is used to model hardware and software parts of the system at the high-level. However, the characteristics of the modeled systems are obtained only at the late stages of the design. In this paper, we present a framework for the estimation of design characteristics at the modeling level of a design. The SystemC class library is extended with new classes describing the computation of area, delay and power characteristics of the SystemC models. The achieved results are illustrated with a case study.

Keywords: design characteristics, power modeling, design space exploration, SystemC.

1. Introduction

The main aim of a designer is to develop a target system while keeping the development time and costs as minimal as possible. The realization of this aim is subject to various performance and functionality constraints such as battery capacity or device response time. As the complexity of embedded systems and Systems-on-Chip (SoC) is rapidly growing, the designers are moving towards higher levels of abstraction in design, such as UML-based design [1], platform-based design [2] and SystemC [3]. Different abstraction levels are used to address different design concerns at the different level of detail. The key objective of the designer is to model the system at each abstraction layer with as little detail as possible and to estimate the design metrics, which are further used to make sound design decisions [4]. Analysis of design characteristics is vital in the early stages of design, where there are many design space exploration options for determining the SoC architecture and selecting or trading off the key hardware (HW) or embedded software (SW) components.

While providers of the commercial synthesis tools are increasingly focusing on SoC design, their tools usually implement a top-down approach that requires the SoC designer to fully define the function of a developed system, repeatedly decompose coarse-grained functions into smaller subfunctions, and then map them into the available library HW cores [5].

Using such a methodology, the physical design characteristics can be estimated only in the final stages of design. Consequently, in case of any mismatch with the imposed design constraints it is very costly and almost impossible to redesign the system in a given time span. That can be one of the reasons, why 85 % of SoC design projects miss their target date [6]. Even minor modifications require large design efforts and much time. It may take two or more weeks to rebuild a moderately modified SoC to a physical implementation ready for verification [6]. As design processes move to the higher levels of abstraction, estimation of design characteristics must follow them, too.

One of the most important design characteristics for a wide range of electronic systems, starting from battery-powered mobile appliances to smart devices, is power (energy) consumption [7, 8], though other characteristics such as a chip area or system delay (latency) remain as important as ever. With rising embedded system complexity, it is becoming increasingly critical to address the power consumption early in the design cycle, i.e., at the system-level, when significant opportunities exist for optimizing the system architecture and for meeting the design constraints [9, 10, 11].

The novelty of this paper is the framework for the estimation of area, power and delay characteristics of HW systems modeled at the register-transfer level (RTL) using the SystemC modeling language. The framework also allows for dynamic power profiling and analysis based on the state of the modeled circuit.

The outline of the paper is as follows. Section 2 overviews the related work. Section 3 presents the description of the model estimation framework.

Section 4 presents a case study. Finally, Section 5 evaluates the results and presents the conclusions.

2. Related work

Recently, with the move towards system-level specifications and design methodologies in SoC design, there has been a significant research interest in power estimation and modeling. A detailed survey of high-level power modeling and optimization techniques is presented was [12].

According to [4], the consumed power can be estimated at four different abstraction levels:

(1) Transistor-level methods simulate the circuit at the transistor or switch level and monitor the supply current [13].

(2) Gate-level methods simulate a design at the logic-gate level and calculate power using switching activity and node capacitance [14].

(3) RTL methods model the power consumption of middle-grained components such as muxes, adders, multipliers, and registers [15, 16, 17].

(4) System-level methods estimate power consumption based on simple high-level descriptions of the systems using abstract models of capacitance and switching [11, 18, 19].

At the system-level of abstraction, SystemC is becoming widely accepted as a unified modeling tool for modeling embedded systems. SystemC, which in fact is an extension of C++ with a library of C++ classes for HW simulation, lacks the semantics to capture energy and other HW-related physical information. However, the object-oriented nature of SystemC allows to extend it for providing to the designer more information than just plain waveforms. Several related works deal with the high level modeling power and other design characteristics using SystemC or other C-based models.

For example, Darringer *et al.* [5] described an approach for developing high-level performance models for SoC designs and outline how this performance analysis capability can be integrated into an overall environment for the efficient SoC design.

Abril *et al.* [18] proposed a simulation framework for the energy consumption estimation and optimization of high-level behavioral C models of SoC.

Lajolo *et al.* [20] described power estimation techniques for HW/SW designs based on concurrent and synchronized execution of multiple power estimators that analyze different parts of SoC. Power estimators operate at different levels of abstraction (e.g., RTL/gate-level power simulator for application-specific HW parts, ISS-based power estimator for embedded SW, and a behavior-based power estimator for SoC integration architecture).

Orinoco [21] is a commercial high level power estimation tool, which allows to compare power consumption of different algorithms running in different architectures in SystemC.

Brandolese *et al.* [22] presented a SystemC-based area estimation methodology for design space exploration of FPGA designs. The methodology is based on a two-level model described in behavioral style and the area is expressed in terms of number of flip-flops and look-up tables. The methodology focuses on a specific tool chain and a specific target device, but allows semi-automated retargeting towards different combinations.

Bansal *et al.* [23] proposed a framework for integrating power models within a system-level simulation environment to achieve a superior trade-off between overall power estimation accuracy and efficiency. A framework is based on a network of power monitors, which observe component-level and system-level execution and power statistics at run time.

Fornaciari *et al.* [24] introduced power metrics as a part of the HW/SW co-design environment to guide the process of system-level partitioning. Power evaluation metrics were defined to widely explore the architectural design space at the high abstraction level.

Krishna *et al.* [16] proposed RTL estimates for power exploration at the behavioral level of design using RTL templates, which characterize the area, delay and power of the design. The templates are based on some knowledge of the logic block such as the number of nodes, levels and their interconnections. The average switching activity of a module is calculated based solely on its functionality.

Xanthos *et al.* [25] proposed a modification to the SystemC library to enable power estimation of digital systems built upon a set of primitive logic gates. Minor modifications of SystemC modules enable the calculation of the dynamic power component due to logic transitions on the nodes of a digital circuit.

This paper was primarily inspired by the work of [25] and is an extension of it. The model estimation framework proposed in this paper (1) is more accurate, because it takes the different input capacitance of the technological library components into account. (2) The model was extended for area and delay estimation of SystemC models. (3) The presented model allows for dynamic power profiling and analysis based on the state of the circuit.

In the next section, we present a description of the proposed framework.

3. Description of framework

3.1. Basic principles

We have defined three main criteria for our model estimation framework: power consumption, delay and chip area. Delay is the amount of time passed between the supply of input signals to the circuit and the receipt of output signals. The area is an estimation of the final circuit area occupied by the implemented

system on a chip. The power consumption defines the amount of power consumed by a component (system) during the execution of its functions. The power consumption in ICs has three main components [26]:

(1) Power consumption during the switching of CMOS gates, when the complementary parts are open simultaneously.

(2) Power consumption caused by leakage currents during the non-conducting state of gates. Parameters influencing leakage are the supply voltage, the transistor threshold voltage, transistor dimensions like width and length, temperature, IR drop effects, manufacturing tolerances and the state of the gate.

(3) Dynamic power consumption during data dependent switching of transistors and connections between them. The dynamic power consumption depends upon switching activity and the size of switched capacities. Dynamic power consumption accounts for the largest portion of the total consumption of power in digital circuits [27].

In this paper, we analyze only the dynamic power consumption of a HW system. We assume that each node of a digital circuit is an input and an output of a logic gate. Therefore, the capacitance being switched consists of two components [25]:

(1) The input capacitance C_{in} of the gate, which consists of the gate capacitances of the input signal receiving transistors.

(2) The output capacitance C_{out} of the gate which consists of the diffusion capacitances of all transistors connected to the output node of the driving gate.

The estimated dynamic power E_{dyn} consumed by the model of a system M during one cycle of operation is calculated as the sum of all switched input and output capacitances multiplied by the power supply voltage V_{DD} [28]:

$$E_{dyn} = \sum_{m \in M} (c_{in} t_{in} + c_{out} t_{out}) V_{DD}^2 . \quad (1)$$

where:

t_{in} , t_{out} are the number of input and output transitions from logic "0" to logic "1",

c_{in} , c_{out} are the input and output capacitances that depend upon the gate type and the technology used,

m is a component of a system M.

The chip area A of the model of a system M is estimated as a sum of the area of all components of a modeled system as follows:

$$A = \sum_{m \in M} A_m . \quad (2)$$

Note that the area of wiring is not estimated here.

The delay of a model of a system M is the longest delay (critical path delay $D_{cr.path}$) of an output of a model M, and it is calculated as the maximal delay of its inputs d_i, $i \in I_m$ increased by the delay D_m of its component m as follows:

$$D_{cr.path} = \sum_{m \in M} \left(\max_{i \in I_m} d_i + D_m \right). \quad (3)$$

where:

I_m – is the set of inputs of the component m.

3.2. Extension of SystemC library

Using the Eq. (1)-(3), we have extended the SystemC library in order to allow for the estimation of the physical design characteristics. Figure 1 shows the extension metamodel. We extended the SystemC library with 3 additional classes: (1) *sc_areaHandler* class for calculation of the area characteristic of the estimated system, (2) *sc_delayHandler* class for calculation of delay of the estimated system, and (3) *sc_powerHandler* class for estimating power consumption of a modeled system.

The references to these classes are inserted into the *sc_module* class, which is a super-class for all SystemC models. Therefore, each SystemC model gets its own area, delay and power handler classes by inheritance. The estimated HW system is represented using the *Composite* design pattern. Each module of a system is either itself a composite of the lower-level components or a lowest level component. Each SystemC module is provided with the additional propagate and delay signals required for power and delay modeling, respectively. The estimation takes place during the modeling of the system and the results of estimation are printed to the standard output.

For the power estimation, for each component of the modeled system we need to add the *estimation()* method to calculate the number of 0-to-1 transitions at its input and output ports, calculate the corresponding dynamic energy consumption using Eq. (1) and save the results to a global variable. The values of the input and output signals are stored and used during the next estimation of the component's power consumption. The calculation of power consumption depending upon the number of input and output transitions and the type of the component is performed by the *calcPower*(…) method of the *sc_powerHandler* class. For each composite module we just need to add an additional propagate signal required to propagate calculation of power to the lower levels of a model during modeling.

For the calculation of area, we just add to the *estimation()* method a call to the *calcArea(...)* method of the *sc_areaHandler* class. This method increases the value of a variable that is used to store the value of chip area depending upon the component type.

For the calculation of critical path delay, for each signal in a SystemC model an additional delay signal is introduced, which is used to propagate the delay value of each signal across the circuit. For the composite models, the delay signals are just wired to the corresponding delay signals of the lower level components, while for the components, the values of

input delay signals are passed to the *calcDelay(...)* method of the *sc_delayHandler* class and the values of output delay signals are calculated depending upon the type of the component.

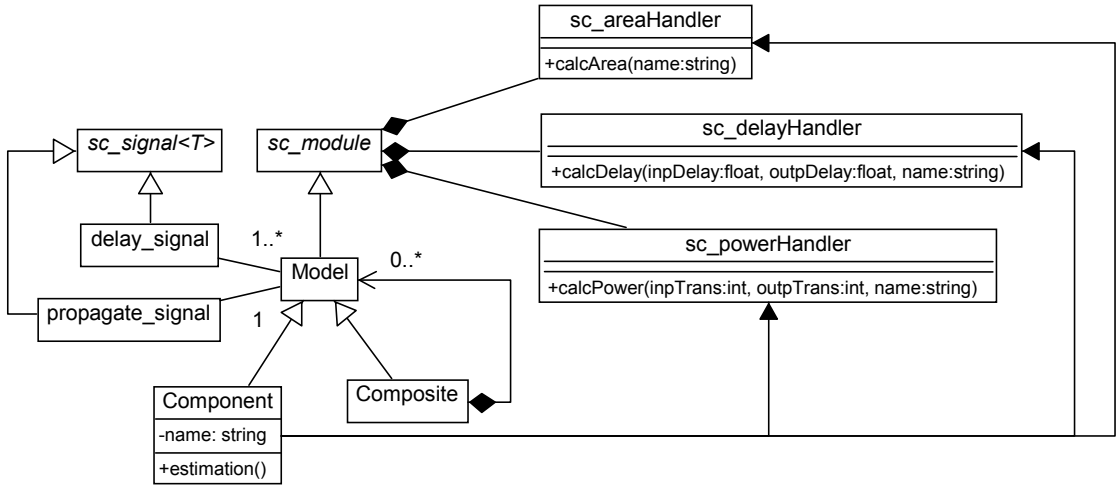

Figure 1. SystemC extension metamodel

For the calculation of area, we just add to the *estimation()* method a call to the *calcArea(...)* method of the *sc_areaHandler* class. This method increases the value of a variable that is used to store the value of chip area depending upon the component type.

For the calculation of critical path delay, for each signal in a SystemC model an additional delay signal is introduced, which is used to propagate the delay value of each signal across the circuit. For the composite models, the delay signals are just wired to the corresponding delay signals of the lower level components, while for the components, the values of input delay signals are passed to the *calcDelay(...)* method of the *sc_delayHandler* class and the values of output delay signals are calculated depending upon the type of the component.

The pseudocode of the *estimation()* method is given in Figure 2.

```
calculate outputs of model
for each input signal in model
    increase input 0-to-1 transitions counter
end for
for each output signal in model
    increase output 0-to-1 transitions counter
end for
call power handler (input transitions counter,
    output transitions counter, model name)
call area handler (model name)
call delay handler (input delay signals,
                    output delay signals)
store current values of inputs and outputs
```

Figure 2. Pseudocode of the estimation method

The extension of SystemC modules describing a particular HW system model is summarized using pseudocode in Figure 3.

```
for each model in system
    add propagate signal
    for each signal in model
        add delay signal
    end for
    if model is composite
        for each component in model
            wire delay signal
            wire propagate signal
        end for
    else
        add estimation method
    end if
end for
```

Figure 3. Pseudocode of modification of SystemC models for estimation of characteristics

4. Case study

As a case study for estimation of design characteristics, we consider three 1-bit full adder implementations: 1) naive, 2) NAND-based, and 3) AOI-based.

The naive implementation (Figure 4) consists of 2 XOR gates, 2 AND gates and 1 OR gate.

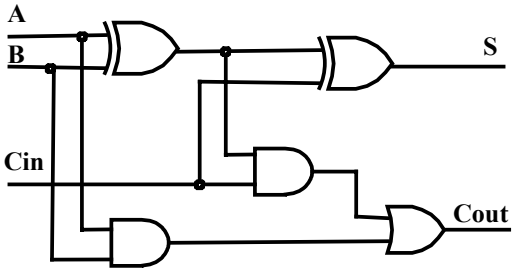

Figure 4. Schematics of naive 1-bit adder implementation

The NAND-based implementation (Figure 5) consists of 3 NAND2 gates, 5 NAND3 gates, 1 NAND4 gate and 3 INV gates.

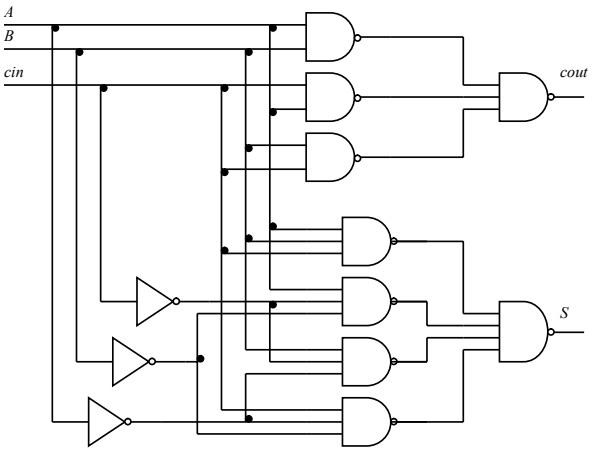

Figure 5. Schematics of NAND-based adder implementation

The AOI-based implementation (Figure 6) consists of 2 NOR2 gates, 3 AOI (AND-OR-INV) gates and 1 INV gate.

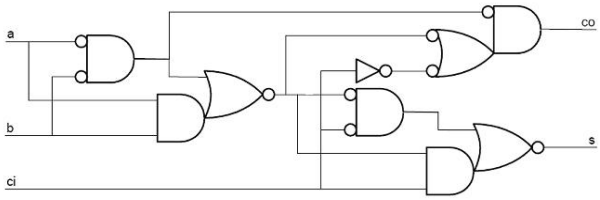

Figure 6. Schematics of AOI-based adder implementation

The results of estimation of the physical design characteristics are presented in Table 1. The characteristics of 0.8 μm CMOS technology; VDD = 3.3V library gates were used [29].

As we can see, the most efficient adder implementation in terms of area, delay and average power consumption per cycle is the AOI-based implementation. However, due to its slower speed, the naive implementation actually consumes less power per second. Thus, if the designer needs to meet area or delay constraints, he should select the AOI-based adder, whereas for low-power devices, where power is the biggest concern, and the delay is not so important, the naïve implementation may be selected.

The estimation of power consumption is a complex problem, because power dissipated in circuits depends not only upon the inputs of the circuit, but also upon the previous inputs of the circuit. Thus, the power consumption numbers given by most of circuit synthesizers are only averaged values based on a sample of circuit inputs. However, power consumption of the circuit is not constant and varies over time.

In Figure 7, we present the power consumption profile of the NAND-based adder model. Power consumption here is considered as a function that depends upon two parameters: current input values and previous input values of the circuit: $P = f(I_t, I_{t-1})$.

Here, input values are coded as particular states of component input, e.g., if A = 0, B = 0, cin = 0, then Input state = "1", etc. As we can see, if the input is not changing, the circuit remains in the same state and no power is dissipated. If the input of the circuit changes, the state of the circuit also changes and power is dissipated depending upon particular combination of current and previous state of the circuit. In this case, the largest power dissipation values are observed, when most of the input bits are switched, e.g., in "S8" -> "1" transition, when all circuit inputs are switched from logic "1" to logic "0".

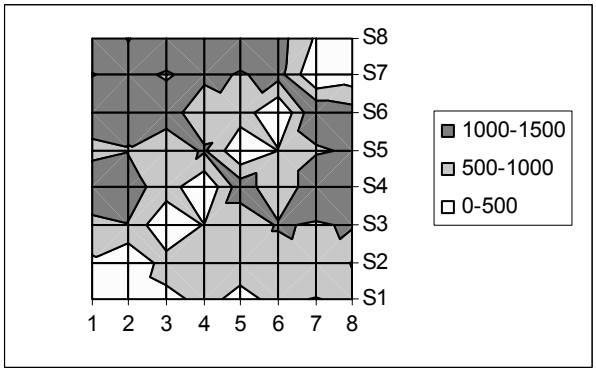

Figure 7. Power consumption profile of the AOI-based adder implementation

Power modeling also provides another possibility for a designer: power consumption breakdown by component type. The designer can use this option to discover power consumption "hotspots" in his design, i.e., components that are consuming much power and try to find an alternative solution using different components that consume less power. For example, in Figure 8 power consumption breakdown by component type for two adder implementations is given. As we can see, most of power is consumed by the NAND3 gates in NAND-based implementation and the AOI gates in AOI-based implementation.

Table 1. Characteristics of the adder implementations

Adder implementation	Area, μm^2	Delay, ns	Av. power consumption/cycle, pW	Av. power consumption/s, mW
Naive	7043	2.49	9.649	3.875
NAND-based	7413	1.43	8.826	6.172
AOI-based	4819	1.38	7.331	5.312

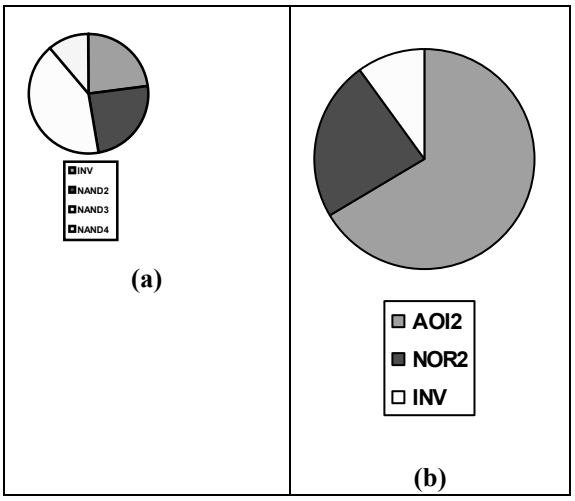

Figure 8. Power consumption breakdown by component type: (a) NAND-based adder implementation, (b) AOI-based adder implementation

The usage of the model estimation framework in SystemC however, for a designer, has some penalty. Since during modeling of a system more computations are performed, the system is modeled slower than without estimation of design characteristics. The modeling slowdown results for our framework are presented in Table 2.

The results show that slowdown caused by estimation is about 68 %, which is a satisfactory result, considering that other papers report up to 8.5 slowdown factor of modeling incurred by estimation of design characteristics [23] (note that direct comparison is not possible due to the different complexity and functionality of the estimation frameworks and modeled systems).

Table 2. Decrease of modeling speed

Adder implementation	Modeling time without estimation, ms	Modeling time with estimation, ms	Slowdown factor
Naive	10.0	16.6	66 %
NAND-based	10.3	17.2	67 %
AOI-based	7.5	12.8	71 %

5. Evaluation and Conclusions

The main benefits of the framework are as follows. First, the framework allows to estimate design characteristics at a higher abstraction level, which allows faster modeling and testing of a design. Second, the framework allows estimating design characteristics at an early design stage, thus allowing a designer to select more efficient implementations or modify design architecture that does not satisfy design constraints with less pain and cost, and consequently allows to decrease time-to-market and increase overall designer productivity.

The framework, however, also has some drawbacks. First of all, the models must be modified for introducing new signals and methods. We hope to solve this problem in future by transforming the analyzed models into estimatable models automatically using generation/transformation techniques, e.g., metaprogramming.

Another problem is that modeling of a system becomes slower. In our case study, modeling with estimation of design characteristics is about 68% slower than without estimation. However, since we perform system modeling at a higher level of abstraction using SystemC, which in turn by some reports allows modeling more than 3 times faster than, e.g., modeling at VHLD level [30], the overall modeling process is performed faster and at a higher abstraction level, than e.g., performing modeling at VHDL level and obtaining design characteristics using a commercial synthesizer.

The presented model estimation framework allows for the early estimation and analysis of the physical design characteristics. Such an analysis can already at the early stage of design provide an answer whether the designed system would match the design constraints. Based on the results of the analysis, the designer can select the system architecture that can lead to a more efficient implementation. Dynamic energy profiling helps to better understand dynamic properties of the designed system, which may be helpful for power optimization of mobile devices.

Future work will address estimation of behavioral SystemC model characteristics, design space exploration by providing estimation of design characteristics for different technological libraries, further development of power estimation models in SystemC to allow more sophisticated power consumption analysis of SystemC models.

References

[1] L. Lavagno, G. Martin, B.V. Selic (Eds.). UML for Real: Design of Embedded Real-Time Systems. *Springer*, 2003.

[2] G. Martin, H. Chang (Eds.). Winning the SoC Revolution: Experiences in Real Design. *Springer*, 2003.

[3] T. Grötker, S. Liao, G. Martin, S. Swan. System Design with SystemC. *Springer*, 2002.

[4] **C. Talarico, J.W. Rozenblit, V, Malhotra, A. Stritter.** A New Framework for Power Estimation of Embedded Systems. *IEEE Computer* 38(2), 2005, 71-78.

[5] **J.A. Darringer, R.A. Bergamaschi, S. Bhattacharya, D. Brand, A. Herkersdorf, J.K. Morrell, I.I. Nair, P. Sagmeister, Y. Shin.** Early analysis tools for system-on-a-chip design. *IBM Journal of Research and Development*, *Vol*.46, *No*.6, 2002, 691-707.

[6] **L. Albanese.** Restoring predictability in SoC integration. *EETimes, 10/11/2004, http://www.eetimes.com /showArticle.jhtml?articleID=49900416.*

[7] **Y. Li, J. Henkel.** A Framework for Estimating and Minimizing the Energy Dissipation of HW/SW Embedded Systems. *Proc. of* 35*th Conference on Design Automation* (DAC 1998), 15-19 *June* 1998, *San Francisco, CA, USA*, 188–193.

[8] **K. Lahiri, A. Raghunathan, S. Dey.** Efficient Power Profiling for Battery-Driven Embedded System Design. *IEEE Trans. on Computer-Aided Design, Vol*.23, June 2004, 919–932.

[9] **J. Rabaey, M. Pedram (Eds.).** Low Power Design Methodologies. *Kluwer Academic Publishers, Norwell, MA*, 1996.

[10] **A. Raghunathan, N.K. Jha, S. Dey.** High-level Power Analysis and Optimization. *Kluwer Academic Publishers, Norwell, MA*, 1998.

[11] **M. Nemani, F.N. Najm.** High-Level Area and Power Estimation for VLSI Circuits, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, *Vol*.18, *No*.6, June 1999, 697-713.

[12] **E. Macii, M Pedram, F. Somenzi.** High-Level Power Modeling, Estimation, and Optimization. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, *Vol*.17, *No*.11, 1061-1079.

[13] **S.M. Kang.** Accurate Simulation of Power Dissipation in VLSI Circuits. *IEEE Journal on Solid-State Circuits*, *Vol*.21, *No*.5, Oct. 1986, 889-901.

[14] **T.H. Krodel.** PowerPlay—Fast, Dynamic Power Evaluation Based on Logic Simulation, *Proc. of IEEE Int. Conf. on Computer Aided Design: VLSI in Computers & Processors* (ICCD 91), 14-16 *October* 1991, *Cambridge, MA, USA*, 96-100.

[15] **S. Ravi, A. Raghunathan, S.T. Chakradhar.** Efficient RTL Power Estimation for Large Designs. *Proc. of* 16*th Int. Conf. on VLSI Design*, 4-8 *January* 2003, *New Delhi, India*, 431-439.

[16] **V. Krishna, N. Ranganathan.** A methodology for high level power estimation and exploration. *Proc. of Great Lakes Symp. on VLSI*, *Feb*. 1998, 420–425.

[17] **A. Raghunathan, S. Dey, N. K. Jha.** Register-transfer level estimation techniques for switching activity and power consumption, *Proc. of Int. Conf. on Computer-Aided Design* (ICCAD 96), *Nov*. 1996, 158–165.

[18] **A. Abrila, H. Mehreza, F. Petrota, J. Gobertb, C. Mirob.** Energy estimation and optimization in architectural descriptions of complex embedded systems. *Proc. of SPIE – Vol.* 5837 *VLSI Circuits and Systems* II, *June* 2005, 456-466.

[19] **D. Brooks, V. Tiwari, M. Martonosi.** Wattch: A Framework for Architectural-Level Power Analysis and Optimizations. *Proc. of* 27*th Int. Symp. on Computer Architectures* (ISCA 2000), 10-14 *June* 2000, *Vancouver, British Columbia, Canada*, 83-94.

[20] **M. Lajolo, A. Raghunathan, S. Dey, L. Lavagno.** Efficient Power Co-Estimation Techniques for System-on-Chip Design. *Proc. of* 2000 *Design, Automation and Test in Europe* (DATE 2000), 27-30 March 2000, Paris, France, pp. 27-34.

[21] ChipVision. *Orinoco: A high-level power estimation and optimization tool suite, http://www.chipvision. com.*

[22] **C. Brandolese, W. Fornaciari, F. Salice.** An area estimation methodology for FPGA based designs at systemc-level. *Proc. of the* 41*th Design Automation Conference* (DAC 2004), *San Diego, CA, USA, June* 7-11, 2004, 129-132.

[23] **N. Bansal, K. Lahiri, A. Raghunathan, S.T. Chakradhar.** Power Monitors: A Framework for System-Level Power Estimation Using Heterogeneous Power Models. *Proc. of Intl. Conf. on VLSI Design*, Kolkata, India, January 2005, 579-585.

[24] **W. Fornaciari, P. Gubian, D. Sciuto, C. Silvano.** Power Estimation of Embedded Systems: A Hardware/Software Codesign Approach. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, *Vol*.6, *No*.2, *June* 1998, 266-275.

[25] **S. Xanthos, A. Chatzigeorgiou, G. Stephanides.** Energy Estimation with SystemC: A Programmer's Perspective. *Proc. of the* 7*th WSEAS Int. Conf. on Systems, Computational Methods in Circuits and Systems Applications*, *WSEAS Press, Corfu, Greece, July* 7-10, 2003, 1-6.

[26] **F. Schirrmeister.** Design for Low-Power at the Electronic System Level. *ChipVision Design Systems*, *White paper*.

[27] **L. Benini, G. De Micheli.** Dynamic Power Management: Design Techniques and CAD Tools. *Kluwer Academic Publishers, Norwell, MA*, 1997.

[28] **F. Najm.** A survey of power estimation techniques in VLSI circuits. *IEEE Transactions on VLSI Systems*, *Vol*.2, *No*.4, *Dec*. 1994, 446-455.

[29] Austriamicrosystems. 0.8 *um CMOS Digital Standard Cell Databook*, *Feb*. 2002.

[30] **M. Steinert, S. Buch.** Using SystemC for Hardware Design Comparison of results with VHDL. *Cossap and CoCentric*, *SNUG Europe* 2002.

Received January 2006.