Creation of ESL Power Models for Communication Architectures using Automatic Calibration

Stefan Schürmans, Diandian Zhang, Dominik Auras, Rainer Leupers, Gerd Ascheid
Institute for Communication Technologies and Embedded Systems
RWTH Aachen University, Germany
{schuerma,zhang,auras,leupers,ascheid}@ice.rwth-aachen.de

Xiaotao Chen, Lun Wang
Huawei Technologies Co., Ltd.
Bridgewater, NJ, USA / Plano, TX, USA
{xiaotaochen,lun.wang}@huawei.com

ABSTRACT

Power consumption is an important factor in chip design. The fundamental design decisions drawn during early design space exploration at electronic system level (ESL) have a large impact on the power consumption. This requires to estimate power already at ESL, which is usually not possible using standard ESL component libraries due to missing power models. This work proposes a methodology that allows extension of ESL models with a power model and to automatically calibrate it to match a power trace obtained by gate-level simulation or measurements. Two case studies show that the methodology is suitable even for complex communication architectures.

Categories and Subject Descriptors

B.8.2 [**Performance and Reliability**]: Performance Analysis and Design Aids; I.6.5 [**Simulation and Modeling**]: Model Development—*Modeling methodologies*

General Terms

Design, Experimentation, Performance

Keywords

Electronic System Level, Power Estimation, Power Model

1. INTRODUCTION

Nowadays, the computational performance of embedded devices is increasing steadily. This increase is enabled by the advances in silicon technology according to Moore's Law [13], which allows to pack an exponentially increasing number of transistors onto a single chip. In recent years, this development has led to Multi-Processor Systems on Chip (MPSoCs), which integrate several processors together with complex communication architectures and large memories on a single piece of silicon. The large number of transistors causes high power consumption, which heats up the chip, drains the battery in mobile devices and has thus become a major design concern.

Exploration of the design options for a future MPSoC usually starts at electronic system level (ESL) using SystemC [6] in order to allow for easy adaptions and fast simulations. Common building blocks are either supplied as black box intellectual property (IP) components by commercial vendors like Synopsys [4] or are available as open source like SoClib [3]. However, those allow only to simulate functionality and timing, but not power consumption. Unfortunately, the fundamental design decisions drawn at ESL have a large impact on the power consumption, which becomes visible only in later design stages when it is very difficult to revise those decisions.

Therefore, it is highly desired to obtain first power consumption estimates during the ESL simulations, even if those are not as accurate as power simulations at later design stages. This work proposes a methodology to extend ESL models with a power model. The state contained in the internals of the ESL model is made available by simple manual instrumentation in the form of state traces, which are used as input to a parametrized power model. Its parameters, called *power state factors*, can be obtained automatically by a process called *calibration*, which takes a power trace and corresponding state traces as input. The output of the calibration process are the power state factors resulting in the best match of the power estimate with the provided power trace. Afterwards, the calibrated power model can be used to estimate the power consumption in different scenarios.

The key differentiator of the proposed methodology is to *simultaneously support the following features*:

- Creation of power models for existing ESL models is **fast**, supported by **automatic tools** and requires only little manual work.
- Both **source-based** models and **black box IP components** are supported.
- Identification of **different power consumption phases** is possible, as the result of the power estimation includes a power trace over time.
- Power models can be created for different types of components, even for **communication architectures**.

- Calibration of power models needs **only a power trace** of a suitable scenario and no details about the hardware structure or the technology library.
- The **ESL simulation speed only drops slightly** when adding power models.

This paper is organized as follows: After discussing related work in section 2, the proposed methodology and the new power model will be introduced in section 3. The calibration process is described in section 4. Section 5 introduces our case studies for ESL power estimation and presents the results. The paper is concluded in section 6.

2. RELATED WORK

Commercial power estimation tools like Synopsys PrimeTime [5] typically operate at post-synthesis or post-layout gate-level and can thus be only used at late stages in MPSoC design. Additionally, the long simulation time is prohibitive for simulation of many options and thus does not allow for a thorough design space exploration.

To alleviate this problem, researchers have raised the abstraction level of power estimation. Early system-level power estimation approaches like [8] were still tailored towards a fixed platform and included manually created power models based on switching activity of RTL signals. Recent tools like Docea Aceplorer [2] have eliminated those limitations and can be used for a wide range of systems. The tool separates the functional description from the power behavior by managing user-defined, high-level power models in parallel to the existing models. This allows to keep the existing design flow for timed functional ESL simulation unchanged.

A comprehensive approach from academia is the ESL framework for rapid prototyping presented in [9]. It supports non-functional system properties like power consumption estimated by pluggable external models for the different system components. Starting from separate application and platform descriptions, it constructs a timing and power aware virtual prototype for analysis of different options in the design space. High flexibility is achieved by only defining interfaces and relying on external tools for creation of the actual timing and power models of the components.

Wattch [7] and SimplePower [18] are popular approaches for creating the required power models for processors. They perform simulations on the architecture level and use the state of the signals in every cycle to drive power models that have been carefully parametrized. The abstraction level can be raised further by running an instruction set simulator and estimating the power consumption just from the executed instructions according to [16]. However, due to inter-instruction effects, a lot of different training instruction sequences have to be analyzed in order to obtain a power model of suitable accuracy.

Functional Level Power Analysis (FLPA, [11], [12]) does not abstract the architecture completely, but avoids the need for simulating all architectural details during power estimation. The power model contains the power consumptions of the different functional units of a processor. Those have been obtained by running training instruction sequences activating only some of the different units and then applying curve fitting in order to obtain a value for each unit. During simulation, the power consumption is derived from the utilization of the units.

All of the approaches discussed so far are processor-centric and will only work for simple buses when applied in a system context. Because modern MPSoCs with a large number of processing elements require complex communication architectures, the power consumption of Networks on Chip (NoCs) is estimated in [14]. A significant increase in accuracy is achieved by using a rate-based model instead of a volume-based one as in earlier works. However, this limits the approach to NoCs and makes it not applicable to other communication architectures like crossbars.

A generalization of FLPA is used to create power models for different types of components in [15]. The code of timed functional models used in common ESL simulations is instrumented to output information about their states to power models. Those power models are created using several manually created training instruction sequences and linear regression. The methodology presented in this paper is also based on the regression approach. In contrast, it does not rely on hand-crafted inputs as it calibrates the power model automatically from a power trace of a known scenario.

The approach from [15] has been extended in [17] to also support black box IP components, i.e. models without access to the source code. As the IP block internals cannot be observed by extension of the code, their inputs and outputs are observed by newly created *estimators*. First, those estimators make the actions on the ports available for power estimation. Second, they can also contain state machines to track the internal state of the IP block, which is not visible from outside the model. Our approach adopts the estimator concept, but generalizes it to be suitable also for complex communication architectures instead of only processors and simple buses. Nevertheless, it does not require manual work during calibration of the power model.

PowerDepot [10] instruments ESL models to export some signals from the ESL simulation for driving a power model. Those signals can be exported directly by an extended ESL model or by a *monitor* observing the ports of an IP block, which is the same concept as the estimator from [17]. The power models are created in a multi-step process from the cells of the standard library and the netlist of the components. This process selects a few hardware signals as *key signals*, which the power model will expect as input. The results obtained by estimation of an H.264 application look very promising, but it is not clear from [10] if this application has also been involved in the characterization of the power model. A major advantage compared to the works discussed before is its capability to output a power trace from the ESL simulation instead of only an average at its end. However, the power model dictates which key signals have to be provided by the ESL model. This limits the approach to models in which information about the state of those signals is available. The methodology presented in this paper does not have this limitation and is still able to generate a power trace at ESL.

3. POWER EXTENSION OF ESL MODELS

For ESL design space exploration, the focus is rather on low setup effort and high simulation speed than on full accuracy. Therefore, the methodology presented in this paper allows to create power models for existing ESL models by little manual instrumentation work and automatic calibration. Instrumentation is used to capture the internal states of the ESL models and to make them available for power estimation. The calibration process creates the parameters of the power model from a reference power trace. Although this work focuses on communication architectures, the methodology is general enough to be applicable to all types of models.

3.1 Information at ESL

ESL models are usually SystemC models written in transaction level modeling (TLM) style, i.e. their internals are mainly modeled using member function calls and member variables. The models can be functional models or pure performance models, which implement only the timing and use dummy data on their ports. However, even a performance model has to implement some control functionality for tracking its state, which is needed for correct timing behavior. In general, ESL models do not contain detailed information about the implementation structure of the component or the type of circuit connected to their outputs, like the length of the wires or the load the component has to drive.

In actual hardware, power consumption is caused by leakage and switching. Leakage is present whenever a circuit is powered and does not depend on its activity. Switching depends on the clock signal for registers and on data signals changing their value along combinational paths. Thus, switching power is mainly influenced by clock gating and activity in different parts of the circuit, which are both dependent on controls signals, i.e. the state of the component.

The actual data signals of the data path also influence switching as different sequences of bits pass through them. However, the control signals have a larger impact on switching than the data bits, because the data bits usually exhibit a nearly fixed switching rate and the control bits determine if data bits are passing through a certain submodule.

In total, the major part of the power consumption depends on the control information, which is available in typical ESL models.

3.2 State Tracing

The state of an ESL model can be hidden within the internal data structures. Manual instrumentation is needed to make the state available for power estimation as depicted in Figure 1, because it is not easily possible to detect the member variables and functions that provide the relevant information using automatic methods. The instrumentation should be simple and have a low overhead at simulation run time. The work presented in this paper uses a singleton class *StateTracker*. It allows to register state traces in the constructor of the model (Listing 1, lines 6-8) and then log state changes at any location in the module code using a single line.

ESL models can contain two types of states: *natural states* and *events*. Natural states, like the activity of a sub-block or the number of pending requests, are usually available in member variables and can be traced by recording the updates (Listing 1, line 14). Events, like data arriving or a register being updated are usually modeled as a function call. As the event models a single-cycle action in hardware, the corresponding state trace has to change from 0 to 1 when the function is called and back to 0 in the next cycle. This can also be accomplished with a single line of code (Listing 1, line 12).

ESL simulation models obtained from commercial ven-

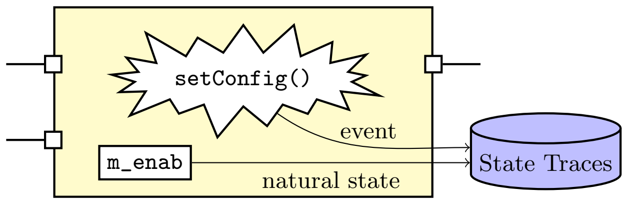

Figure 1: State tracing for ESL module

```
1  class MyModule: public sc_module
2  {
3    MyModule(const sc_module name &name):
4      sc_module(name)
5    {
6      m_tr = StateTracker::get();
7      m_state = m_tr.create(name() + ".enab", 0);
8      m_event = m_tr.create(name() + ".cfg", 0);
9    }
10   void setConfig(bool enab)
11   {
12     m_tr.event(m_event, 1, m_cycle_time, 0);
13     m_enab = enab;
14     m_tr.update(m_state, m_enab ? 1 : 0);
15   }
16 };
```

Listing 1: Instrumentation for tracing a natural state and an event using StateTracker

dors are often delivered as black box IP components. Those blocks can be used in arbitrary ESL simulations, but it is not possible to add state tracing to them, as the source code is not accessible and cannot be instrumented. For those blocks, the approach presented in [17] is chosen. As shown in Figure 2, the activity on all ports is monitored by a small ESL block that just forwards its inputs to its outputs, but additionally traces the states.

If the internals of the IP model involve state information that cannot be tracked at the single ports, a *power state machine* (PSM) is added next to the IP block. This state machine is fed with the observed information and will keep track of the IP block state. A state trace is created for each state of the PSM. The trace is set to 1 while the PSM is in this state and to 0 otherwise.

3.3 Power Model

A linear power model with a constant part is used in the proposed methodology.

Let $\mathbf{s}_i \in \mathbb{N}^T$ for $i \in \{2, \ldots, N\}$ be the state traces provided by an instrumented ESL model during a simulation of T cycles of duration t_{cyc}. Further, let $\mathbf{s}_1 = \mathbf{1}$ be an artificial state trace which is always 1, modeling the constant part of the power consumption, which is caused by leakage, the clock network, etc. For each state trace $\mathbf{s}_i$, a so-called *power state factor* $f_i \in \mathbb{R}$ is defined, which will be determined during the calibration of the power model.

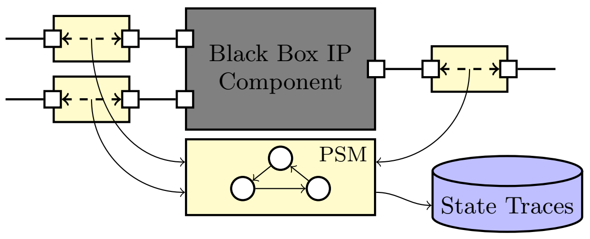

Figure 2: State tracing for IP component: observing ports, PSM to track internal state

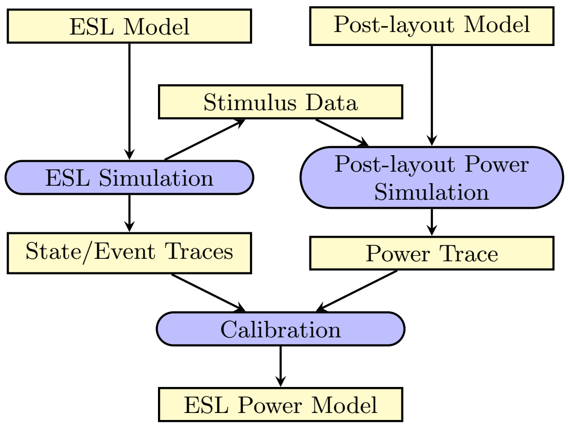

Figure 3: Work flow for ESL power model calibration using post-layout power simulation

The estimated ESL power trace $\mathbf{P}_{\mathrm{est}}$ of the model is then the weighted sum of the state traces. The energy E_{est} and the average power $\overline{P_{\mathrm{est}}}$ can be calculated from it:

$$\mathbf{P}_{\mathrm{est}} := \sum_{i=1}^{N} f_i \mathbf{s}_i \quad E_{\mathrm{est}} := t_{\mathrm{cyc}} \sum_{j=1}^{T} P_{\mathrm{est},j} \quad \overline{P_{\mathrm{est}}} := \frac{1}{T} \sum_{j=1}^{T} P_{\mathrm{est},j}$$

4. AUTOMATIC CALIBRATION

The remaining task in order to use the power model together with an existing ESL model is to determine values for the power state factors f_i. Because the ESL model is abstract, it is not possible to derive the power state factors directly from it. Instead, some information about the power consumption of the component is needed for this purpose.

The methodology presented here relies on a power trace recorded in a reference scenario and uses this for *calibration*. It does not matter if the power trace has been obtained by power simulation at RTL, post-synthesis or post-layout level, by measurements of actual hardware or by any other means. The only requirement is that the reference scenario is known and can also be simulated at ESL in order to obtain the corresponding state traces.

Without loss of generality, the remainder of this section describes calibration to data obtained from post-layout simulations.

4.1 Obtaining a Power Trace using Post-layout Gate-Level Simulation

If an implementation of the component is available, post-layout power simulation can be used to generate the power trace for calibration of the ESL power model. This allows to achieve a high accuracy also for large components, in which wires and the clock network have a big impact on power consumption.

The work flow is depicted in Figure 3. First, the ESL simulation is run using the non-power extended ESL model to obtain both the state traces and the traces of the data on the ports of the ESL model. The port data are used as stimulus data for a post-layout gate-level simulation, whose outputs are processed by a time-based power simulation in order to obtain a cycle-accurate power trace.

4.2 Calculation of Power State Factors

Because the power model is linear, the minimization of the mean square error is a natural approach for calculation of power state factors f_i that provide a best fit of the estimated power $\mathbf{P}_{\mathrm{est}}$ to the reference power trace $\mathbf{P}_{\mathrm{ref}}$ for given state traces $\mathbf{S} = (\mathbf{s}_1 \;\ldots\; \mathbf{s}_N)$:

$$\mathbf{f} := (\mathbf{S}^\top \mathbf{S})^{-1} \cdot \mathbf{S}^\top \mathbf{P}_{\mathrm{ref}}$$

Due to redundancies in the recorded state traces, the matrix $\mathbf{S}^\top \mathbf{S}$ might be singular or unstable (i.e. almost singular), making the inversion impossible or imprecise, respectively. In order to avoid this, some of the state traces have to be excluded. The trace selection is started with just the constant state trace s_1 and then iteratively selects further state traces as long as the matrix does not become singular. If there is some information about the relevance priorities of the state traces for power consumption, it is possible to sort the state traces $\mathbf{s}_2, \ldots, \mathbf{s}_N$ in order of descending relevance before starting the selection.

The calibration will deliver values of the power state factors for selected traces. All values for excluded traces are set to zero. The power model presented in section 3.3 can then be used to estimate the power consumption of other scenarios at ESL. This allows to avoid running the slow power simulations at lower level.

5. CASE STUDIES

The communication architecture is one of the parts of an MPSoC for which power estimation is most difficult, because it contains not only combinational logic and registers like processors or peripherals, but typically also long wires consuming high amounts of switching power. Therefore, two different communication architectures with considerable complexity have been selected as case studies for the ESL power estimation methodology presented here.

As communication architectures cannot be operated properly without connecting them to subsystems, 8 master subsystems and 8 slave subsystems have been modeled on ESL, without including them in the power estimation. The master subsystems consist of processors and local memories while the slave subsystems contain only memories.

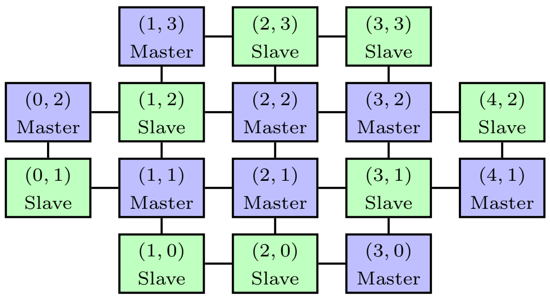

Figure 4: 2D Mesh NoC based system

5.1 Network on Chip

A Network on Chip (NoC) is used as the first case study. It is organized as a semi-regular, 2-dimensional mesh of size 5x4, as shown in Figure 4. Each node contains either a master or a slave subsystem, an NoC router and a network interface connecting the subsystem to the router. Each router has up to 5 ports of width 128 bit with 4 virtual channels (VC), each containing a buffer for 8 flits. The packets are limited to 4 kB of data, contain a 4 bit priority value and are transmitted in flits of 128 bit using wormhole routing according to an adapted x/y-routing scheme. For post-layout simulations used as reference, the NoC has been implemented

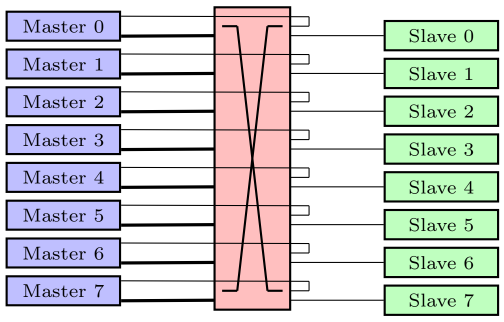

Figure 5: AXI based system

scenario	# active subsys.	traffic per subsys. [MB/s] min.	average	max.
A	16	332	1078	3318
B	16	1296	1296	1296
C	7	55	1345	3988

Table 1: Traffic scenarios: synthetic (A, B), basestation of mobile communication services (C)

at RTL and a layout has been created using a proprietary 65 nm standard-cell library, Synopsys Design Compiler [5] for synthesis and Cadence Encounter [1] for place & route.

The ESL models of the NoC router and the network interface have been modeled in SystemC from scratch. After timing verification against the RTL models, state tracing has been added. The number of buffered flits and the number of pending requests to routing, switch allocation and VC allocation are traced as natural states. Event tracing is used for flits entering and leaving the buffer and for performing routing, switch allocation and VC allocation. The manual one-time instrumentation took approximately two hours.

5.2 AXI Crossbar

An AXI crossbar, which is commercially available in Synopsys DesignWare [4], has been chosen as a second case study. It is configured to have a fully registered data path of 128 bit width and to provide 8 master and 16 slave ports. Figure 5 presents a diagram of the AXI system. Each master subsystem is connected to a master and a slave port, which is being used for remote access to the local memory. Each slave subsystem is connected to a slave port. The layout for AXI has been created using the same 65 nm library and the same tools as the NoC.

At ESL, a black box IP model from Synopsys [5] is used to model the AXI. Because the internals of the IP block are not accessible for state tracing, the approach depicted in Figure 2 has been used. Sending or receiving an address, a data word or a ready flag on a master or slave port is traced as an event. The necessary implementation work was completed within a single work day.

5.3 Results

For evaluation of the ESL power estimation methodology, three periodic traffic scenarios have been simulated for 3 periods of 1.58 ms. The simulations were done on both the NoC and the AXI communication architecture at ESL and post-layout gate-level using a clock frequency of 316 MHz. The ESL simulations have been performed with Synopsys tools [5]: Platform Architect at ESL and VCS/PrimeTime at gate-level. For each case, the ESL power model has been calibrated and was then used for ESL power estimation for

calibration scenario	estimation scenario A	B	C
A	212.17 mW 0.00 %	299.91 mW −4.48 %	168.20 mW −1.05 %
B	222.38 mW 4.81 %	313.96 mW 0.00 %	131.45 mW −21.03 %
C	173.73 mW −18.12 %	296.63 mW −5.52 %	166.45 mW 0.00 %
post-layout	212.17 mW	313.96 mW	166.45 mW

Table 2: ESL power estimation results for NoC

calibration scenario	estimation scenario A	B	C
A	59.35 mW 0.00 %	90.48 mW −3.92 %	63.54 mW −2.44 %
B	55.39 mW −6.67 %	94.17 mW 0.00 %	71.41 mW 9.64 %
C	50.80 mW −14.41 %	89.85 mW −4.59 %	65.13 mW 0.00 %
post-layout	59.35 mW	94.17 mW	65.13 mW

Table 3: ESL power estimation results for AXI

all of the scenarios.

The power estimates $\overline{P_{\text{est}}}$ were compared to the post-layout power consumption $\overline{P_{\text{ref}}}$ and the relative error e was calculated:

$$e := \frac{\overline{P_{\text{est}}} - \overline{P_{\text{ref}}}}{\overline{P_{\text{ref}}}} \cdot 100\,\%$$

The results for the NoC are shown in Table 2 and for AXI in Table 3. It can be seen that the overall maximum error is 21 %. This accuracy allows to **substitute gate-level power estimation with much faster ESL power estimation** during design space exploration. Thus, either more configurations can be simulated or the exploration is finished earlier.

When comparing the accuracies achieved by different calibration scenarios, it shows that scenario A leads to power models resulting in estimation errors below 5 %, which is remarkable considering the large difference of abstraction levels between post-layout gate-level and ESL. The reason is that this scenario is suited very well for the calibration process. First, it uses all subcomponents of the communication architectures. This allows to calibrate the power state factors for all of them. Second, the traffic produced by the subsystems is different enough to avoid a high correlation between the state traces. This allows the regression during calibration to work well.

The post-layout and ESL power traces are plotted in Figure 6. To facilitate readability, their resolution has been reduced to 5 k cycles by averaging. The plots show that the proposed methodology is able to **predict the phases of different power consumption** correctly. This is extremely beneficial for design space exploration, as the designer is enabled to identify which computations cause the highest power consumption and focus on those during optimization.

To investigate the tradeoff between power estimation accuracy and simulation speed, the execution times of the post-layout power simulations and the ESL simulations with and without power estimation have been measured. The numbers in Table 4 show that the ESL simulations are only slightly slower for AXI when including ESL power estima-

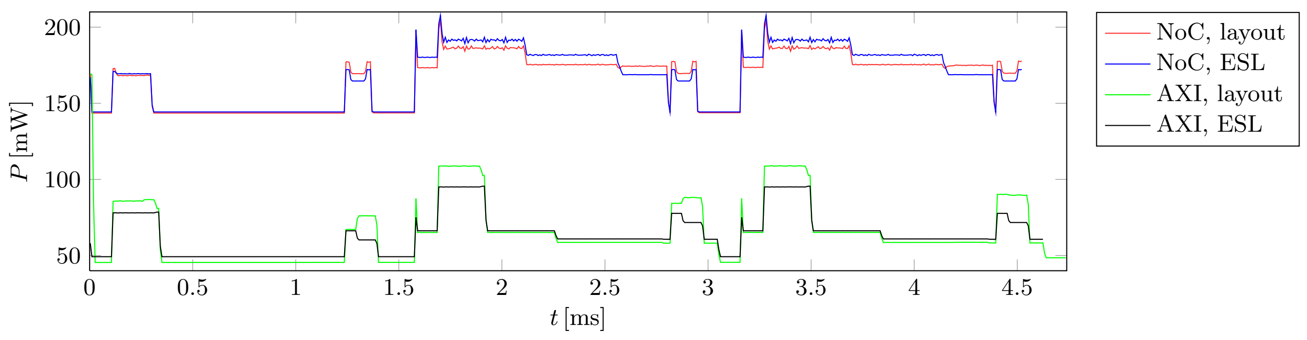

Figure 6: Power traces of scenario C (calibration of ESL model using scenario A)

system & scenario		post-layout power	ESL sim.	ESL & power	speed-up vs. layout	overhead vs. ESL sim.
NoC	A	45 h	101 s	274 s	591	2.71
	B	91 h	152 s	479 s	684	3.15
	C	34 h	39 s	107 s	1144	2.74
AXI	A	38 h	166 s	196 s	698	1.18
	B	75 h	167 s	195 s	1385	1.17
	C	27 h	108 s	124 s	784	1.15

Table 4: Speedup of ESL power estimation vs. post-layout power estimation, and overhead of ESL power estimation vs. ESL simulation

tion. The slowdown of about factor 3 for NoC is mainly caused by disk I/O for writing the traces for each NoC component. Compared to time-based post-layout power simulation a speedup of factor 880 is achieved on average.

6. CONCLUSIONS

The presented methodology for ESL power estimation extends available ESL models with power models, which are automatically calibrated to match power traces obtained by low-level power simulation or measurements. This enables inclusion of the important design criterion of power consumption in early design space exploration and thus development of more power-efficient systems. Two case studies have shown the applicability of the methodology to complex communication architectures. Depending on the calibration scenario, the accuracy ranges between 5 % and 21 % compared to post-layout power simulations while achieving a gain of multiple orders of magnitude in simulation time.

Currently, the quality of the generated power model depends on the calibration scenario. This requires a subsequent validation of the model. Further, the proposed methodology has only been applied to communication architectures yet, although it was created with applicability to all kinds of models in mind. An evaluation for other components like processors, accelerators and peripherals will be performed in the future.

7. REFERENCES

[1] Cadence digital implementation. [Online] `http://www.cadence.com/products/di/` (accessed 11/2012).

[2] Docea Aceplorer. [Online] `http://www.doceapower.com/products-services/aceplorer.html` (accessed 11/2012).

[3] SoClib. [Online] `http://www.soclib.fr` (accessed 11/2012).

[4] Synopsys IP. [Online] `http://synopsys.com/IP` (accessed 11/2012).

[5] Synopsys tools. [Online] `http://synopsys.com/Tools` (accessed 11/2012).

[6] SystemC. [Online] `http://www.accellera.org/downloads/standards/systemc` (accessed 11/2012).

[7] D. Brooks, V. Tiwari, and M. Martonosi. Wattch: A framework for architectural-level power analysis and optimizations. In *Proceedings of the 27th Annual International Symposium on Computer Architecture*, ISCA '00, New York, NY, USA, 2000. ACM.

[8] W. Fornaciari, P. Gubian, D. Sciuto, and C. Silvano. Power estimation of embedded systems: A hardware/software codesign approach. In *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, volume 6, Jun. 1998.

[9] K. Gruttner, K. Hylla, S. Rosinger, and W. Nebel. Towards an ESL framework for timing and power aware rapid prototyping of HW/SW systems. In *Specification Design Languages (FDL 2010), 2010 Forum on*, Sep. 2010.

[10] C.-W. Hsu, J.-L. Liao, S.-C. Fang, C.-C. Weng, S.-Y. Huang, W.-T. Hsieh, and J.-C. Yeh. Power depot: Integrating IP-based power modeling with ESL power analysis for multicore SoC designs. In *Proceedings of the 48th Design and Automation Conference*, ACM, New York, NY 10121, Jun. 2011. ACM.

[11] N. Julien, J. Laurent, E. Senn, and E. Martin. Power consumption modeling and characterization of the TI C6201. *Micro, IEEE*, 23(5), Sep. 2003.

[12] J. Laurent, N. Julien, E. Senn, and E. Martin. Functional level power analysis: An efficient approach for modeling the power consumption of complex processors. In *Proceedings of the Conference on Design, Automation and Test in Europe*, DATE '04, Washington, DC, USA, 2004. IEEE Computer Society.

[13] G. E. Moore. Cramming more components onto integrated circuits. *Electronics*, 38(8), Apr. 1965.

[14] L. Ost, G. Guindani, F. Moraes, L. Indrusiak, and S. Määttä. Exploring NoC-based MPSoC design space with power estimation models. *IEEE Design and Test*, 28, Mar. 2011.

[15] S. K. Rethinagiri, R. ben Atitallah, and J.-L. Dekeyser. A system level power consumption estimation for MPSoC. In *2011 International Symposium on System on Chip*. IEEE, Nov. 2011.

[16] V. Tiwari, S. Malik, and A. Wolfe. Power analysis of embedded software: A first step towards software power minimization. *Very Large Scale Integration Systems, IEEE Transactions on*, 2(4), Dec. 1994.

[17] C. Trabelsi, R. Ben Atitallah, S. Meftali, J.-L. Dekeyser, and A. Jemai. A model-driven approach for hybrid power estimation in embedded systems design. *EURASIP Journal on Embedded Systems*, 2011.

[18] W. Ye, N. Vijaykrishnan, M. Kandemir, and M. J. Irwin. The design and use of SimplePower: A cycle-accurate energy estimation tool. In *Proceedings of the 37th Annual Design Automation Conference*, DAC '00, New York, NY, USA, 2000. ACM.