ESL Power Estimation using Virtual Platforms with Black Box Processor Models

Stefan Schürmans, Gereon Onnebrink, Rainer Leupers, Gerd Ascheid
Institute for Communication Technologies and Embedded Systems
RWTH Aachen University, Germany
Email: {schuerma,onnebrink,leupers,ascheid}@ice.rwth-aachen.de

Xiaotao Chen
Huawei Technologies Co., Ltd.
Bridgewater, NJ, USA
Email: xiaotaochen@huawei.com

***Abstract*—Processor models for electronic system level (ESL) simulations are usually provided by their vendors as binary object code. Those binaries appear as black boxes, which do not allow to observe their internals. This prevents the application of most existing ESL power estimation methodologies. To remedy this situation, this work presents an estimation methodology for the case of black box models. The evaluation for the ARM Cortex-A9 processor shows that the proposed approach is able to achieve a high accuracy. In comparison to hardware power measurements obtained from the OMAP4460 chip on the PandaBoard, the ESL estimation error is below $5\,\%$.**

I. Introduction

Power consumption has become a very important metric for the design of electronic systems. According to [25], the design decisions made at early stages of design, i.e. at electronic system level (ESL), typically have a higher impact on this metric than the decisions made at later stages. For example, the selection of processor types and interconnect architectures has a high influence on the power consumption. Once this selection is made, further design stages are performed at lower abstraction levels lacking the broad view required for revising the decisions. Consequently, power consumption improvements focus on details and thus their effect is limited.

Classical industry standard tools for design space exploration at ESL, like SystemC [4] or Synopsys Platform Architect [3], model only the functionality and timing of the components and lack information about the power consumption. It is desirable to extend all models in the ESL library with power models. This would allow to predict the power consumption during early design space exploration at ESL.

Several efforts have been made in academia and industry to create ESL power models. Because processor cores are the most complex building blocks of systems as well as the major contributors to the overall power consumption, most work has focused on power estimation for processor cores. Almost all of those approaches require either deep insight into the internals or manual annotation of the ESL model with power consumption information.

Modern ESL processor models are usually provided by their vendors as proprietary binary object code, which is basically a black box that does not allow observation of the internal details. This prohibits application of the existing tool-supported approaches for creating ESL power models. To alleviate the situation, a different method is suggested in this work. It allows to create power models for non-cycle-accurate black box processor models based on a few reference power curves.

The contributions of this work[1] are as follows:

- Efficient tracing of black box processor models to derive sufficient information for ESL power estimation.
- Enabling calibration-based ESL power estimation supporting non-cycle-accurate simulators and reference power curves obtained by hardware measurements.
- Evaluation of the ESL black box power estimation method for ARM Cortex-A9 processors.

II. Related Work

Power estimation at levels above register transfer level (RTL) has been investigated by academia for several years. Early system level approaches like [11] were strongly inspired by RTL power estimation approaches and tried to predict switching activity of RTL signals from ESL simulations. There are also recent works like [15] that rely on the RTL code. However, application of such methods is not possible if the RTL code is not available.

One of the first works that drop the dependency on RTL completely is [8]. More recent approaches, e.g. [23] and [13], focus on the integration of various ESL power models into the workflow, while other approaches also provide a recipe of how to create the power models, e.g. [19]. Commercial frameworks like Mentor Graphics Vista [25] and Docea Aceplorer [1] show that there is also a growing demand for ESL power estimation in industry.

Another group of works from academia has investigated the creation of power models suitable for ESL simulations for different kinds of system components, e.g. dynamic memories (DRAMs) [16] or peripheral cores [12]. A method targeted at communication architectures in general is proposed in [21]. Its approach of *calibrating* a linear power model to a reference power trace is reused in this work.

Processors are the main elements of electronic systems and have therefore attracted a lot of attention also regarding high level power models. Early approaches have started to estimate the power from information about the executed instructions. Examples are SimplePower [28], Wattch [9] and derivatives like [10]. The estimation accuracy has been improved by including information about the functional units, e.g. in [18].

[1]This work has been supported by Huawei Technologies Co., Ltd.

978-1-4673-7311-1/15/$31.00 ©2015 IEEE

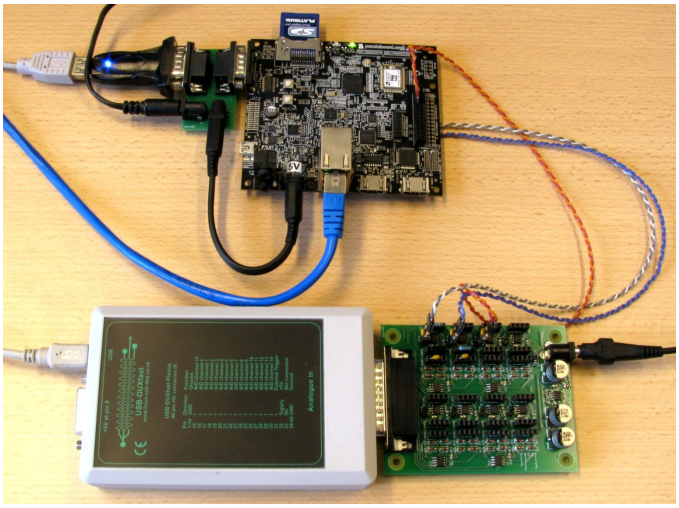

Fig. 1: PandaBoard and power measurement setup using custom voltage converter board and USB-DUXfast

All of the above approaches depend either on manual creation of a power model based on information from a detailed data sheet or on observation of the internals of the processor model. This is not possible for most modern processor simulators. In contrast, this work shows how to estimate power consumption using an instruction-accurate black box processor model from Open Virtual Platforms (OVP) [2], which does not allow to observe its internals.

III. Reference System

The PandaBoard [7] features a Texas Instruments OMAP4460 chip [6], which contains an ARM Cortex-A9 subsystem with a separate power domain. This subsystem has been chosen for reference timing and power measurements.

The ARM subsystem is based on two ARM Cortex-A9 cores clocked at 1.2 GHz. Both cores have separate level 1 (L1) caches for instruction and data. All L1 caches are 4-way associative and have a capacity of 32 kB and a line size of 32 B. The two data caches are kept coherent by the so-called *Snoop Control Unit* (SCU). In contrast, the instruction caches are not participating in the coherency protocol, as those are only read and never written.

All L1 caches are connected via a shared level 2 (L2) cache of size 1 MB to the DRAM interface and the global interconnect of the OMAP4460. The DRAM memory interface provides access to a 1 GB LPDDR2 (low-power, double data rate, version 2) memory. The global interconnect allows to access the peripherals, for example the Universal Asynchronous Receiver and Transmitter (UART), the Ethernet interface and the general purpose input/output (GPIO) pins.

A. Power Measurement

The ARM Cortex-A9 power domain is driven by a separate switched mode power supply (SMPS) located on the bottom side of the PandaBoard, directly under the OMAP4460 chip. The wires from its output to the power input of the OMAP4460 are not easily accessible. Therefore, the power is measured at the input of this SMPS, although this might introduce a small measurement error, as the efficiency of the SMPS is less than 100 % and might vary with its load.

The data logger USB-DUXfast [5] is used to record the voltage and the voltage drop over a shunt at the SMPS input with a resolution of 12 bit and a sampling frequency of 5 kHz. A custom circuit board converts the voltage levels to the valid range before feeding them to the data logger. The entire setup is shown in Figure 1. The power consumption is computed from the measured values in software on a PC. Additionally, the state of a GPIO pin is recorded to measure the start time and the end time of the `main()` function of the benchmark.

B. Benchmarks

The benchmarks used for creation and evaluation of the power models have been taken from four different sources. The first group are standard benchmarks and standard benchmark suites: Dhrystone [27], LTE uplink receiver PHY benchmark [22] (abbreviated `lte-bench` in this paper), telecomm package of MiBench [14] (`mib/t`), StreamIt [24] (`stit`) and WiBench [29] (`wib`).

Most of the standard benchmarks perform floating point computations. In order to create benchmarks putting load on the integer functional units of the processor, some of the benchmarks have been converted to fixed-point integer operations. The resulting benchmarks form the second group. They are named like the floating-point versions, but with `_int` appended.

The third group of benchmarks is used to investigate the differences between load on one and two cores, while keeping communication between the two cores at a minimum. Those benchmarks have been created by running a benchmark from the first two groups on both cores. The original benchmark is first run on both cores at the same time and then two times on the first core while the second core is idle. This sequence is iterated three times. The names of the resulting benchmarks are formed by appending `2co` to the name of the original benchmark.

In-house parallel benchmarks form the fourth group. They make use of multiple threads communicating to each other. In contrast to the third group, this causes communication between both cores. The names of those benchmarks starts with `mt`.

The benchmarks are run in *bare-metal* mode, i.e. without an operating system. The runtime environment for C applications is based on the GNU C Compiler [26] version 4.8.1, binutils version 2.23.2 and newlib version 2.0.0 and does not provide support for the vector coprocessor. Reading from files is emulated using static arrays included in the application binary. Data written to standard output is buffered in memory and transmitted using the UART on benchmark end. A minimal cooperative scheduler has been implemented for executing multithreaded applications.

IV. Virtual Platform

This work investigates the accuracy of ESL power estimation using black box processor models. Therefore, the ESL simulator has been based on an instruction-accurate black box ARM Cortex-A9 processor model from OVP [2]. This model is a typical example of proprietary models that are delivered

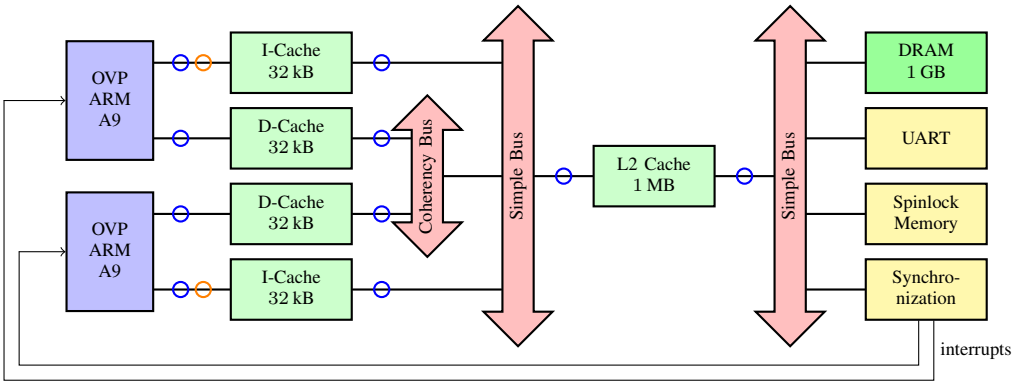

Fig. 2: Virtual platform for the OMAP4460 ARM Cortex-A9 subsystem on the PandaBoard, Blue circles indicate the locations of full TLM tracing, Orange circles indicate locations of simple activity tracing

as a binary object containing all the functionality plus some limited interface source code for integrating it into a SystemC simulation.

Based on the OVP ARM model, a virtual platform replicating the OMAP4460 ARM subsystem from the PandaBoard has been created using Transaction Level Modeling (TLM) 2.0 and SystemC 2.3 [4]. Figure 2 shows a block diagram of the VP. All connections except for the interrupt wires are modeled using blocking TLM transactions.

The data caches (D-Cache) and the *Coherency Bus* have been written from scratch in order to model the coherent data caches and the SCU of the OMAP4460. They make use of TLM extensions to implement the cache coherence protocol including direct cache-to-cache transfer, as it is used in the real hardware. In contrast, the instruction caches, the L2 cache, the simple buses and the UART were taken from an in-house component model library. The DRAM model is basically a simple TLM memory model, but it accounts for additional delay times for page switches and for switching from write to read. There are more detailed approaches like [17], but a high degree of abstraction is sufficient for our purpose. The spinlock memory is only a simulation vehicle used to implement spinlocks on the VP, as the load-link and store-conditional instructions are not supported by the OVP ARM model and the TLM buses. The synchronization block is used to wake up cores from the sleep state.

Although the source code of all models except for the processor models is available, all models are treated as black boxes. This means that the inner details are not observed or modified. Only constructor parameters are used to configure their timing, and only their ports are observed for power estimation.

V. Timing Annotation

The timing of the VP has to approximate the real hardware timing as much as possible in order for ESL power estimation to be feasible. The reason is that the timing is used to correlate processes in the VP to the corresponding part of the power consumption curve. Therefore, the VP timing annotations have been tuned to make the benchmark execution times match those on the reference hardware. The left column of Table I shows the parameters available for timing annotation.

timing parameter	value
processor instruction	465 ps
additional delay for MUL/MLA	586 ps
additional delay for LDRH/LDRB	1172 ps
delay of buses	0
L1 instruction cache read	275 ps
L1 data cache read	8 ps
L1 data cache write	0
L2 cache read	20 ns
L2 cache write	0
DRAM read	41.7 ns
DRAM write	0
DRAM write/read switch	13.3 ns
DRAM page switch	66.7 ns

TABLE I: Timing annotations of the virtual platform

First, the timing of the core and the L1 instruction cache has been adapted using small loops of simple instructions not accessing the L1 data cache. Once their timing was acceptable, benchmarks issuing reads and writes to the L1 data cache have been run. The L1 data cache parameters were tweaked until the timing of the new benchmarks were acceptable. Next, the L2 cache was included in the benchmarks. This allowed to find suitable timing parameters for the L2 cache. Finally, the DRAM timing could be determined using benchmarks also using the DRAM. Once initial values for all parameters were available, a refinement of the timing has been done using the benchmarks described in Section III-B. One or two timing parameters were changed at a time until the best configuration was found. This process has been iterated for all parameters several times.

The resulting timing annotations are listed in Table I. The processor instruction time is a lot shorter than the 833 ps cycle time of the real hardware processor. This compensates the effect that the real ARM Cortex-A9 is superscalar out-of-order and thus is able to execute multiple instructions at the same time while the model executes exactly one instruction per cycle. As some instructions require more time on the real processor an extra delay is introduced when a fetch transaction indicates a MUL, MLA, LDRH or LDRB instruction, following the idea from [20].

The bus timing and the write times are set to zero, which reflects the fact that the real hardware provides sufficiently sized write buffers that are able to hide almost all write latencies. The read times are non-zero, but the L1 data read

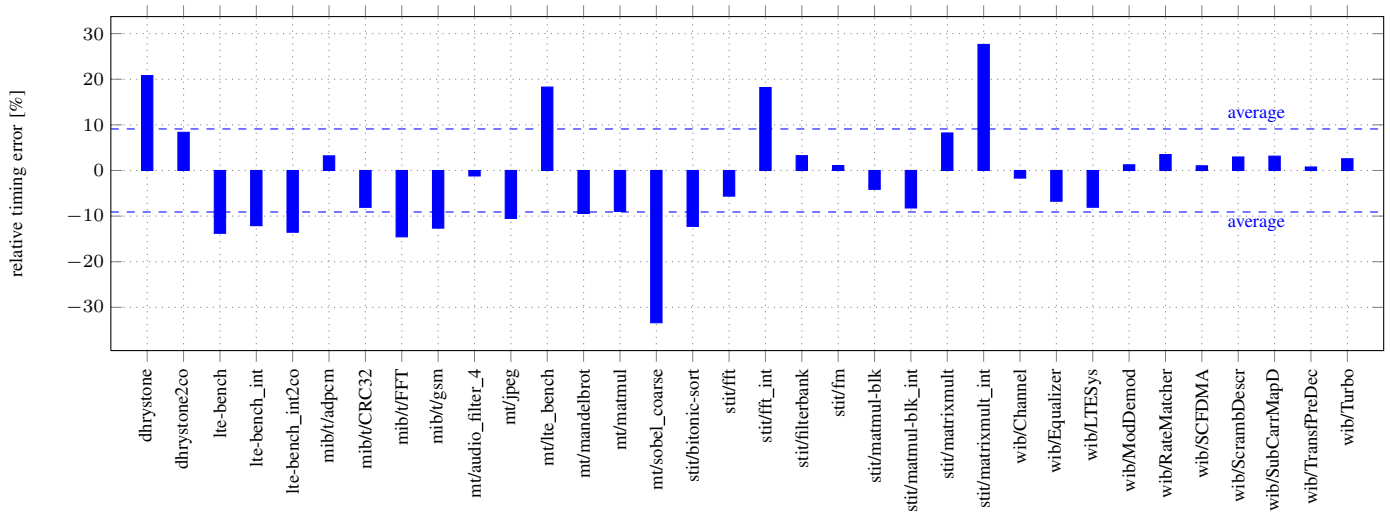

Fig. 3: Relative timing error of the virtual platform

time is very short, probably due to the out-of-order feature hiding many read latencies. Finally, the DRAM model timing introduces extra delays for page switches and for switching from write to read.

Figure 3 shows the relative timing errors of the final virtual platform. It can be seen that an average timing error of only $9.1\,\%$ was achievable even using an instruction-accurate processor model and blocking TLM transactions. The maximum timing error was only $33.4\,\%$.

VI. Power Estimation

The ESL power estimation methodology proposed in [21] records N traces of T cycles duration from an ESL simulation as a matrix $\mathbf{S} \in \mathbb{N}^{T\times N}$. The first trace is always 1 (i.e. $\forall t:\ s_{t,1} = 1$) in order to model the constant part of the power consumption. The power estimate is then calculated from the traces as follows:

$$\mathbf{P}_{\text{est}} = \mathbf{S} \cdot \mathbf{a} \quad \text{with} \quad \mathbf{a} \in \mathbb{R}^N \tag{1}$$

The factor vector $\mathbf{a}$ is determined by *calibration*: a reference power trace $\mathbf{P}_{\text{ref}} \in \mathbb{R}^T$ has to be obtained for one scenario. The methodology assumes that an RTL or gate-level implementation is available for this calibration scenario and a lower-level simulation can be used to obtain the reference power trace.

As the estimate should match the reference as closely as possible, the factor vector $\mathbf{a}$ can be computed using the pseudo-inverse matrix $\mathbf{S}^+$:

$$\mathbf{a} := \mathbf{S}^+ \cdot \mathbf{P}_{\text{ref}} \tag{2}$$

The power estimation approach proposed in this work reuses the linear power model and the approach of calibration. However, it supports non-cycle-accurate ESL simulations based on black-box models as well as reference power traces obtained from hardware measurements. The requirement for cycle-accurate traces is dropped. Instead, sampled traces will be used in the following. This leads to a lower number of entries T in the traces. To compensate for this, multiple reference scenarios are used for calibration by concatenating their traces.

A. Black Box Tracing

The virtual platform described in Section IV is instrumented to record traces. Because the processor model is a black-box model and all the other models are treated as black-boxes, the traces are recorded on the TLM connections between the models, as indicated by the blue and orange circles in Figure 2. The connections to the DRAM and to the peripherals are not traced, as those components are not part of the ARM subsystem and their power consumption is out of scope for this work.

The reference power trace to be used in calibration provides one sample value per $0.2\,\text{ms}$. Therefore, the *TLM traces* are also recorded with a resolution of $t_{\text{samp}} = 0.2\,\text{ms}$ in simulated time. The read and write transactions passing a TLM connection are counted as well as the extended TLM transactions used to model the coherence protocol. Every $0.2\,\text{ms}$ of simulated time, the counters are written to the trace file and are then reset for the next interval.

Besides the TLM traces, a second, even more abstract tracing approach has been applied: the so-called *activity traces*. For those traces, only the instruction ports of the processor models are instrumented. A single trace is recorded for each processor. This trace is set to zero when the processor fetches a wait instruction, i.e. when it is halted. On fetch of any other instruction type, the processor is assumed to be active again and the trace is set back to one. In total, the activity traces consists of three traces: the constant 1 trace plus an activity trace for each processor.

The timing overhead imposed on the simulation by the tracing is marginal, due to the few instrumentation points and the sampling intervals being large compared to the cycle times of the processor models. Exact numbers cannot be given here, because the license of the OVP processor model does not permit to publish simulation performance data.

B. Timing Mismatch Compensation

Two effects cause the timing of the traces from the ESL simulation not to match the reference power traces. First, the timing error discussed in Section V leads to different lengths

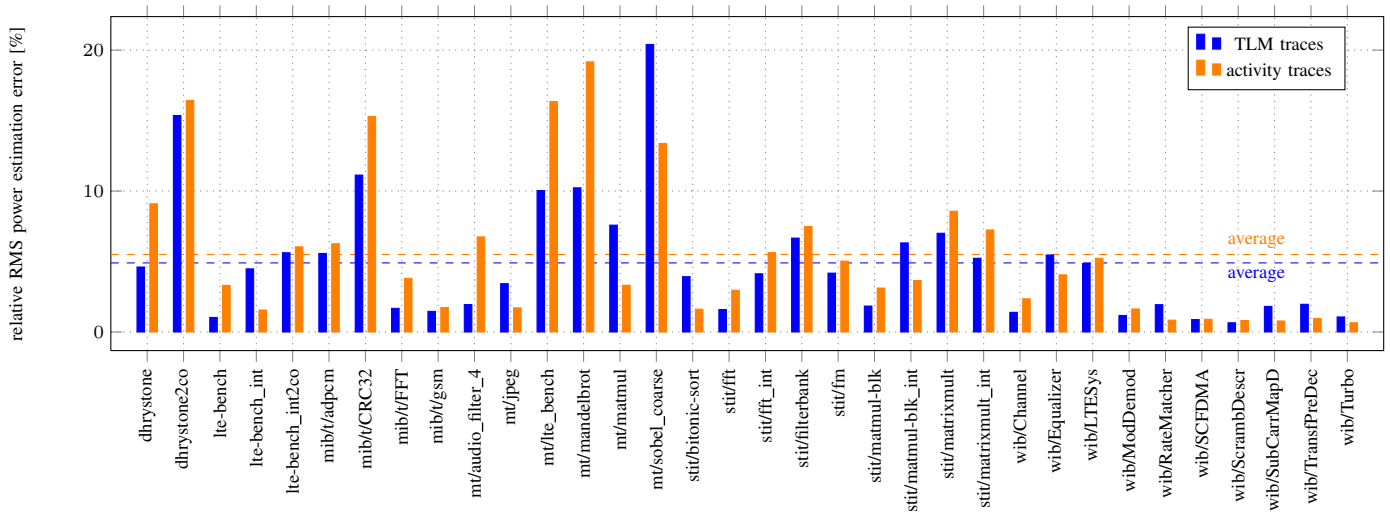

Fig. 6: Relative RMS power estimation error

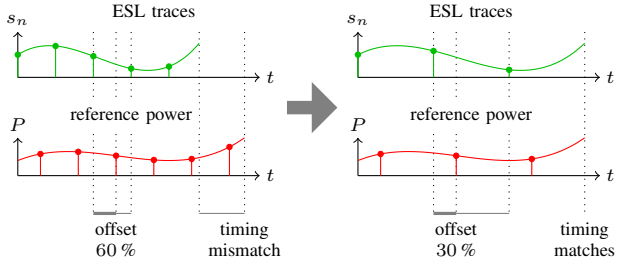

Fig. 4: Compensation of timing mismatch by scaling and reduction of relative timing offset by sample rate reduction (shown for $k = 2$)

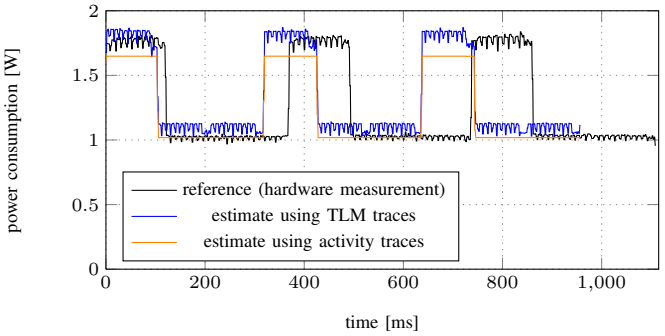

Fig. 5: Estimated power consumption curves and reference power curve for `lte-bench_int2co`

of both traces. Second, the synchronization of the hardware power consumption curve to the ESL traces may introduce a time offset of up to one sampling period: The samples of the ESL traces are recorded at precisely known simulation times $t \cdot t_{\text{samp}}$ with $t \in \mathbb{N}$. The power consumption sampling of the hardware starts when the power of the board is turned on, but due to bootloader and benchmark transfer to the board, the benchmark code starts at a later time t_{start}. There is a t with $t \cdot t_{\text{samp}} < t_{\text{start}} \leq (t+1) \cdot t_{\text{samp}}$. Thus, the change of the GPIO pin signaling the start of the benchmark will be detected at the following sample time, i.e. with an offset of $0 < t_{\text{ofs}} \leq t_{\text{samp}}$. Both effects are shown in the left part of Figure 4.

The calibration approach requires the number of samples in the ESL traces to match the number of samples in the power consumption curve exactly. Therefore, the timing error has to be compensated. This is possible by resampling the ESL traces, which can be interpreted as a scaling in time. It will ensure that the end of the benchmark in the ESL simulation is calibrated to the end of the benchmark on the real hardware.

The timing offset t_{ofs} is bounded by the sample time t_{samp}, but its exact value is unknown. If it is close to zero, it will not affect calibration. If it is close to the sample time, power sample t will almost correspond to ESL sample $t + 1$ and calibration will result in an inaccurate power model. To alleviate this effect, the sampling rate is reduced by a factor k. This does not change the absolute offset t_{ofs}, but compared to the new sampling time $k \cdot t_{\text{samp}}$, the relative offset is limited to $\frac{1}{k}$. The factor $k := 10$ has been chosen for this work, limiting the relative offset to 10 %.

After compensating the timing mismatch and offset, the original calibration approach according to Formula 2 is used to create the power model.

C. Power Estimation Results

In order to maximize the number of test cases while providing sufficient input to calibration, the evaluation of the extended power estimation methodology has been performed using leave-one-out cross-validation. This means that the power model used for estimating a benchmark has been created using all other benchmarks for calibration.

An example of estimated power curves is shown in Figure 5. It can be seen that both estimates follow the shape of the reference curve closely, but the timing error leads to an estimate scaled in time. Besides this timing effect, the magnitude of the power estimate matches the reference very closely when using the TLM traces. Using the simple activity traces fails to predict the fine substructure of the power consumption, but predicts the main phases correctly.

Figure 6 shows the resulting black box ESL power estimation errors for both variants of ESL tracing. The errors are computed as relative root-mean-square (RMS) value of the

difference between estimate and reference:

$$e_{\mathrm{rel}} = \frac{\sqrt{\frac{1}{T} \cdot \sum_{t=1}^{T} \left(P_{\mathrm{est},t} - P_{\mathrm{ref},t}\right)^2}}{\overline{P_{\mathrm{ref}}}} \tag{3}$$

To enable the above power comparison, the timing error has to be compensated by scaling the estimated curve in time to match the length of the reference curve, as it is done for calibration. Using the RMS value provides strong evidence that the estimate is always close to the reference.

The black box TLM traces result in a maximum error of $20.4\,\%$ and an average error of only $4.9\,\%$. The even more abstract activity tracing method shows an average error of $5.5\,\%$, which is just slightly larger. The maximum error of $19.2\,\%$ is even lower than for the TLM traces.

VII. Conclusions

A method for ESL power estimation using virtual platforms based on instruction-accurate black box processor models has been created based on the calibration method from [21], which is originally targeted at cycle-accurate ESL platforms. The presented case study of the ARM Cortex-A9 subsystem has shown that black box ESL power estimation can provide estimates with $4.9\,\%$ average error compared to hardware measurements. Even just tracing the activity of each core in a single trace allows to estimate the power consumption with an average error of $5.5\,\%$.

The evaluation using a general purpose ARM processor does not permit general statements about all processor types. Further research is required to investigate the applicability of the proposed approach to different processor types, e.g. complex digital signal processors.

References

[1] Docea Aceplorer. [Online] `http://www.doceapower.com/products-services/aceplorer.html` (accessed 11/2014).

[2] Open virtual platforms. [Online] `http://ovpworld.org` (accessed 11/2014).

[3] Synopsys tools. [Online] `http://synopsys.com/Tools` (accessed 11/2014).

[4] SystemC. [Online] `http://www.accellera.org/downloads/standards/systemc` (accessed 11/2014).

[5] USB-DUXfast: Technical specification. [Online] `http://www.linux-usb-daq.co.uk/tech2_duxfast/` (accessed 11/2014).

[6] OMAP4460 multimedia device silicon revision 1.x technical reference manual. [Online] `http://www.ti.com/product/ omap4460` (accessed 11/2014), 2011.

[7] OMAP4460 Pandaboard ES system reference manual. [Online] `http://pandaboard.org/content/resources/references` (accessed 11/2014), 2011.

[8] L. Benini, R. Hodgson, and P. Siegel. System-level power estimation and optimization. In *Intl. Symposium on Low Power Electronics and Design*, ISLPED '98. ACM, 1998.

[9] D. Brooks, V. Tiwari, and M. Martonosi. Wattch: A framework for architectural-level power analysis and optimizations. In *27th Intl. Symposium on Computer Architecture*, ISCA '00. ACM, 2000.

[10] L. Eeckhout and K. D. Bosschere. Early design phase power/performance modeling through statistical simulation. In *Intl. Symposium on Performance Analysis of Systems and Software*. IEEE, 2001.

[11] 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*, 1998.

[12] T. Givargis, F. Vahid, and J. Henkel. Instruction-based system-level power evaluation of system-on-a-chip peripheral cores. *Very Large Scale Integration (VLSI) Systems*, 2002.

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

[14] M. R. Guthaus, J. S. Ringenberg, D. Ernst, T. M. Austin, T. Mudge, and R. B. Brown. MiBench: A free, commercially representative embedded benchmark suite. In *Workload Characterization, WWC-4*. IEEE CS, 2001.

[15] 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 *48th Design and Automation Conference*. ACM, 2011.

[16] M. Jung, C. Weis, P. Bertram, and N. Wehn. Power modelling of 3D-stacked memories with TLM2.0 based virtual platforms. In *Synopsys User Group Conference (SNUG)*, 2013.

[17] M. Jung, C. Weis, N. Wehn, and K. Chandrasekar. TLM modelling of 3D stacked wide I/O DRAM subsystems: A virtual platform for memory controller design space exploration. In *Workshop on Rapid Simulation and Performance Evaluation: Methods and Tools*. ACM, 2013.

[18] 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 *Design, Automation and Test in Europe*, DATE '04. IEEE CS, 2004.

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

[20] F. Rosa, L. Ost, R. Reis, and G. Sassatelli. Instruction-driven timing CPU model for efficient embedded software development using OVP. In *Electronics, Circuits, and Systems (ICECS)*, 2013.

[21] S. Schürmans, D. Zhang, D. Auras, R. Leupers, G. Ascheid, X. Chen, and L. Wang. Creation of ESL power models for communication architectures using automatic calibration. In *50th Design Automation Conference*, DAC '13. ACM, 2013.

[22] M. Själander, S. McKee, P. Brauer, D. Engdal, and A. Vajda. An LTE uplink receiver PHY benchmark and subframe-based power management. In *Performance Analysis of Systems and Software (ISPASS)*. IEEE, 2012.

[23] M. Streubühr, R. Rosales, R. Hasholzner, C. Haubelt, and J. Teich. ESL power and performance estimation for heterogeneous MPSoCs using SystemC. In *Forum on Specification and Design Languages (FDL)*, 2011.

[24] W. Thies, M. Karczmarek, and S. Amarasinghe. StreamIt: A language for streaming applications. In *Intl. Conference on Compiler Construction*, Grenoble, France, 2002.

[25] Y. Veller and S. Matalon. Why you should optimize power at the ESL. [Online] `http://go.mentor.com/cvtq` (accessed 11/2014), 2010.

[26] W. Von Hagen. *The definitive guide to GCC*. Apress, 2006.

[27] R. P. Weicker. Dhrystone: A synthetic systems programming benchmark. *Comm. ACM*, 1984.

[28] W. Ye, N. Vijaykrishnan, M. Kandemir, and M. J. Irwin. The design and use of SimplePower: A cycle-accurate energy estimation tool. In *37h Design Automation Conference*, DAC '00. ACM, 2000.

[29] Q. Zheng, Y. Chen, R. Dreslinski, C. Chakrabarti, A. Anastasopoulos, S. Mahlke, and T. Mudge. WiBench: An open source kernel suite for benchmarking wireless systems. In *Workload Characterization (IISWC)*, 2013.