Power Monitors: A Framework for System-Level Power Estimation Using Heterogeneous Power Models

Nikhil Bansal Kanishka Lahiri Anand Raghunathan Srimat T. Chakradhar
{nbansal,klahiri,anand,chak}@nec-labs.com
NEC Laboratories America, Princeton, NJ

Abstract—**Power analysis early in the design cycle is critical for the design of low-power systems. With the move to system-level specifications and design methodologies, there has been significant research interest in system-level power estimation. However, as demonstrated in this paper, the addition of power estimation capabilities to system-level simulation tools can significantly degrade simulation efficiency *(upto 8.5X)* , limiting the use of power estimation during long simulation runs, and the ability to perform extensive design space exploration.**

Some power modeling techniques for system components provide "local" trade-offs between power estimation accuracy and computational cost. This work addresses a complementary problem — the optimized integration and usage of heterogeneous component power models within a system-level simulation framework. We view system-level power estimation as a *global deployment of computational effort* (the effort required to perform power estimation) *over space* (the different components) and *time* (the duration of the simulation). We illustrate the advantages of optimizing the allocation of power estimation effort based on run-time variations of component-level, as well as system-level power consumption characteristics. To achieve this, we have developed a novel power estimation framework, based on a network of *power monitors*. Power monitors observe component- and system-level execution and power statistics at run time, based on which they (i) select between multiple alternative power models for each component, and/or (ii) configure the component power models, to best negotiate the trade-off between efficiency and accuracy. In effect, the power monitor network performs a co-ordinated, adaptive, spatio-temporal allocation of computational effort for power estimation.

Experiments conducted on a commercial system-level simulation framework and System-on-Chip platform demonstrate that the proposed techniques yield large reductions in power estimation overhead (nearly an order of magnitude), while minimally impacting power estimation accuracy.

I. Introduction

Power consumption has emerged as a primary design metric for a wide range of electronic systems, ranging from battery-powered appliances to high-performance computing systems. With rising system complexity, it is becoming increasingly critical to address power consumption early in the design cycle, namely, at the system-level, when significant opportunities exist for optimizing the system architecture and application for improved power efficiency [1], [2], [3].

Recent years have seen significant research interest in system-level power estimation. Most of this research has focused on power modeling techniques for individual system components (*e.g.*, processors, memories, on-chip buses, peripherals, user-defined logic, *etc.*). These power models can be integrated into system-level simulation frameworks to provide power estimation capabilities. Due to the inherent diversity of System-on-Chip (SoC) components and their design styles, system-level simulation is typically performed using a collection of heterogeneous simulation models for the different components. Consequently, power models for different system components are also heterogeneous in nature (*e.g.*, instruction-level power modeling techniques that may be used for a processor differ significantly from analytical power models that may be used for an on-chip memory, and from transaction-level power models used for an on-chip bus). Notwithstanding the inherent efficiency of system-level simulation (when compared to lower levels of abstraction), power estimation typically results in a substantial decrease in simulation efficiency compared to functional simulation (upto 8.5X, as demonstrated in this paper). This slowdown is due to the overhead of extracting the necessary data from the component (functional) simulation models, evaluating the power models, and performing power aggregation and reporting.

In this paper, we address a problem that is complementary to previously proposed power modeling techniques for individual system components, *i.e.,* the optimized integration and usage of heterogeneous component power models in a system-level simulation framework.

A. Paper Overview and Contributions

In this paper, we propose 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. The intuition behind our approach is that the computational effort for power estimation should be distributed spatially (across different system components) and temporally (over the duration of simulation) in a manner that maximizes the resulting estimation accuracy. Conversely, for a desired level of accuracy, power estimation effort should be deployed judiciously so as to minimize the resulting simulation overhead.

In order to motivate our work, we first study the effect of integrating power models for the components of a commercial SoC platform into a state-of-the-art system-level simulation framework. In particular, we analyze the computational overhead for power estimation, and breakdown the overhead across the different SoC components, and over the simulation period. Our study demonstrates several opportunities for reducing power estimation effort for each component, and underscores the need for managing power estimation effort by taking into account (i) global (system-level) information, as well as (ii) dynamic variations in component-level power consumption characteristics.

Based on these observations, we propose a novel power estimation framework that integrates heterogeneous component power models using a network of "power monitors". The monitor-based framework provides an intelligent interface, facilitating the seamless integration of component simulation models on one hand, and a variety of heterogeneous power models on the other. Power monitors enable each component model to be associated with multiple (distinct) power models of differing accuracy and efficiency, or with configurable power models that can be tuned to different accuracy/efficiency levels. The power monitor exercises fine-grained control over the different power models through dynamic selection and configuration of power models based on information gathered during simulation. We demonstrate the effectiveness of the proposed framework through experiments on a system-level model of a commercial SoC platform. Experiments indicate that the monitor based power estimation technique helps achieve large reductions in power estimation overhead (upto 9.5X) while resulting in less than 5% error on average, when compared to conventional techniques.

The rest of this paper is organized as follows. In Section I-B, we describe related work in system-level power estimation. In Section II, we analyze the computational effort spent in performing system-level power estimation, and point out the opportunities for improving power estimation efficiency. In Section III, we provide an overview of the proposed monitor-based power estimation framework. In Section IV, we present details of the framework, including techniques for dynamic power model selection. In Section V we present the experimental evaluation of the proposed framework using a commercial SoC platform.

Proceedings of the 18th International Conference on VLSI Design held jointly with 4th International Conference on Embedded Systems Design (VLSID'05)
1063-9667/05 $20.00 © 2005 **IEEE**

B. Related Work

Extensive work has been performed in power estimation at the transistor, gate and register-transfer levels [1], [2], [4], [5]. While these techniques are invaluable at later stages of the design cycle, they are too inefficient for use in system-level design.

A significant body of research has focused on developing power models for individual system components. For programmable processors, power models can be divided into structural and instruction-based techniques. Structural models monitor the activity of the various components in the processor's micro-architecture, and use this information to estimate power consumption (*e.g.,* [6], [7]). Instruction-level power models associate power consumption with the sequence of instructions executed by the processor, with corrections to account for architectural effects such as cache misses and pipeline stalls. Techniques for instruction-level power modeling were first proposed in [8], following which several enhancements have been proposed to improve efficiency, accuracy, and characterization effort (*e.g.,* [9], [10], [11]).

For components with regular implementations, such as memories and caches, various analytical models have been proposed to estimate power consumption under given access patterns [12], [13]. For user-defined logic, techniques for power estimation at the behavioral and cycle-accurate functional levels of abstraction have been proposed, and can be used in the context of system-level power estimation [14], [15], [16]. Power models for on-chip buses are described in [17], [18].

A limited amount of research has addressed the integration of power models for different system components. Energy models for a processor and memory sub-system were integrated in [19], and used to explore system-level energy trade-offs. A HW/SW co-simulation framework was enhanced with power models for a processor, memory, and custom hardware in [20]. In order to enable efficient power estimation for design space exploration, trace-based analysis techniques for power estimation and profiling were proposed in [17], [21]. Trace-based techniques are efficient for repeated power estimation under a given test bench, since they perform system simulation only once and use the information captured in traces to perform power estimation.

The proposed techniques are complementary to the work described above. Given a system-level power estimation framework consisting of heterogeneous component power models, our work addresses the issue of how to achieve the best trade-off between efficiency and accuracy by optimizing the allocation of power estimation effort spatially, between the different system components, and temporally, over the duration of the simulation. New power modeling techniques for system components can be easily integrated with our approach. Our approach can also be combined with trace-based techniques to provide further efficiency improvements for repeated estimation runs.

II. System-Level Power Estimation: Allocation of Computational Effort

In this section, we analyze the computational overheads incurred when a system-level simulation framework is enhanced with power estimation capabilities, and illustrate opportunities that exist for reducing this overhead by carefully optimizing the spatio-temporal allocation of computational effort.

A. Example System-on-Chip Platform

For this study, we consider the System-on-Chip platform illustrated in Figure 1. The platform consists of an embedded processor (the ARM946E-S [22]) that includes instruction and data caches, and tightly coupled (scratch-pad) memories, an on-chip bus (the AMBA bus [23]), a dedicated hardware component for performing application-specific tasks (labeled "Image Filter HW"), as well as standard peripherals, such as timer, interrupt controller, DMA controller, and interfaces to external devices. The platform implements an image processing application that runs on the processor, retrieves image data from off-chip memory, uses the image filter hardware to perform basic image processing operations (smoothing, color enhancement, *etc.*) at the pixel-level, and stores the resulting image in memory.

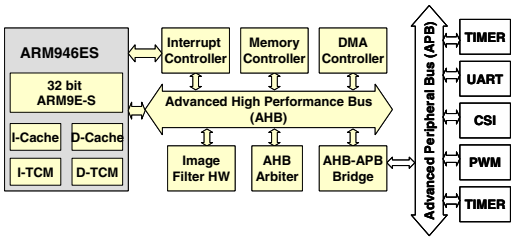


Fig. 1. Example System-on-Chip platform.

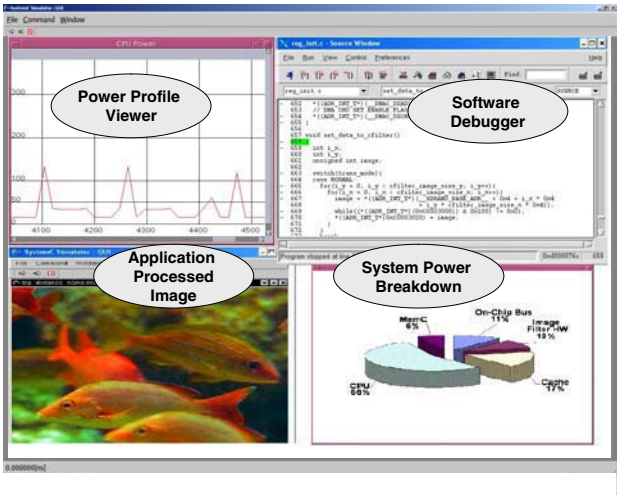
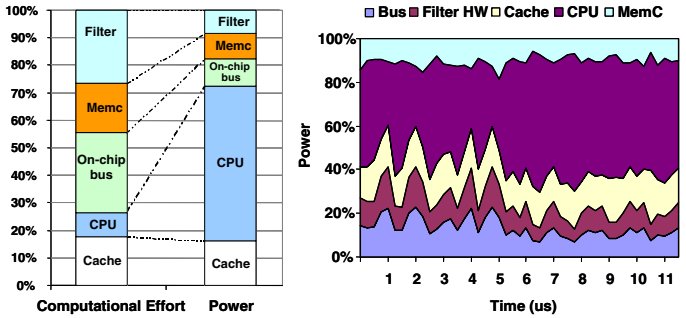


Fig. 2. System-level simulation and power estimation tool interface.

The platform was modeled using cycle-accurate functional models for the hardware components, transaction-level models [23] for the on-chip bus (both implemented using SystemC), and an instruction-level model for the processor. For power estimation, all the functional models were enhanced with corresponding power models (details of which are described in Section IV). System-level simulation and power estimation is performed using the OSCI reference simulator [24]. The interactive framework allows users to set breakpoints, obtain power vs time profiles for different components, and a breakdown of system power consumption. A screenshot illustrating these capabilities is presented in Figure 2.

B. Power Estimation Effort and Impact on System-Level Power Estimation Accuracy

We first analyze the computational overhead measured while performing power analysis at the system level. To quantify this overhead, we performed pure functional simulation of the entire system as it executed the image application, and then repeated the experiment with all the component power models included. Both simulations were run on a Dell PowerEdge Server with a 2.8 Ghz Intel Xeon processor and 4 GB RAM. A comparison of measured execution times revealed that the inclusion of power models causes a reduction in simulation efficiency of more than 8.5X. This slowdown was observed in spite of using hardware power models for certain components (*e.g.,* Image Filter HW) that operate at the cycle-accurate functional level [16], which are



(a) (b)

Fig. 3. (a) Comparison of power estimation effort and contribution to system power for different system components; (b) Dynamic variation of relative contributions to system power.

significantly more efficient than commercially available RT-level hardware power estimators. Clearly, the incorporation of power models has a large impact on overall simulation efficiency.

To better understand the computational effort associated with power estimation, consider the results presented in Figure 3(a). The first column presents a breakdown of the computational effort (CPU time) expended in performing power estimation for different system components. The second column presents the percentage contributions of the corresponding components to overall power consumption. We observe that the allocation of computational effort poorly tracks the manner in which power is consumed by the different system components. For example, while the image filter and bus architecture together account for only 18% of the total power, their power models account for 55% of the computational effort towards power estimation. In contrast, while the processor accounts for 56% of the total power, its power model consumes only 10% of the computational effort.

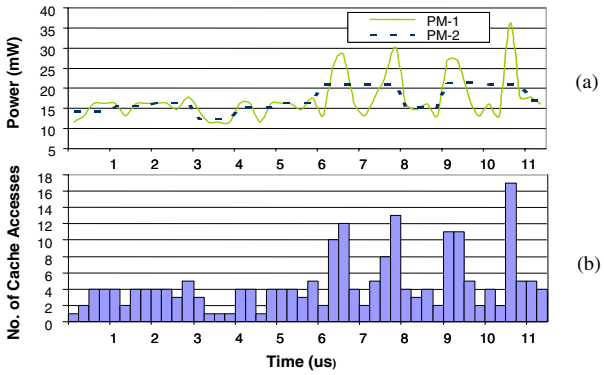
These results illustrate that a large discrepancy exists between the complexity associated with power estimation of certain components, and the impact that these components have on total system power. This suggests that a more optimized allocation of computational effort may result in a superior trade-off between overall power estimation accuracy and efficiency. Based on these insights, it might be tempting to optimize the selection of power models by using more accurate, but computationally expensive models for components that on average, consume a larger fraction of total system power, and less accurate, but more efficient ones for components that consume a smaller fraction of system power. However, such an approach may not accurately track system power consumption over time, since components may exhibit significant dynamic variation in their individual power consumption characteristics. From Figure 3(b), which illustrates the variation of each component's contribution to total power, we observe that the processor's contribution varies between 27% and 63% of system power. This motivates the use of techniques in which power estimation effort for a component is adapted, based on its (variable) contribution to total system power over time.

C. Power Estimation Effort And Component-Level Variations

The previous example illustrated the need for careful allocation of power estimation effort from a system-level standpoint: *i.e.*, by taking total system power, and the relative contributions of each component, into account. We next illustrate that, in addition to system-level considerations, it is also necessary to consider variations in component-level behavior.

Figure 4(a) illustrates the power consumption of the processor instruction cache as it executes a portion of the application described in Section II-A. The figure displays the power profile of the cache obtained using two different power models that have contrasting accuracy/efficiency characteristics. The profile marked "PM-1" is obtained using a per-access power model, which computes cache power on every clock cycle, taking activities in the address lines, bit lines, and word lines into account. The second profile (marked "PM-2") is obtained using periodic application of a more efficient (but less accurate) analytical model, which estimates average power using an aggregate count of the number and types of accesses seen during a certain interval. From the figure, we observe that upto 6 μs, the power profile generated by the analytical model tracks the profile obtained using the per-access model with an error of 9%. However, after 6 μs, the cache exhibits high variation in its power consumption, increasing the error to 26%. Using the analytical model alone throughout the simulation would significantly compromise power profiling accuracy, whereas only using the per-access model could result in large estimation overhead. Clearly, a technique that can dynamically vary the power model based on component-level information may result in a superior accuracy-efficiency trade-off.

Since the recorded history of power consumption of a component is only as accurate as the power model that is used to obtain it, using it as a basis for predicting future power consumption characteristics can be misleading. Instead, in our approach, we track component-specific parameters from the functional model of the component, that provide an indication of component power characteristics. For example, the cache power profile [Figure 4(a)] is to a large extent determined by the rate of accesses [Figure 4(b)]. Hence, we consider a scheme in which the variance in the rate of accesses to the cache is monitored. In intervals where the variance exceeds a certain threshold, the accurate, per-access model is used, whereas elsewhere, the more efficient, analytical model is used. In our study, the resulting compromise in accuracy was observed to be less than 5%, while the reduction in power estimation effort was 3.4x.



(a) (b)

Fig. 4. (a) Comparison of power profiles under different cache power models; (b) Corresponding profile of cache accesses.

In summary, these examples illustrate the following issues:

- The availability of alternate (or configurable) power models that provide distinct accuracy/efficiency characteristics can facilitate fine-grained, dynamic allocation of computational effort towards power estimation.
- Customizing the spatial allocation of computational effort among different components keeping in mind their relative contributions to total system power can improve power estimation efficiency without significantly compromising accuracy.
- It is also important to customize the temporal allocation of computational effort to each system component (over the duration of the simulation), based on run-time variations in the component's power consumption characteristics.

In the next two sections, we describe the proposed monitor-based power estimation framework that addresses the issues described above.

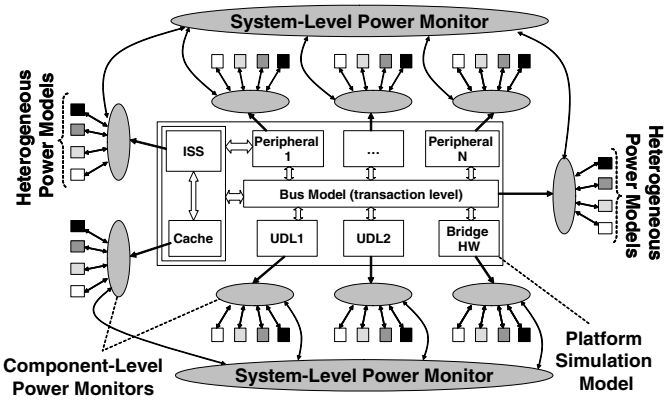


Fig. 5. Monitor-based system-level power estimation framework using heterogeneous power models.

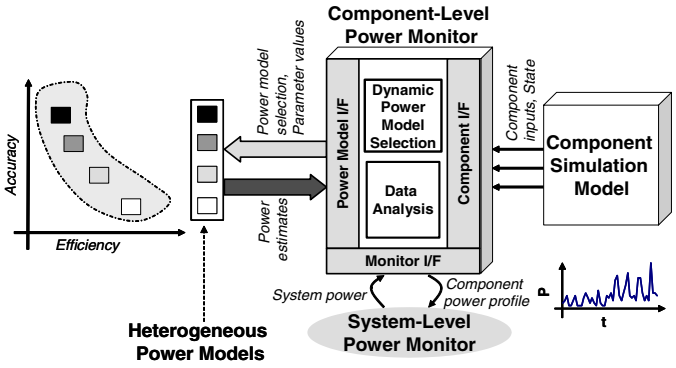


Fig. 6. Monitor-enhanced simulation model of an SoC component with heterogeneous power models.

III. Monitor-Based Power Estimation: Overview

In this section, we present an overview of the proposed monitor-based power estimation framework. We highlight the various components associated with the framework, and briefly describe their functions. Figure 5 illustrates how the proposed framework enhances a simulation model of the target system. The framework consists of (i) simulatable, functional models of each system component, (ii) for each system component, a set of heterogeneous power models, differing in terms of their accuracy and efficiency, and (iii) a network of power monitors (shown as shaded ellipses), which in turn, consists of (a) component-level power monitors for each system component, and (b) a system-level power monitor. Figure 6 illustrates in detail, the integration of a simulation model of a component, a set of heterogeneous power models, the corresponding component-level power monitor, and the system-level power monitor. We next describe each of these components in turn.

Component Simulation Model: The level of abstraction at which each component is modeled may vary, depending on the complexity of the component, ranging from pin-accurate, register-transfer level models to more abstract models. The power monitor network is not tied to any specific modeling abstraction, and in general, could be used to bridge potential gaps between the abstraction levels at which functionality is modeled, and power is estimated. In our work, we make use of popular abstractions for system-level simulation: cycle-accurate, functional models for custom hardware, transaction-level models for the on-chip bus, and instruction-level models for embedded processors.

Heterogeneous Power Models: Each system component is associated with a set of alternative power models that provide trade-offs between power estimation accuracy and efficiency, as illustrated in Figure 6. In the proposed framework, this heterogeneity is leveraged to achieve gains in power estimation accuracy and efficiency through run-time power model selection and/or configuration. Further details about heterogeneous power models are provided in Section IV-A.

System-Level Power Monitor: The system-level power monitor accumulates power estimates from the component-level monitors, generates system-level power statistics (*e.g.,* a system power profile, total energy consumed), and provides feedback to component-level power monitors.

Component-Level Power Monitor: Component-level power monitors form the core of our power estimation framework. These power monitors facilitate the integration of a simulatable functional model of a system component with a set of heterogeneous power models. The key functions of each component-level power monitor are as follows:

Power Model / Component Separation: The component-level power monitor provides a clean separation between the functional model of the component and the set of corresponding power models, facilitating the seamless addition of new power models, while minimizing the changes to the functional models. This separation is achieved through three interfaces. The *component interface* enables the extraction of data from the component simulation model for (i) guiding the process of power model selection, and (ii) computing the values of power model parameters. The *power model interface*, is in fact, a set of interfaces, one for each alternative power model, which permits the exchange of power model parameters, and power estimates, between the monitor and the power model. The *system-level monitor* interface enables the exchange of power consumption estimates between component- and system-level monitors.

Dynamic Power Model Selection: The component-level power monitor is responsible for optimizing the selection and usage of power models that are associated with the component, based on conditions observed during simulation. It performs this task based on information gathered via (i) the component interface, which extracts current state and input values from the functional model of the component, (ii) the power model interface, which provides power estimates for the component, and (iii) the monitor interface, which provides an running estimate of the total system power consumption. Details of this process are described in the Section IV-B.

Data Analysis: Certain power models may require aggregate data as parameters, such as rate of cache accesses, or number of instructions executed of a certain type. The data analysis module observes the necessary values at the monitor's component interface, and computes the required model parameters. In addition, the data analysis module also computes metrics that monitor temporal variation in component activity, to help guide dynamic power model selection.

Power Profile Generation: Finally, the power monitor is responsible for generating a component-level power profile, which is made available to the system-level power monitor, as well as to graphical display utilities.

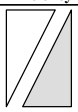
In summary, the power monitors exercise dynamic, regulatory control over a set of power models in order to perform optimized power estimation for different parts of the system. The dynamic management of power models is performed both locally, making use of component-level power consumption characteristics, as well as globally, making use of system-level information.

IV. Run-Time Management of Heterogeneous Power Models

In this section, we first describe a set of heterogeneous power models for each system component, and then describe techniques used by the component-level power monitors for their dynamic configuration and management.

TABLE I

HETEROGENEOUS POWER MODELS FOR SYSTEM-LEVEL POWER ESTIMATION

Platform Component		Power Model Description	Accuracy/ Efficiency
Processor	1	Pipeline state aware	
	2	Instruction-level	
	3	Analytical, Instruction-class based	
	4	Function-level Macro-models	
	5	Mode-based	
On-Chip Bus	1	Distributed RC models	
	2	Lumped capacitance	
	3	Analytical, transaction based	
Cache	1	Structure-aware, access level	
	2	Analytical, access-statistics based	
Application Specific HW	1	Gate-level	
	2	Register-transfer level	
	3	Cycle-accurate functional level - State based sampling OFF	
	4	Cycle-accurate functional level - State based sampling ON	

A. Heterogeneous Power Models

We consider different system components in turn, and describe a set of alternative power models for each. Note that, we only describe an illustrative set of power models: the framework can be easily extended to support other power models as well.

Embedded processors: Several techniques have been developed for modeling processor power consumption at the system level. The complexity of these models vary, depending on the volume of, and frequency with which, information is extracted from a simulation model of the processor. Table I lists several alternatives, in decreasing order of computational effort. In the ~~first~~ model, for every clock cycle, the complete pipeline state of the processor is captured, and the combination of instructions found in the different stages is used to estimate power consumption [6], [7]. In the ~~second~~ model, for each cycle, only the instruction that is currently being executed is extracted [11]. In the ~~third~~ model, over discrete time intervals, only the number of instructions of different predefined types are counted, to compute total energy or average power [11]. The ~~fourth~~ approach, software energy macro-modeling, involves monitoring code sequences of larger granularity (*e.g.*, function calls) [25]. The ~~fifth~~ and simplest power model we consider is based on parameters such as power modes, operating voltage, and frequency [11].

On-Chip Buses: Numerous models have been proposed for estimating the power consumption of global buses. Examples of such power models that we have considered in our framework are listed in Table I. In the ~~first~~ approach, on every cycle, transition activity is examined on individual bus lines, and is used to estimate power, using transmission line models that capture deep sub-micron effects, and effects of the drivers and repeaters [18]. In the ~~second~~ model, for each cycle, aggregate transition activity is used to estimate power consumed on global buses, using a lumped capacitance to model driver, repeater, line, and parasitic capacitances. The ~~third~~ model is an analytical one, in which over a certain time interval, the number and types of bus transactions are monitored, and used to estimate average transition activity, which can then be used to estimate average power.

Caches: Cache power models include those that are targeted towards cycle-level simulation environments [6], as well as more efficient analytical models that are targeted towards exploring alternative cache architectures [12], [17]. For our framework, we consider the two models listed in Table I. In the ~~first~~ model, on every access to the cache, the power consumed by the cache is computed based on the type of access (read/write), the result of the access (hit/miss), and transition activity on the bit and word lines. In the ~~second~~ model, over a certain time interval, statistics that capture the number and types of cache accesses are used as inputs to an analytical model that computes average cache power, using lumped capacitances for different cache components, and estimated transition activity.

Application Specific Logic and Platform Infrastructure Hardware: Power analysis of hardware, including both application-specific hardware, as well as standard components such as memory controllers, timers, and other peripherals, has traditionally been performed at the logic- and register-transfer levels (RTL). Recently, advances have been made in estimating the power consumed at the cycle-accurate functional and behavioral levels [14], [15], [16]. While each abstraction level in itself represents a potential trade-off between power estimation accuracy and computational effort, approaches based on logic-/RT-level power estimation are unacceptably slow. In our framework, we consider power models at the cycle-accurate, functional level. We use a technique that embeds structural information obtained from an RTL description of a component into the corresponding cycle-accurate, functional model. While details are available in [16], it bears mentioning that the technique provides support for statistical sampling, which provides a knob for regulating power estimation accuracy and efficiency.

B. Dynamic Power Model Selection

The procedure for power model selection that is executed by each component-level power monitor is presented in Figure 7. The procedure receives inputs information from the three interfaces of the power monitor. Over the course of the simulation run, the procedure repeatedly computes an optimized power model selection for purposes of power estimation. The procedure operates in two steps. In the first step (encircled using dotted lines), system-level criteria are used to reduce the number of available choices, by optimizing the spatial allocation of computational effort. Next, component-level criteria are used to choose a unique power model, which helps optimize the temporal allocation. We next discuss each of these steps in detail.

B.1 System-Level Criteria

Let $P_C = \{P_1, P_2, \ldots, P_N\}$ be the set of power models associated with a component C, sorted in terms of decreasing accuracy and related computational effort. Let the average power consumed by the system over a time interval τ be given by $P_{sys}(\tau)$. A component-level monitor can observe $P_{sys}(\tau)$ via the system-level monitor interface. Let $P_C(\tau, P_i)$ denote the power consumed by a component C over the same interval, as estimated by power model P_i. The monitor computes $F_C(\tau) = P_C(\tau, p_i)/P_{sys}(\tau)$, which is the average contribution of C to the total system power over the interval. For a component C, a set of threshold values for $F_C(\tau)$ are pre-determined: $C_T = \{C_1, C_2, \ldots, C_M\}$, representing discrete values of component C's contribution to total system power. A lookup-table (LUT) is used to store a one-to-many mapping between C_T and P_C. For example, if $C_i \leq F_C(\tau) \leq C_j, (1 \leq i, j \leq M)$, the LUT returns a set of power models $\tilde{P_C} = \{P_k \mid 1 \leq k \leq N\} \subseteq P_C$. In the lookup table, larger values of C_i are mapped to more accurate (but more computationally expensive) power models, while smaller values are mapped to more abstract, efficient ones.

Note that, the number of predefined thresholds M can can be controlled to regulate the sensitivity of power model selection policies to system-level information. A small value of M results in higher number of power models being provided to the component-level step, making component-level criteria play a more significant role.

B.2 Component-Level Criteria

In the remainder of the methodology, as shown in Figure 7) a power model P_i is selected from $\tilde{P_C}$. The intuition behind this process is as follows. For some intervals during system execution, components may

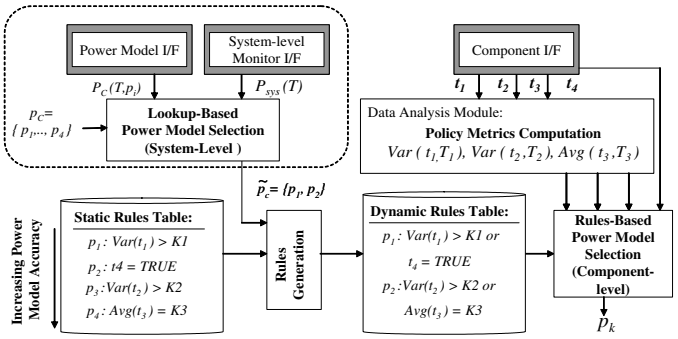


Fig. 7. Methodology for power model selection considering system-level and component-level information

experience workloads that lend predictability to their power consumption characteristics, enabling the application of more abstract, efficient power models without significantly compromising accuracy. However, for other intervals, more accurate power models may be required.

For each component C, a set of variables t_i are monitored, that we denote as triggers. The choice of triggers is component-specific, and depends on the component's functionality. Examples of triggers are the execution of a specific type of instruction, access to a cache, *etc*. The component interface receives information about the execution of triggers, and the data analysis module computes specific metrics that are then used by the power model selection algorithms. Figure 7 illustrates some of these metrics. For example, if t_1 represents the number of cache accesses in an interval, $Var(t_1, T_1)$ represents the variance of cache access rates over a sequence of intervals of length T_1. Note that, triggers may by themselves be regarded as metrics *i.e.,* they may not require further analysis (*e.g.*, t_4 in Figure 7). For example, an application specific co-processor may have a "start" pin that is asserted each time a particular computation intensive operation is initiated. In such a case, the value of the pin is a useful metric to drive power model selection. The computed metrics are then used by the power model selector to evaluate a set of mutually exclusive conditions, which are stored in a "Dynamic Rules Table". For example, in Figure 7, one of the illustrated rules specifies that power model p_1 is to be used if the variance in the value of trigger t_1 measured over T_1 exceeds a threshold K_1, or if trigger t_4 is currently true. The mutually exclusive nature of the rules guarantees selection of a unique power model.

We next explain the rationale for the dynamic rules table. At run-time, the number of power models available to the component-level policy may vary, since it depends on system-level criteria. In order to address this issue, a set of rules must be developed statically for each component, assuming that all the power models associated with this component may be available at run-time. These rules are stored in the "Static Rules Table", in order of increasing accuracy of the associated power model. At run-time, a "Rules Generation" step transforms this table, taking the actual number of power models available into account. It constructs new rules by applying boolean OR operators on consecutive rules in the static rules table, ensuring a one-one map from rules to models. The figure illustrates the operation of the rules generator for a component that has a maximum of 4 power models (p_1 through p_4), but at a certain time, is restricted by the system-level policy to choose between p_1 and p_2.

V. Experimental Results

In this section, we describe the implementation of the proposed monitor-based power estimation framework. We also present results that analyze the accuracy and efficiency of the framework for an example SoC platform.

A. Experimental Methodology

We implemented the proposed monitor network as an extension of the functional models of the SoC platform illustrated in Figure 1. The processor was modeled using an instruction set simulator. SystemC [24] was used to implement cycle-accurate functional models for the peripheral hardware and the image filter, as well as transaction-level models for the AMBA bus [23]. The image processing application (implemented in C) was cross-compiled using the CrossGCC toolchain [26]. System simulation was performed using the OSCI reference simulator [24]. The user interface [Figure 2] was provided via the GNU Insight debugger, which communicates with the target system using the remote GDB protocol [27]. An interactive Qt-based graphical user interface [28] was developed to provide additional control over the simulation and power estimation features, and to provide display capabilities (*e.g.,* power vs time waveforms).

In order to analyze the effectiveness of the proposed techniques, we compared the accuracy and efficiency of the monitor-based power estimation technique relative to a base case. In the base case, the power model selection was fixed. Power models numbered 2, 1, 2, and 3 in Table I were used for the processor, cache, bus, and hardware respectively. In the monitor-based framework, the selected power models were automatically varied among models 2 and 3 for the processor, models 1 and 2 for the cache, models 2 and 3 for the bus, and models 3 and 4 for other platform hardware. The base case utilizes the most accurate power model that we implemented for each component, hence, it represents a bound on the accuracy achievable by the monitor-based framework. We considered three variants of the base architecture. In *Arch 1*, Direct Memory Access (DMA) was disabled, but both instruction and data caches were enabled. In *Arch 2*, DMA was enabled but the caches were disabled. In *Arch 3*, DMA and caches were both enabled. The metrics used for analyzing the monitor-based framework are as follows:

Average Power Error: This refers to the error in estimating the average power consumed by the system during the entire simulation run using the monitor-based framework, relative to the base case.

Profiling Error: To quantify the accuracy of the power profile (power versus time curve) as generated by the monitor-based framework, we compute the profiling error as follows. The entire duration of the simulation was divided into intervals of equal length (for our experiments, we used an interval of 100 cycles). The estimation error in each interval was computed (relative to the base case), and then averaged (using absolute values, to prevent positive and negative errors from cancelling) over all the intervals.

Efficiency Gain: Improvements in power estimation efficiency were computed relative to the base case using execution time measurements on a Dell PowerEdge Server with a 2.8 Ghz Intel Xeon processor, and 4 GB RAM, running Red-Hat 8.0.

B. Power Profiling Accuracy

We first compare the system-level power profile as obtained by the monitor-based power estimation framework ("monitor") with the base case ("original"). Figure 8 illustrates the power profiles as generated by the two techniques for the architecture denoted by *Arch 3* as it executes the image processing application. The figure shows that the power profile generated by the monitor-based framework is very close to the base case. Note that, in the callout, the y-axis has been appropriately scaled in order to make the difference between the two profiles visible. For this architecture, the profiling error was observed to be 1.08%.

C. Power Estimation Accuracy Versus Efficiency

The following experiments analyze the overall power estimation accuracy and efficiency achieved by the monitor-based framework rela-

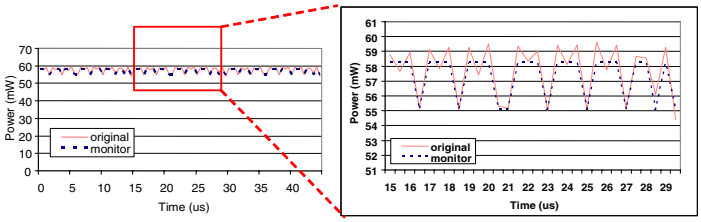


Fig. 8. Comparison of power profiles obtained using the monitor-based framework ("monitor"), and conventional system-level power estimation ("original")

tive to the base case described in Section V-A. For each of the three architectural variants, we compare the profiling error, average error, and efficiency gain achieved by the monitor-based framework. Table II presents the results of these experiments. The second and third columns denote the profiling errors and average power errors respectively. The last column in this table represents the reduction in power estimation overhead. From the table, we observe significant reductions in power estimation overhead, upto almost an order of magnitude (9.5x in the first row). Upon analysis, we found that the efficiency gains were mainly due to intervals of time where more abstract power estimation models were used for different parts of the system. For significant parts of the simulation run, the monitor-based framework uses abstract models for the bus, filter HW, and memory controller. It occasionally takes advantage of more abstract CPU and cache power models, but at most times uses more accurate models for the CPU (since it consumes a large fraction of system power), and for the cache (since it exhibits large dynamic variation). We observe from the table that the impact on the overall accuracy is negligible across all the architectures.

TABLE II

ACCURACY/EFFICIENCY OF THE MONITOR-BASED POWER ESTIMATION FRAMEWORK FOR DIFFERENT SYSTEM ARCHITECTURES

Architecture Model	Profiling Error (%)	Average Power Error (%)	Efficiency Gain
Arch 1	1.36	0.91	9.5x
Arch 2	3.83	3.83	5.3x
Arch 3	1.08	0.73	5.2x

Table III analyzes the estimation results for *Arch 1* in further detail. The results in this table illustrate the contribution of different components to the overall power estimation speedup and accuracy loss. For some components, the accuracy loss is relatively high (*e.g.,* the memory controller exhibits a profiling error of 7.5%). However, since the contribution of these components to total system power is small, the system-level profiling error is insignificant — in this case, 1.36%. However, the benefits in terms of speedup are substantial. From the table, we observe a reduction of almost 11x in terms of power estimation overhead for the memory controller. This goes a long way in achieving the overall speedup of almost an order of magnitude. The above results demonstrate that the monitor-based framework successfully uses abstract power models for efficiency gains, whenever it is possible to do so without significantly compromising system-level power estimation accuracy.

VI. Conclusions

In this paper, we demonstrated the importance of judiciously allocating computational effort among different components while performing power estimation at the system level. We described a monitor-based power estimation framework that enables the integration and dynamic management of a heterogeneous set of power models. Our experiments showed that the framework is capable of optimizing the allocation of computational effort, achieving substantial speedups in power estimation efficiency, at very little cost in terms of accuracy.

TABLE III

ACCURACY/EFFICIENCY OF THE MONITOR-BASED POWER ESTIMATION FRAMEWORK FOR INDIVIDUAL SYSTEM COMPONENTS

Component	Profiling Error (%)	Average Power Error (%)	Efficiency Gain
Cache	2.64	2.99	6.8x
CPU	0.93	0.12	4.9x
On-Chip Bus	3.89	2.46	4.7x
MemC	7.52	0.97	10.9x
Filter HW	2.16	0.008	5.1x
System	**1.36**	**0.91**	**9.5x**

References

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

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

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

[4] J. Monteiro and S. Devadas, *Computer-Aided Design Techniques for Low Power Sequential Logic Circuits*. Kluwer Academic Publishers, Norwell, MA, 1996.

[5] E. Macii, M. Pedram, and F. Somenzi, "High-Level Power Modeling, Estimation, and Optimization," in *Proc. Design Automation Conf.*, pp. 504–511, June 1997.

[6] D. Brooks, V. Tiwari, and M. Martonosi, "Wattch: A Framework for Architectural-Level Power Analysis and Optimizations," in *Int. Symp. on Computer Architecture*, 2000.

[7] W. Ye, N. Vijaykrishnan, M. Kandemir, and M. J. Irwin, "The Design and Use of SimplePower: A Cycle-Accurate Energy Estimation Tool," in *Proc. Design Automation Conf.*, pp. 340–345, 2000.

[8] V. Tiwari, S. Malik, and A. Wolfe, "Power Analysis of Embedded Software: A First Step Towards Software Power Minimization," *IEEE Trans. VLSI Systems*, vol. 2, pp. 437–445, Dec. 1994.

[9] A. Sama, M. Balakrishnan, and J. F. M. Theeuwen, "Speeding Up Power Estimation of Embedded Software," in *Proc. Int. Symp. Low Power Electronics & Design*, pp. 191–196, Aug. 2000.

[10] S. Steinke, M. Knauer, L. Wehmeyer, and P. Marwedel, "An Accurate and Fine Grain Instruction-Level Energy Model Supporting Software Optimizations," in *Proc. Int. Wkshp Power & Timing Modeling, Optimization & Simulation (PATMOS)*, Sept. 2001.

[11] A. Sinha and A. P. Chandrakasan, "JouleTrack - A Web Based Tool for Software Energy Profiling," in *Proc. Design Automation Conf.*, pp. 220–225, June 2001.

[12] M. B. Kamble and K. Ghose, "Analytical Models for Energy Dissipation in Low Power Caches," in *Proc. Int. Symp. Low Power Electronics & Design*, pp. 143–148, Aug. 1997.

[13] M. Mamidipaka, K. S. Khouri, N. Dutt, and M. Abadir, "IDAP: A Tool for High-Level Power Estimation of Custom Array Structures," in *Proc. Int. Conf. Computer-Aided Design*, Nov. 2003.

[14] R. Mehra and J. Rabaey, "Behavioral Level Power Estimation and Exploration," in *Proc. Int. Wkshp. Low Power Design*, pp. 197–202, Apr. 1994.

[15] L. Kruse, E. Schmidt, G. Jochens, A. Stammermann, A. Schulz, E. Macii, and W. Nebel, "Estimation of Lower and Upper Bounds on the Power Consumption From Scheduled Data Flow Graphs," *IEEE Trans. VLSI Systems*, vol. 9, pp. 3–14, Feb. 2001.

[16] L.Zhong, S. Ravi, A. Raghunathan, and N. K. Jha, "Power Estimation for Cycle-Accurate Functional Descriptions of Hardware," in *Proc. Int. Conf. Computer-Aided Design*, 2004.

[17] T. D. Givargis, F. Vahid, and J. Henkel, "Evaluating Power Consumption of Parameterized Cache and Bus Architectures in System-on-a-Chip Designs," *IEEE Trans. VLSI Systems*, vol. 9, pp. 500–508, Aug. 2001.

[18] P. P. Sotiriadis and A. P. Chandrakasan, "A Bus Energy Model for Deep Sub-Micron Technology," *IEEE Trans. VLSI Systems*, vol. 10, pp. 341–350, June 2002.

[19] Y. Li and J. Henkel, "A Framework for Estimating and Minimizing the Energy Dissipation of HW/SW Embedded Systems," in *Proc. Design Automation Conf.*, pp. 188–193, June 1998.

[20] M. Lajolo, A. Raghunathan, S. Dey, L. Lavagno, and A. Sangiovanni-Vincentelli, "Co-Simulation Based Power Estimation for System-on-Chip Design," *IEEE Trans. VLSI Systems*, vol. 10, pp. 253–266, June 2002.

[21] K. Lahiri, A. Raghunathan, and S. Dey, "Efficient Power Profiling for Battery-Driven Embedded System Design," *IEEE Trans. Computer-Aided Design*, vol. 23, pp. 919–932, June 2004.

[22] "ARM946E-S: Embedded Core with Flexible Cached Memory System and DSP Instruction Set Extensions." `http://www.arm.com/products/CPUs/ARM946ES.html`.

[23] "AMBA CLI Specification." `http://www.arm.com/products/solutions/ahbcli_spec.html`.

[24] "The Open SystemC Initiative." `http://www.systemc.org`.

[25] T. K. Tan, A. Raghunathan, G. Lakshminarayana, and N. K. Jha, "High-Level Software Energy Macromodeling," in *Proc. Design Automation Conf.*, pp. 605–610, 2001.

[26] "CrossGCC." `http://www.billgatliff.com`.

[27] "The Free Software Foundation." `http://www.gnu.org`.

[28] `http://www.trolltech.com`.