Design Automation for Embedded Systems, 7, 183–203, 2002.

A Framework for Modeling and Estimating the Energy Dissipation of VLIW-Based Embedded Systems

L. BENINI
Università degli Studi di Bologna, Bologna, Italy

D. BRUNI
Università degli Studi di Bologna, Bologna, Italy

M. CHINOSI
STMicroelectronics, Agrate B. (MI), Italy

C. SILVANO
Università degli Studi di Milano, Milano, Italy

V. ZACCARIA
Politecnico di Milano, Milano, Italy

R. ZAFALON
STMicroelectronics, Agrate B. (MI), Italy

Abstract. This paper describes a technique for modeling and estimating the power consumption at the system-level for embedded VLIW (Very Long Instruction Word) architectures. The method is based on a hierarchy of dynamic power estimation engines: from the instruction-level down to the gate/transistor-level. Power macro-models have been developed for the main components of the system: the VLIW core, the register file, the instruction and data caches. The main goal is to define a system-level simulation framework for the dynamic profiling of the power behavior during the software execution, providing also a break-down of the power contributions due to the single components of the system. The proposed approach has been applied to the Lx family of scalable embedded VLIW processors, jointly designed by STMicroelectronics and HPLabs. Experimental results, carried out over a set of benchmarks for embedded multimedia applications, have demonstrated an average accuracy of 5% of the instruction-level estimation engine with respect to the RTL engine, with an average speed-up of four orders of magnitude.

Keywords: Embedded systems, power estimation, VLIW-architectures.

1. Introduction

The demand for low-power VLSI circuits and systems for portable applications is steadily increasing. At the same time, design methodologies are evolving to deal with the complexity of systems-on-chip (SoC, from now on) which integrates on a single die one or more processors, a significant amount of memory, and other functional modules. The system-level design approach requires to effectively manage huge design complexity and to

support specification and analysis at high abstraction levels. High-level power estimation and optimization is a crucial issue in the early determination of the power budget for SoCs. Accuracy and efficiency must be traded-off to meet the overall power and performance requirements, avoiding expensive design respins.

In this scenario, our work focuses on software power estimation for embedded applications, where an embedded core (with a memory hierarchy) is integrated in a SoC. The main contribution of our estimation engine consists of providing power consumption figures for *software running on a given hardware architecture*, and to help optimizing the target application for energy efficiency. While relative accuracy is certainly useful, absolute accuracy is the ultimate target.

State-of-the art processor power estimators are based either on *instruction* [4], [24] or *on micro-architectural* [22], [23], [25], [26] modeling methodologies. The proposed approach aims at exploiting the advantages of both techniques, however some differences can be remarked. With respect to instruction-level power analysis (ILPA), our approach gives better insight on the power bottlenecks during software execution (and optimization), because it is based on a detailed micro-architectural model of the core. In fact, to the best of our knowledge, previous micro-architectural models have not been validated on a complete, real-life processor.

The proposed system-level power estimation methodology is based on a hierarchy of dynamic power estimation engines at several abstraction levels: from the instruction-level down to the gate/transistor-level. The main goal is twofold: to profile dynamically the power behavior during software execution and to provide a break-out of the power contributions due to the single components of the system. The proposed approach is adopted in an industrial environment, where detailed description of the processor hardware architecture is available. We prove the viability of high-level power estimation for processor cores both from the *efficiency* and from the *accuracy* standpoints. The main contributions of our work can be summarized as follows:

— the development of novel power macro-models for the main components of the system, namely the VLIW core, the register file and the caches;

— the validation methodology to evaluate the accuracy of the macro-models against post-layout circuit and gate-level simulation;

— the integration of the power macro-models within a hierarchy of simulators, from RT-level (cycle-accurate) to the instruction-level.

We describe the application of the proposed modeling and estimation framework to support the system-level power analysis for the Lx core, a high-performance embedded VLIW processor for multimedia and signal processing applications, jointly developed by STMicroelectronics and Hewlett-Packard [17]. The Lx core is based on a scalable and customizable VLIW processor technology platform and it supports tightly-coupled multiprocessor clusters, as well as on-chip instruction and data caches. In the Lx processor, a very long instruction (*bundle*) is composed of four explicitly parallel instructions

(*syllables*). A complete software environment is being developed concurrently with the hardware [17]. Software development support includes an aggressive ILP compiler, instruction-set simulators and the power estimation environment described in this paper. Extensive validation of the proposed power estimation methodology has been carried out over a set of multimedia benchmarks for embedded applications. The experimental results have demonstrated an average accuracy of 5% of the instruction-level estimation engine with respect to the RTL engine, with an average speed-up of four orders of magnitude.

The rest of the paper is organized as follows. In Section 2, we review some of the most relevant approaches on system-level power estimation appeared so far. Section 3 describes the overall power estimation framework based on an instruction-level engine characterized by using an RT-level engine. The energy models for the VLIW core, the register file and the instruction and data caches are discussed in Section 4, while experimental results derived from the application of the proposed methodology to the Lx case study are described in Section 5. Finally, Section 6 outlines some concluding remarks and future developments of our research.

2. Previous Work

Aggressive energy optimization should focus not only on hardware resources, but also on optimizing their usage by software applications. For this reason, accurate and efficient power analysis at the software-level is a key issue in the development of low-power systems on silicon.

Processors and memories are large hardware blocks, and their power consumption during software execution cannot be assessed by circuit-level or gate-level tools for obvious efficiency reasons. High-level power/energy estimation techniques [2], [3] based on macro-modeling can be used for software power estimation, by leveraging fast cycle-accurate HDL simulators. Unfortunately, HDL simulation speed for complex processor cores and memory systems is still insufficient to estimate the power/energy consumed by realistic applications. For this reason, higher-abstraction approaches for software-level power estimation have been proposed in the last few years. Probably, the best-known technique in this class is *instruction-level power analysis* [4], [24]. This approach defines a power consumption value for each instruction (or instruction pair) in the instruction set, and elaborates average power by weighted averaging of power costs with instruction execution frequency (obtained by instruction-level simulation).

Instruction-level power analysis has been successful in estimating power for relatively simple embedded cores (SPARC, ARM), as well as off-the-shelf processors. The main limitations of ILPA are: (i) it does not provide any insight on the causes of power consumption within the processor core, which is seen as a black box; (ii) it does not account for the power consumed in the memory system, which is often dominant. To address the second limitation, researchers have developed power estimation frameworks which integrate processor and memory models [5]–[8] and are built around instruction set simulators. Instruction set simulators produce both the instruction profiles for ILPA, and address traces to drive memory system simulators, augmented by memory power models [9]–[13]. These

integrated core-and-memory simulators are fast enough to run complex applications for millions of cycles. Their accuracy has not been fully validated for system-on-chip designs, but it has been shown to be satisfactory for board-level designs built with commercial off-the-shelf components [6].

The lack of insight on the sources of power consumption within the processor core has been recently addressed by a new generation of micro-architectural power estimation tools, targeting high-end processors with complex micro-architectures [14]–[16] and on-chip caches. The main purpose of these tools is to support exploration of micro-architectural tradeoffs in processor design, when energy (power) is one of the metrics of interest. Similarly to ILPA, these estimators are built around an instruction set simulator, but they feature a detailed micro-architectural model of the processor, with separate power models for its main functional units (e.g., execution units, pre-fetch buffers, register files, etc.). Analytic energy models for caches are also provided [12], where energy per access is automatically scaled depending on cache organization (e.g., number of cache lines, associativity, etc.). Micro-architectural power modeling is a tool for processor architects, aiming at exploring the design space of processor and cache organizations. Absolute accuracy with respect to the final implementation is not required, and it is hardly achievable, since the detailed circuit design and optimization are completed much later than architectural exploration.

Previous work on micro-architectural power estimation [22]–[26] has emphasized design exploration capabilities through the user of scalable power models, and power estimates have not been validated against data obtained from low-level simulation of a complete, placed and routed processor implementation. Hence, absolute accuracy of available micro-architectural estimators is still not fully assessed. On the contrary, our work emphasizes absolute accuracy and its quantitative assessment, as outlined in the following sections. The absolute accuracy of the model is needed to perform power optimization of software applications running on the given platform, that is ready for silicon and is used to develop the power macro-models.

3. Power Estimation Framework

In this section, we describe the proposed power estimation framework, based on a hierarchy of dynamic power estimation engines.

The cornerstone of our framework is the instruction-level power estimation (ILPE) module (see Figure 1). The ILPE module is based on an instruction set simulator (ISS) connected with a set of RT power macromodels and is characterized by a very high performance: 1.7 millions of bundles per second on average.

The ISS interprets the executable program by simulating and profiling the effects of each instruction on the main components of the architectural state of the system (e.g., the register and program counter values, the state of the memory hierarchy etc.) and provides a cumulative report of the main events occurred within the system (cache misses, mispredicted branches and other statistics on the instruction stream).

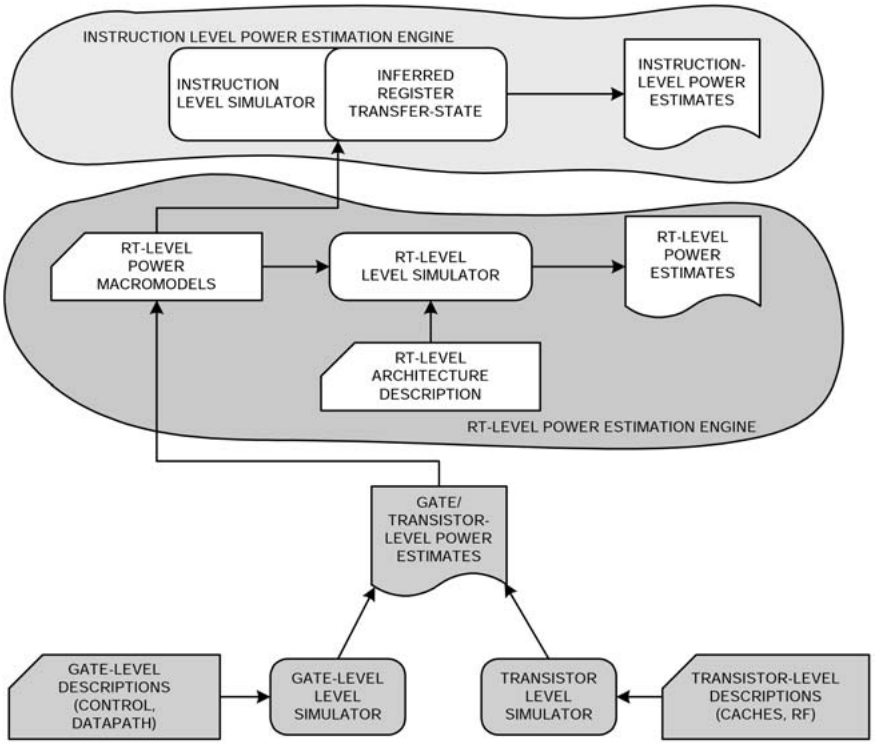

Figure 1. The power estimation framework.

Eventually, the ISS can be modified in order to derive a fast estimate of other RTL parameters (used as inputs for the RT power macro models) that would be otherwise not observed during a normal instruction set simulation. Among these parameters we can find the Hamming distance between the consecutive values on the cache buses and register file input and output ports.

Both the instruction level parameters and the inferred RT-level parameters can be elaborated at run-time by plugging the RT power model into the source code of the ISS) or *off-line*, i.e., by post-processing the statistics of the output report of the ISS. Obviously, in the first case, the information is processed, averaged and represented instruction-by-instruction (not cycle-by-cycle) giving an approximate figure of the instantaneous power consumption $P(t)$. In the second case (*off-line* elaboration or post-processing), the power consumption is computed at the end of the simulation time as an average value of the function $P(t)$.

The accuracy of the IL power estimation engine depends on how well the ISS infers the correct RT-state and must be traded off with the ISS speed. Experimental results have shown an average accuracy of approximately 5% of the IL engine with respect to the RTL engine while the performance improvement is of four orders of magnitude. As can be seen from Figure 1, the RTL power models are also embedded within a functional RTL

description of the core, written in Verilog, that is used as a reference for the instruction-level (IL) engine.

The RTL power macro-models have been characterized by either gate-level analysis (with back-annotation of wiring capacitances extracted from layout) for synthesized modules, or by transistor-level power analysis for post-layout full-custom modules, such as cache memory banks and RF. All macro-models are linked to a cycle-accurate RTL simulation model of the core through the standard PLI interface. Thus, RTL power estimation is obtained as a by-product of RTL functional simulation.

3.1. Target System Architecture

We applied our experimental framework to the scalable and customizable Lx processor technology [17] designed for multimedia and signal processing embedded applications. The Lx processor is a statically scheduled VLIW architecture designed by STMicroelectronics and Hewlett-Packard supports a multi-cluster organization based on a single PC and a unified I-cache. The single-cluster is 4-issue VLIW core composed of four 32-bit integer ALUs, two 16×32 multipliers, one load/store unit and one branch unit. The cluster also includes a register file of 64 32-bit general purpose registers and an 8 1-bit branch register file. The register file has 8 read ports and 4 write ports. Lx supports an in-order six-stage pipeline and a very simple integer RISC ISA. For the first generation, the scalable Lx architecture is planned to span from one to four clusters (i.e., from 4 to 16 issued instructions per cycle).

Lx comes with a commercial software toolchain, where no visible changes are exposed to the programmer when the core is scaled and customized. The toolchain includes a sophisticated ILP compiler technology (derived from the Multiflow compiler [21]) coupled with GNU tools and libraries. The Multiflow compiler includes most traditional high-level optimization algorithms and aggressive code motion technology based on trace scheduling.

The synthesizable RTL and the gate-level netlists of the processor core have been used to perform the power measurements of the core module. The experiments have been carried out by using Synopsys VCS 5.2 and a set of PLI routines to elaborate toggle statistics over the whole gate-level netlist. PowerCompiler, by Synopsys, has been used to combine the toggle statistics with the power models of the standard cells library provided by STMicroelectronics.

Finally, the instruction and data caches as well as the register file power models were characterized by simulating at the transistor level the corresponding full-custom layout descriptions with an extensive set of input patterns.

4. Power Macro-Modeling

In this section, we describe the macro-models developed to describe the power behavior of the main resources of the target system architecture, namely the VLIW core, the RF, and

the separated I- and D-caches. The main issues of the proposed power macro-models can be summarized as follows:

— they are tightly related to the micro-architectural details of each system module;

— they accurately consider the processor-to-memory communication in terms of read/write accesses to each level of the memory hierarchy;

— they can be used at both RTL and IL to estimate the power consumption.

4.1. VLIW Core Model

For VLIW architectures, an instruction-level energy model should account for all possible combinations of instructions (syllables) in a very long instruction (bundle), thus the problem complexity is $O(N^{2K})$ where N is the number of syllables in the ISA and K is the number of syllables in a bundle.

The analytical energy model proposed in this section aims at reducing the complexity of the instruction-level energy model we proposed in [19], while preserving a good level of accuracy in the estimates with respect to energy estimates derived from gate-level description of the core. The original power model is accurately calculated by looking at the constituent blocks of the processor by taking into account the effects that the single very long instruction (namely bundle) can produce on them. This level of detail cannot be achieved by using a simple black-box instruction-level energy model such as those presented in the literature so far. In fact, the original model decomposes the energy contributions of a single macro-block in the energy contribution of the macro-block functional units that work separately on each operation of the bundle. This property, introduced as the *spatial additive property*, is of fundamental importance to deal with the complexity of the *IL* power model for VLIW cores, which grows exponentially with the number of possible operations in the *ISA*. The proposed decomposition provides a way to create a mapping between bundles and micro-architectural functional units involved during the simulation. This instruction-to-unit mapping is used to retrieve energy information for each unit that is elaborated together with the stall and latency information to obtain run-time power estimates.

The model proposed in [19] starts by considering a stream $\mathcal{W}$ composed of N very long instructions:

$$\mathcal{W} = \langle \mathbf{w_1}, \ldots, \mathbf{w_{n-1}}, \mathbf{w_n}, \ldots, \mathbf{w_N} \rangle \tag{1}$$

where $\mathbf{w_n}$ represents the n-th very long instruction composed of K parallel operations carried out by multiple and independent functional units working in parallel:

$$\mathbf{w_n} = [w_n^1 \ldots w_n^k \ldots w_n^K]^T \tag{2}$$

where w_n^k is the k-th operation (issued on the k-th lane of the processor) of the n-th bundle of the stream.

The power model starts from the assumption that the energy associated with $\mathbf{w_n}$ is dependent on the properties of $\mathbf{w_n}$ (e.g., class of the instruction, data values involved in its evaluation and so on) as well as on its *execution context*, i.e., the preceding instruction $\mathbf{w_{n-1}}$ and the additional stall/latency cycles introduced during the execution of the instruction:

$$E(W) \approx \sum_{1 \le n \le N} \sum_{\forall s \in S} \left[U_s(0|0) + \sum_{\forall k \in K} \nu_s(w_n^k | w_{n-1}^k) + m_s^n * p_s^n * S_s + l_s^n * q_s^n * M_s \right]. \quad (3)$$

Where the term $U_s(\mathbf{0}|\mathbf{0})$ is the base energy cost that represents the energy consumed by stage s during an execution of a bundle constituted entirely by NOPs ($\mathbf{0} = [\texttt{NOP} \ \ldots \ \texttt{NOP}]^{\mathrm{T}}$), $\nu_s(w_n^k | w_{n-1}^k)$ is the additional energy contribution due to the change of operation on the same lane k, m_s^n (l_s^n) is the average number of additional cycles due to a data (instruction) cache miss during the execution of the $\mathbf{w_n}$ in s, p_s^n (q_s^n) is the probability that this event occurs, and S_s (M_s) is the energy consumption per stage of the processor modules that are active due to a data (instruction) cache miss.

To reduce the complexity of the model expressed by equation (3), while preserving its accuracy, some basic observations can be outlined. Although, in general, the target VLIW processor is a pipelined processor, in some cases it could be difficult to recognize within the processor database the evident structure of the pipeline and the modules belonging to the different pipeline stages. In these cases, it could be quite difficult to isolate the power contributions due to the different processor's modules. This aspect could represent a limit for the application of equation (3) but, as a matter of fact, the model can be reduced to manage also such cases. The term $\sum_s U_s(\mathbf{0}|\mathbf{0})$ corresponds to the power consumption of the core while it is executing NOPs and can be substituted by the average base cost $U(\mathbf{0}|\mathbf{0})$. Besides, $\sum_s \sum_k \nu_s(w_n^k | w_{n-1}^k)$ can be substituted by a cost dependent only on the pair of instructions $(\sum_k \nu(w_n^k | w_{n-1}^k))$ that corresponds to the energy consumption of the core while it is executing the same pair of instructions $(\mathbf{w_n}, \mathbf{w_{n-1}})$. Regarding instruction and data cache misses, we assume that after a transient state, the probabilities per stage (p and q) and their penalties (m and l) become stationary and can be averaged for each instruction of the stream. So we can assume that:

$$\sum_s (m_s^n * p_s^n * S_s + l_s^n * q_s^n * M_s) \rightarrow (m * p * S + l * q * M) \quad (4)$$

where $m(l)$ is the average data (instruction) cache miss length, $p(q)$ is the average probability per stage and per instruction that a data (instruction) cache miss can affect one instruction and S (M) are the average energy consumption of the whole processor during these events.

The model is thus reduced to:

$$E(\mathcal{W}) \approx \sum_{1 \le n \le N} \left[U(\mathbf{0}|\mathbf{0}) + \sum_{\forall k \in K} \nu(w_n^k | w_{n-1}^k) + m * p * S + l * q * M \right]. \tag{5}$$

Note that, for a k-issue VLIW core, the complexity of the model is quadratic with respect to the number of operations within the instruction set ($O(K * |ISA|^2)$), while a black-box model would have a complexity of $O(|ISA|^K)$. This leads to a reduction of the time necessary to perform the characterization, since the number of experiments to be done is reduced exponentially.

To further reduce the complexity of the model expressed by equation (5) we assumed that the inter-instruction effect due to the switching activity between two adjacent instructions is not as important as the one produced by stalls and misses within the core. This assumption can be confirmed by the analysis of the Lx architecture: almost the same modules (except for function units) are involved in the execution of two different long instructions. With this assumption, the term $\nu(w_n^k | w_{n-1}^k)$ can be rewritten as:

$$\nu(w_n^k | w_{n-1}^k) \approx \nu(w_n^k) = \begin{cases} \overline{\nu} & \text{if } w_n^k \neq \texttt{NOP} \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

where $\overline{\nu}$ is a fixed cost associated to an operation different from a `NOP`. The term $U(0|\mathbf{0})$ becomes also $U(\mathbf{0})$.

Under these assumptions, the model of equation (5) can be rewritten as:

$$E(W) \approx \sum_{1 \le n \le N} [U(\mathbf{0}) + \alpha_n * \overline{\nu} + m * p * S + l * q * M] \tag{7}$$

where α_n is the number of syllables different from NOPs within the bundle w_n. Globally, given equation (7), the average power associated with the stream W can be expressed as:

$$P(\mathcal{W}) = \frac{E(\mathcal{W})}{N * (1 + m * p + l * q) * T_c} \tag{8}$$

where T_c is the clock period. From equation (7) we can derive the final model:

$$P(\mathcal{W}) = (1 - f_S - f_M) \frac{(U(\mathbf{0}) + \alpha * \overline{\nu})}{T_c} + f_S \frac{S}{T_c} + f_M \frac{M}{T_c} \tag{9}$$

where T_c is the clock period, f_S is the fraction of time spent by the processor stalling the pipeline (i.e., $\frac{m*p}{1+m*p+l*q}$), f_M is the fraction of time spent by the processor during an I-cache miss (i.e., $\frac{l*q}{1+m*p+l*q}$), and $\overline{\alpha}$ is the average number of syllables per bundle different from NOPs. Globally, the average power is therefore linear with respect to three power

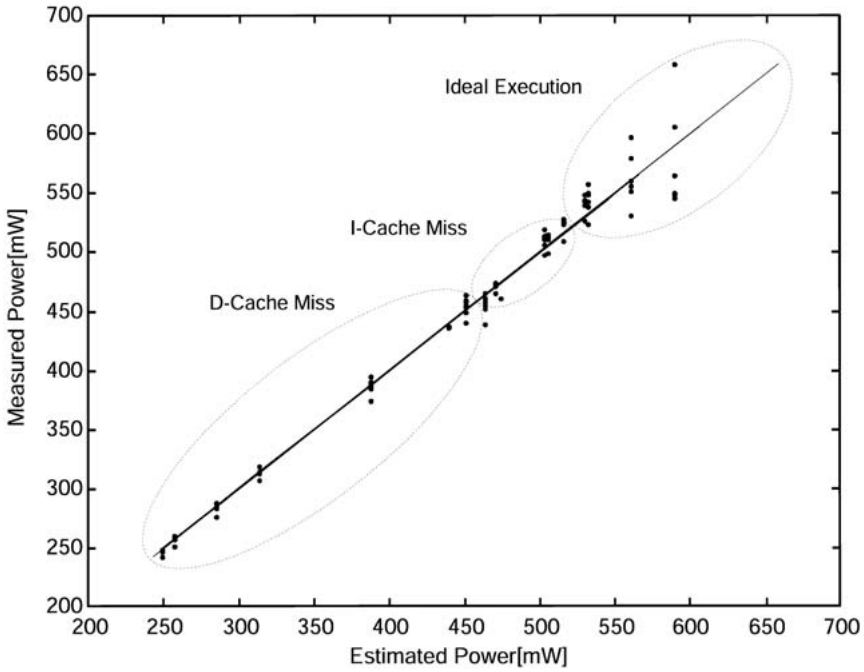

Figure 2. Agreement between measured Gate-Level vs. model-estimated power values for the Lx core (maximum error within ±10%).

contributions: the one-cycle-per-instruction ideal power consumption, the power due to a pipeline stall and the power due to an I-cache miss.

4.1.1. RT-Level Model Validation

The core model has been characterized by means of a gate-level simulation of a huge set of programs with very different instruction and data cache miss probabilities and instruction composition. Linear regression has been applied to the data-set to achieve this goal.

The macro-model for the core has then been validated against gate-level simulation on a new set of benchmarks. The agreement between predicted and measured power values is shown in Figure 2. The plot clearly illustrates three different regions where power consumption is dominated by D-cache misses, I-cache misses and ideal execution.

The power model has shown a maximum error of 10% with an RMS of 4.1%.

4.1.2. ISS Model Validation

Once the RT-level model has been validated, we can use this model to setup complex simulations of the entire LX processor in order to validate the ISS power estimation

Table 1. Comparison Between Instruction-Level Power Estimates and RTL Power Estimates for the Benchmark Set for the Core Processor

	gauss	fir1	fir2	fast_dct	fast_idct	dct_idct
ISS Power [W]	0.468	0.528	0.519	0.401	0.438	0.486
RTL Power [W]	0.439	0.497	0.536	0.448	0.467	0.436
Percentage Error	6%	6%	−3%	−12%	−7%	10%

engine. For this purpose, we used a set of common benchmarks (the same used in Section 5) and, for each benchmark, we performed both RT-level and instruction-level simulations. Then we compared the power consumption results as reported in Table 1, which represents the accuracy obtained by the ISS power model compared to RTL power model. We can observe that the maximum error is approximately −12% while the average error is around 0%.

4.2. Register File Model

The general problem of evaluating the power consumption of RFs has recently been addressed in [20]. The paper compares various RF design techniques in terms of energy consumption, as a function of architectural parameters such as the number of registers and the number of ports.

In our work, we propose a parametric power model of a multi-ported RF: the power behavior is linear with respect to the number of simultaneous read/write accesses performed on the different ports:

$$P_{RF} = P_i + \frac{1}{T} \sum_{1 \le n \le N} (E_{r,n} + E_{w,n})$$

where P_i is the RF base power cost measured when neither read nor write accesses are performed, T is the total simulation time, $E_{r,n}$ ($E_{w,n}$) is the energy consumption of a read (write) access occurred during bundle w_n, and f_S has been defined above.

The energy contribution $E_{r,n}$ is defined as:

$$E_{r,n} = \sum_{1 \le i \le N_{rp}} H(RR_{i,n},\, RR_{i,n-1}) * E_{rb}$$

where N_{rp} is the number of read ports of the RF, H is the Hamming distance function, $RR_{i,k}$ is the data value read from the RF output port i by the k-th bundle and E_{rb} is the energy consumption associated with a single bit change on a read port.

The energy contribution $E_{w,n}$ is defined as:

$$E_{w,n} = \sum_{1 \le i \le N_{wp}} H(RW_{i,n},\, old_{i,n}) * E_{wb}$$

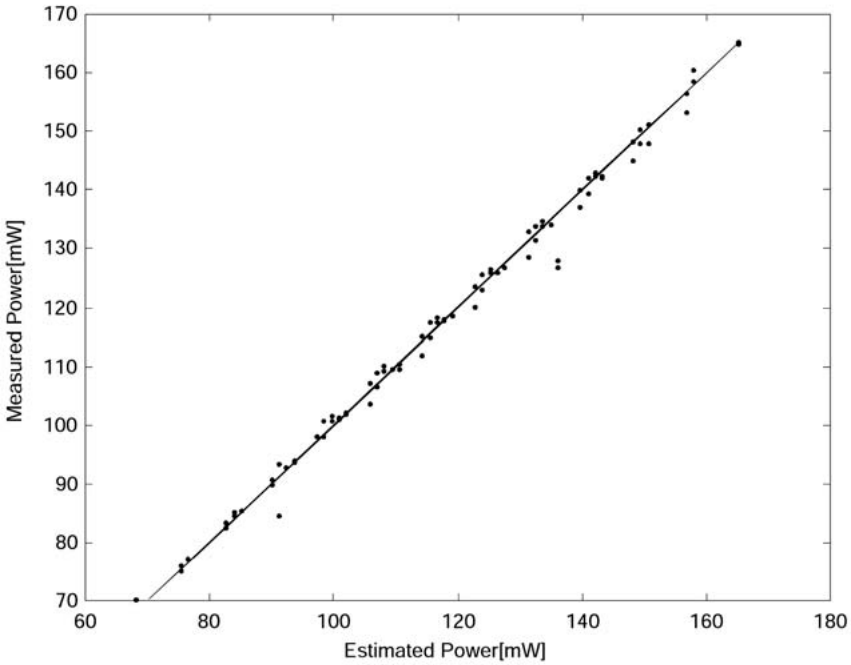

Figure 3. Agreement between measured transistor-level power values and estimated power values for the register file (maximum error within ±8%).

where N_{wp} is the number of write ports of the RF, H is the Hamming distance function, $RW_{i,n}$ is the new data value written by the n-th bundle on input port i, $old_{i,n}$ is the previous data value contained in the same RF location and E_{wb} is the energy consumption associated with a single bit change on a write port.

4.2.1. RT-Level Model Validation

Model characterization and validation has been carried out by transistor-level simulation of the register file circuit extracted from layout including parasitics. The simulation has been performed by generating a set of sequences of input vectors characterized by different switching activity on the read and write ports.

The agreement between predicted and measured power values is shown in Figure 3 (maximum error of 8%, RMS of 1.75%).

4.2.2. ISS Model Validation

Similarly to the core model validation, we compared the accuracy of the RT-level and instruction-level simulations for the Register File. The results are reported in Table 2, which represents the accuracy obtained by the ISS power model compared to RTL power model.

Table 2. Comparison Between Instruction-Level Power Estimates and RTL Power Estimates for the Benchmark Set for the Register File

	gauss	fir1	fir2	fast_dct	fast_idct	dct_idct
ISS Power [W]	0.088	0.115	0.121	0.075	0.071	0.09
RTL Power [W]	0.089	0.098	0.115	0.087	0.09	0.09
Percentage Error	−1%	15%	5%	−16%	−27%	0%

We can observe that the maximum error is approximately −27% while the average error is around 4%. The maximum error can be due to the fact that the ISS does not infer the switching activity associated with the accesses to the specific register file ports, but only an average value depending on the type of instructions. In particular, the maximum error found can be considered acceptable in our target system architecture since, as we can see in Section 5, the contribution of the Register File to the overall system power is maintained within 5%.

4.3. Cache Model

Most published analytic cache models [12], deal with relatively simple cache organizations, and they are not suitable for modeling complex cache architectures based on multiple SRAM memory banks, with a significant amount of control logic. The multi-banked structure is dictated mainly by performance constraints, because cache access time is critical for overall processor performance.

The modeling approach proposed in this paper is hierarchical: We first built power macro-models for all the various types of SRAM banks contained in the caches, then we compose these models in a single logical model that generates the correct access patterns for every bank according to the cache organization. Composition of the atomic macro-models in the complete cache model is trivial at the RT level, because the RTL description of the cache subsystem does contain the behavioral description of every SRAM module. Building the cache model for instruction-level simulation is not as straightforward, because the ISS simply provides cache accesses per instructions, but it does not infer any knowledge of internal cache organization. Hence, we re-constructed the pattern of accesses to the various SRAM sub-modules in response to every type of cache access. Unfortunately, it is not possible to fully reconstruct SRAM access patterns from cumulative counts, and some loss of accuracy occurs in the process.

The instruction cache is a 32Kb direct mapped architecture reported in Figure 4(a). This topology enables the extraction of an entire bundle (4 syllables) from the cache memory in the same clock cycle, in case of an I-cache hit. Bit 14 of the address distinguishes the two banks and enables to activate only the bank needed, the others remaining in the sleep state. The 32Kb D-cache (shown in Figure 4(b)) is a 4-way set associative structure with a FIFO replacement policy. Memory blocks used in the D-cache enable data write of 64-bit blocks and data read of 32-bit blocks at a time. This architecture has been chosen by the designers to increase block replacement speed by exploiting the burst access capability of the main memory.

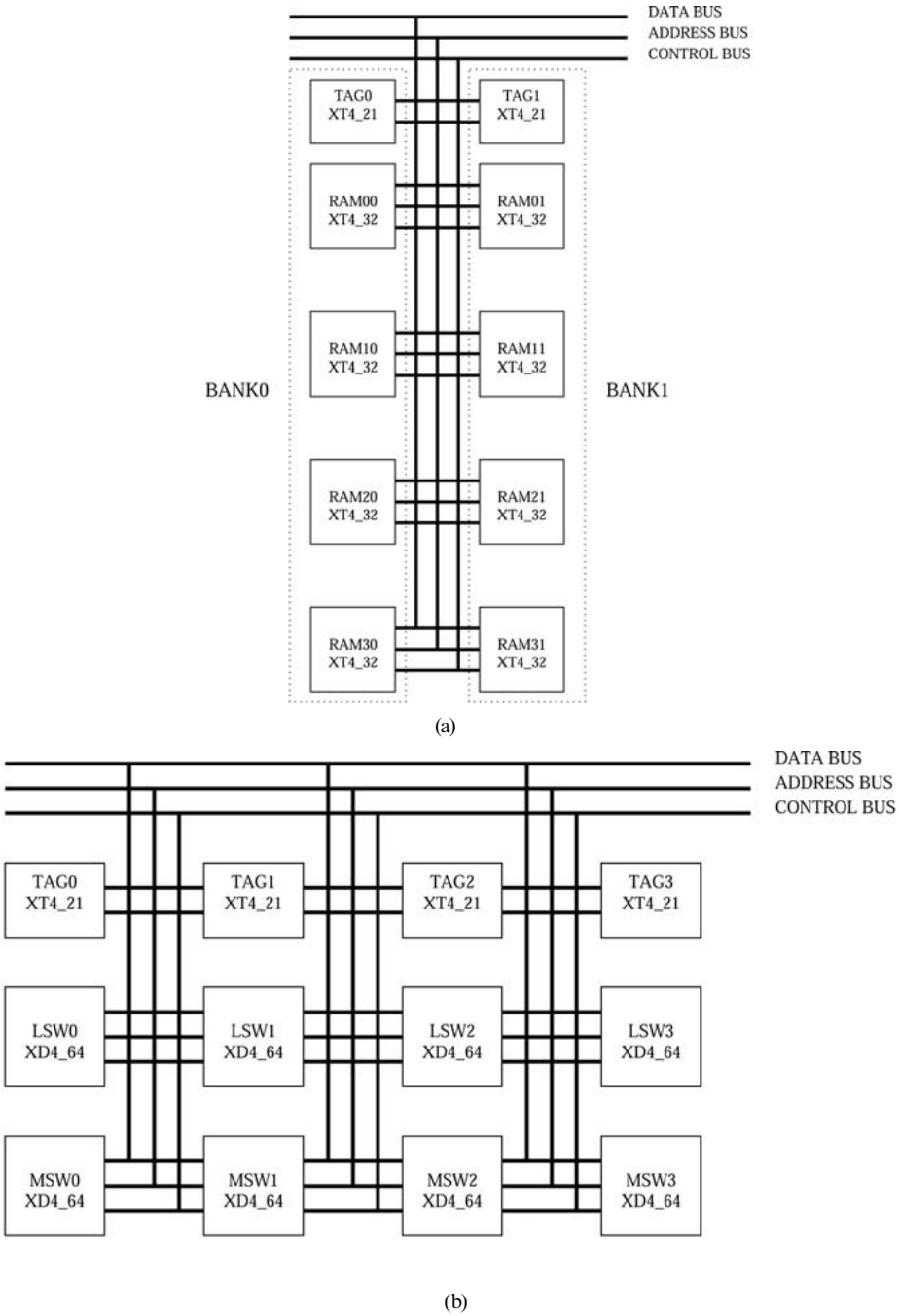

Figure 4. Simplified structures of (a) I-Cache and (b) D-Cache.

4.3.1. RTL Power Models for Caches

The macro-models for the atomic SRAM modules are *mode-based*: power consumption depends on the mode of operation (i.e., `read`, `write`, `idle`). More precisely, since the SRAM modules are synchronous, the energy consumed in a given clock cycle is mainly a function of the *mode transition* between the previous and the current cycle. Thus, we characterized energy as a function of the nine possible mode transitions (e.g., `read`−`read`, `read`−`write`, etc.). For a given mode transition, energy is weakly dependent on the number of transitions on the address lines. Accounting for this dependency leads to a macro-models with $9 \cdot (N_{addr} + 1)$ characterization coefficients, where N_{addr} is the number of address lines. Thus, we can model the energy consumed by cache modules with the following equations:

$$E_{tag} = E_s N_s + \sum_{n=0}^{8} \sum_{x=s,r,w} \sum_{y=r,w} E_{xy}[n] N_{xy}[n] \tag{10}$$

$$E_{Imem} = E_s N_s + \sum_{n=0}^{10} \sum_{x=s,r,w} \sum_{y=r,w} E_{xy}[n] N_{xy}[n] \tag{11}$$

$$E_{Dmem} = E_s N_s + \sum_{n=0}^{10} \sum_{x=s,r,w} \left\{ E_{xr}[n] N_{xr}[n] + \sum_{B=0}^{8} E_{xw}[B,\ n] N_{xw}[B,\ n] \right\} \tag{12}$$

where E_s is the energy consumed during an idle cycle for the selected module, N_s is the number of idle cycles during the entire RTL simulation, the parameter $E_{xy}[n]$ is the energy consumed during one access operation depending on the number $[n]$ of bits that change, the current cycle operation y and the access operation x, performed during the previous cycle.

For the D-cache, E_{Dmem} contains an additional parameter $(\sum_{B=0}^{8} E_{xw}[B,\ n] N_{xw}[B,\ n])$, because the D-cache line can be written byte per byte with a bus composed of byte selection signals. In such case, the term $E_{xw}[B,\ n]$ depends on both the previous access address n and the current byte selection B.

This type of model can be efficiently implemented into the Verilog description, with a Look-Up Energy table. Each row of this table can be selected with an access code that depends on the parameters x, y and n previously described. The coefficients has been characterized by simulating the back-annotated transistor-level netlist of the SRAM modules with the MACH-PA circuit simulator by Mentor Graphics. Average accuracy of the SRAM macro-models can be considered satisfactory (percentage average errors are within 5%), as shown in the next sub-section.

4.3.2. RT-Level Model Validation

The scatter plot of Figure 5 represents the energy consumption evaluated at the RT-level compared to the energy measured with MachPA. Since an electrical simulation of the entire processor is impractical, for the validation we use the same decomposition approach used in the characterization phase. The scatter plot of Figure 5 show the agreement between the RT-level and transistor level power measurements.

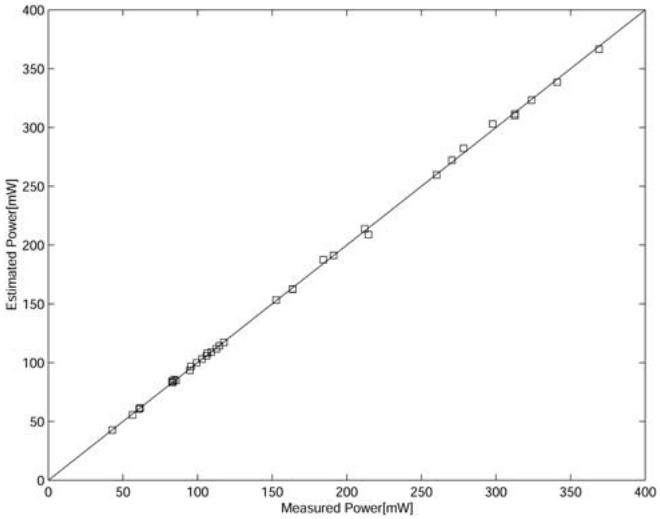

Figure 5. Agreement between measured transistor-level power values and estimated RT-level values for the cache blocks.

4.3.3. ISS Level Power Models for Caches

The ISS profiles the accesses to the cache resources. As the Figure 4(a) shows, the I-cache is split into two sub blocks: $bank0 = \{ram00 \ldots ram30\}$ and $bank1 = \{ram01 \ldots ram31\}$ associated with tag modules $tag0$ and $tag1$. In the first cycle of a hit or a miss, we have a read access on both $tag0$ and $tag1$, a read on one of the two *ram* sets (say ram_x) and an idle on the other *ram* bank (say ram_y). In the case of a miss the cache is refilled with the behavior represented in the Table 3.

To model the power consumption of I-cache, we consider the average energy consumed by the module for a given type of access. Since the ISS is instruction-accurate, we do not have the visibility of access addresses on a cycle-by-cycle basis, as in the RTL model. For this reason, we use the average values of associated with the given type of access (`sleep`, `read`, `write`) (gathered from the Look Up Energy table).

The power consumption in D-cache is computed in a similar fashion. The main difference is the capability to perform writes in 64bit blocks during a cache refill, and the different topology of the architecture that is 4-way set associative. In this case, we distinguish only between read or write accesses. The behavior of the data cache is summarized in Table 4.

4.3.4. ISS Model Validation

Table 5 represents the accuracy obtained by the ISS power model compared with the RTL power model. The maximum error in the estimation D-cache power consumption is approximately 18%. This value is due to the fact that the D-cache has a very complex behavior compared to the I-cache and it is more difficult to gather the parameters needed for the model.

Table 3. The Refill Behavior of the I-Cache

Cycle	ram_x	ram_y	tag_x	tag_y
1–33	read	idle	read	read
34	2 band write, 2 bank read	idle	read	read
35	read	idle	read	read
36	2 bank write, 2 bank read	idle	read	read
37	read	idle	read	read
38	2 bank write, 2 bank read	idle	read	read
39	read	idle	read	read
40	2 bank write, 2 bank read	idle	read	read
41–45	read	idle	read	read
46	2 bank write, 2 bank read	idle	read	read
47	read	idle	read	read
48	2 bank write, 2 bank read	idle	read	read
49	read	idle	read	read
50	2 bank write, 2 bank read	idle	read	read
51	read	idle	read	read
52	2 bank write, 2 bank read	idle	write	read

Table 4. The Behavior of the D-Cache

Event	Cycle	Tag	Bank	Comment
Read hit	1	4 tags read	8 banks read	
Write hit	1	4 tags read	1 banks read	
Read miss	1	4 tags read	8 banks read	(such as read hit)
	2	4 tags read	2 banks read	dirty bit extract
	4	/	4 banks write (64bit switching)	refill action
	5	1 tag write	/	TAG write
	6	4 tag read	8 bank read	pending read action
Write miss	1	4 tags read	/	
	2	4 tags read	2 banks read	dirty bit extract
	4	/	4 banks write (64bit switching)	refill action
	5	1 tag write	/	TAG write
	6	4 tag read	8 bank read	pending write action
	7	/	1 bank write	pending write action

Table 5. Comparison Between Instruction-Level Power Estimates and RTL Power Estimates for the Benchmark Set for the Caches

	gauss	fir1	fir2	fast_dct	fast_idct	dct_idct
I-cache ISS Power [W]	0.635	0.640	0.636	0.626	0.618	0.635
I-cache RTL Power [W]	0.614	0.619	0.617	0.589	0.602	0.617
Percentage Error	3.4%	3.4%	3.1%	6.3%	2.7%	2.9%
D-cache ISS Power [W]	0.430	0.766	1.05	0.314	0.284	0.398
D-cache RTL Power [W]	0.429	0.774	0.905	0.266	0.265	0.433
Percentage Error	0.2%	−1.0	16.0%	18.0%	7.2%	−8.1%

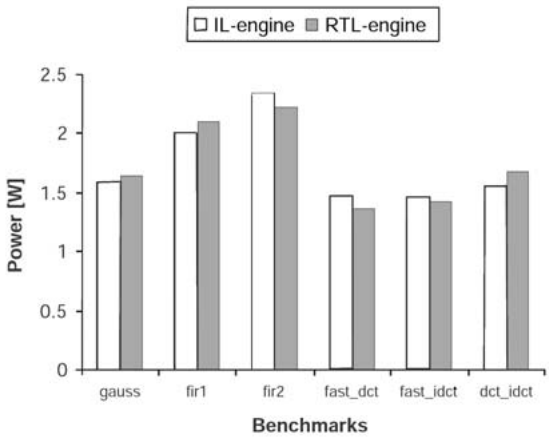

Figure 6. Comparison between Instruction-Level power estimates and RTL power estimates for the benchmark set. (Average Error 5.2%—Maximum Error 7.9%.)

5. Experimental Results

In this section, we show some experimental results derived with the instruction-level power estimation engine proposed in this work.

The IL engine is based on the ISS available in Lx toolchain. The Lx ISS has been purposely modified to gather a fast estimate of the RTL status and parameters. These values are then linked to the power macro-models to get power estimates. The experiments have been carried out over a set of selected benchmark applications including *C* language implementation of digital filters, discrete cosine transforms, etc., especially tuned for the Lx processors.

Figure 6 shows the comparison results between the IL power estimates with respect to RTL estimates based on the same models. For the benchmark set, the average error (IL vs. RTL) is 5.2%, while the maximum error is 7.9%. On a Sun Ultra Sparc 60 at 450 MHz (1 GB RAM), the RTL engine simulates 160 bundles per second on average, while the IL engine simulates 1.7 millions of bundles per second on average, thus providing a speed-up of four orders of magnitude approximately.

Figure 7 shows an example of the ISS-based static power profiling for the same benchmarks of above. As can be seen from the figure, the power consumption associated to the caches is more than a half of the total power consumption. The contribute of the D-cache is however highly dependent on the benchmark while the I-cache power consumption is almost stable. This is due to the fact that the I-cache is accessed in every cycle, even during an I-cache miss, while the D-cache is not enabled during inactivity periods.

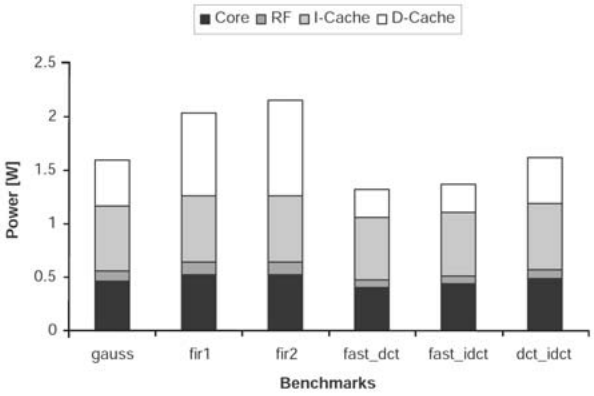

Figure 7. Break-out of the RTL power contributions due to core, RF, I-cache, and D-cache for the benchmark set.

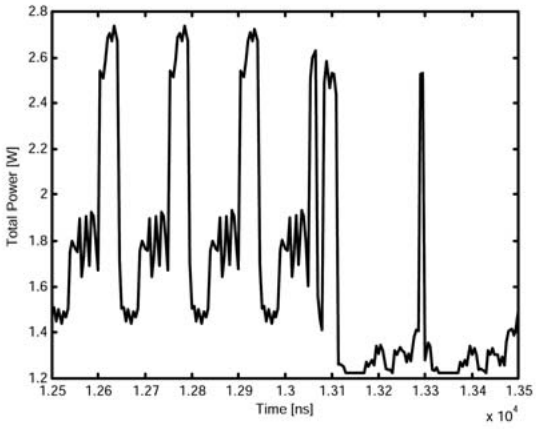

Figure 8. An example of ISS-based system-level dynamic power profiling capability.

Figure 8 shows an example of ISS-based dynamic power profiling applied to a FIR filter. During the first 600 ns there is a high total power consumption that is due to an intensive D-cache activity (in this phase the data is fetched from the memory hierarchy into the registers). The remaining part of the plot shows a period in which the data is elaborated and only the core, the RF and the I-cache consume power.

6. Conclusions

In this paper, we have presented an efficient and accurate framework for embedded core power modeling and estimation. The method is based on a hierarchy of dynamic power estimation engines: from the instruction-level down to the gate/transistor-level. Power macro-models have been developed for the main components of the system: the VLIW core, the register file, the instruction and data caches. The main goal consisted in defining a system-level simulation framework for the dynamic profiling of the power behavior during the software execution, providing also a break-down of the power contributions due to the single components of the system. The proposed approach has been applied to the Lx family of scalable embedded VLIW processors, jointly designed by STMicroelectronics and HP. Experimental results, carried out over a set of benchmarks for embedded multimedia applications, have demonstrated an average accuracy of 5% of the instruction-level estimation engine with respect to the RTL engine, with an average speed-up of four orders of magnitude. Future directions of our work aim at defining: (i) power efficient instruction scheduling opportunities, (ii) a more general exploration methodology to evaluate power-performance tradeoffs of a given set of software applications running on the target platform at the system-level, and (iii) techniques to optimize the software code from the power standpoint. The last point consists in developing some power optimization techniques that can decrease the average number of operation per bundle (eventually increasing the latency) in order to minimize the power consumption.

References

1. Nishitani, T. Low-Power Architectures for Programmable Multimedia Processors, *IEICE Transactions Fundamentals*, vol. E82-A, no. 2, pp. 184–196, Feb. 1999.
2. Landman, P. High-Level Power Estimation. In *Proceedings of ISLPED*, pp. 29–35, 1996.
3. Macii, E., M. Pedram, and F. Somenzi. High-Level Power Modeling, Estimation and Optimization, *IEEE Transactions on CAD*, vol. 17, no. 11, pp. 1061–1079, 1998.
4. Tiwari, V., S. Malik, and A. Wolfe. Power Analysis of Embedded Software: A First Step Towards Software Power Minimization, *IEEE Transactions on VLSI Systems*, vol. 2, no. 4, pp. 437–445, Dec. 1994.
5. Li, Y. and J. Henkel. A Framework for Estimating and Minimizing Energy Dissipation of Embedded HW/SW Systems. In *Design Automation Conference*, pp. 188–193, June 1998.
6. Simunic, T., L. Benini, and G. De Micheli. Cycle-Accurate Simulation of Energy Consumption in Embedded Systems. In *Design Automation Conference*, pp. 867–872, June 1999.
7. Lajolo, M., A. Raghunathan, S. Dey, and L. Lavagno. Efficient Power Co-Estimation Techniques for System-on-Chip Design. In *Design, Automation and Test in Europe Conference*, pp. 27–34, March 2000.
8. Givargis, T., F. Vahid, and J. Henkel. Fast Cache and Bus Power Estimation for Parameterized System-on-Chip Design. In *Design Automation and Test in Europe Conference*, pp. 333–338, March 2000.
9. Liu, D. and C. Svensson. Power Consumption Estimation in CMOS VLSI Chips, *IEEE Journal of Solid-State Circuits*, vol. 29, no. 6, pp. 663–670, June 1994.
10. Evans, R. and P. Franzon. Energy Consumption Modeling and Optimization for SRAM's, *IEEE Journal of Solid-State Circuits*, vol. 30, no. 5, pp. 571–579, May 1995.
11. Landman, P. and J. Rabaey. Architectural Power Analysis: The Dual Bit Type Method, *IEEE Transactions on VLSI*, vol. 3, no. 2, pp. 173–187, June 1995.

12. Kamble, M. and K. Ghose. Analytical Energy Dissipation Models for Low Power Caches. In *International Symposium on Low Power Electronics and Design*, pp. 143−148, August 1997.
13. Coumeri, S. and D. Thomas. Memory Modeling for System Synthesis, *IEEE Transactions on VLSI*, vol. 8, no. 3, pp. 327−334, June 2000.
14. Ye, W., N. Vijaykrishna, M. Kandemir, and M. Irwin. The Design and Use of SimplePower: A Cycle-Accurate Energy Estimation Tool. In *Design Automation Conference*, pp. 340−345, June 2000.
15. Brooks, D., V. Tiwari, and M. Martonosi. Wattch: A Framework for Architectural-Level Power Analysis and Optimization. In *International Symposium on Computer Architecture*, pp. 83−94, June 2000.
16. Brooks, D. et al. Power-Microarchitecture: Design and Modeling Challenges for Next-Generation Microprocessors, *IEEE Micro*, vol. 20, no. 6, pp. 26−44, Nov./Dec. 2000.
17. Faraboschi, P., G. Brown, J. Fisher, G. Desoli, and F. Homewood. Lx: A Technology Platform for Customizable VLIW Embedded Processing. In *International Symposium on Computer Architecture*, pp. 203−213, June 2000.
18. Zafalon, R., M. Rossello, E. Macii, and M. Poncino. Power Macromodeling for a High Quality RT-Level Power Estimation. In *ISQED 2000: IEEE International Symposium on Quality Electronic Design*, San Jose, CA, pp. 59−63, March 2000.
19. Sami, M., D. Sciuto, C. Silvano, and V. Zaccaria. Power Exploration for Embedded VLIW Architectures. *ICCAD-2000: IEEE/ACM Int. Conference on Compuer Aided Design*, San Jose, CA, pp. 498−503, Nov. 5−9, 2000.
20. Zyuban, V. and P. Kogge. The Energy Complexity of Register Files. In *International Symposium on Low Power Electronics and Design*, pp. 305−310, August 1998.
21. Lowney, P. G. The Multiflow Trace Scheduling Compiler, *Journal of Supercomputing*, vol. 7, no. 1/2, pp. 51−142.
22. Vijaykrishnan, N., M. Kandemir, M. Irwin, H. Kim, and W. Ye. Energy-Driven Integrated Hardware-Software Optimizations Using SimplePower. In *ISCA 2000: 2000 International Symposium on Computer Architecture*, Vancouver BC, Canada, 2000.
23. Brooks, D., V. Tiwari, and M. Martonosi. Wattch: A Framework for Architectural-Level Power Analysis and Optimizations. In *ISCA 2000: 2000 International Symposium on Computer Architecture*. Vancouver BC, Canada, pp. 83−94, 2000.
24. Lee, M. T.-C., V. Tiwari, S. Malik and M. Fujita. Power Analysis and Minimization Techinques for Embedded DSP Software, *IEEE Trans. VLSI System*, vol. 5, no. 1, pp. 123−135, Mar. 1997.
25. Joseph, R. and M. Martonosi. Run-Time Power Estimation in High Performance Microprocessors. In *Low Power Electronics and Design, International Symposium on, 2001*, pp. 135−140, August 2001.
26. Stanley-Marbell, P. and M. S. Hsiao. Fast, Flexible, Cycle-Accurate Energy Estimation. In *Low Power Electronics and Design, International Symposium on, 2001*, pp. 141−146, August 2001.