Processor Power Estimation Techniques: A Survey

Hameedah Sultan

Department of Electrical Engineering
Indian Institute of Technology,
New Delhi, India
E-mail: jvl132811@ee.iitd.ac.in

Gayathri Ananthanarayanan

School of Information Technology
Indian Institute of Technology,
New Delhi, India
E-mail: gayathri@cse.iitd.ac.in

Smruti R. Sarangi

Computer Science and Engineering,
Indian Institute of Technology,
New Delhi, India
E-mail: srsarangi@cse.iitd.ac.in

Abstract: Since the end of the nineties, power dissipation has been regarded as a first order design constraint in processors. Increased power dissipation along with the resultant rise in die temperature is considered as the single largest bottleneck for increasing processor frequency and complexity. Consequently, it is very important from both a technical as well as commercial perspective to accurately estimate processor power such that designers can tailor their architecture, software and systems to minimize power consumption. In this paper, we provide a survey of most of the processor specific power estimation techniques proposed after the mid nineties. In specific, we look at estimating power both at design time as well as runtime. The former approach is more suitable for early stage architectural exploration, and the latter approach is more germane to creating power efficient application software. We broadly focus on estimating power using system level models, architectural simulation, hardware performance counters, on-chip temperature profiles, and program execution profiles. We showcase a broad range of methods for power estimation using simulators, compilers, profilers, and sophisticated mathematical analysis routines.

Keywords: Survey of Power Estimation, Performance Counters, Architectural Tools, SystemC

Biographical notes: Hameedah Sultan is pursuing her Master's degree in VLSI Design Tools and Technology from the Indian Institute of Technology, Delhi, India.
Gayathri Ananthanarayanan is pursuing her Ph.D degree in the School of Information Technology, IIT Delhi.
Dr. Sarangi obtained his Ph.D in computer architecture from the University of Illinois at Urbana Champaign(UIUC), USA in 2006, and a B.Tech in computer science from IIT Kharagpur in 2002. He has subsequently worked in Synopsys Research, and IBM Research Labs. Currently, he is an Assistant Professor in the Computer Science Department at IIT Delhi. His primary interests are in computer architecture, and parallel algorithms. He is a member of both the IEEE and ACM.

1 Introduction

With the advent of large multicore processors, and a wide variety of embedded processors for tablets and smart phones, power consumption has become a very crucial factor in the design and marketability of processors. Consequently, research efforts in both industry and academia are trying to scale up to provide solutions to mitigate the problem of high power consumption. Over the past ten years, researchers have proposed a plethora of schemes in the area of power optimization and management for a wide range of processors. Given the importance of the problem from both a technical as well as commercial angle, we expect that research efforts to reduce power consumption will continue to be very important in the future also. Secondly, over the last

decade, the related issue of temperature management is becoming increasingly important primarily because of the fact that high on-chip temperatures have grossly deleterious effects in terms of lifetime reliability and leakage power.

This survey paper focuses on a very key aspect of research in processor power optimization and management namely *power estimation*. It is important to have a very accurate power estimation framework because of several technical and commercial reasons. If we are able to quickly estimate the power of an architecture for a suite of benchmarks, then we can rapidly sift through a large number of designs and make optimal choices. This can be done well before the actual processor is designed, fabricated, and tested. Likewise, for software, we can choose appropriate algorithms. From a commercial point of view, accurate power estimation in the design stage avoids costly re-design cycles. Secondly, to carry out power optimization, a rapid estimate of the power consumption is needed. Hence, a good power estimation methodology can possibly lead to a product with better power consumption characteristics, and thus ensure higher profitability.

Consequently, researchers have been looking at a multitude of techniques to estimate power at both the level of software as well as hardware. Some of the simple approaches include measuring the power through digital multimeters, loop ammeters, and embedded power counters. However, because of the limited applicability and accuracy of these methods, a more comprehensive and extensive body of research work has emerged over the last decade. We provide some background information as well as a broad taxonomy of approaches in Section 2.

Researchers have focused on accurate architecture level power modeling from very high level architectural specifications and models. These approaches predict general trends in power consumption. They have also proved their worth in being credible tools for research in computer architecture as well as early stage design exploration as described in Sections 3 and 4. Since these are the only tools available at the design stage, so, although they do not offer very good accuracy, these methods are indispensable to avoid re-design cycles and reduce the time to market.

To estimate the power consumption of an application more accurately, it is possible to use the information provided by performance counters. Performance counters are registers built into processors that store hardware activity-related statistics. Approaches based on performance counters have been shown to have an error limited to 5% for some benchmarks (see Section 5).

It is further possible to estimate power, especially leakage power, by using a temperature map of a chip as described in Section 6. Lastly it is possible to estimate the power of an application by a combination of power characterization, profiling, and static program analysis. These predominantly compiler and profiler driven approaches are described in Section 7. Finally, in Section 8, we provide a comparison of the various approaches discussed as well as the positive and negative aspects of each technique. We conclude by enunciating a set of possible future research directions in this field in Section 9.

2 Overview

2.1 Preliminaries

2.1.1 Dynamic Power

In a CMOS inverter, the energy stored across a capacitor is equal to $\frac{1}{2}CV^2$, where C is the load capacitance. In a complete charge/discharge cycle the total amount of energy dissipated is CV^2. Since most circuits do not have a charge and discharge event every cycle, this effect is quantified by introducing an extra term namely the activity factor, α. The maximum total power is obtained by multiplying this value with the clock frequency and the energy dissipated in a complete charge/discharge cycle is shown in Equation 1.

This equation holds for all kinds of circuits from small transistors to large functional units. We just need to calculate the load capacitance, C, to accurately use this equation.

$$P_{dynamic} = \alpha C V^2 f \tag{1}$$

2.1.2 Static Power

Because of the increasing amount of miniaturization in transistor technology, some amount of current continually leaks through transistors even if they are in the *off* state. This is known as the static or leakage power and is estimated to be about 30-50% of the overall power in state of the art technologies (45 nm and beyond) (Borkar, 2001). Researchers have found several sources of static power (see Figure 1). We list some of them here.

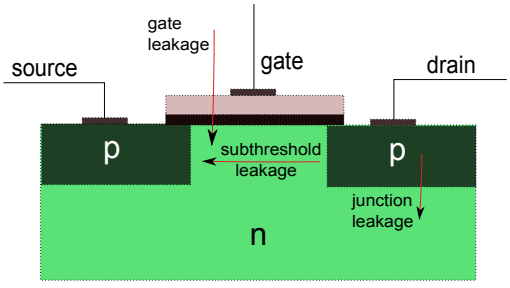


Figure 1: Different types of leakage Power

Subthreshold leakage happens when the gate to source voltage is less than the threshold voltage of the transistors. Ideally, the current flow should be zero. However, some amount of current flows between the drain and the source. An equally important source

of leakage is *gate leakage*, wherein, some amount of current flows between the gate and the channel due to quantum mechanical tunneling (Fowler-Nordheim tunneling). Lastly, for nanometer scale CMOS circuits, *junction leakage* is emerging as an important source of leakage power. In this case, some current leaks from the drain or source to the substrate, when the associated p-n junction is in reverse bias. There are some other sources of leakage such as gate induced drain leakage, and hot carrier injection, which we do not discuss here.

All the sources of leakage can be cumulatively summed up using the BSIM3 (Cheng and Hu, 1999) leakage equation given by Equation 2.

$$P_{static} = A \times \nu_T^2 \times e^{\frac{V_{GS}-V_{th}-V_{off}}{n \times \nu_T}} \left(1 - e^{\frac{-V_{DS}}{\nu_T}}\right) \quad (2)$$

Here, A is a constant of proportionality denoting the area of the transistor, ν_T is the thermal voltage (kT/q), V_{GS} is the gate to source voltage, V_{th} is the threshold voltage, V_{off} is a fixed offset voltage, n is a swing constant, and V_{DS} is the drain to source voltage, k is the Boltzmann constant, and q is equal to 1.6×10^{-19} coulombs. Now, the threshold voltage, V_{th}, is also a function of the temperature. V_{th} at temperature T is equal to $V_{th}^0 - k_1 \Delta T$. Here, V_{th}^0 is the threshold voltage at temperature T_0. k_1 is typically equal to $2.6mV/°C$ (Martin et al., 2002).

Equation 2 can be slightly simplified and approximated to yield Equation 3.

$$P_{static} \propto T^2 \times e^{\frac{qk_1\Delta T - qV_{th}^0}{kT}} \quad (3)$$

Note that the **leakage power is exponentially dependent on the temperature differential**. Consequently, we can conclude that the static component of power is extremely sensitive to changes in temperature, and thus estimating temperature is vitally essential in estimating power.

2.1.3 Short-Circuit Power

When a typical CMOS gate switches between the on-off states, there is a short period of time in which both the transistors are in the linear region of operation. At this point of time there is some current flow and resulting power dissipation. This short circuit power is typically modeled through a set of equations proposed by Nose and Sakurai (2000).

2.2 Power Estimation and Measurement

Formal definition of the problem

The problem of power estimation is defined as the task of calculating the power that a processor will consume for a certain program or representative input, given the characteristics of the processor, structure of the reference program, and possibly details about its execution trace. Note that we do not differentiate between a real processor and the model of a processor unless stated explicitly. We refer to this program as the *test program* henceforth.

It is necessary to estimate the power from secondary sources of information because there are numerous practical difficulties in trying to measure it directly. The most common solution (Isci and Martonosi, 2003; Joseph and Martonosi, 2001; Bellosa, 2000) is to calculate the power dissipation by measuring the current at the mains and multiplying it by the supply voltage. However, in this case we are measuring the power consumed by the entire system inclusive of the processor, motherboard, hard disk, and the peripheral I/O cards. We can refine this approach by tapping the power lines that go to the CPU on the motherboard (see Bertran et al. (2010)). However, this approach requires invasive changes to the motherboard, and does not work if there are other units like the graphics processor in the CPU package. To ameliorate this problem, the Intel Sandybridge processor, which includes an on-chip graphics card provides a power counter (Rotem et al., 2011). Unfortunately, the power counter provides the power usage for all the cores in a CMP, has limited applicability, and cannot be used in the design stage.

We desire to have a vast portfolio of approaches that can be used in either the design stage, or in the field.

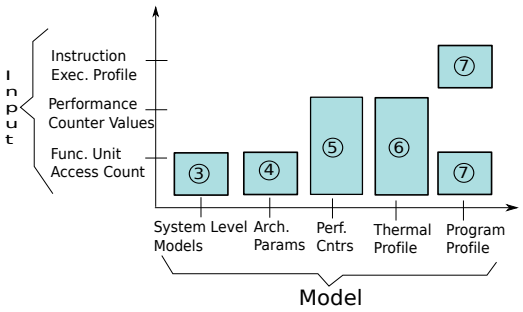


Figure 2: Model and Input

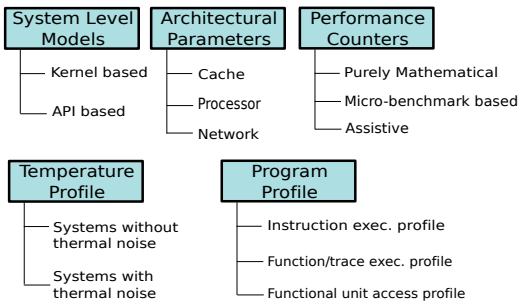


Figure 3: Further classification of the model

2.3 Taxonomy of Solutions

Every power estimation technique can be divided into two parts – model and input. The *model* is independent

of the test program, and is typically derived from either first principles or by using a learning based approach. The *input* is an aspect of the execution of the test program. See Figure 2.

A *model* captures the relation between power consumption and the source of information. We classify the model based on the type of information it uses – system level models, architectural parameters, performance counters, thermal profiles, and program profile information. A model is typically derived from first principles such as circuit simulation, program CFG analysis, or by solving heat flow equations. If it is not possible to do so, then we can empirically derive a model based on a large amount of training data by using machine learning based approaches.

The *input* captures some aspects of the execution of the test program. As shown in Figure 2, the input can either be (1) functional unit access counts, (2) instruction execution profiles, or (3) performance counter values. We can estimate the power consumption of the test program by applying the input to the model.

The generic approach for most techniques have the following structure:

1. Calculate the energy per invocation, E_i, for unit i
2. Generate activity factors (accesses per cycle), α_i.
3. Compute the total power using Equation 4.

$$P = \sum_i \alpha_i \times E_i \times f + P_{idle} \tag{4}$$

For system level models, the energy per invocation is calculated for each transaction, while for architectural model, it is calculated for each functional unit. For predominantly software approaches, such as program execution profile based models, it is calculated for each instruction. Performance counter based approaches do not strictly adhere to this structure. Some performance counter based techniques do follow this generic method, since the energy associated with a performance counter may be calculated by exciting that counter separately. However, this is not always possible. For thermal models, the temperature data collected from thermal sensors or an infrared photograph of a die serves as activity factor.

We have a two level classification in each section. First, we classify the different techniques based on the model, and talk about each model in a different section (see Figure 2). The different types of models are fairly different from each other and do not adhere to a common structure. Consequently, we use different high level criteria for further classifying them (level 2) as shown in Figure 3. Models using architectural parameters are classified structurally based on the type of functional unit they are meant for modeling. Other models that use a thermal profile, or performance counter based inputs, are classified based on the mathematical flavor of the approach.

Lastly, it is necessary to specially mention an orthogonal direction here. Instead of using the entire program, or large portions of it, it is possible to perform power estimation on a significantly reduced portion of the original program. There are methods to create a gist of a large program such that its power profile is significantly correlated with the power profile of the original program in a statistical sense. Starting from the seminal work of Hsieh and Pedram (1998), there have been many proposals ranging from formal program reduction to statistical sampling. Some of the important proposals in the area of statistical sampling of programs for power estimation are SimPoint (Perelman et al., 2003), and SMARTS (Wunderlich et al., 2003). Since power estimation techniques are oblivious of the genesis of the program, we do not consider these schemes henceforth.

Interested readers can refer to an older survey paper by Najm et al. (Najm, 1994) on generic low level power estimation techniques published in 1994 for some early work in this field. A later survey by Macii et al. (1998) published in 1999, describes some more power estimation techniques based on software techniques, gate level, behavioral, and RTL level models. We mostly focus on recent results and specialize in power estimation at the level of processors and their major components.

2.4 Scope of the Survey

We focus on important results in the area of processor power estimation proposed in the last 10-15 years, and do not consider generic power estimation techniques based on gate level and RTL level approaches. Instead, we focus on research contributions that are at a higher level, and are aware of at least some aspect of a processing device. In the interest of brevity, we do not focus on power estimation of all aspects of large systems such as storage, peripherals, and networks.

3 Approaches based on System Level Models

3.1 Preliminaries

In this section, we focus on estimating power through system level (high level) models. Such kind of models typically model different entities of a system of processing elements, and the interaction between them. An important subset of such kind of primarily communication oriented models are *Transaction Level Models* (TLMs). TLMs model each event or a message between modules as a basic transaction. SystemC is one of the most common TLM based languages for high level modeling. It provides an interface for modeling system level designs and, additionally, it contains a basic event driven simulation engine (kernel). Traditionally, SystemC has only been used to estimate the execution time.

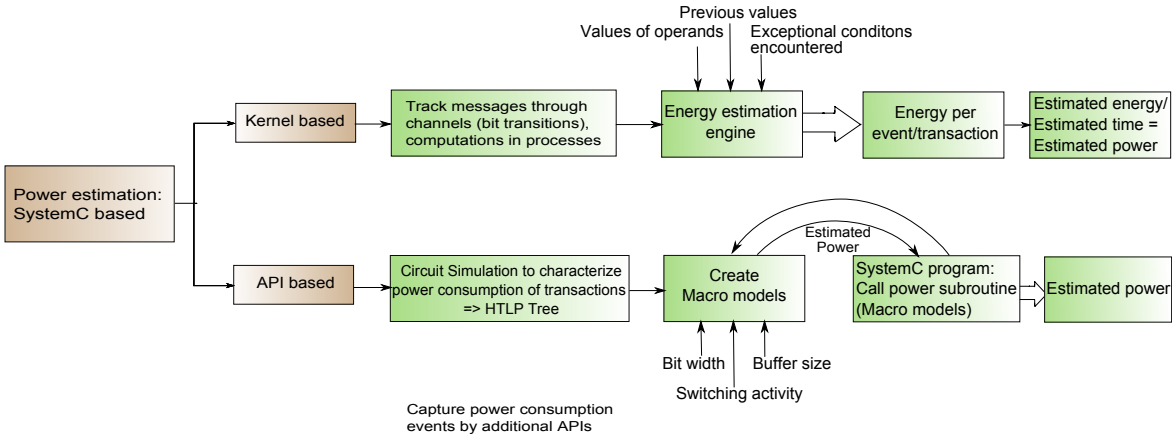


Figure 4: Generic flow of System-level model based approaches

Proposal		Category	Error
Year	Authors		(%)
2005	Dhanwada et al.	API based	3-11%
	Evaluated on a PowerPC/ CoreConnect System		
2006	Damasevicius	Kernel based	
	Designed for simple circuits.		
2007	Beltrame et al.	Kernel based	$< 20\%$
	Fast simulation : 20-40k transactions per second		
2010	Liu et al.	API based	$\approx 18\%$
	Uses aspect oriented programming		
2011	Kuehnle et al.	API based	13%
	VHDL to SystemC translation		

Table 1 Summary of system level approaches

Prior work in power estimation for system level models has mainly looked at power/energy estimation for SystemC based designs. The first set of approaches modify the SystemC kernel. The second set of approaches capture events relevant to power consumption by using additional custom APIs. No modification is made to the kernel. The kernel based approach is definitely more user friendly and generic; however, it is not very flexible. The generic flow for both these methods is given in Figure 4. Table 1 shows a list of some of the major proposals in this area. The *error* is computed by measuring the absolute value of the difference between estimated power and measured power for the entire chip. Power is typically measured by physical means. Most proposals measure the current flowing into the CPU by either using a loop ammeter connected to the mains, or by tapping power lines on the motherboard. In the rest of the paper, we shall use this definition of error unless specified otherwise.

3.2 Kernel based Approaches

Damasevicius et al. (Damasevicius and Stuikys, 2007; Damasevicius, 2006) provide the basic structure of a kernel based approach as shown in Figure 4. Kernel based approaches typically instrument the different types of channels to capture the values that are transmitted across them. Each transition (0→1 and 1→0) is associated with the flow of current, and the consequent resistive heat dissipation. Lebreton and Vivet (2008) adopt this methodology in designs that have voltage scaling. Giammarini et al. (2011) propose a tool called *PowerSim* on the same lines. This tool uses extensive energy models that are collected using empirical measurements on embedded micro-controllers. Ahuja et al. (2009) specify the model at the system level and generate cycle accurate description using a high level synthesis tool. They then map the system level design to the RTL design and obtain the activity factors using a combination of probabilistic approaches and simulation.

Beltrame et al. (2007) extend the basic approach to include a set of estimators with varying levels of accuracy in the kernel. The kernel incorporates different versions of channels and modules, differing in their power estimation accuracy. There is a tradeoff between accuracy and performance. Channels range from simple FIFO queues to complicated multi-cycle split transaction buses. The channels are mostly stateless, whereas for modules, different versions maintain shared state. At the outset, each simulation object (channel/module) needs to register itself with an *object request broker* (ORB). At runtime, all requests are passed to the ORB, and the ORB subsequently chooses the right version of the channel/module to process the request.

PowerSC and BlueSpec are two commercial tools that support power estimation for SystemC. PowerKernel is one of the prominent open source tools in this domain.

3.3 API based Approaches

Dhanwada et al. (Narayanan et al., 2005; Dhanwada et al., 2005) proposed a generic flow for power estimation at the SystemC level (Figure 4). The output of circuit simulation is a multi-level tree (called HTLP tree), where the granularity of the transaction gradually increases as we move towards the root, from power consumption values for primitive transactions, to a complex sequence of actions. This work assumes that most of the basic

circuit blocks are already designed at the time of power estimation. The main advantage over circuit simulation is that power estimation is done for significantly larger blocks, and the final power consumption can be characterized using some basic parameters (macro-models). The tradeoff is between speed and accuracy.

Liu et al. (2010) use aspect oriented programming concepts in SystemC for power estimation. An *aspect oriented programming* style proposes to divide a program into two parts. One part of the program contains the core logic and some specialized markers called *aspects*. The second part contains a set of sub-routines corresponding to the aspects. Based on the global configuration, and sometimes runtime configuration, the aspect compiler weaves both parts of the code together to produce one program. We can think of it as a specialized pre-processor. In the context of SystemC, the aspects contain calls to power calculation, and reporting routines. The second part consists of implementations of different types of power models.

All the approaches considered up till now are applicable after we have exact power consumption figures for different functional units of a processor. However, Grammatikakis et al. (2011) propose an approach that just accumulates the number of bit transitions for different functional units. They assign a relative cost to each transition, and empirically demonstrate for a NOC chip that the total cost is proportional to the actual power dissipation.

Kuehnle et al. (Kuehnle et al., 2011, 2012) describe a VHDL to SystemC translator that can insert power simulation routines in the SystemC code. These special API calls do not need input values as previous approaches. They use stochastic approaches to estimate the average number of bit transitions per operation. Consequently, they are significantly faster, and for large SystemC programs, have enviable accuracy.

4 Approaches based on Architectural Parameters

4.1 Preliminaries

Power estimation at the architecture level is typically done for three types of structures – caches, processors, and on-chip networks. The model is parameterized by high level architectural parameters such as the sizes of the caches, number of registers, and entries in the instruction window. The generic approach for estimating power at the architectural level consists of the following steps:

1. Calculate the load capacitance for each functional unit by using either analytic equations, empirical data, or circuit simulation.

2. Generate functional unit activity factors (accesses per cycle), α_i, through simulation.

3. Compute the total power using Equation 1.

There are two main things to be computed: load capacitance and activity factors. To determine the load capacitance, researchers have mostly used analytical models for regular structures like caches. However, for more complex units like full scale processors, prior work has used a combination of empirical data based modeling and circuit simulation. Note that some structures within a processor, such as L1 caches, register files and TLBs, are essentially caches themselves and do not need to be considered separately. Likewise, for on-chip networks, components such as buffers, and queues are caches. However, units such as crossbars and arbiters require detailed analytical modeling, and subsequent verification using circuit simulation. Table 2 shows a list of major contributions in this area.

4.2 Cache

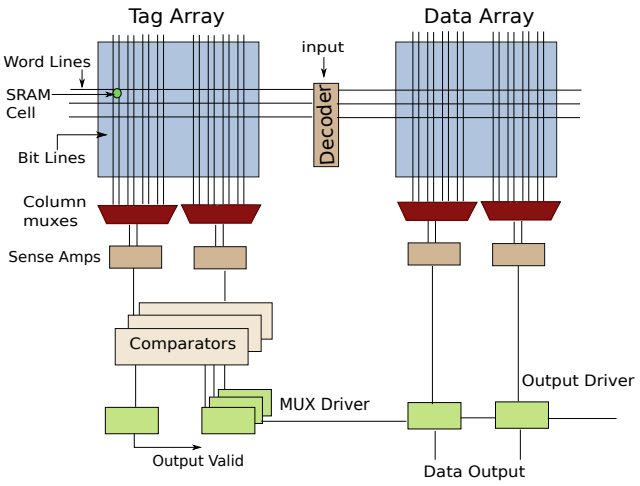


Figure 5: Structure of a cache

The earliest techniques in power estimation focussed on the cache, since it has a regular structure and is easier to model. Figure 5 shows the structure of a cache.

To estimate cache power, power dissipation is computed for the (1) SRAM cell, (2) bit line, (3) word line, (4) comparators/sense amps, and (5) input/output drivers.

Prior work in the area of cache power estimation is summarized in Figure 6 . One of the earliest models was proposed by Kamble and Ghose (1997) called CAPE. Ko et al. (1998) observed that the CAPE model is agnostic to the nature of inputs. In reality, the energy consumed in a cache access is also dependent on the existing state of the cache. For example, if we write the same data twice, we do not need to charge the drivers and input lines the second time.

The most influential work in this area is the Cacti model and tool designed by Reinman and Jouppi (2000) in 2000. It was released as the first publicly

Proposal		Name of	Category		Error
Year	Authors	tool (if any)	Main	Sub	(%)
1997	Kamble and Ghose	CAPE	Cache	Activity sensitive	
1998	Ko et al.		Cache	Transition sensitive	
1999	Cai and Lim		Processor	Activity sensitive	5-25%
		Classifies blocks by circuit type			
2000	Reinman and Jouppi	Cacti	Cache	Transition sensitive	20%[1]
		1) [1] source: Mamidipaka and Dutt (2004) 2) Widely used, open source, [1]Mamidipaka and Dutt (2004)			
2000	Brooks et al.	Wattch	Processor	Activity sensitive	<10%
		1) Classifies blocks by functionality 2) Widely used, open source, bundled with SimpleScalar			
2001	Chen et al.		Processor	Transition sensitive	4-7%
2002	Wang et al.	Orion 1.0	NOC	Activity sensitive	0-84%[2]
		1) [2] source: Kahng et al. (2009) 2) Basic NOC model, not validated extensively			
2009	Li et al.	McPat	Processor	Activity sensitive	≈ 23%
		Integrated XML based power, area, timing model			
2009	Kahng et al.	Orion 2.0	NOC	Activity sensitive	≈ 20%
		Validated with an Intel 80 core processor			

Table 2 Summary of architecture level approaches

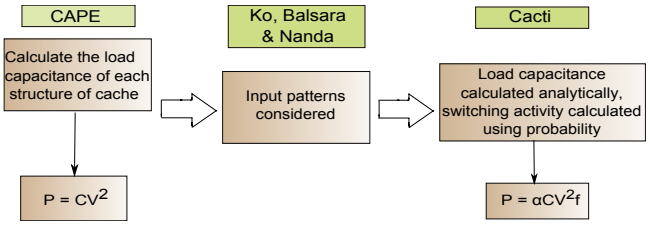


Figure 6: Cache power estimation techniques

available cache power modeling tool. It is also used as a part of the processor power modeling toolkit, Wattch (see Section 4.3.1). The procedure followed by Cacti is outlined in Figure 7.

4.3 Processor Power

There are two major kinds of models for processor power – activity sensitive models, and transition sensitive models. The former assumes that the energy consumed by a given functional unit is proportional to the number of accesses. It is oblivious of the input values. The latter model takes the input values into account. This is a slightly more accurate approach since we can correctly account for the energy consumed in I/O line capacitance charging/discharging by considering line states and transitions.

4.3.1 Activity Sensing Models

Cai and Lim (1999) divide all the functional units by the circuit type. The types of circuits that they consider are: static, dynamic, SRAM, programmable logic arrays (PLAs), clock trees, synthesized logic blocks, and custom logic blocks. They characterize the power of each type of circuit by doing extensive RTL level power simulations. They consider four parameters: *activity*, *area*, *active_power_density*, and *inactive_power_density*. *Activity* refers to the fraction of cycles a block is busy. Cai and Lim model both dynamic power and leakage power. When a block is busy, it is consuming dynamic power. The power density is dependent on the circuit type. This relationship is captured by the term *active_power_density* (power dissipated per unit area). Likewise, we can define a term, *inactive_power_density*, for leakage power. The active power of a block (functional unit) is computed by multiplying the activity, with the area and active_power_density.

As compared to the Cai-Lim model, Wattch (wat, 2000) released by Brooks et al. classifies different blocks by their structure and functionality. Some examples of hardware structures considered by Wattch include the instruction cache, wakeup logic, instruction window, branch predictor, register file, load/store queue and the global clock. Wattch is coupled with the SimpleScalar simulator, and uses the simulator to get functional unit access counts. The working of Wattch is illustrated in Figure 8.

As compared to Wattch, the McPAT (Li et al., 2009) model proposed by Li et. al. is an integrated timing, area, and power model. Moreover, McPAT models the structures of an OOO pipeline in great detail including the effect of leakage and short-circuit power. Secondly, instead of using linear scaling across technologies, it uses data directly from the latest ITRS reports (itr, 2007). Lastly, from the point of view of software engineering, McPAT is completely standalone and is not hardwired to any simulator. It takes an XML specification of the architecture and returns a combined area, timing, and power model.

To model power, McPAT analyzes each basic circuit block to determine the load capacitance. To calculate the activity factor, it relies on access statistics obtained from architectural simulation. Short circuit power is computed using equations given by Nose and Sakurai (2000) and leakage current is determined by MASTAR

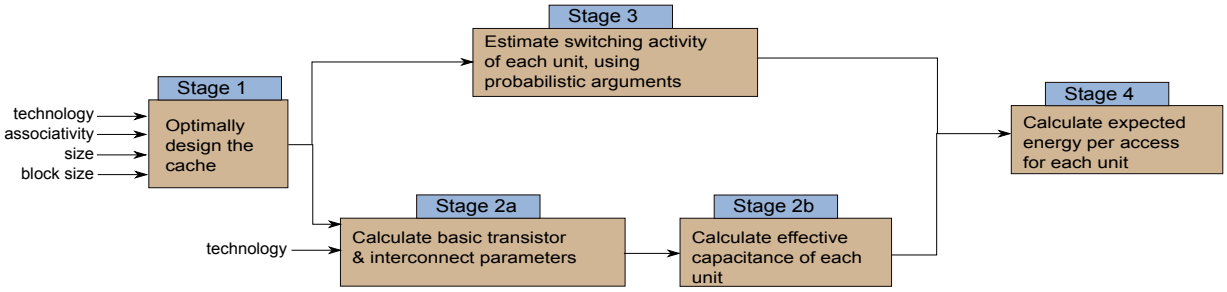


Figure 7: Functioning of the Cacti tool

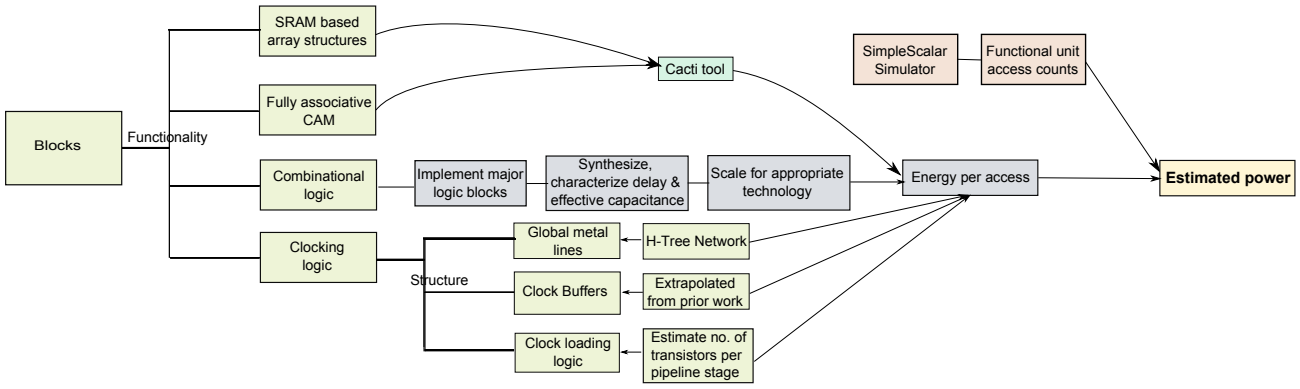


Figure 8: Work flow of Wattch

and Intel's data. For complex structures, McPAT adopts an empirical modelling approach.

4.3.2 Transition Sensitive Models

Chen et al. (2001) and Ye et al. (2000) developed a transition sensitive power model to estimate the power of a 32 bit RISC processor having an embedded 16 bit DSP core. The functional units are classified into two categories: *bit dependent* and *bit independent*. In bit dependent units, the value of a bit affects the operations of the rest of the bits. Examples of such functional units are adders, multipliers, and decoders. For such units, the power consumed is a function of the new input vector and the old input vector. Such units are pre-characterized by the model using circuit level simulation and the results are put in the form of a large matrix for each input-output pair. To reduce the size of this matrix the authors use clustering techniques.

In contrast, bit independent units have no functional dependence across bits such as register files, and memories. They calculate the energy consumption by dividing a large unit into a group of subcells. One subcell might correspond to one input bit. The model calculates the load capacitance, and then the power consumed. For the control path, the model tries to deduce the power from the instruction format. It starts out by characterizing the control path power consumption for a given instruction in a reference processor. This value is saved in a table, which is used later to compute the power consumption for processing a given instruction.

4.4 On-chip Network

The Orion tool (Wang et al., 2002) designed by Wang et al. was the first widely available framework to estimate the power of the on-chip network (NOC). The NOC is estimated to account for about 20% (Kahng et al., 2009) of the total power budget in modern processors.

Figure 9 shows the different types of units modeled by Orion. The first class of units are message transport agents such as links, crossbars, and routers. The second class of units have message storage capacity, and can also generate and modify messages. Examples include buffers, arbiters, message sending and receiving circuits. The Orion tool can faithfully model most interconnection, network topologies, and routing schemes. It models FIFO buffers as traditional SRAMs using the Cacti tool (see Section 4.2).

The crossbar is a matrix of inputs and outputs, where each unique input-output pair is connected by a specialized *connector* unit, typically a pass transistor. Orion gets the values of the effective capacitance of each of the interconnect lines and transistors (gate and drain capacitance) from the Cacti tool. The energy per access is then computed as CV^2. Orion considers three kinds of arbiters (Figure 10(b)): matrix, round-robin, and queuing. The matrix arbiter is similar to a crossbar

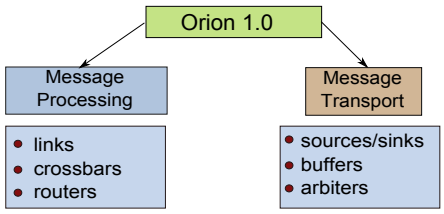
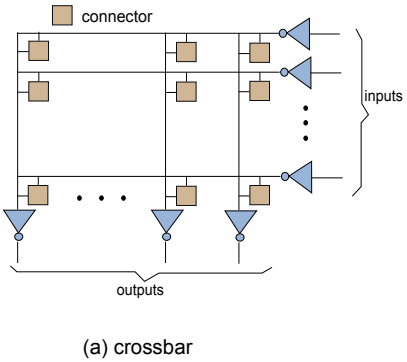
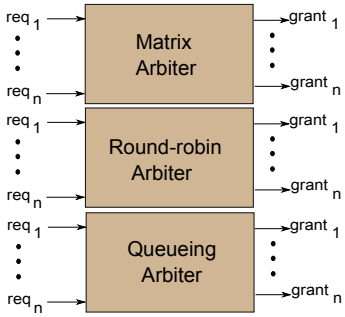


Figure 9: Major units in the on-chip network



(a) crossbar



(b) arbiter

Figure 10: Crossbars and arbiters

with some extra logic to effect a priority encoder. The round-robin and queuing arbiters use SRAM arrays and combinational logic circuits.

The Orion 2.0 model and toolkit (Kahng et al., 2009) was proposed in 2009. The authors observed that Orion 1.0 did not model power very accurately for some recently proposed multicore chips. Orion 2.0 models some extra structures namely latch based FIFO queues, clocks, and on-chip links. Secondly, instead of being dependent on Cacti, it takes transistor parameters directly from process technology files, or from the ITRS report (itr, 2007). Orion 2.0 also models leakage power. First, the tool characterizes different basic circuits for leakage power using HSpice based models. For larger circuits, Orion 2.0 estimates the leakage power using linear regression. The authors tried to calibrate Orion with an Intel 80 core processor and found it to be accurate within ± 20%.

5 Performance Counter Based Methods

5.1 Preliminaries

5.1.1 Performance Counters

Hardware performance counters are a set of machine specific registers, which store statistics about the activity of different subsystems of a processor. These registers are typically readable by user level or kernel level software modules.

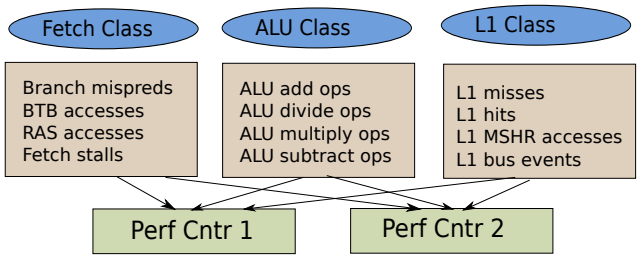


Figure 11: Performance counters

Figure 11 shows a conceptual diagram of performance counters. Most architectures divide the set of counters into a set of disjoint classes. For example, one class might contain a set of events for just the load-store queue. To reduce the hardware overhead, at most one event is chosen from each class. For further reduction, designers provision for a lesser number of performance counter registers than the number of event classes, allowing monitoring of only a subset of events a processor can generate at any moment of time. There are methods to increase the set of available counters by combining the data from multiple separate runs of the program (Joseph and Martonosi, 2001).

5.1.2 The Problem

$$P = \mathcal{F}(\mathbf{V}) \tag{5}$$

$$P = \mathbf{V}\mathbf{W} + P_{idle} \tag{6}$$

We provide two formulations of the problem in Equations 5 and 6. Here P represents the total power, and the vector $\mathbf{V}$ represents the sampled values of n performance counters. The first equation is a generic formulation that simply says that the total power is a function of the sampled performance counter values. The second equation specializes the function $\mathcal{F}$ by considering a a linear relationship. Here, $\mathbf{W}$ is a vector of weights, where W_j refers to the weight associated with the j^{th} performance counter. P_{idle} represents the static power. If we want to estimate the power for each functional unit, then we can treat $\mathbf{W}$ as a matrix.

There are two main research problems:

1. Choose the appropriate set of counters.
2. Accurately estimate the unknowns – function $\mathcal{F}$

For the first problem, prior work has mainly considered counters that have a good correlation with the measured power. However, estimating $\mathcal{F}$ is a far more challenging problem. The first set of approaches are *purely mathematical*. They view the problem as an optimization problem and typically try to find a least squares based estimate. Since regression based

Proposal		Name of	Category	Error
Year	Authors	tool (if any)		(%)
2000	Bellosa et al.	JouleWatcher	Microbenchmark	$< 10\%$
		Limited scope		
2001	Joseph and Martonosi	Castle	Assistive	$< 6\%$
		Uses Wattch, SimpleScalar for getting FU access counts, energy per access		
2003	Isci and Martonosi		Microbenchmark	$< 10\%$
2005	Contreras and Martonosi		Mathematical	<4%
		Model for Intel XScale		
2006	Wu et al.		Assistive	$< 5\%$
		Clustering with phase based analysis		
2009	Singh et al.		Assistive	$4 - 6\%$
		Non-linear techniques		
2009	Powell et al.	CAMP	Mathematical	8%
		Per-component power in Intel processors		
2010	Bertran et al.		Microbenchmark	$1.9 - 6\%$
		Very accurate		

Table 3 Summary of approaches

techniques are oblivious of the underlying architecture, they sometimes produce inaccurate or infeasible results.

Consequently, prior work has looked at architecture based approaches that try to derive **W** from first principles. **W** can be assigned a physical connotation. W_j might be made to represent the average power consumption associated with each instance of the j^{th} performance counter event. It is thus possible to estimate the value of W_j by measuring the total power dissipated by a micro-benchmark that exclusively generates events for the j^{th} performance counter. Note that it might not always be possible to isolate a given set of functional units associated with a performance counter.

Hence, recent work has looked at combining the micro-benchmark based approaches with purely mathematical approaches. We call such approaches – *assistive approaches*. These approaches typically enforce additional constraints in the optimization process by taking inputs from micro-benchmark based approaches or data from architectural simulators.

Table 3 presents a summary of some of the major contributions in this area, along with the broad class of techniques that they belong to.

5.2 Purely Mathematical Approaches

Contreras and Martonosi (2005) developed a power model for the Intel PXA255 processor using purely linear regression. Out of the 15 events that can be monitored in the target processor, the authors choose 5 performance events. To get the value of the number of cycles lost due to data dependencies, the authors estimate the number of cycles it would have taken for the application to run without any data dependencies.

Bircher et al. (Bircher and John, 2007; Bircher et al., 2005) use linear regression to create power models for the entire system including SMPs. At the outset, they classify the target system in to five subsystems: memory, chip set, I/O, disk and processors, and then separately estimate the power for each subsystem.

CAMP (Powell et al., 2009) proposed by Powell et al., considers **W** to be a matrix, and uses it to compute the power dissipation for each individual component using linear regression. For each component, the authors train a regression model by collecting samples across a broad suite of 73 applications. Instead of measuring power directly, they use Intel's power simulator to yield reference values. They further observe that just nine performance counters are enough to accurately account for 95% of the total dynamic power.

5.3 Micro-benchmark based Approaches

The earliest work in this field called *JouleWatcher* was done by Bellosa (2000). They consider four types of events namely integer micro-ops, floating point operations, second level cache accesses and memory transactions. The first task is to find the power consumed by an integer/control flow micro-operation. The authors run the integer micro-benchmark repeatedly. Additionally, they put in a variable sleep interval between consecutive runs. Subsequently, they plot the power consumed as a function of the number of micro-operations. The slope of this line is equal to the power consumption per integer micro-operation. It is hard to trigger floating point operations alone. Hence, JouleWatcher interleaves floating point and integer operations, and uses the same procedure for computing the average power per operation. Since the power consumption of an integer operation is known from the first step, it is possible to deduce the power associated with a floating point operation. We can extend this procedure to deduce the values for L2 accesses and memory transactions. This scheme is rather coarse grained in nature and does not consider a lot of performance counters.

Isci and Martonosi (2003) compute the j^{th} entry of the vector **W** as follows. Here, C_j corresponds to the j^{th} component in the core.

$$W_j = architecture\_scaling(C_j) \times max\_power(C_j) + non\_gated\_clock\_power(C_j) \quad (7)$$

architecture_scaling is a piece-wise linear function if the concerned component exhibits a non linear behavior; otherwise, it is a constant factor. The initial assumption for the value of *max_power* for each component is determined by scaling the value of the maximum total power documented in the processor specifications by the physical area occupied by the component in the die. The authors further refine this approach by running a set of micro-benchmarks. A micro-benchmark exercises a given set of components in a pre-determined manner. They try to minimize the error by adjusting the values of *max_power* for the concerned components.

Bertran et al. (2010) find the power usage of each architectural component. The authors group

related performance counters into sets. For example, the number of fetched instructions, and the number of decoded instructions show a very high degree of correlation. Consequently, the authors treat the fetch and decode stages as one set. However, the fetch stage and the L2 cache can be treated as separate sets. Subsequently, the authors create targeted micro-benchmarks, which only exercise a particular set of performance counters. These need to be extremely specific such that the power estimates are very accurate. Subsequently, they compute the power usage based on performance counter values using an iterative technique (similar to JouleWatcher (Bellosa, 2000)).

5.4 Assistive Approaches

In this section we consider approaches that try to combine a set of two or more basic approaches.

Joseph and Martonosi (2001) observed that it is hard to deduce the activity factors of many units such as the select logic, register file, and the instruction window from performance counters alone. To solve this problem, they calibrated a version of the Simplescalar simulator to accurately model the Alpha 21264 processor along with its performance counters. Subsequently, they model the correlation between functional unit access counts and performance counter values using a regression based model. Then, they use the architecture level power modeling tool, *Wattch* (wat, 2000), to get the average energy per access for different functional units. They subsequently compute the average total power by using Equation 4.

Wu et al. (2006) observe that for small benchmarks that have a highly homogeneous behavior, linear regression produces good results. However, most programs especially the SPEC benchmarks do not yield favorable results. Hence, they divide their execution into 1 to 100 milliseconds long phases (Sherwood et al., 2003), where each phase has its unique power characteristics. They propose an approach that uses only one representative sample from each phase to derive the regression line. This helps in reducing noise. Consequently, Wu et al. start out by isolating all the power phases from different runs of representative programs (SPEC benchmarks). For each phase they collect the following information: (1) power dissipation over time as measured by a digital multimeter, and the (2) values of 18 different performance counters. Subsequently, they cluster the measured power vectors of different phases based on a novel similarity metric, shown by Equation 8, into K different clusters using the K-means clustering algorithm. *Correlation* refers to the Karl Pearson's correlation coefficient, and D_{man} is the Manhattan distance between two vectors($D_{man} = \sum_i \mid a_i - b_i \mid$).

$$Sim(\mathbf{a}, \mathbf{b}) = Correlation(\mathbf{a}, \mathbf{b}) \times (1 - D_{man}(\mathbf{a}, \mathbf{b})) \quad (8)$$

They only consider the centroids of each cluster while deriving the linear relationship between performance counter values and the power dissipation. They further propose to optimize this process by using additional constraints for the coefficients by considering empirical data obtained through targeted micro-benchmarks.

Singh et al. (2009) introduce a more sophisticated model to improve the accuracy of regression based analysis. The authors look at linear, generalized non-linear, and exponential transformations of performance counter values. They conclude that piecewise linear functions are the best transformation functions. They figure out the parameters of each function using results gathered from the execution of targeted micro-benchmarks. They subsequently calculate the total power as a weighted linear sum of the transformed performance counter values. The weights are calculated using linear regression.

6 Thermal Profile based Approaches

6.1 Preliminaries

6.1.1 Thermal Equivalent of an Electrical Circuit

$$q = -k\nabla T \quad (9)$$

Equation 9 shows the Fourier law of conductance. q is the heat flux (Watts per unit area). k is the thermal conductivity and ∇T is the temperature gradient. This law has the same structure as Ohm's law ($V = IR$). We can thus map the temperature to voltage, the heat flow to current, and we can denote the quantity $1/k$ as the *thermal resistance*. Likewise, we can define *thermal capacitance* and proceed to make a thermal circuit of a semiconductor package. Each core is represented as a current (thermal power) source.

6.1.2 Inverse Heat Conduction Problem (IHCP) and Research Challenges

The problem of finding the power distribution map of a die, given the temperature map is known as the inverse heat conduction problem(IHCP). If we consider steady state values, then we have a linear system of equations as shown in Equation 10. Here $\mathbf{P}$ and $\mathbf{T}$ are column vectors representing the power and temperature of each core. The challenge is to estimate the matrix, $\mathbf{A}$, which we refer to as the *conductance matrix*.

$$\mathbf{P} = \mathbf{AT} \quad (10)$$

We can have a transient version of the same equation as given by Equation 11. Here C is a diagonal matrix, which contains the thermal capacitance of each node.

$$\mathbf{P} = \mathbf{AT} + C.\frac{d\mathbf{T}}{dt} \quad (11)$$

The main research challenge in this area as succinctly stated by Cochran et al. (2010) is to minimize the error in power estimation given the two main sources of inaccuracies:

Proposal		Name of the	Category	Error
Year	Authors	tool (if any)		(%)
2006	Hamann et al.	SIMP	w/o noise	$< 1\%$
		CFD analysis + laser based excitation		
2007	Martinez et al.		w/o noise	$< 1\%$
		IR photography (oil based heat sink)		
2009	Wang et al.		with noise	$< 10\%$
		Uses image processing		
2010	Qi et al.	PowerTrace	with noise	$< 7\%$
		Uses constrained optimization		
2010	Oh et al.		w/o noise	$< 3\%$
		Single thermal sensor		

Table 4 Summary of approaches

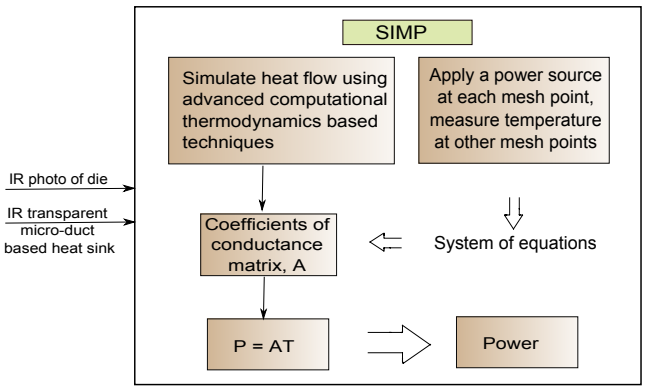


Figure 12: SIMP methodology

1. The first source of inaccuracy owes its basis to the physics of heat transfer. The silicon substrate and the packaging have a low pass filtering effect. This blurs the temperature map significantly. The low pass filter effect is inversely proportional to the lateral heat conductance.

2. The second source is the discretization introduced in the process of collecting thermal measurements.

6.1.3 Organization

To compute solutions to the IHCP problem, the first step is to collect the temperature data. This can be done either through embedded performance counter based thermal sensors (Oh et al., 2010), or an IR (InfraRed) photograph of a die (Hamann et al., 2006; Mesa-Martinez et al., 2007; Wang et al., 2009). The latter represents functional unit activity. In either case, the mathematical techniques are the same. Prior work has mostly looked at two variants of this problem. The first set of approaches try to directly solve Equation 10 or Equation 11 by assuming that the temperature values are exact. The latter set of approaches consider some thermal noise, which can arise due to measurement error or the fundamental limits of heat transfer. Table 4 shows a summary of approaches.

6.2 Systems without Thermal Noise

In the SIMP project (Hamann et al., 2006), the authors try to solve IHCP from an infrared photograph of a die in operation. To compute the conductance matrix the authors propose two approaches as shown in Figure 12.

In 2007, Martinez et al. (Mesa-Martinez et al., 2007) extended SIMP. Unlike SIMP, Martinez et al. use a very accurate IR camera and an IR transparent oil based heat sink. The authors start out by creating an analytic model of chip power consumption by considering the standard equations for dynamic and leakage power as defined in Section 2. For modeling the thermal RC circuit, they use parameters from the popular HotSpot (Skadron et al., 2003) temperature modeling tool. They consider both dynamic and leakage power. In their system of equations for each functional unit, they have four unknowns, P_{dyn} (dynamic power), P_{leak0}, P_{leak1}, and P_{leak2}. The latter three are constants in the leakage power equation (see Equation 2). The inputs are the values of temperature for each floorplan block and the total power as measured by a digital multimeter. To compute the dynamic and leakage power for each block, we need to solve a complex non-linear optimization problem. The authors start out by collecting a lot of samples for 14 different applications (mostly SPEC benchmarks) across many program phases, and use these samples to train a genetic algorithm. To calculate the fitness value of a candidate solution, the authors first compute the predicted total power and temperature of each block using analytical models. The discrepancy between the measured and computed values of die temperature and total power is the fitness value. Based on the fitness value of each solution, they discard some solutions.

An orthogonal approach as described by Oh et al. (2010) considers power estimation with just one thermal sensor. In this case, we do not have sufficient information to compute the power dissipation for every block. The authors show that with some simplifying assumptions, it is possible to estimate the total chip power by considering the transient form of the power equation (Equation 11). Subsequently, the authors estimate the power per functional unit, by apportioning the total power on the basis of performance counter data.

6.3 Systems with Thermal Noise

The approaches described in this section consider some degree of temperature measurement error. They use Equation 12 as the governing equation. Here, $\mathbf{N}$, represents the measurement error/ Gaussian noise.

$$\mathbf{T} = \mathbf{BP} + \mathbf{N} \tag{12}$$

As a first line approach, we can use a least squares based estimate to compute the optimal power map given the Gaussian noise. The formulation is shown in Equation 13. This method has stability issues (Wang et al., 2009) and does not consider the diffusive nature of heat flow.

$$\hat{\mathbf{P}} = \underset{\mathbf{P}}{\text{ArgMin}} \mid\mid \mathbf{BP} - \mathbf{T} \mid\mid^2 \tag{13}$$

Wang et al. (2009) adopt an approach from the world of image processing known as BTV regularization. They observe that the power map of a chip contains crisp boundaries, whereas a temperature map is very blurry in nature. Consequently, they adopt an approach that conceptually deblurs the temperature map to obtain the power map. They modify Equation 13 to produce Equation 14.

$$\begin{aligned}\hat{\mathbf{P}} = &\underset{\mathbf{P}}{\text{ArgMin}}(|| \mathbf{BP} - \mathbf{T} ||^2 \\ &+ \lambda \sum_{m,l=-p}^{p} \alpha^{|m|+|l|} || \mathbf{P} - S_x^m S_y^l \mathbf{P} ||)\end{aligned} \tag{14}$$

Here, the term λ is a weight for the regularization. It properly balances the mean square error with the regularization. S_x^m and S_y^l are two operations, which effectively shift the power map m positions in the horizontal direction, and l positions in the vertical direction, to cancel out the blurring effect of the image. α is another scalar term that properly weights this shift. The authors solve this equation using steepest gradient descent techniques. This technique proves to be very effective in canceling out noise and in giving a fairly accurate and crisp power map.

Qi et al. (2010) take the help of architectural simulation to obtain stable solutions for Equation 13. They add two more sets of constraints: (1) One for the power range of each block, (2) and the other for the total power. These constraints substantially improved the accuracy of the solution.

Cochran et al. Cochran et al. (2010) study the IHCP problem in FPGAs. They augment a standard FPGA with a set of micro-heaters, which are small circuits with a known power dissipation profile. Using representative power and temperature samples, they calibrate a regularization function (see (Wang et al., 2009)) to deblur the temperature map. Nowroz et al. Nowroz et al. (2011) add to this work by exciting the micro-heaters with an AC signal rather than a fixed DC signal. The authors observe a significantly enhanced accuracy.

7 Program Execution Profile Based Approaches

7.1 Preliminaries

In this section, we look at approaches that use a group of instructions as the basic atomic unit. The generic approach is to characterize the energy consumption of each instruction, and then profile the code to get instruction counts. This is done at the basic block level. We describe such approaches in Section 7.2. It is possible to do characterization and profiling at a higher level of granularity – functions or traces (see Section 7.3). Lastly, we describe a set of approaches that try to predict the functional unit access counts by using program analysis or from instruction access counts in Section 7.4. Subsequently, they compute the total energy by multiplying the access counts with pre-characterized energy values. Table 5 shows a summary of approaches in this area.

7.2 Instruction Level Power Estimation

All the approaches presented here have a similar structure. The first phase contains a profiling run, which executes different snippets of code repeatedly, and measures their energy usage. This allows us to estimate the energy associated with the set of instructions. In the second phase, we need to embed counters in software that give the execution frequency of each basic block. Lastly, we can obtain the total energy consumption by multiplying the number of executed instructions of each type by their corresponding energy values. Energy divided by the execution time yields the average power.

7.2.1 Data Independent Approaches

Jouletrack (Sinha and Chandrakasan, 2001) estimates power at the basic block/instruction level. Additionally, using equations presented in Section 2, it calculates the leakage power. Fournel et al. (2009) use a similar approach but do not model leakage power explicitly. Their model clubs all sources of power that cannot be classified as dynamic power into a lumped constant.

Sami et al. (2002) extend the instruction level profiling schemes to VLIW processors. Since these processors execute a group of instructions as a bundle, the paper characterizes power at the level of each bundle of instructions. Here, the power consumed by an instruction is dependent on three parameters – opcode/operands of the instruction, other instructions in the bundle, and the pipeline/circuit state. Benini et al. (2001) adopt a simpler approach for VLIW processors. They pass the instruction trace to an architectural power simulator that is calibrated with RTL models.

Brandolese et al. (2010) propose a very low level method for modeling the power and timing of C programs. They propose to break every statement in the C language to a set of micro-instructions that resembles a very primitive RISC ISA. For each such micro op, the authors characterize the power consumption. Subsequently, they embed software counters in a block of C statements corresponding to a high level construct such as a loop or a switch case statement. Based on the access counts, they estimate the total power.

7.2.2 Semi-data Dependent Approaches

Tiwari et al. (1994, 1996) and Mehta et al. (1996) were the first to propose a systematic methodology for calculating the power cost associated with the execution of each instruction as well as account for inter instruction effects in a stochastic sense. They take three types of inter-instruction effects into account – change in

Proposal		Category		Error
Year	Authors	Main	Sub	(%)
1994	Tiwari et al.	Instruction Level	Semi-Data Dependent	$< 3\%$
		For Intel 486 and Fujitsu SparcLite		
1999	Sarta et al.	Instruction Level	Data Dependent	$< 8\%$
		Considers a host of dependences across instructions		
2000	Qu et al.	Function Level		$\approx 3\%$
		Embedded Toshiba core		
2001	Tan et al.	Function Level		$< 5\%$
		Considers algorithmic parameters also		
2001	Sinha and Chandrakasan	Instruction Level	Data Independent	$< 3\%$
	(Jouletrack)	StrongArm and Hitachi microprocessors		
2004	Muttreja et al.	Function Level		$< 1\%$
		Automated model creation		
2005	Senn et al.	Functional Unit Based	Compiler Directed	$1 - 4\%$
	(SoftExplorer)	Uses simulators to convert code access profiles to functional unit access profiles		
2005	Kadayif et al.	Functional Unit Based	Compiler Directed	$< 6\%$
		Elaborate compiler framework		
2007	Blume et al.	Functional Unit Based	Hybrid	$< 9\%$
		Uses instruction profiles also		
2010	Brandolese et al.	Instruction Level	Data Independent	$< 1 - 5\%$
		Tailored for C programs		

Table 5 Summary of architecture level approaches

circuit state, pipeline stalls, and cache misses. The authors try to measure the average number of bits that need to change their state for every consecutive pair of instructions using extensive simulations. They model other effects by adding a constant to the overall instruction energy/power.

The comprehensive instruction level power model including all the above components gives the overall energy cost, E_P, of program P.

$$E_P = \sum_i (B_i * N_i) + \sum_{i,j} (O_{i,j} * N_{i,j}) + \sum_k E_k \qquad (15)$$

Here, B_i is the base cost of each instruction i, N_i is the number of times instruction i is executed, $O_{i,j}$ is the circuit state overhead associated with each pair of consecutive instructions (i,j), $N_{i,j}$ is the number of times the instruction pair is executed and E_k is the energy contribution due to all other inter instruction effects.

Russell and Jacome (1998) slightly simplify the model by proposing to use a constant power dissipation per instruction for homogeneous programs. This includes the effects of some inter instruction relationships.

7.2.3 Data Dependent Approaches

Sarta et al. (1999) extended the work of Tiwari et al. (1994, 1996). They primarily focus on the execution unit (EX + MEM stage) of the ST20-C1 embedded processor in their work. They use Equation 16 to estimate power.

$$P_j = K_0 + C_{ij} + K_1 * n_1 + K_2 * n_2 + ... + K_n * n_n \qquad (16)$$

P_j is the power cost for instruction, j. K_0 is the base power cost of a functional unit. For example, the authors found that the fetch unit consumes at least 13mA of current in their target processor irrespective of the amount of work. Thus, for the fetch unit, K_0 is equal to the fixed current (13 mA) times the supply voltage. Likewise, the authors measure the fixed power cost for the execution unit also. C_{ij} is the cost of executing instruction j after instruction i due to the changes introduced in the datapath/control path when we try to execute a new instruction. Lastly, K_i and n_i are the weights and number of transitions of the activity indices respectively. Activity indices are the elements that have a strong influence on the power consumption. The activity indices used in this work are the address bus, data write bus, and ALU bus, as they are representative of the switching activity inside the processor. K_i and n_i are measured by simulating the program on a VHDL model of a processor.

Wendt et al. (2010) propose an automated method for characterizing the instruction set. They decompose the energy per clock cycle into four parts: instruction-dependent, data-dependent, cache energy dissipation and the dissipation of other external components. For instruction dependent energy dissipation, they use a method similar to Tiwari et al. (1994).

Park et al. (2011) propose a multigranularity power estimation model for three design stages, with each stage more finely grained than the previous one. They create a 3D lookup table that stores the power consumption of each instruction, at each pipeline stage, for each functional unit in the processor.

7.3 Function/Program Level Power Analysis

Function level power analysis aims at estimating the power consumption of the processor at the granularity of functions and library calls. The main idea is to put software counters at the entry point of functions

that we expect to be invoked very frequently. The power dissipation of these functions needs to be pre-characterized.

Qu et al. (2000, 2002) observed that a majority of machine code executed on embedded microprocessors is part of either library functions or large user defined routines. There is some intermittent *glue code*, which is part of the core application logic. The power characteristics of each routine is more or less constant across benchmarks. Russell and Jacome (1998) had also arrived at a similar conclusion. Consequently, it is possible to estimate the power consumption of a large embedded program by just measuring the number of instances, the average duration, and energy of execution of each library call/user defined routine. The authors observed a good agreement between predicted and measured values.

Tan et al. (2001) factor in the effect of algorithmic parameters also. The total number of steps in an algorithm that has $\theta(n^2)$ time complexity can be represented as $c_1 + c_2 n + c_3 n^2$, where n is the input size and c_1, c_2, and c_3 are constants. These co-efficients can be found by measuring the total time it takes to run the algorithm for different values of n. We can further use linear regression techniques to minimize the least square error. The total energy is highly correlated with the total number of steps. Note that such kind of methods have limited applicability.

For the profiling based estimation techniques, Tan et al. look at several ways of collecting profiles in software – basic block counts, basic block correlation profiles and trace profiles. Like Tiwari et al. (1994), the authors first count the number of invocations of different basic blocks. Basic block A can be succeeded by B or C. The sequence AB might have a very different power profile from the sequence, AC. The authors consider a second order model that takes such correlations into account. They further extend this to consider long traces consisting of tens of basic blocks. Like the case of complexity based macro-modeling, they try to create a relationship between the parameters obtained through profiling and the energy consumption. Using regression based techniques, the authors arrive at a set of coefficients that efficiently weight the different profiling parameters.

Muttreja et al. (2007) observe that a program such as gzip mostly calls C libraries or well defined subroutines that have predictable power behavior. Only 2% of its code can be characterized as glue code. They provide an automated method to characterize these functions/library calls by figuring out the right set of parameters, collecting data through simulation, and creating models using symbolic regression. This method can be used for many embedded and DSP based codes.

7.4 Functional Unit based Approaches

The generic structure of such approaches is to estimate the activity of relevant functional units, and to compute the total energy by multiplying the activity by the corresponding energy per access obtained through architectural analysis, or from empirical data. Here, estimating functional unit activity is the main research challenge.

7.4.1 Compiler Directed Approaches

Senn and Laurent (Laurent et al., 2004; Senn et al., 2005, 2004; Julien et al., 2003) propose a family of approaches to estimate power for generic C programs. These efforts have culminated in a tool called *SoftExplorer* for digital signal processors (DSPs). The authors define two kinds of parameters – algorithmic and architectural. Examples of algorithmic parameters are IPC, issue rate, and cache miss rate. They represent the flow of data across functional units. The frequency setting or memory mode is an example of an architectural parameter. The authors create a regression model between a sample set of parameters (representative of functional unit activity), and the energy consumption from empirical data. Subsequently, to use the model, they create a specialized compiler that can automatically compute the algorithmic parameters by statically analyzing C programs meant to be run on DSPs. It starts out by estimating the average number of memory accesses, and estimated number of bank conflicts and cache misses. Since most DSP programs fit in the cache, do not use pointers, and have well defined memory requirements, it is fairly easy to estimate these parameters using static analysis. It is hard to do so for generic programs. Based on these, the tool computes the pipeline stall rate (PSR) due to memory accesses. Assuming that memory is the main bottleneck, it is possible to calculate the fetch rate, α, and the execution rate, β. SoftExplorer computes the final power via the already derived regression based model using the parameters – PSR, α, β, and some generic parameters such as the memory mode and frequency. This method is also known as Functional Level Power Analysis (FLPA). Schneider et al.(Schneider et al. (2004)) show that using FLPA, it is possible to achieve accurate results for digital signal processors. They were able to bound the error within 3%.

Kadayif et al. (2005, 2002) propose an elaborate compiler framework for energy estimation. Using static analysis they find the number of data path accesses, cache misses, memory transactions, and bus transactions. Their method is also able to estimate the total execution time for simple programs, giving an estimate of the total energy dissipated by the clocking network. For each high level construct in C, they estimate the number of assembly instructions that are required to implement it. Subsequently, for each assembly instruction, they try to find out how many times it will access different functional units. For example, for a load instruction, they estimate the number of cache misses. For a branch instruction in a loop, they estimate the number of iterations. For an

ALU instruction, they estimate the number of register file accesses. They multiply these estimates with the corresponding energies per access to get the total estimated energy.

Kremer et al. (Kremer et al., 2003; Heath et al., 2004) look at programs that have a lot of interaction with devices especially disk drives. They instrument the program to track the time that different devices including the CPU are active. For example, the compiler instruments a disk read/write system call to find out the approximate duration of disk activity. The model uses average values of CPU and disk power to compute the total power.

7.4.2 Hybrid Approaches

Blume et al. (2007) describe a hybrid approach between FLPA and instruction level techniques. The first step of the algorithm is to collect detailed instruction level traces using methods similar to those described in Section 7.2. In the next step, the authors try to derive the functional unit activities from the instruction execution profile. This can be done either analytically by taking a look at the specifications of the processor, or through regression based arithmetic models. The rest of the approach is similar to classic FLPA analysis as described by Senn and Laurent (see Section 7.4.1).

8 Comparison of Approaches

8.1 Preliminaries

Even though most of the proposals that we have presented, vary widely in terms of content, approach, experimental methodology, PVT (process, voltage, temperature) parameters, technology, and year of publication, there are some striking similarities. The first is that the method of reporting errors is very similar for all the proposals. They first estimate the power using custom algorithms and then compare the total estimated power of the chip with power that is measured by physical means. Here the aim is to measure the current and multiply it by the voltage to calculate power. The second important trend is that different proposals in the same family of approaches have more or less similar amounts of error. For example, all architecture level approaches have an error equal to almost 20%, whereas all performance counter based approaches have an error between 3-10%. Consequently, we can infer some degree of consistency across the results.

We use the following parameters for comparing the various approaches: (1) Granularity, (2) Leakage power modelling, (3) Intrusiveness, (4) Accuracy. A summary of the various approaches is shown in Table 6.

Granularity refers to the level of detail used by a technique to estimate power. A fine-grained power estimation technique that requires details of the underlying hardware cannot be used at higher levels of abstraction. Similarly, a coarse grained approach when applied to a lower level, may not be able to utilize all the information available at that design stage.

Some approaches discussed in this paper have an inherent ability to model *leakage power*, some have to be augmented for this purpose by additional methods, while others cannot model it at all. So, the techniques are compared on the basis of their ability to model leakage power. The next parameter, *intrusiveness* tells about the invasiveness of the approach.

All these factors, combined with the average accuracy obtained, primarily determine the applicability of a technique at a particular design stage. To some extent accuracy is not an independent factor, since methods applicable at higher levels of abstraction usually have a lower accuracy. The targeted accuracy is different at each level of design.

8.2 Features of Various Approaches

8.2.1 Granularity

Thermal models (based on IR photography) have the finest granularity, architecture level power models have an intermediate level of granularity, performance counter-based and program execution profile-based models are relatively coarsely grained. However, if thermal sensors are used to obtain the temperature map of a chip, then the model becomes coarse grained and depends on the number of sensors. In comparison, an IR photograph can give as much detail as required by setting the resolution of the IR camera. The granularity of system level models tends to vary based on the level of instrumentation.

8.2.2 Leakage power modelling

Since a substantial portion of the power consumption in modern processors is due to leakage power, hence methods that can accurately model leakage power preponderate over the other methods. Performance counter-based and program execution profile-based techniques can fundamentally model all kinds of power without any distinction between them. Thermal models also take leakage power into account. Architectural parameters-based techniques need to include some additional method to model leakage power. System level techniques usually do not consider it.

8.2.3 Intrusiveness

Intrusiveness of a technique limits its applicability and ease of use. A highly intrusive technique can be used at the design and fabrication stage but may not be used after that, and additionally may not be useful at the customer's end. Software techniques are the least intrusive, while architectural and system level model based techniques are the most intrusive, since they are dependent on the structure of the processor. Thermal models need the temperature map of a chip and hence,

Technique	Granularity	Intrusiveness	Leakage power modelling	Accuracy(%)	Primary advantage
Architectural-level	Low	High	No	80-90%	Availability at design stage
System-level	Low	High	No	85-95%	Availability at design stage
Performance counter based	Intermediate	Low	Yes	90-95%	Ease of use
Thermal	High	Intermediate	Yes	92-97%	Very accurate
Program profile based	Intermediate	Low	Yes	97-99%	At higher abstraction

Table 6 Comparison of approaches

some mechanism to obtain this must be incorporated while fabricating the processor. If we are using in-built temperature sensors, then the technique is almost non-intrusive; otherwise, if it is necessary to take an IR photograph, then it is necessary to remove the packaging. The latter technique is moderately intrusive. Similarly, for performance counters-based techniques, the appropriate performance counters are most often embedded in hardware, and we do not require additional support for using them. We thus conclude that thermal and performance counter based techniques fall in the middle of the spectrum.

8.2.4 Accuracy

Table 6 shows the average error rates of each of the broad family of approaches. We can conclude that early stage exploration approaches using system level models and high level architectural parameters have the least accuracy (about 80-90%). System level power estimation is more accurate than purely architecture level power estimation because it uses a more detailed implementation of hardware. Nevertheless, both the sets of approaches have been found to be extremely useful in predicting high level trends. In comparison, performance counter based methods have an error between 5-10%, and most temperature profile based methods have an error typically less than 1-3%. This is because these methods are much more specific and intrusive. Secondly, these methods are only applicable to processors that have already been fabricated and are in operation. Lastly, we observe that program profile based methods are in the middle with an error rate between 3-8%.

8.3 Explanation of Trends

Architecture level power models such as Wattch are designed with a given processor and technology in mind. Most of the time, researchers try to apply standard scaling rules to get the power dissipation values for a target technology. However, (Govindan et al., 2009) establish that scaling methodologies underestimate the latch capacitance by up to 40%. As the design methodology evolves, the original assumptions regarding the design of a certain functional unit cease to hold. Moreover, architectural models need cycle accurate micro-architectural detail and can not be used until this information is available. Also, all accesses to a given set of functional units do not consume the same amount of power, and the mean value is not sufficiently representative. Lastly, architecture level power estimators do not accurately take clock gating and leakage into account.

For system level models, there can be different physical realizations of a TLM based model. Each such design will have its own set of idiosyncrasies. Since system level models are formulated early in the design process, the implementation of different blocks keeps getting modified and refined till the last tape-out. Every such minor modification increases the risk of a modeling error. Secondly, system level models are at a fairly high level, and thus they suffer from abstraction errors.

As compared to architectural and system level modeling based approaches, the other three types of estimation (performance counter/ thermal/ program profile) have fewer errors, because these are in-vivo methods that operate directly on the target system while it is running.

Out of the three approaches, performance counter based methods are the most inaccurate. This is so because it is not possible to read all the performance counters of interest simultaneously. Most architectures, typically allow the user to read one counter from each class at any time. Consequently, it is necessary to read the performance counters in several rounds and stitch the values together. It is hard to achieve synchrony, and secondly there are several sources of jitter such as operating system and cache activity that can skew results. Lastly, most works use linear models. However, proposals such as Singh et al. (2009) have reported significant gains by using non-linear models.

Program profile based approaches have been reported to be slightly more accurate. However, it would not be wise to make an apples-to-apples comparison with performance counter based methods because, most instruction/function profile based approaches have been evaluated on custom embedded codes or snippets of large programs. Whereas, performance counter based methods have been evaluated using SPEC benchmarks.

Approaches that look at instructions in isolation suffer from large abstraction errors because tens of instructions are simultaneously considered for execution in a complex superscalar processor. Hence, it is necessary to consider large sequences of instructions for accurate power modeling. The basic problem is that it is hard to infer important statistics such as cache miss

rates, branch misprediction rates, and TLB miss rates. These figures can have a significant impact on power dissipation. It is also very difficult to model clock gating and leakage power using such approaches. Temperature profile based approaches are clearly the best in terms of accuracy. They capture leakage and clock gating very accurately. IR camera based approaches have an error less than 1% because they do not suffer from significant abstraction errors. It is possible to accurately infer the temperature at any point on the die. The relationship between power (both dynamic and leakage) and temperature is fairly stable, and depends on the thermal properties of the package and the technology. Once this relationship is determined, power estimation is very accurate. The main issue arises when we have a limited number of sensors. In this case, inferring the total power becomes a difficult problem because it is not possible to accurately guess the temperature values at all the points on a die.

9 Future Research Directions

The next decade is expected to throw up some new challenges in power estimation. We are moving to an ultra-nanoscale CMOS era, which will be characterized by extreme process variation, and high leakage. It will be necessary to bring in a lot of statistical techniques to measure the power dissipation. We are simultaneously moving into an era of 3D chips. Estimating the leakage power for such chips will be very difficult because of the extra dimension of heat flow. The on-chip interconnects are expected to incorporate disruptive technologies such as photonics, wireless technologies, and free space networks. They will have novel power dissipation characteristics. Probably towards the end of the next decade, we will move to non CMOS based technologies such as quantum or biological computing, which will radically change the way conventional power estimation is done.

References

(2000). *Wattch: a framework for architectural-level power analysis and optimizations*, ISCA '00, New York, NY, USA. ACM.

(2007). International technology roadmap for semiconductors. Technical report, In Technical Report 04-250, UCLA Engr.

Ahuja, S., Mathaikutty, D. A., Singh, G., Stetzer, J., Shukla, S. K., and Dingankar, A. (2009). Power estimation methodology for a high-level synthesis framework. In *Proceedings of the 2009 10th International Symposium on Quality of Electronic Design*, ISQED '09, pages 541–546, Washington, DC, USA. IEEE Computer Society.

Bellosa, F. (2000). The Benefits of Event Driven Energy Accounting in Power-Sensitive Systems. In *Proceedings of the 9th workshop on ACM SIGOPS European workshop: beyond the PC: new challenges for the operating system*, EW 9, pages 37–42, New York, NY, USA. ACM.

Beltrame, G., Sciuto, D., and Silvano, C. (2007). Multi-Accuracy Power and Performance Transaction-Level Modeling. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 26(10):1830 –1842.

Benini, L., Bruni, D., Chinosi, M., Silvano, C., Zaccaria, V., and Zafalon, R. (2001). A Power Modeling and Estimation Framework for VLIW-based Embedded Systems . In *ST Journal of System Research*, page 118.

Bertran, R., Gonzalez, M., Martorell, X., Navarro, N., and Ayguade, E. (2010). Decomposable and responsive power models for multicore processors using performance counters. In *Proceedings of the 24th ACM International Conference on Supercomputing*, ICS '10, pages 147–158, New York, NY, USA. ACM.

Bircher, W. and John, L. (2007). Complete system power estimation: A trickle-down approach based on performance events. In *Performance Analysis of Systems Software, 2007. ISPASS 2007. IEEE International Symposium on*, pages 158 –168.

Bircher, W. L., Valluri, M., Law, J., and John, L. K. (2005). Runtime identification of microprocessor energy saving opportunities. In *Proceedings of the 2005 international symposium on Low power electronics and design*, ISLPED '05, pages 275–280, New York, NY, USA. ACM.

Blume, H., Becker, D., Rotenberg, L., Botteck, M., Brakensiek, J., and Noll, T. (2007). Hybrid functional- and instruction-level power modeling for embedded and heterogeneous processor architectures. *Journal of Systems Architecture*, 53(10):689 – 702.

Borkar, S. (2001). Low power design challenges for the decade (invited talk). In *Proceedings of the 2001 Asia and South Pacific Design Automation Conference*, ASP-DAC '01, pages 293–296.

Brandolese, C., Fornaciari, W., and Salice, F. (2010). Timing and energy estimation of c programs. *Special Issue on Power Aware Embedded Computing*, 1.

Cai, G. and Lim, C. H. (1999). Architectural level power/performance optimization and dynamic power estimation. In Cool Chips Tutorial: An Industrial Perspective on Low Power Processor Design, ACM/IEEE Micro32, Haifa Israel.

Chen, R. Y., Irwin, M. J., and Bajwa, R. S. (2001). Architecture-level power estimation and design experiments. *ACM Trans. Des. Autom. Electron. Syst.*, 6:50–66.

Cheng, Y. and Hu, C. (1999). *MOSFET Modeling and Bsim3 User's Guide*. Kluwer Academic Publishers, Norwell, MA, USA.

Cochran, R., Nowroz, A. N., and Reda, S. (2010). Post-silicon power characterization using thermal infrared emissions. In *Low-Power Electronics and Design (ISLPED), 2010 ACM/IEEE International Symposium on*, pages 331 –336.

Contreras, G. and Martonosi, M. (2005). Power prediction for intel xscale® processors using performance monitoring unit events. In *Proceedings of the 2005 international symposium on Low power electronics and design*, ISLPED '05, pages 221–226, New York, NY, USA. ACM.

Damasevicius, R. (2006). Estimation of design characteristics at RTL modeling level using SystemC. *Information Technology and Control*, 35.

Damasevicius, R. and Stuikys, V. (2007). Estimation of power consumption at behavioral modeling level using systemc. *EURASIP J. Emb. Sys.*, 1.

Dhanwada, N., Bergamaschi, R., Dungan, W., Nair, I., Gramann, P., Dougherty, W., and Lin, I.-C. (2005). Transaction-level modeling for architectural and power analysis of powerpc and coreconnect-based systems. *Design Automation for Embedded Systems*, 10:105–125.

Fournel, N., Fraboulet, A., and Feautrier, P. (2009). Embedded software energy characterization: Using non-intrusive measures for application source code annotation. *J. Embedded Comput.*, 3(3):179–188.

Giammarini, M., Conti, M., and Orcioni, S. (2011). System-level energy estimation with powersim. In *ICECS'11*, pages 723–726.

Govindan, M. S. S., Keckler, S. W., and Burger, D. (2009). End-to-end validation of architectural power models. In *ISLPED*, pages 383–388.

Grammatikakis, M., Politis, S., Schoellkopf, J.-P., and Papadas, C. (2011). System-level power estimation methodology using cycle- and bit-accurate tlm. In *Design, Automation Test in Europe Conference Exhibition (DATE), 2011.*

Hamann, H. F., Lacey, J., Weger, A., and Wakil, J. (2006). Spatially-resolved imaging of microprocessor power (simp): hotspots in microprocessors. In *ITHERM.*

Heath, T., Pinheiro, E., Hom, J., Kremer, U., and Bianchini, R. (2004). Code transformations for energy-efficient device management. *IEEE Trans. Comput.*, 53(8):974–987.

Hsieh, C.-T. and Pedram, M. (1998). Microprocessor power estimation using profile-driven program synthesis. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 17(11):1080 –1089.

Isci, C. and Martonosi, M. (2003). Runtime power monitoring in high-end processors: Methodology and empirical data. In *Proceedings of the 36th annual IEEE/ACM International Symposium on Microarchitecture*, MICRO 36, pages 93–, Washington, DC, USA. IEEE Computer Society.

Jevtic, R. and Carreras, C. (2012). A complete dynamic power estimation model for data-paths in {FPGA} {DSP} designs. *Integration, the {VLSI} Journal*, 45(2):172 – 185.

Joseph, R. and Martonosi, M. (2001). Run-time power estimation in high performance microprocessors. In *Low Power Electronics and Design, International Symposium on, 2001.*, pages 135 –140.

Julien, N., Laurent, J., Senn, E., and Martin, E. (2003). Power consumption modeling and characterization of the ti c6201. *IEEE Micro*, 23(5):40–49.

Kadayif, I., Kandemir, M., Chen, G., Vijaykrishnan, N., Irwin, M. J., and Sivasubramaniam, A. (2005). Compiler-directed high-level energy estimation and optimization. *ACM Trans. Embed. Comput. Syst.*, 4:819–850.

Kadayif, I., Kandemir, M. T., Vijaykrishnan, N., Irwin, M. J., and Sivasubramaniam, A. (2002). Eac: A compiler framework for high-level energy estimation and optimization. In *DATE*, pages 436–442.

Kahng, A., Li, B., Peh, L.-S., and Samadi, K. (2009). Orion 2.0: A fast and accurate noc power and area model for early-stage design space exploration. In *Design, Automation Test in Europe Conference Exhibition, 2009. DATE '09.*, pages 423 –428.

Kamble, M. B. and Ghose, K. (1997). Energy-efficiency of vlsi caches: A comparative study. In *Proceedings of the Tenth International Conference on VLSI Design: VLSI in Multimedia Applications*, VLSID '97, pages 261–, Washington, DC, USA. IEEE Computer Society.

Ko, U., Balsara, P., and Nanda, A. (1998). Energy optimization of multilevel cache architectures for risc and cisc processors. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 6(2):299 –308.

Kremer, U., Hicks, J., and Rehg, J. (2003). A compilation framework for power and energy management on mobile computers. In *Proceedings of the 14th international conference on Languages and compilers for parallel computing*, LCPC'01, pages 115–131.

Kuehnle, M., Wagner, A., and Becker, J. (2011). A statistical power estimation methodology embedded in a systemc code translator. In *Proceedings of the 24th symposium on Integrated circuits and systems design*, SBCCI '11, pages 79–84.

Kuehnle, M., Wagner, A., Brito, A. V., and Becker, J. (2012). Modeling and implementation of a power estimation methodology for systemc. *International Journal of Reconfigurable Computing*, 2012.

Laurent, J., Julien, N., Senn, E., and Martin, E. (2004). Functional level power analysis: An efficient approach for modeling the power consumption of complex processors. In *Proceedings of the conference on Design, automation and test in Europe - Volume 1*, DATE '04, pages 10666–, Washington, DC, USA. IEEE Computer Society.

Lebreton, H. and Vivet, P. (2008). Power modeling in systemc at transaction level, application to a dvfs architecture. In *Symposium on VLSI, 2008. ISVLSI '08. IEEE Computer Society Annual*, pages 463 –466.

Li, S., Ahn, J. H., Strong, R. D., Brockman, J. B., Tullsen, D. M., and Jouppi, N. P. (2009). Mcpat: an integrated power, area, and timing modeling framework for multicore and manycore architectures. In *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO 42, pages 469–480.

Liu, F., Tan, Q., Song, X., and Abbasi, N. (2010). Aop-based high-level power estimation in systemc. In *Proceedings of the 20th symposium on Great lakes symposium on VLSI*, GLSVLSI '10, pages 353–356.

Macii, E., Pedram, M., and Somenzi, F. (1998). High-level power modeling, estimation, and optimization. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 17(11):1061 –1079.

Mamidipaka, M. and Dutt, N. (2004). ecacti: An enhanced power estimation model for on-chip caches. Technical Report 04-28, Center for Embedded Computing, University of California, Irvine.

Martin, S. M., Flautner, K., Mudge, T., and Blaauw, D. (2002). Combined dynamic voltage scaling and adaptive body biasing for lower power microprocessors under dynamic workloads. In *Proceedings of the 2002 IEEE/ACM international conference on Computer-aided design*, ICCAD '02.

Mehta, H., Owens, R., and Irwin, M. (1996). Instruction level power profiling. In *Acoustics, Speech, and Signal Processing, 1996. ICASSP-96. Conference Proceedings., 1996 IEEE International Conference on*, volume 6, pages 3326 –3329.

Mesa-Martinez, F. J., Nayfach-Battilana, J., and Renau, J. (2007). Power model validation through thermal measurements. In *Proceedings of the 34th annual international symposium on Computer architecture*, ISCA '07, pages 302–311, New York, NY, USA. ACM.

Muttreja, A., Raghunathan, A., Ravi, S., and Jha, N. K. (2007). Automated energy/performance macromodeling of embedded software. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 26(3):542 – 552.

Najm, F. N. (1994). A survey of power estimation techniques in vlsi circuits. *IEEE Trans. Very Large Scale Integr. Syst.*, 2(4):446–455.

Narayanan, V., Lin, I.-C., and Dhanwada, N. (2005). A power estimation methodology for systemc transaction level models. In *Hardware/Software Codesign and System Synthesis, 2005. CODES+ISSS '05. Third IEEE/ACM/IFIP International Conference on*, pages 142 –147.

Nose, K. and Sakurai, T. (2000). Analysis and future trend of short-circuit power. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 19(9):1023 – 1030.

Nowroz, A. N., Woods, G., and Reda, S. (2011). Improved post-silicon power modeling using ac lock-in techniques. In *Proceedings of the 48th Design Automation Conference*, DAC '11, pages 101–106, New York, NY, USA. ACM.

Oh, D., Kim, N. S., Chen, C. C. P., Davoodi, A., and Hu, Y. H. (2010). Runtime temperature-based power estimation for optimizing throughput of thermal-constrained multi-core processors. In *Proceedings of the 2010 Asia and South Pacific Design Automation Conference*, ASPDAC '10, pages 593–599, Piscataway, NJ, USA. IEEE Press.

Park, Y.-H., Pasricha, S., Kurdahi, F. J., and Dutt, N. (2011). A multi-granularity power modeling methodology for embedded processors. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 19(4):668–681.

Perelman, E., Hamerly, G., Biesbrouck, M. V., Sherwood, T., and Calder, B. (2003). Using simpoint for accurate and efficient simulation. In *SIGMETRICS*, pages 318–319.

Powell, M., Biswas, A., Emer, J., Mukherjee, S., Sheikh, B., and Yardi, S. (2009). Camp: A technique to estimate per-structure power at run-time using a few simple parameters. In *High Performance Computer Architecture, 2009. HPCA 2009. IEEE 15th International Symposium on*, pages 289 – 300.

Qi, Z., Meyer, B., Huang, W., Ribando, R., Skadron, K., and Stan, M. (2010). Temperature-to-power mapping. In *Computer Design (ICCD), 2010 IEEE International Conference on*, pages 384 –389.

Qu, G., Kawabe, N., Usami, K., and Potkonjak, M. (2002). Code coverage-based power estimation techniques for microprocessors. *Journal of Circuits, Systems, and Computers*, 11(5):557–.

Qu, G., Kawabe, N., Usarni, K., and Potkonjak, M. (2000). Function-level power estimation methodology for microprocessors. In *Design Automation Conference, 2000. Proceedings 2000. 37th*, pages 810 –813.

Reinman, G. and Jouppi, N. (2000). Cacti 2.0: An integrated cache.

Rotem, E., Naveh, A., Rajwan, D., Ananthakrishnan, A., and Weissmann, E. (2011). Power management architecture of the2nd generation intel core microarchitecture. In *HotChips*.

Russell, J. and Jacome, M. (1998). Software power estimation and optimization for high performance, 32-bit embedded processors. In *Computer Design: VLSI in Computers and Processors, 1998. ICCD '98. Proceedings. International Conference on*, pages 328 –333.

Sami, M., Sciuto, D., Silvano, C., and Zaccaria, V. (2002). An instruction-level energy model for embedded vliw architectures. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 21(9):998 – 1010.

Sarta, D., Trifone, D., and Ascia, G. (1999). A data dependent approach to instruction level power estimation. In *Low-Power Design, 1999. Proceedings. IEEE Alessandro Volta Memorial Workshop on*, pages 182 –190.

Schneider, M., Blume, H., and Noll, T. G. (2004). Power estimation on functional level for programmable processors. *Advances in Radio Science*, 2:215–219.

Senn, E., Laurent, J., Julien, N., and Martin, E. (2004). Softexplorer: Estimation, characterization, and optimization of the power and energy consumption at the algorithmic level. In *PATMOS*, pages 342–351.

Senn, E., Laurent, J., Julien, N., and Martin, E. (2005). Softexplorer: estimating and optimizing the power and energy consumption of a c program for dsp applications. *EURASIP J. Appl. Signal Process.*, 2005:2641–2654.

Sherwood, T., Perelman, E., Hamerly, G., Sair, S., and Calder, B. (2003). Discovering and exploiting program phases. *Micro, IEEE*, 23(6):84 – 93.

Singh, K., Bhadauria, M., and McKee, S. A. (2009). Real time power estimation and thread scheduling via performance counters. *SIGARCH Comput. Archit. News*, 37:46–55.

Sinha, A. and Chandrakasan, A. P. (2001). Jouletrack: a web based tool for software energy profiling. In *Proceedings of the 38th annual Design Automation Conference*, DAC '01, pages 220–225.

Skadron, K., Stan, M. R., Huang, W., Velusamy, S., Sankaranarayanan, K., and Tarjan, D. (2003). Temperature-aware microarchitecture. In *ISCA*, pages 2–13.

Tan, T., Raghunathan, A., Lakshminarayana, G., and Jha, N. (2001). High-level software energy macro-modeling. In *Design Automation Conference, 2001. Proceedings*, pages 605 – 610.

Tiwari, V., Malik, S., and Wolfe, A. (1994). Power analysis of embedded software: a first step towards software power minimization. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 2(4):437 –445.

Tiwari, V., Malik, S., Wolfe, A., and Lee, M. T.-C. (1996). Instruction level power analysis and optimization of software. In *Proceedings of the 9th International Conference on VLSI Design: VLSI in Mobile Communication*, VLSID '96, pages 326–, Washington, DC, USA. IEEE Computer Society.

Wang, H.-S., Zhu, X., Peh, L.-S., and Malik, S. (2002). Orion: a power-performance simulator for interconnection networks. In *Proceedings of the 35th annual ACM/IEEE international symposium on Microarchitecture*, MICRO 35, pages 294–305, Los Alamitos, CA, USA. IEEE Computer Society Press.

Wang, X., Farsiu, S., Milanfar, P., and Shakouri, A. (2009). Power trace: An efficient method for extracting the power dissipation profile in an ic chip from its temperature map. *Components and Packaging Technologies, IEEE Transactions on*, 32(2):309 –316.

Wendt, M., Grumer, M., Steger, C., Weiss, R., Neffe, U., and Muehlberger, A. (2010). Tool for automated instruction set characterization for software power estimation. *Instrumentation and Measurement, IEEE Transactions on*, 59(1):84–91.

Wu, W., Jin, L., Yang, J., Liu, P., and Tan, S.-D. (2006). A systematic method for functional unit power estimation in microprocessors. In *Design Automation Conference, 2006 43rd ACM/IEEE*, pages 554 –557.

Wunderlich, R. E., Wenisch, T. F., Falsafi, B., and Hoe, J. C. (2003). Smarts: Accelerating microarchitecture simulation via rigorous statistical sampling. In *ISCA*, pages 84–95.

Ye, W., Vijaykrishnan, N., Kandemir, M., and Irwin, M. (2000). The design and use of simplepower: a cycle-accurate energy estimation tool. In *Design Automation Conference, 2000. Proceedings 2000. 37th*, pages 340 –345.