To appear in the 15th International Symposium on High-Performance Computer Architecture (HPCA 2009)

CAMP: A Technique to Estimate Per-Structure Power at Run-time using a Few Simple Parameters

Michael D. Powell, Arijit Biswas, Joel S. Emer, Shubhendu S. Mukherjee, Basit R. Sheikh[1], Shrirang Yardi[2]
Intel Massachusetts *Computer Systems Laboratory, Cornell University[1]* *Department of Electrical and Computer Engineering, Virginia Tech[2]*
{michael.d.powell, arijit.biswas, joel.emer, shubu.mukherjee}@intel.com, basit@csl.cornell.edu, yardi@vt.edu

Abstract

Microprocessor power has become a first-order constraint at run-time. Designers must employ aggressive power-management techniques at run-time to keep a processor's ballooning power requirements under control. Effective power management benefits from knowledge of run-time microprocessor power consumption in both the core and individual microarchitectural structures, such as caches, queues, and execution units. Increasingly feasible per-structure power-control techniques, such as fine-grain clock gating, power gating, and dynamic voltage/frequency scaling (DVFS), become more effective from run-time estimates of per-structure power. However, run-time computation of per-structure power estimates based on utilization requires daunting numbers of input statistics, which makes per-structure monitoring of run-time power a challenging problem.

To address the challenges of estimating per-structure power in hardware, we propose a new technique, called Common Activity-based Model for Power (CAMP), to estimate activity factors and power for microarchitectural structures. Despite using a relatively few input parameters—specifically nine—based on general microprocessor utilization statistics (e.g., IPC and load rate), our linear-regression-based model estimates activity and dynamic power for over 100 structures in an out-of-order x86 pipeline and core power with an average error of 8%. Because the computations utilize few inputs, CAMP is simple enough to implement in hardware, providing run-time structure and core power estimates for dynamic power management. Because the input statistics are generic in nature and the model remains accurate across incremental microarchitectural refinements, CAMP provides simple intuitive equations relating global microarchitectural statistics to structure activity and power. These equations provide a simple technique that can equate changes in one structure's activity to power variations in other structures across the pipeline.

1 Introduction

Microprocessor power has become a first-order constraint at run-time. Transistor densities have increased exponentially over successive process technologies, but supply voltage has not decreased proportionately. Consequently, transistor count has increased faster than per-transistor power has decreased. Designers must employ aggressive power-management techniques at run-time to keep a processor's ballooning power requirements under control. A challenge to effective run-time power management is knowing the run-time power consumption of the microprocessor.

Microprocessor power estimates may be computed at several levels, including system, die, core, and per-structure power. System and die power estimates can be used to monitor and control system-level attributes like disks, fans, and case temperature. Core-level power estimates are a popular topic of recent research because these estimates may be used to address power and power-density concerns on a multi-core die. Beyond the core level, run-time power estimates for individual microarchitectural structures, such as caches, ALUs, and queues, would be useful for fine-grain management of package temperature and core power requirements. We refer to estimates at this level as *"per-structure"* power estimates. Per-structure power estimates will be useful for controlling selective enabling and disabling of microarchitectural resources. As power-management becomes increasingly important in microprocessors, coarse-granularity core-level power estimates are likely to become inadequate to manage and to reallocate continuously power budgets for individual microarchitectural structures. Increasingly feasible per-structure power-control techniques, such as fine-grain clock gating, power gating, and dynamic voltage/frequency scaling (DVFS), will also benefit from run-time estimates of per-structure power.

System-level power estimates can be computed by directly sensing current because of the fairly long response time constants (many milliseconds or a few seconds) of the system-level attributes. However direct monitoring of core power in a multi-core die, let alone structure power, faces several challenges. On-die current sensors have been proposed for quiescent current (IDDQ) testing but have rarely been used in production for even that purpose due to problems such as area and performance overhead and calibration drift due to process variations [27]. Other proposed current sensors for IDDQ attempt to address these problems [27,5], but these sensors are still experimental, and no proposal suggests using the sensors for run-time dynamic power monitoring in a deployed microprocessor. In addition, it would be difficult to use such sensors to estimate per-structure power because each structure would need a separate power domain, increasing design and layout complexity.

It might seem possible to estimate structure or core power at run-time by borrowing methodology from design-time power models like Intel's ALPS [8]. These techniques count utilization of structures (e.g., register reads) and compute power estimates based on a per-event power model. However, such direct computation of core and structure power at run-time based on utilization would be complex due to the hundreds of utilization statistics required for the core and tens of statistics required for each structure. For example, in a Intel® Core™-like processor, ALPS would calculate reorder-buffer (ROB) power for a single cycle using structure-specific input statistics, such as ROB reads, register-file reads, double-width register-file reads, number of operations retired, number of

operations written back, occurrence of nuke operations this cycle, and occurrence of integer, floating-point, and branch operation activity this cycle. Tracking such a myriad of cryptic statistics for each structure in hardware would lead to tremendous wire and logic complexity and is thus impractical.

To address the challenges of estimating per-structure power in hardware, we propose a new analytical model, called *Common Activity-based Model for Power* (CAMP), to estimate activity factors and power for microarchitectural structures. This model does not rely on current monitoring or simulating dozens or hundreds of utilization statistics. Instead CAMP is based on a few input statistics—specifically 9—that are general microarchitectural statistics (e.g., IPC, fetch rate, number of loads). In spite of this limited input data, CAMP's linear-regression-based methodology can estimate activity for tens or hundreds of microprocessor structures. It can also estimate the overall core power to within 8%. The reason only a few input statistics are sufficient to estimate the dynamic power of a microprocessor is because the myriad per-structure events are related to a small set of global parameters, such IPC and load rate. We use this key observation to drive the development of CAMP.

CAMP is simple enough to implement in hardware because the computations are based on a few input statistics which can be obtained easily from performance counters. Without CAMP, a hardware power estimator based on structure activity and performance counters (similar to ALPS), would require many tens or hundreds of structure-specific inputs encompassing most pipeline structures, leading to wiring and power overhead from routing and counting the many statistics. With CAMP, we avoid the use of cryptic statistics by using data from a few general performance counters to estimate structure-level and core-level power at extremely fine time intervals (on the order of 100 microseconds). To the best of our knowledge, no previous published work has proposed such fine-grained per-structure power models using global statistics.

An additional benefit of CAMP is that it provides simple intuitive equations relating global microarchitectural statistics to structure activity and power. These equations provide a simple analytical model that can equate changes in one structure's activity to power variations in other structures across the pipeline. For example, CAMP equations might reveal the impact of increasing branch prediction accuracy on decreasing power and activity in the pipeline back-end due to reduced mis-speculation. This type of expression is useful in early design stages for evaluating trade-offs, and the equation-based analysis can be done in spreadsheets prior to development of a detailed performance simulator.

Our common activity-factor-based model differs from previous proposals for run-time power models. Previous work on run-time power monitoring has focused either on embedded designs, microprocessors with little power management, or on directly monitoring current externally [10] and not on run-time monitoring in production microprocessors in the field. We compare CAMP to previous analytical power models and discuss related work in more detail in the next section.

The main contributions of this paper are:

- We propose CAMP, a regression-based model that can estimate activity and power for over 180 structures with high accuracy, using only nine input statistics, that estimates power for about 90% of microarchitectural structures to within 5%.
- We explain how the model can be implemented in hardware for run-time power estimates in the field at either the structure-level or the core level.
- We explain how the model can be used at design-time to explore the impact of microarchitectural changes on individual structure activity and power. CAMP is unique among run-time power estimation techniques in that the same concepts can be applied in early design analysis.

The rest of this paper is organized as follows. In Section 2 we discuss related work. Section 3 covers power-modeling background, and Section 4 describes our infrastructure and methodology. We describe the details of CAMP in Section 5. In Section 6 we discuss using CAMP in hardware and present results. Section 7 discusses using CAMP for power estimates at design time. We conclude in Section 8.

2 Related Work

In this section, we discuss previous proposals for run-time power monitoring and related analytical design-time models for performance and power estimation. Figure 1 summarizes the design space for power estimation, and Table 1 summarizes the categories of run-time techniques. First we discuss techniques that either target or are conceptually suitable for run-time power estimation; these categories are shown in ovals on the figure. Then we discuss techniques that are suitable only for offline power estimation, shown in the table at the left of the figure. Finally, we discuss related analytical performance models that do not target power estimation.

2.1 Run-time Power Monitoring

Several proposals either aim to monitor run-time power or might seem suitable for monitoring run-time power. The first run-time power monitoring category targets power studies using a current-sensor as one of the inputs. Isci and Martonosi [10] implement run-time power monitoring for a single-core Intel® Pentium™ 4 (Willamette) using a combination of performance counters and an inline external ammeter. Performance-counter results and current measurements are fed to a separate monitoring machine which computes power in software on intervals of about 400 milliseconds. The model estimates power for 22 microprocessor structures

Table 1: Run-time estimation techniques for high-performance CPUs with aggressive power management

Technique and inputs	Structures covered	Limitations
Runtime power-monitoring (e.g., [10])	core and few structures	need sensors, limited structure power
Wire or structure specific stats (e.g., [4] [2] [3])	many	many input statistics
Counter-based techniques (a) (e.g., [12][1])	core	no structure power
Counter-based techniques (b) (e.g., [26][20])	few structures	ratio of input statistics to structures not scalable
CAMP	many (180)	

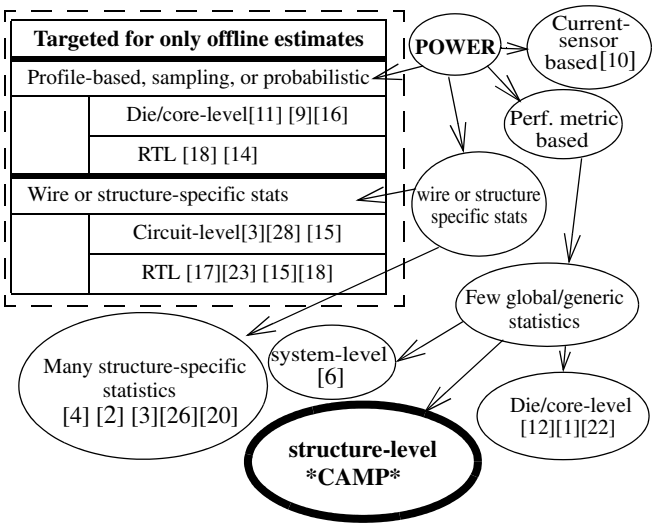

FIGURE 1: Taxonomy of power-estimation techniques: related work and CAMP

using 24 statistics plus the meter. While this model is quite useful, the external hardware requirement for model construction makes it unwieldy for run-time power monitoring in the field, particularly in a multicore microprocessor where each processor core might require an on-package current monitor.

Other power estimation proposals rely solely on performance metrics as input statistics. One category uses a large number of wire-specific or structure-specific statistics relative to the number of structures for which power is estimated. Examples include the Cai-Lim model [4], which uses structure activity and power densities to compute power for 17 structures and PowerTimer [3], which relies on many switching factors as input statistics. Another example is Wattch [2], which is similar to the Cai-Lim model but derives energy costs from wire-delay and circuit models instead of power density. Wattch uses a few dozen input statistics to estimate power for about a dozen structures. Wu et. al [26] estimate power for 15 P4 structures using up to 22 input statistics, and Peddersen et. al [20] estimate power for 5 structures in a small embedded-type core. While these estimation techniques are conceptually suitable for run-time power estimation, the number of input statistics relative to the number of structures for which they compute power makes them impractical for run-time estimation for many structures.

Another category of power estimators rely on a handful of global or generic input statistics for core-level estimates, and thus would be more suitable for run-time power estimation. One example is Joseph and Martonosi's work [12], which uses performance counters to estimate power in the Intel® Pentium Pro™ over 10 millisecond intervals. The authors note that there is limited clock-gating in this microprocessor and thus little variation between minimum and maximum power (about 25%). Estimating run-time power on a complex wide-issue processor with aggressive clock gating is substantially more difficult, as noted by Isci and Martonosi [10]. Due to limited clock gating, Joseph and Martonosi [12] were able to assume constant power for many structures and primarily focus on memory and ALU operation power. Another work, Bellosa [1], estimates core power at run-time in software, using Intel® Pentium™ 4 performance counters. Sharkey [22] also estimates core-level power.

It is also possible to estimate system-level power (e.g., including DIMMS, disks, and other I/O) using global statistics. One example is Economou et. al [6].

CAMP differs from other techniques that use global input statistics to estimate core or die power in that CAMP estimates per-structure power for over 180 structures on an aggressively clock-gated microprocessor.

2.2 Offline Analytical Power Monitoring

There are a number of analytical power models that target offline power estimation using inputs that would be unsuitable for run-time estimation in hardware. These techniques are summarized in the table at the left of Figure 1. One category uses profiling, sampling, and statistical or probabilistic analysis to estimate power. The typical approach is to perform detailed simulations of a few design points, use data from these to fit inference models that relate performance/power to architectural statistics and then predict the performance/power at different design points [16,11,9]. For example, Lee and Brooks [16] use trace-driven simulation of 4000 samples to fit linear regression models that relate several micro-architectural design parameters to performance and full-CPU power. Similar analysis can be done at the RTL level, such as Macii et al. [18], which uses statistical sampling for RTL estimation and Katkoori et al. [14], which generates behavioral profiles from RTL simulations.

Offline estimation at the circuit level or RTL can also use wire-specific or structure specific inputs (e.g., temperature estimates or transistor count) for even finer granularity. For instance, Srinivasan, et al. [23] present an empirical approach to determine the optimal pipeline depth considering both power and performance constraints. Zyuban and Strenski [28] present a mathematical approach based on hardware intensity, which relates delay and average energy consumption. Lee et al. [17] analyze low-level power for embedded DSP software. Landman et al. [15] surveys power-estimation techniques ranging from counting gate equivalents to circuit-level estimation. PowerTimer [3] and Wattch [2] can also be considered in this space when specific values of input bits (e.g., the addends in an addition operation) are considered as one of the inputs.

2.3 Analytical Estimation at Design

Analytical performance models are related to offline power-estimation techniques. These models might be used in conjunction with CAMP to provide estimates for the input parameters to the power model before an architectural simulator is available. Many analytical performance models determine overall performance for an ideal processor (no misses, infinite hardware resources) and then derive performance limits imposed by adding constraints, such as dependencies and hardware limitations. The models are typically parameterized with data obtained from trace-driven simulations. Noonburg and Shen [19] develop such a model based on probability matrices, while Karkhanis and Smith [13] present a more concise model that calculates performance using two components - a constant, ideal component and a performance loss component.

3 Power Modeling Background

CAMP extends an existing design-time power-modeling methodology for simulation to generate activity and power metrics at run-time. In this section, we provide background on design-time

power modeling techniques used in CAMP to estimate run-time metrics.

3.1 Components of Microprocessor Power

Microprocessor power can be broken into three major components: static, dynamic-idle, and dynamic-active. Static power, also called leakage power, is dissipated whenever the microprocessor is connected to a power-supply. Static power is not a function of microprocessor utilization but is a function of manufacturing process technology, circuit topology, temperature, and the power-gating status of the core. The methodology to estimate static power in a microprocessor is now somewhat well-understood. Process technology and circuit topology are constants at runtime, making runtime leakage computation a single equation with two variables: 1) temperature and 2) power-gating status. Temperature can be obtained from on-die sensors [8]. Power gating, not to be confused with clock gating, refers to leakage-reduction techniques such as gated-Vdd [21]. Power-gating status can be obtained from a core's power-control unit. Because estimating leakage at run-time is less complex than estimating dynamic power, we do not discuss it further.

Dynamic-idle power is dynamic power that is not conditionally clock-gated, and thus is not a function of utilization. This power dissipates whenever the microprocessor core is clocked. Thus, as long as global clock-gating conditions are known (i.e., is the entire core gated?), we can also estimate dynamic-idle power in a microprocessor.

The final, largest, and hardest-to-estimate single component, dynamic-active power, is dissipated according to microprocessor utilization. This component represents power for structures that are either not clocked or are conditionally clock-gated when idle. Dynamic-active power is often the largest component of power and often represents over 50% of total power. Estimation of dynamic-active power is the focus of this work; from this point forward, "power" refers to dynamic-active power unless stated otherwise.

3.2 Circuit and Architectural Power Modeling

Microprocessor power estimates for architectural design are typically calculated by extending the energy-modeling methodology for circuits to the architectural level. For a circuit, power (P) for a single-switching event is calculated as the capacitance of the circuit (C) times the square of the voltage (V) times the clock frequency. Average power can be calculated by multiplying energy by the activity factor (a), or fraction of cycles the circuit is switching. This calculation leads to the familiar equation:

$$P = aCV^2f$$

At the architectural level, we know the activities of structures and blocks, such as register files, arithmetic-logic units (ALUs), and caches, but we do not know the activity of individual signals or circuits. However, architectural simulation can abstract the individual signals up to the structure and block level.

Power modeling techniques, such as Intel's Architecture Level Power Simulator (ALPS) [8], Wattch [2], or PowerTimer [3], abstract the power equation from the circuit level to the architectural level. Each method obtains power estimates differently and at different structural granularity, but with the same overall goal. Here we focus on ALPS. ALPS uses power estimates from previous designs and/or circuit simulations on micro benchmarks to estimate power for architectural events (e.g., register reads, ALU operations) down to the functional-unit-block (FUB) level. The power-per-event numbers can then be multiplied by activity to compute power. This computation is equivalent to modifying the circuit-power equation above to use *effective capacitance* ($C_{effective}$) as the equivalent capacitance of architectural events and *architectural activity* (A) as the fraction of cycles a specific event occurs:

$$P = AC_{effective}V^2f$$

Note that in ALPS an individual structure (e.g., a cache) can have multiple FUBs (e.g., data bank, tag bank, decoder) and each FUB can have multiple events (e.g., read, write), each with their own effective capacitance. ALPS estimates dynamic-active power for individual FUBs and an entire core by summing all of the event powers for each FUB and summing all of the FUB powers.

As described, this power-modeling methodology is useful in detailed software-based execution-driven simulators. Wattch [2] uses a similar technique to map power for a few tens of structures to a few tens of events, while Intel has used ALPS to maps power for hundreds of FUBs to hundreds of events in complex products like the Pentium 4 [8].

Table 2: System parameters

Instruction issue	4, out-of-order
I-cache	64KB 4-way
D-cache	64KB 8-way, 2 cycles
Branch Predictor (size)	Bimodal (512)+ Gshare (1024)
Branch Target Buffer	4K entries; 16-way
Fetch / Decode queues	14/24 entries
Reservation Stations	32
Reorder Buffer Entries	96
Load/Store Buffers	50/24 entries
L2 cache	1MB, inclusive, 8-way, 10-cycles
L2 miss latency	200 cycles

3.3 Limitations of Existing Power Methodologies

The key limitation that prevents methodologies, such as ALPS (or Wattch or PowerTimer), from being used to estimate power in hardware or from being used in design before a detailed simulator is available is the sheer number of input statistics required. Tracking many tens or hundreds of statistics in hardware is overly complex. It is equally daunting for an early design-time power model (before a detailed simulator is created) to generate those statistics, as they tend to be highly specific (e.g., read of d-cache bank 0 or access to instruction-decoder 2). A technique both to reduce the number of statistics and to generalize them to be less structure-specific would enable hardware power monitoring and early design-time power estimates.

4 Methodology and Infrastructure

In this section, we describe the infrastructure that we use to model activity factors and power and the system that we model.

We use a detailed execution-driven simulation in the Asim [7] simulation environment to simulate an Intel® Core™-like microprocessor core. The parameters of our core are shown in Table 2 For this core, we have a detailed ALPS-like [8] power model that

covers over 200 architectural structures comprising over 300 FUBs. ALPS is Intel's premier micro-architectural power simulator and has been validated against silicon for several products. It is generally believed to be accurate to within 5% to 10%. This power model is integrated into our simulator, which tracks architectural activities for each of those FUBs.

To make our power results process (voltage and frequency) and logic-style independent, we normalize all power results with respect to voltage and frequency, and the $C_{effective}$ of the macro-instruction-queue (IQ) write event. We call these units *pseudo-Watts.* Recalling the equation from Section 3.2, pseudoWatts are expressed as:

$$PseudoWatts = A\frac{C_{effective}}{C_{effectiveIQWrite}}\frac{V^2 f}{V^2 f} = A\frac{C_{effective}}{C_{effectiveIQWrite}}$$

To show energy, we use the equivalent in similar *pseudoJoules.*

For our implementation of CAMP, we focus on events in 60 architectural structures—comprising 180 FUBs—that cover 95% of the core's dynamic-active power. Due to space limitations, we limit our discussion to the 22 structures shown in Table 3. However, the overall non-structure-specific results do include all 60 structures. Each of these structures corresponds to one or more power *macros*. Power macros are our smallest unit of power computation and correspond to well-defined structures such as register-files or execution units. Each macro has one or more events that are used to compute its power (e.g., the uop buffer may be read or written).

To train our CAMP model, we run a suite of 73 benchmarks including SPEC CPU 2K [24], SPEC CPU 06 [25], multimedia, server, and TPC-C workloads. To test our model, we run a suite of 83 *different* traces from among the same workload classes. For training and testing, detailed simulations are run for 10 million macro-instructions chosen as representative traces. Memory structures are warmed up prior to the detailed runs.

5 CAMP: Activity Factor Estimation

In this section, we describe our methodology for exploring several micro-architectural statistics and selecting the best subset that needs to be directly observed in order to estimate microprocessor power. A reduced number of statistics can be monitored simply in hardware or can facilitate quick power-performance trade-offs at design-time. We focus on general statistics that report performance and which correlate well with the activity factors (a.f.s) of most pipeline structures. We then use these statistics as *predictors* (Terminology note: in this section we mean *predictors* in the linear regression sense, not in the architectural speculation sense) to construct linear regression models that can be used to estimate activity factors for each structure. Note that for our studies, we use simple linear regression models. Other models are possible, including multi-variable regressions and iterative analysis, such as in [26]**.** Analysis of these alternative models is beyond the scope of this work; the goal of CAMP is to show that it is possible to estimate core and fine-grain structure power using few input statistics, not to pick the optimum regression model.

Our studies show that nine general micro-architectural statistics are adequate to measure 95% of dynamic-active power with an average of 90% accuracy for a typical out-of-order pipeline. By using the same set of statistics that are used to summarize performance to also estimate activity factors (and hence, power), we demonstrate that hardware utilization is the key commonality between performance and power.

5.1 Selection of Predictors

The activity factors of most structures in a microprocessor pipeline correlate well to key system utilization statistics. This behavior is not unexpected - activity factor is fundamentally a function of utilization. For example, activity factor in the instruction decoders is expected to correlate well to the number of instructions fetched and activity factor of the retire logic is expected to correlate well to the number of instructions retired. However, some correlations may not be particularly obvious as in these examples. For instance, do writes to the data operands stored in the ROB correlate to instructions fetched, instructions retired, load hits, or some combination of these and other statistics? Further, a key question for design-time decisions is whether such correlations hold over micro-architectural perturbations.

To answer these questions, we performed correlation analysis between activity factors and architectural statistics including macro-instructions-per-cycle (IPC), instructions-fetched-per-cycle (*speculative-IPC,* or SIPC, since the metric includes mis-speculated instructions), number of loads, number of stores, number of branch instructions, and number of 128-bit (SIMD) floating-point instructions. Figure 2 plots the correlation coefficients for the macros (as defined in Section 4) for a subset of these parameters, averaged over the 73 training benchmarks described in Section 4. The x-axis lists the structures in pipeline order, starting with the FETCH_CONTROL_LOGIC to the ROB_RETIRE_LOGIC, and the y-axis plots the correlation coefficients. (Recall that correlation coefficients range from -1 to 1, with values near zero corresponding to no correlation and values near 1 corresponding to strong negative or positive correlation.)

Many activity factors correlate highly (correlation coefficient over 0.9) to either or both of instructions-per-cycle (IPC) and instructions-fetched-per-cycle (SIPC). However we found that 64% of all macros do not correlate well to either of these basic statistics, so others are needed for a complete model. It is interesting to observe that some macros correlate well with only a few statistics - for example, the floating-point ALUs correlate well with only the SIMD instruction rate but with none of the other statistics, and the branch rate is the only statistic to correlate highly to the EXE1 execution unit (which is used to compute branch outcome). It is

Table 3: Microprocessor structures considered

Front End	i-cache tag
fetch control logic	macro-inst (macro-op) decoder
i-cache data	micro-inst (uop) decoder
macro-inst queue (IQ)	micro-instruction (uop) buffer
Back End	
allocate resources (ALLOC)	rename
ROB allocate (ROB_ALLOC)	ROB operand (high & low)
reservation station (RS) cam	RS scheduler
RS source regfile (RS_SRCRF)	Execution units (EXE)
committed PC logic	ROB retire logic
Memory Subsystem	DTLB
data-cache (DCU) tag	data-cache (DCU) data
store buffer (STB)	load buffer (LDB)

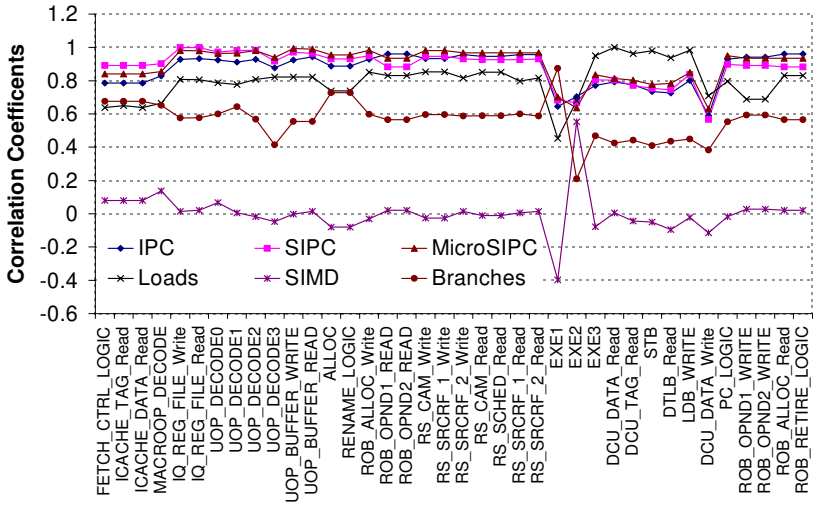

Table 4: Statistics (Predictors) used for each fit

Statistic	Fit
IPC	**Used in all fits**
SIPC	
Load rate	
Store rate	
FP Ld/St rate	
SIMD inst rate	
64-bit FP inst rate	
Micro-SIPC rate	uSIPC, Best
Branch rate	Branches, Best
Load hit rate (per cycle)	DistLoads
Load miss rate (per cycle)	DistLoads

FIGURE 2: Per-structure a.f. correlation with selected micro-architectural statistics

also interesting that the instruction decoders and decode queue correlate fairly well with the number of loads. We also assess the robustness of these correlations across microarchitectural perturbations (e.g., structure size changes); those results are presented in detail in Section 7.

Based on the analysis of which statistics correlate highly to structure activity factors, we narrowed the number of statistics to 9 to use as predictors for the linear regression models described next.

5.2 Model Construction

We fit linear regression models for each structure of interest using the predictors described in the previous section. To simplify CAMP, we aim to keep the number of predictors to about 10, so we performed four fits using a combination of the 11 predictors shown in Table 4. In what follows, we term a specific combination of predictors as a fit and the resulting linear equations as models. Seven statistics are common to all fits. The fit *DistLoads* uses load hits and load misses as additional predictors. The fit *uSIPC* uses the micro-SIPC, and the fit *Branches* uses the branch rate. Finally, the fit *BEST* uses micro-SIPC as well as branch instructions (but not load hits and misses). Each fit uses at most nine predictors.

The accuracy of these models also depends on the workloads used to generate samples for the fit. We expect the workloads that exercise all macros in the pipeline to be good samples. To assess the sensitivity of the a.f. models to different workloads, we use two different sets to generate samples. The first set, termed as *MIX* uses the 73 training benchmarks described in Section 4. The fitted models are then used to predict the activity factors over 83 different traces derived from the same mix. The second set of samples is derived using traces from only the SPEC CPU 2K suite (SPEC2K). The resulting models are then used to predict a.f.s over the same 83 traces as before.

This analysis produces an activity factor model for each structure (i.e., power macro) and the resulting a.f. numbers can be used in the power equation from Section 3.2 to compute power estimates. For example, the activity model for the reservation-station first-source operand register read (RS_SRCRF_1_Read) macro using the Branches fit is shown in Figure 4. Note that the largest

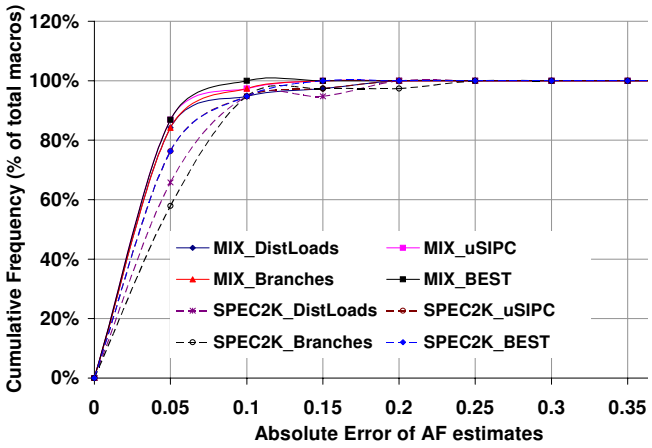

FIGURE 3: CDF of absolute error for all fits

contributing statistic to this activity factor (i.e., largest coefficient) is the branch rate.

As discussed in Section 1, CAMP equations like that in Figure 4 expose the dependence of structure power on global statistics, allowing design-time reasoning about trade-offs regarding specific architecture choices or workload characteristics. For example, the equation indicates that an increase in the rate of floating-point loads and stores will tend to decrease activity of the RS_SRCRF_1 structure, but an increase in 64-bit floating-point instructions will increase activity. This sort of reasoning is more difficult with per-structure activity models that rely entirely on structure-specific statistics (e.g., number of reads to the structure) that are not related to global characteristics.

5.3 Model Evaluation

In this section, we evaluate the quality of the fits used to predict activity factors. We expect high-quality fits because the predictors for the fits were chosen to ensure that statistics that correlate highly to each structure's activity were included. We expect fits that include micro-SIPC and branch rate to perform the best overall, because micro-SIPC correlates highly to activity for many struc-

$$AF = 0.026 + 0.24IPC + 0.016SIPC + 0.079Loads + 0.11Stores - 0.20FPLdSt + 0.17FP64 + 0.32Branches$$

FIGURE 4: RS_SRCRF_1_Read activity factor equation using Branches fit.

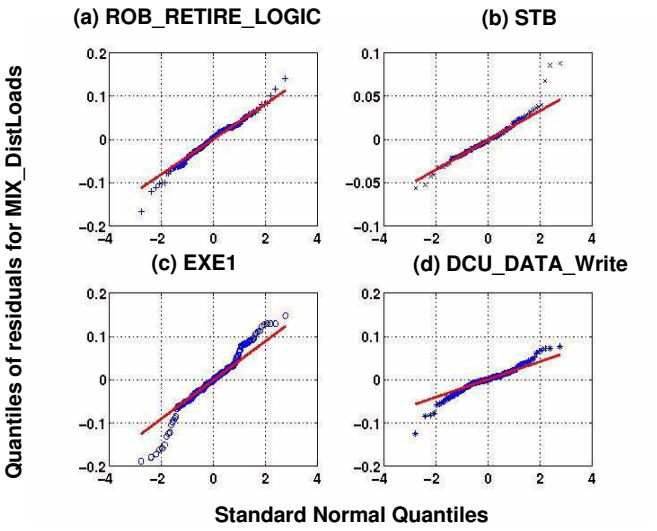

FIGURE 5: MIX_DistLoads fit residuals QQ-plots

tures, and branch rate is the only highly-correlating statistic for one structure, as we saw in Figure 2 from the last subsection.

We use the adjusted R-sq values, distribution of residuals, and the mean absolute error of predictions as metrics to assess the fits. Figure 3 plots the cumulative distribution (CDF) of the absolute error in a.f. predictions, and Table 5 summarizes the metrics used to assess fits. Each CDF curve represents a fit, which is labeled by the workload used to train the fit (MIX or SPEC2K) and the statistics in the fit (DistLoads, uSIPC, Branches, and BEST).

On an average across all fits, more than 77% of the models (each structure has its own model, or equation, within each fit) predict a.f. within 5%, 96% predict within 10% and all predict within 25% of the actual a.f. values. This result shows that a single set of linear models is adequate to predict power over a wide range of workloads. The SPEC2K fits show that even if the models are generated using a previous generation of benchmarks, they are capable of accurately predicting power for future workload generations (e.g., SPEC CPU 06) and even other classes of workloads (e.g., TPC-C). This behavior is not surprising because the equations are fundamentally a function of the hardware and its utilization, not the specific workload run on the hardware. (Of course that is true only if the workloads in the fit fully exercise the structures.)

We also examine fits by studying the distribution of residuals using a quantile-quantile plot where standard normal quantiles are plotted on the x-axis and residual quantiles on the y-axis. A linear trend indicates a model with good prediction capability. These plots also provide insight into selection of predictors. Figure 5(a) and Figure 5(b) indicate good fits because both of these macros correlate well with the selected parameters. Figure 5(c) is a qq-plot for the EXE1 macro using the MIX_DistLoads fit. EXE1 correlates well with the branch instruction rate which is not part of this fit, so the residuals show significant deviation from normal. Similarly, DCU_DATA_Write correlates well with only one parameter (store instruction rate, which is not shown in Figure 2), so all the fits deviate slightly from the normal. In general, we found that the distribution of residuals for the "MIX_BEST" and "SPEC2K_BEST" followed the normal most closely.

Table 5: Summary of Fit Evaluation Metrics

Fit	Adj. R_sq		Mean Relative Error (%)		Mean Absolute Error	
Training Workload:	MIX	SPEC 2K	MIX	SPEC 2K	MIX	SPEC 2K
DistLoads	0.957	0.974	10.2	21.24	0.035	0.050
uSIPC	0.972	0.972	7.84	19.73	0.026	0.041
Branches	0.959	0.974	9.29	22.90	0.031	0.053
Best	0.976	0.975	7.37	20.00	0.025	0.041

The combination of high-quality and simplicity of the CAMP models allows intuition into the relationship between architectural statistics and structure power. Simplicity also allows the models to be used for both run-time and design-time power estimates as explained in the following sections.

6 CAMP for Run-time Power Estimation

Accurate run-time power estimation is valuable in driving a variety of run-time power management techniques in both hardware and software. Prediction accuracy, granularity (both temporal and structural) and the cost (in area and time) are the three main aspects of a run-time power estimator. We would like the energy estimates to be accurate to enable correct power management decisions, and we would like choice in the temporal granularity of estimates. Finer structural granularity of estimates can be used to drive more aggressive, fine-grained power management in hardware, while coarser-granularity power information can be used at higher levels in hardware or software. Finally, estimates should be obtained without a large area or power overhead.

In this section, we explain how CAMP can address these challenges. We outline a low-overhead hardware mechanism that couples the CAMP linear equations with a few hardware counters to construct an accurate, fine-grained, low-overhead run-time energy estimator. We then show that CAMP can produce accurate core and structure energy estimates at different temporal granularities.

6.1 CAMP Hardware for Run-time Estimation

CAMP may be implemented using a set of hardware counters that monitor microprocessor statistics for an interval of time over which we then compute estimated power. The interval size may vary from tens of thousands of cycles, to enable fine-grained hardware power management, to millions of cycles, to enable software-based power management. (If long calculation intervals are desired, larger hardware counters are needed to prevent overflow.)

CAMP uses the output of the hardware counters and pre-computed per-structure linear regression models to compute the estimated power. The effective capacitance used in the power-computation equation from Section 3.2 is provided by an architectural power model for the microprocessor. Figure 6 shows the structure of our run-time power estimator. Because only nine counters are required and the energy computation is performed at relatively infrequent intervals over many cycles, power overhead is negligible.

6.1.1 Hardware Implementation Details

The hardware required to implement the predictors could be complicated because the predictors we use to construct the regression models are generally a count divided by the number of cycles in an interval. To simplify the hardware required, we suggest that interval lengths be limited to a power-of-two number of cycles. This limitation allows the division to be implemented with a simple

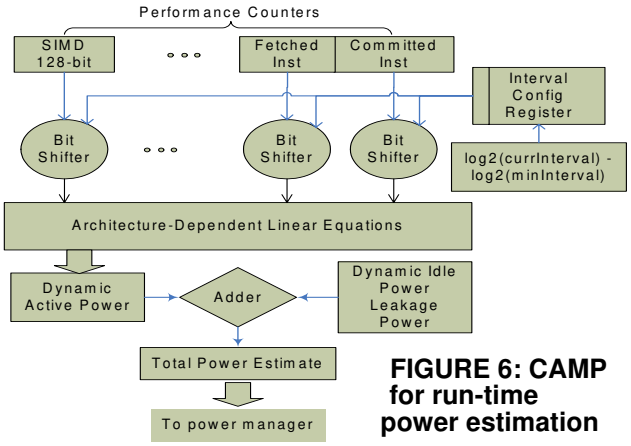

FIGURE 6: CAMP for run-time power estimation

bit shift left by log2 the number of cycles per interval. To limit our mechanism to integer arithmetic, we define a minimum interval size (*minInterval* in Figure 6) of 65536 (i.e., 2^{16}) cycles and express each predictor in terms of events per 2^{16} cycles. Thus, digits beyond the decimal point are not needed to maintain high accuracy. This simplification would be implemented by simply shifting log2(65536) or 16 bits less than would be indicated by the interval size and adjusting the value of the linear-equation coefficients accordingly. This simplification allows us to use only integer arithmetic and has no significant effect on the accuracy of our mechanism. An alternative simplification that would not require defining a minimum interval would be to simply pad the counter values and coefficients with zeros (effectively multiplying by 2 to the number-of-zeros power) and adjust the linear equations accordingly, though this might increase the width of the arithmetic logic.

If programmable interval lengths are not necessary, a final simplification is to fix the interval size, eliminate the dividers (bit shifters) altogether, and simply use the raw counts as input to the linear equations. This simplification requires changing the coefficients of the linear equations to be the original values times the number of cycles in the fixed interval length. (Of course, this change is one-time and static, not a dynamic calculation.)

6.2 Accuracy of Run-time Energy Estimates

In this subsection, we assess the accuracy of energy estimates provided by CAMP using the BEST and SPEC2K_BEST fits from Section 5.2. We use the SPEC2K_BEST fit to demonstrate that a CAMP model built from older workloads is capable of estimating energy in newer workloads, as discussed in Section 5.3. We expect the microprocessor energy estimates to be accurate compared to detailed simulation, because of the high quality of these fits. We expect accurate estimates regardless of interval length, because as we saw in Section 5.3 the fits are functions of utilization and not workload or program phase.

We estimate energy consumption for a 35-trace subset of the 83 traces used for testing in Section 5.3; we do not show results for all 83 traces to avoid clutter on the plots. This subset contains traces from all of the workload categories from Section 4. Energy is estimated at 64K (65536) cycle intervals, 1M (1048576) cycle intervals, and full-simulation intervals. Energy values are normalized using the mechanism described in Section 4.

Figure 7 ((a) through (d)) show the per-interval energy estimates for two traces for 1M and 64K interval lengths, respectively, for both the BEST and SPEC2K_BEST fits. In all cases, the estimates track well with the actual energy consumption (from detailed simulation), also shown in the figure. On an average across all the 35 traces, the mean relative error for BEST and SPEC2K_BEST is 2.7% and 6.2% respectively with the maximum error being 14% and 39% respectively.

Figure 8 is a scatter plot of the actual (x-axis) vs. the estimated (y-axis) total energy for the 35 traces which shows that the estimates are generally accurate regardless of workload. One outlier for both the BEST and SPEC2K_BEST fit is SPEC06-libquantum,

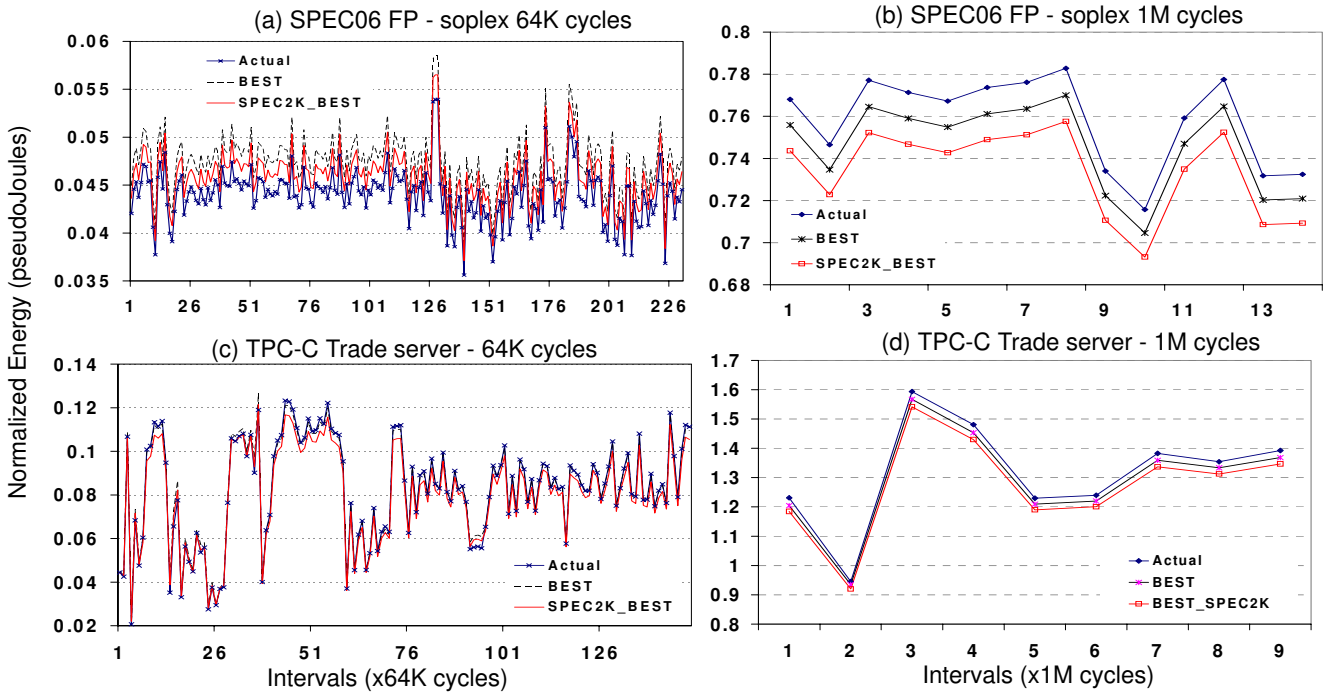

FIGURE 7: Accuracy of per-interval energy estimates for two interval sizes

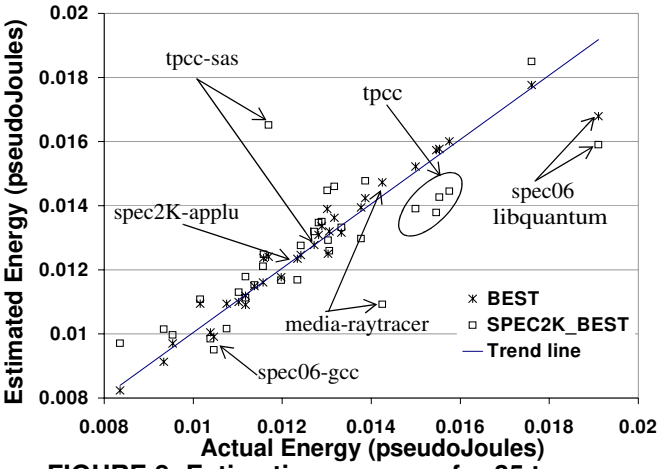

FIGURE 8: Estimation accuracy for 35 traces

although even for this outlier the relative estimation error is only 12%. This application has a large working set (32MB) which strains the memory hierarchy differently from any of our other traces; as a result, the activity estimates for the data cache and L2 cache have substantial errors. Another application, tpcc-sas, is an outlier for the SPEC2K_BEST fit, but not for the BEST fit, indicating that there are advantages to training over a broader set of workloads.

6.2.1 Accuracy of per-structure energy estimates.

CAMP's energy estimates are also accurate for individual structures. Figure 9 shows the 29 macros corresponding to the 22 structures highlighted earlier in Table 3 (some structures such as the uop decoder comprise more than one macro) along the x-axis, normalized actual energy and estimated energy (using the BEST fit and SPEC2K_BEST fit) in the bottom graph, and estimation error for those structures in the top graph. The bottom graph shows that estimated energy tracks actual energy well. The top graph shows that for the BEST fit, the error is less than 5% for all but one highlighted structure, the store buffer (STB). The generally high accuracy of CAMP's per-structure estimates extends beyond the highlighted structures to the 60 power-macro structures discussed in Section 4 that comprise 180 FUBs and 95% of core power. Although not shown in Figure 9, the additional structures were included in the distribution shown in Figure 3, which shows that nearly 90% of structures had estimation errors less than 5%.

The SPEC2K_BEST fit, which uses a much more limited training set than the BEST fit, still achieves reasonable per-structure estimates. All but 6 of the 29 estimates are within 20% of the actual value. In terms of normalized energy (the bottom graph), even the 6 outliers are not extreme. This demonstration of an extreme case where a fit based on SPEC2K is used to estimate energy for a much broader suite of workloads (including server and media workloads not in SPEC2K or SPEC06) shows the versatility of CAMP in estimating per-structure power.

In summary, these results indicate that CAMP is an effective solution to the accuracy and granularity challenges in run-time power estimation. These results also show that the predictive models that make up CAMP capture the inherent relationship between hardware utilization, performance and power irrespective of the workload being run. Such run-time power estimation can be used to facilitate a variety of hardware and/or software-based power-

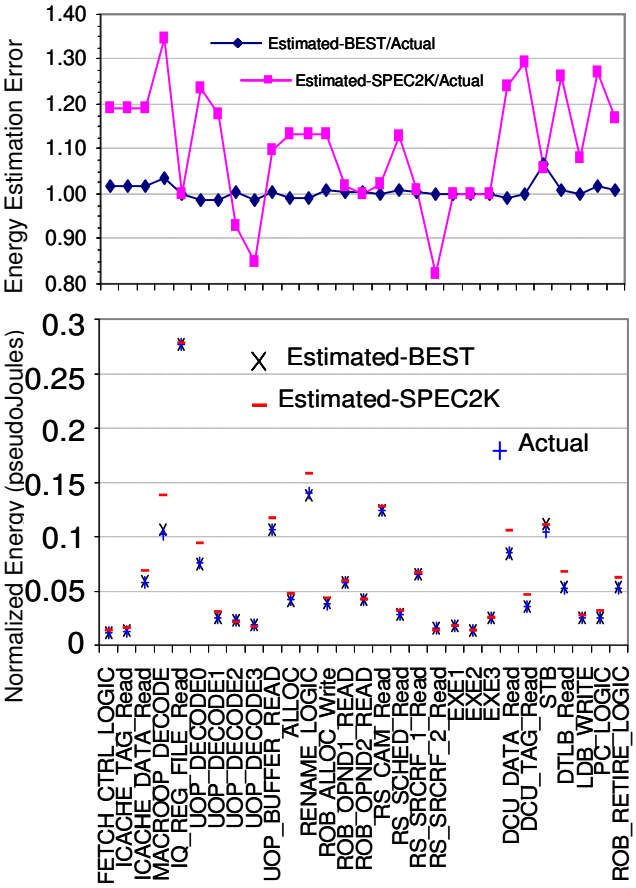

FIGURE 9: Per-structure energy estimate accuracy

management techniques such as voltage-frequency scaling, clock gating and selective disabling of resources. In the following section, we describe how the same models can be used during design time to guide early power-performance trade-off analysis.

7 CAMP for Design-time Power-Perf. Analysis

Power estimation is a key aspect of early-stage micro-architectural design space exploration. In addition to estimates of core power, it is useful to understand variation in per-structure power to drive power-aware design. Designers typically rely on detailed, cycle-accurate simulation coupled with complex power models to obtain detailed power estimates, creating a bottleneck for exploring the power impact of design changes. CAMP aims to reduce the bottleneck by providing an intuitive model that accurately estimates energy using the same parameters that summarize performance.

Our key observation is that the accuracy of CAMP's activity estimations is accurate across incremental micro-architectural structure-size changes. The CAMP model constructed for a base architecture continues to provide accurate energy estimates for different micro-architectural changes across all the workloads we consider. This consistency across different designs allows CAMP to be used as an effective early-stage power-performance analysis tool. We show that CAMP provides useful insights and facilitates reasoning about the power-performance behavior across the pipeline using only a single set of performance parameters and without requiring a detailed power model.

In Section 7.1 we provide results to validate CAMP's robust-

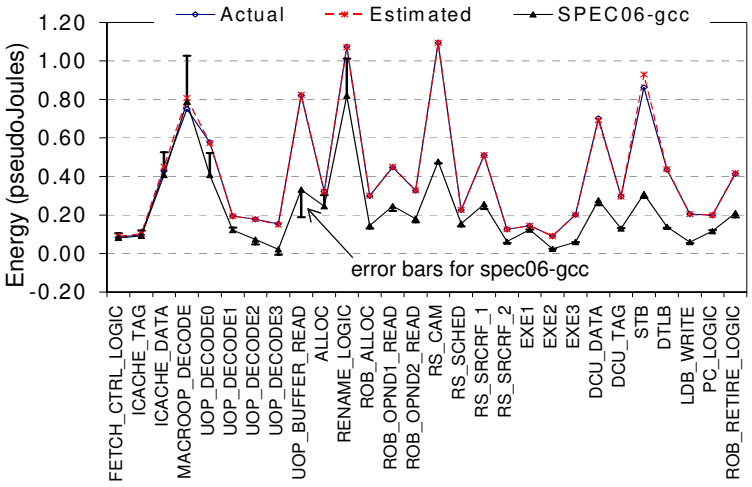

FIGURE 10: (a) Per-macro accuracy of energy estimates across all benchmarks and all structure size changes

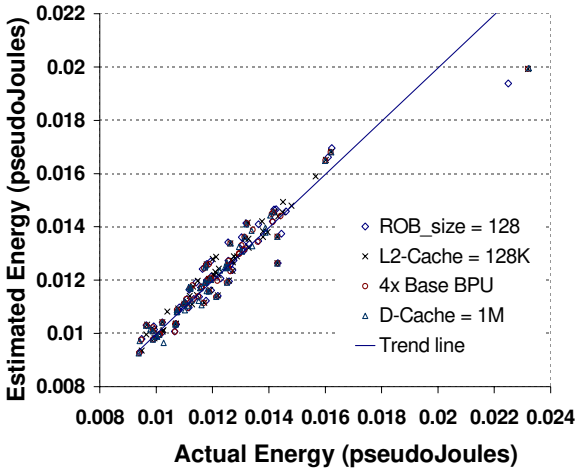

FIGURE 10: (b) Accuracy of energy estimates across structure-size changes

ness across micro-architectural perturbations. In Section 7.2 we give an example of how CAMP can be used to compare different micro-architectural choices at early design stages.

7.1 Robustness of CAMP to structure size changes

In Section 5.2, we demonstrated the close correlation between activity factors of selected structures with several micro-architectural statistics. We performed a similar analysis for 12 different micro-architectural configurations obtained by varying the data cache, L2 cache, IDQ, RS, ROB and BPU sizes. We expect activity correlations to remain strong under perturbations because the relationship between structures is not changing, only their sizes.

We found that correlations of these statistics to per-structure a.f. remain consistent across structure size changes. Across all macros, all micro-architectural perturbations and all statistics, the standard deviation of correlation coefficients was 15% with a maximum deviation of 26% (observed for micro-SIPC). To assess the predictive power of the a.f. models of a single fit across different micro-architectural configurations, we plotted the CDF of their absolute prediction error (not shown). Analogous to the correlation analysis, the regression models also hold fairly consistently across structure size perturbations - 82% of the models predict a.f. within 5%, 93% within 10%, and all within 40% of the actual a.f.s.

Given the values of a few micro-architectural statistics for different designs, CAMP can be used to obtain energy estimates instead of using a detailed power model. Figure 10(b) illustrates a scatter plot for specific cases where the RS-ROB, L1 d-cache, L2 cache, and BPU sizes were varied. The plot shows the actual vs. estimated total energy using the BEST fit for a set of 58 traces from the SPEC06-FP, SPEC06-INT, SPEC2K-FP, SPEC2k-INT, TPC-C and multimedia suites. As before, each trace is 10M instructions long. A linear trend line is included to show how the estimated values track the actual energy consumption. Across the 12 designs, we observed a mean relative error of 3.3% and maximum error of 14.6% (observed for IDQ size set to 16) over all the traces.

Figure 10(a) plots the average estimated energy and actual energy (from detailed simulation) for individual macros; each point is averaged across the 58 traces and across all 12 micro-architectures. The estimated and actual points frequently overlap, and due to the small prediction error, the error bars when averaged over all traces were too small to show up on this plot. Instead we add an additional series for estimated SPEC06-gcc energy, and we plot error bars for that benchmark to verify that the error is small even for individual traces.

These results support our intuition behind CAMP that both the power and performance of a micro-architectural configuration can be obtained by tracking a common set of utilization parameters across the micro-architecture. This analysis implies that a CAMP model for a previous generation microprocessor may be used for initial estimates in the next-generation processor in the common case where the new design builds from the previous and most of the differences are structure-size changes. For example, the Intel® Pentium™ Pro, Intel® Pentium™ II, Intel® Pentium™ III, Intel® Pentium™ M, and Intel® Core™ each derived from the previous generation P6 processor. Of course, a CAMP model from a previous generation would not be useful for an entirely new microarchitecture (e.g., Intel® Pentium™ 4), but all-new micro-architectures are not the common case.

Next, we show how this property of CAMP can be used to facilitate early-stage power-performance analysis.

7.2 Examples of Early Design Analysis with CAMP

In this subsection, we discuss a case study that uses CAMP to study the power-performance trade-off of several designs obtained by varying the RS and ROB sizes. We assume that values for performance statistics used in the CAMP BEST fit (IPC, SIPC, loads, etc.) are provided to us for a base design using cycle-accurate performance simulation. These values could also be obtained from analytical performance models as explained in Section 2. We expect that, given a set of performance statistics for a base design and a modified design, CAMP can provide accurate total as well as per-structure power across different designs to compare their power-performance behavior.

For each design, we show the IPC values obtained from simulation and the per-structure energy and the overall energy-delay^2 (ED^2) product obtained using CAMP. Processor Energy-delay^2 results are based on the approximation that the per-event power of the structure being perturbed, such as the ROB, did not change with the perturbation. This approximation is acceptable for early design experiments and acceptable to the first order because: 1) Only one or two structures are being perturbed, and no single structure makes up a majority of the processor power; and 2) Our main

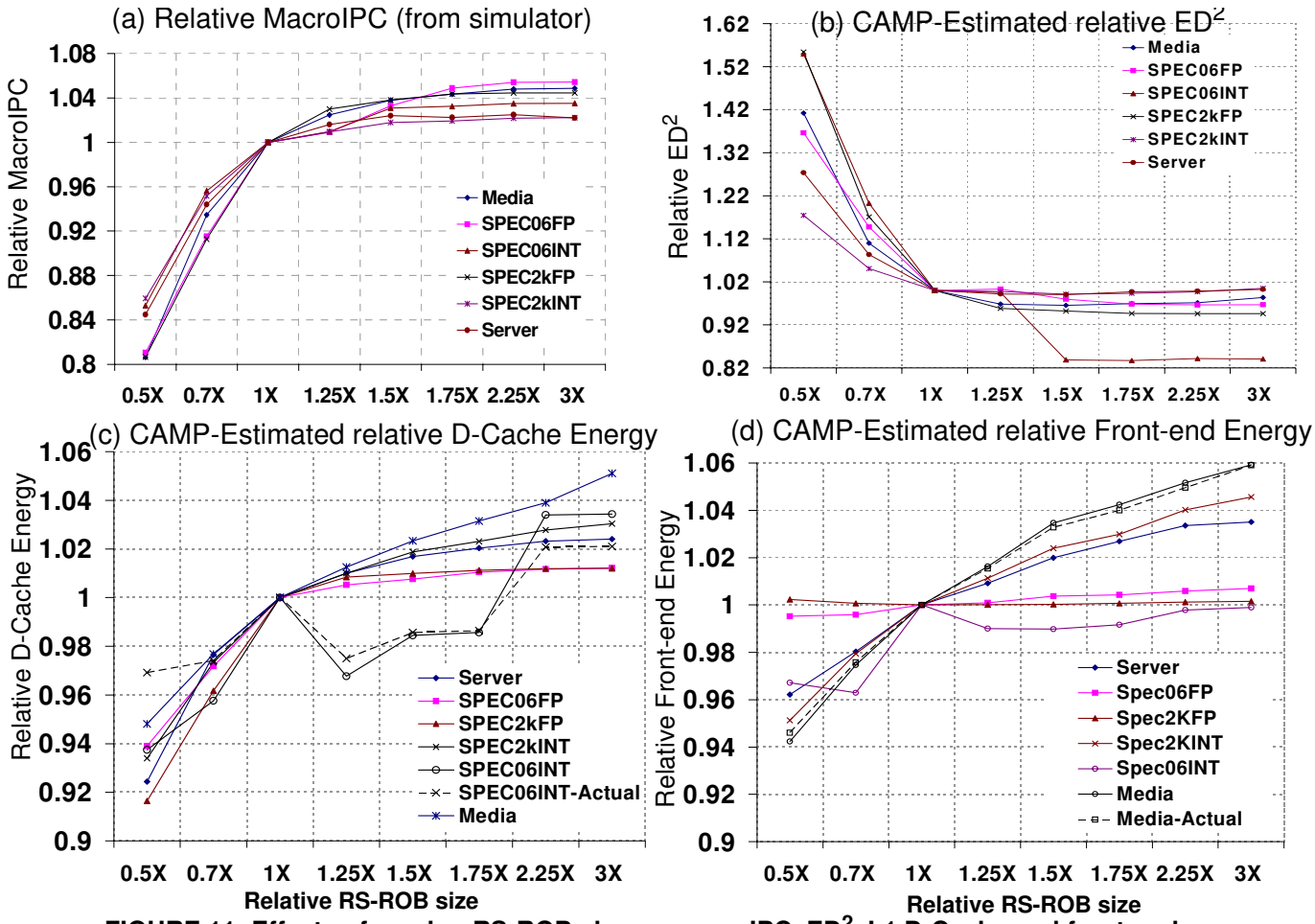

FIGURE 11: Effects of varying RS-ROB size on macroIPC, ED^2, L1 D-Cache and front-end energy

interest is in the impact of the architectural change on the power of *other* parts of the pipeline, not the structure being perturbed; obtaining an event-power analysis for an individual structure is a circuit-power-modeling concern orthogonal to CAMP.

Each point is obtained by averaging across 58 traces from among the six categories described in Section 4 and normalizing with the corresponding value from the base micro-architecture.

7.2.1 Varying RS and ROB size.

The ROB is a key structure that determines the total number of instructions in the pipeline. Though we refer to these variations as ROB size changes, we vary the RS and ROB sizes together because of their inter-dependence. Figure 11(a) and Figure 11(b) show the effect of different ROB sizes on the macro-IPC and the ED^2 for the six benchmark suites. The "1X" point corresponds to the base architecture. From the figure, it can be seen that there is a slight performance gain by increasing the ROB sizes. However, the ED^2 curves show that, except for SPEC06-INT, the energy-efficiency remains fairly flat for ROB sizes greater than 1X. To understand the behavior of SPEC06-INT, we looked at the energy expended by other structures, specifically the L1 D-cache and the front-end for different ROB sizes.

Figure 11(c) and Figure 11(d), plotting estimated energy for the L1 data cache and front-end respectively, show that SPEC06-INT displays an unusual behavior where both the L1 d-cache and front-end energy actually *decrease* as ROB size is increased. This behavior also explains the drop in SPEC06-INT Energy-delay^2 in Figure 11(b). To validate that this result is not an experimental anomaly, we compared our estimates against actual energy numbers from detailed simulation. Figure 11 (c) plots this comparison for SPEC06-INT (dotted line), and shows that the CAMP estimator tracks the results from detailed simulation. (In addition, Figure 11(d) plots both estimated and actual energy for Media, indicating that increasing trend is not an anomaly either.)

Typically, a detailed power model would be required to uncover such behavior, but CAMP can provide valuable insight by tracking the hardware utilization across the entire pipeline due to changes in a single structure. This case study also reinforces that it is necessary to vary multiple structures in synchrony and consider their interaction when making micro-architectural design decisions.

8 Conclusion

Effective CPU power management would benefit from knowledge of run-time microprocessor power consumption in both the core and individual microarchitectural structures, such as caches, queues, and execution units. Increasingly feasible per-structure power-control techniques, such as fine-grain clock gating, power gating, and dynamic voltage/frequency scaling (DVFS) would benefit from run-time estimates of per-structure power. However, run-time computation of structure power estimates based on utilization would seem to require daunting numbers of input statistics.

To address the challenges of estimating per-structure power in

hardware, we propose a new technique, called Common Activity-based Model for Power (CAMP), to estimate activity factors and power for microarchitectural structures. In spite of using a relatively few nine input parameters based on general microprocessor utilization statistics (e.g., IPC and load rate), our linear-regression-based model estimates activity and dynamic power for over 100 structures in an out-of-order x86 pipeline and estimates core power with an average error of 8%. Because the computations utilize few inputs, CAMP is simple enough to implement in hardware, providing run-time structure and core power estimates for dynamic power management. Because the input statistics are generic in nature and the model remains accurate across microarchitectural changes, CAMP provides simple intuitive equations relating global microarchitectural statistics to structure activity and power. These equations provide a simple technique that can equate changes in one structure's activity to power variations in other structures across the pipeline.

Because the input statistics are general and the accuracy of the model is maintained across incremental microarchitectural changes, design can use CAMP to estimate the power impact of design choices. Before a detailed architectural simulator is available, designers can use intuition and the CAMP equations to estimate how architectural changes will impact per-structure as well as overall chip power. Our validation suggests that these results are accurate as compared to detailed, cycle-accurate simulation. For instance, across 12 different micro-architectural configurations obtained by varying sizes of multiple structures, CAMP estimates energy to within 3.3% of the actual energy across traces from six benchmark suites.

References

[1] F. Bellosa, A. Weissel, M. Waitz, and S. Kellner. Event-driven energy accounting for dynamic thermal management. In *Proceedings of the Workshop on Compilers and Operating Systems for Low Power (COLP03)*, Sept. 2003.

[2] D. Brooks, V. Tiwari, and M. Martonosi. Wattch: A framework for architectural-level power analysis and optimizations. In *Proceedings of the 27th Annual International Symposium on Computer Architecture*, pages 83–94, June 2000.

[3] D. M. Brooks, P. Bose, V. Srinivasan, M. K. Gschwind, P. G. Emma, and M. G. Rosenfield. New methodology for early-stage microarchitecture-level power-performance analysis of microprocessors. *IBM Journal of Research and Development*, 47(5/6):653–670, Sept. 2003.

[4] G. Cai and C. H. Lim. Architectural level power/performance optimization and dynamic power estimation. In *Cool Chips Tutorial colocated with MICRO32*, Nov. 1999.

[5] C.-S. Chen, J.-C. Lo, and T. Xia. An indirect current sensing technique for iddq and iddt tests. In *Great Lakes Symposium on VLSI*, pages 235–240, Apr. 2006.

[6] D. Economou, S. Rivoire, C. Kozyrakis, and P. Ranganathan. Full-system power analysis and modeling for server environments. In *2006 Workshop on Modeling, Benchmarking and Simulation*, June 2006.

[7] J. Emer, P. Ahuja, E. Borch, A. Klauser, C.-K. Luk, S. Manne, S. S. Mukherjee, H. Patel, S. Wallace, N. Binkert, R. Espasa, and T. Juan. Asim: A performance model framework. In *IEEE Computer 0018-9162:68-76*, pages 68–76, Feb. 2002.

[8] S. H. Gunther, F. Binns, D. M. Carmean, and J. C. Hall. Managing the impact of increasing microprocessor power consumption. *Intel Technolgoy Journal Q1 2001*, 5(1), Feb. 2001.

[9] E. Ipek, S. McKee, B. de Supinski, M. Schulz, and R. Caruana. Efficiently exploring architectural design spaces via predictive modeling. In *Proceedings of the Twelfth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS XII)*, Oct. 2006.

[10] C. Isci and M. Martonosi. Runtime power monitoring in high-end processors: Methodology and emprical data. In *36th International Symposium on Microarchitecture (MICRO 36)*, pages 93–104, Dec. 2003.

[11] P. J. Joseph, K. Vaswani, and M. J. Thazhuthaveetil. Construction and use of linear regression models for processor performance analysis. In *Twelfth International Symposium on High Performance Computer Architecture (HPCA)*, pages 99–108, Feb. 2006.

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

[13] T. S. Karkhanis and J. E. Smith. A first-order superscalar processor model. In *Proceedings of the 31st International Symposium on Computer Architecture (ISCA 31)*, pages 338–349, June 2004.

[14] S. Katkoori and R. Vemuri. Architectural power estimation based on behavior level profiling. *Journal on VLSI Design, Special Issue on Low Power*, 1996.

[15] P. Landman. High level power estimation. In *Proceedings of the International Symposium on Low Power Electronics and Design*, pages 29–35, Aug. 1996.

[16] B. C. Lee and D. M. Brooks. Accurate and efficient regression modeling for microarchitectural performance and power prediction. In *Proceedings of the Twelfth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS XII)*, Oct. 2006.

[17] M. T.-C. Lee, V. Tiwari, S. Malik, and M. Fujita. Power analysis and minimization techniques for embedded dsp software. *IEEE Transactions on VLSI Systems*, 5(1):123–135, Mar. 1997.

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

[19] D. B. Noonburg and J. P. Shen. Theoretical modeling of superscalar processor performance. In *Proceedings of the 27th International Symposium on Microarchitecture (MICRO 27)*, pages 52–62, Nov. 1994.

[20] J. Peddersen and S. Parameswaran. CLIPPER: counter-based low impact processor power estimation at run-time. In *Asia and South Pacific Design Automation Conference*, Jan. 2007.

[21] M. D. Powell, S.-H. Yang, B. Falsafi, K. Roy, and T. N. Vijaykumar. Gated-Vdd: A circuit technique to reduce leakage in cache memories. In *Proceedings of the 2000 International Symposium on Low Power Electronics and Design (ISLPED)*, pages 90–95, July 2000.

[22] J. Sharkey, A. Buyuktosunoglu, and P. Bose. Evaluating design tradeoffs in on-chip power management for cmps. In *Proceedings of the International Symposium on Low Power Electronics and Design*, pages 44–49, Aug. 2007.

[23] V. Srinivasan, D. Brooks, Michael Gschwind, P. Bose, V. Zyuban, P. N. Strenski, and P. G. Emma. Optimizing pipelines for power and performance. In *Proceedings of the 35th International Symposium on Microarchitecture (MICRO 35)*, pages 333–344, Nov. 2002.

[24] The Standard Performance Evaluation Corporation. Spec CPU2000 suite. http://www.specbench.org/osg/cpu2000/.

[25] The Standard Performance Evaluation Corporation. Spec CPU2006 suite. http://www.specbench.org/osg/cpu2006/.

[26] W. Wu, L. Jin, J. Wang, P. Liu, and S. X.-D. Tan. A systematic method for functional unit power estimation in microprocessors. In *Proceedings of the 43rd Conference on Design Automation*, July 2006.

[27] B. Xue and D. M. H. Walker. Built-in current sensor for iddq test. In *IEEE International Workshop on Defect Based Testing (DBT)*, pages 3–9, Apr. 2004.

[28] V. Zyuban and P. Strenski. Unified methodology for resolving power-performance tradeoffs at the microarchitectural and circuit levels. In *Proceedings of the International Symposium on Low Power Electronics and Design*, pages 166–171, Aug. 2002.