# New Methodology For Early-Stage, Microarchitecture-Level Power-Performance Analysis Of Microprocessors

**Article** *in* Ibm Journal of Research and Development · October 2003

DOI: 10.1147/rd.475.0653 · Source: IEEE Xplore

CITATIONS
103

READS
420

**6 authors**, including:

Pradip Bose
IBM
**295** PUBLICATIONS **7,388** CITATIONS

SEE PROFILE

Michael Gschwind
Meta
**148** PUBLICATIONS **2,744** CITATIONS

SEE PROFILE

# New methodology for early-stage, microarchitecture-level power–performance analysis of microprocessors

D. Brooks
P. Bose
V. Srinivasan
M. K. Gschwind
P. G. Emma
M. G. Rosenfield

*The **PowerTimer** toolset has been developed for use in early-stage, microarchitecture-level power–performance analysis of microprocessors. The key component of the toolset is a parameterized set of energy functions that can be used in conjunction with any given cycle-accurate microarchitectural simulator. The energy functions model the power consumption of primitive and hierarchically composed building blocks which are used in microarchitecture-level performance models. Examples of structures modeled are pipeline stage latches, queues, buffers and component read/write multiplexers, local clock buffers, register files, and cache array macros. The energy functions can be derived using purely analytical equations that are driven by organizational, circuit, and technology parameters or behavioral equations that are derived from empirical, circuit-level simulation experiments. After describing the modeling methodology, we present analysis results in the context of a current-generation superscalar processor simulator to illustrate the use and effectiveness of such early-stage models. In addition to average power and performance tradeoff analysis, PowerTimer is useful in assessing the typical and worst-case power (or current) swings that occur between successive cycle windows in a given workload execution. Such a characterization of workloads at the early stage of microarchitecture definition helps pinpoint potential inductive noise problems on the voltage rail that can be addressed by designing an appropriate package or by suitably tuning the dynamic power management controls within the processor.*

## 1. Introduction

Power dissipation limits have emerged as a first-order design constraint for microprocessors. In fact, at the low end of the performance spectrum, that is, in the world of handheld and portable devices or systems, it is not unusual for power to dominate performance as the *primary* design constraint. Battery life and system cost constraints therefore force a microprocessor design team to consider power over performance. Increasingly, however, power is a key design aspect in the workstation and server market as well [1]. For high-end applications, increasing microarchitectural complexities, clock frequencies, and die sizes are pushing the chip-level (and hence, system-level) power consumption, literally to the edge. If designs continue to pursue a performance-centric, power-unaware approach, traditional air cooling for multiprocessor servers may soon have to be replaced by refrigeration or liquid cooling. This would cause a break point (with a step



**0018-8646/03/$5.00 © 2003 IBM**

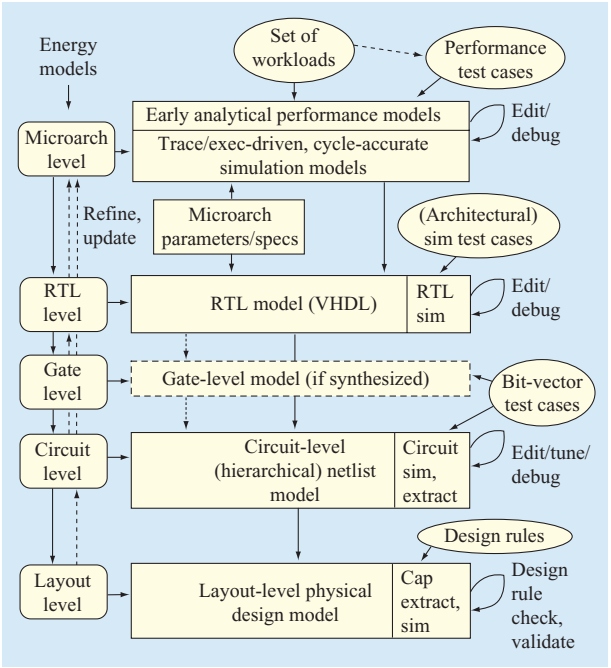


**Figure 1**

Typical modeling and analysis tool flow in microprocessor design.

*upward*) in the ever-decreasing price–performance ratio curve. Hence, a modern microprocessor system design approach that takes into account power consumption and dissipation limits early in the design cycle and maintains a power-centric focus across all levels of design abstraction should have an edge over competing approaches.

**Figure 1** shows a typical modeling and analysis tool flow in microprocessor design. There are variations in the flow and the exact levels of design description, depending on the product [e.g., high-end microprocessor, application-specific integrated circuit (ASIC) or embedded processor system, digital signal processor (DSP)]; however, the essential elements of the overall methodology are the same.

At the very early stages of design definition, microarchitects start with analytical CPI (cycles-per-instruction) performance models that lead into trace or execution-driven, cycle-by-cycle simulators. Full or sampled benchmark traces are processed through such simulators, driven by a microarchitecture parameter file. The goal of this design space exploration phase is to optimize the choice of microarchitectural parameters for CPI performance under design constraints known at that stage. The performance model is typically written in a standard systems programming language such as C/C++, and is designed to project execution times (in cycles) for input application traces; it typically does not model the actual execution of the instructions, but only the execution timing. At the end of the high-level design phase, the register transfer level (RTL) model is developed using a hardware description language such as VHDL [VHSIC (very-high-speed integrated circuit) Hardware Description Language]. This model, when ready and validated, incorporates the full register-transfer-level functionality with accurate cycle-by-cycle timing flow behavior of the modeled machine. The main objective of developing this RTL model is to verify the functional integrity of the full logic-level description of the machine. For designs that are synthesized, the VHDL description is also used for generating a gate-level model as a precursor to the circuit-level refinement. For custom designs, the circuit-level netlist description is developed independently, in parallel with or after RTL coding, starting with schematic entry. Beyond the circuit netlist model, which is subject to timing verification and tuning, the design description is further refined into the physical design level, with layout, placement, and routing information included.

As the design model is refined from the highest level of abstraction to the lowest, the accuracy and detail of functional and timing information increase. Validation of the model at each level and between successive levels becomes more time-consuming and complex in proceeding down the design hierarchy. In current processor design, energy models are defined and used at virtually all levels of this hierarchy in order to include power as a constraint in the design optimization process.

In this paper, we describe *PowerTimer*, initially introduced in [2], a toolset that is used for microarchitecture-level power–performance modeling. Although currently targeted for high-end microprocessors, its methodology should also be applicable to other domains (e.g., embedded or special-purpose microprocessor systems). PowerTimer uses a variety of sources for power models, such as output from a circuit-level power analysis and extraction tool or analytical models derived in a bottom-up modeling methodology. Microarchitecture-level energy models can be based on either A) technologically scaled projections from low-level (e.g., circuit-simulation-based, RTL-simulation-based, or actual hardware-measurement-based) energy characterization data available from *previous* designs; or B) analytical models that attempt to characterize the power on the basis of the implementation structure (at the gate level or circuit level with or without interconnect effects) of each microarchitectural entity or event (e.g., an issue queue entry or a cache access event). The models are in the form of equations driven by parameters available from a given CMOS technology.

In Section 2, we describe the overall structure and operation of PowerTimer, with simple examples to illustrate how it is used in practice.

Section 3 describes the generation of energy models that are used by PowerTimer. We include a brief description of Common Power Analysis Methodology (CPAM) [3], a circuit-level power analysis tool developed at IBM. We use this tool for deriving type-A energy models. We also describe the analytical modeling techniques for type-B energy models.

In Section 4, we report specific examples of applications of PowerTimer, with experimental results and analysis.

Section 5 covers related prior work in some detail. In Section 6 we provide a summary of the main content of this paper and a brief discussion of future work. In particular, we point to the need for developing robust pre-chip-fabrication silicon model validation methodologies that are appropriate for current early-stage tools such as PowerTimer. We refer briefly to prior approaches and explain how we plan to adapt and extend those techniques for PowerTimer.

## 2. PowerTimer: Overall structure and operation

**Figure 2** is a block diagram of the PowerTimer toolset—currently in use during the early-stage definition of future high-end PowerPC* processors. The base cycle-accurate performance simulator that is used depends on the target processor. For general research studies, we use the code base of an available parameterized simulation toolset designated as Turandot/MET [4]. (MET denotes a microarchitecture evaluation toolset, and Turandot denotes the cycle-accurate performance simulator within the MET.) A cycle-accurate simulator such as Turandot is capable of reading instructions from a program's executable code (or a trace of the same) and simulating the timing flow within the target processor. Pipeline latencies, instruction flow bandwidths, and stall/flush occurrences (for example, in the case of cache misses and branch mispredictions) are all modeled as accurately as possible, with the goal of projecting the execution time (in processor cycles) of an input application program. The key new capability added to early-stage performance simulators in such a toolset is provided in the form of processor-specific energy models that can be used during the simulation run. The research microarchitecture power models (RMAP) function suite is designed to meet this need. An input file that defines the microarchitectural parameters for the base performance simulator also helps configure the associated unit-level energy functions. For example, the size, type, and configuration parameters defining an issue queue unit within the simulation model also determine the energy function to be called to compute power usage for that unit during the simulation run.

The other input to the RMAP suite is the set of circuit style and technology parameters that are specific to a particular target processor. Examples are latch types and the CMOS technology generation with associated voltage and scaling parameters.

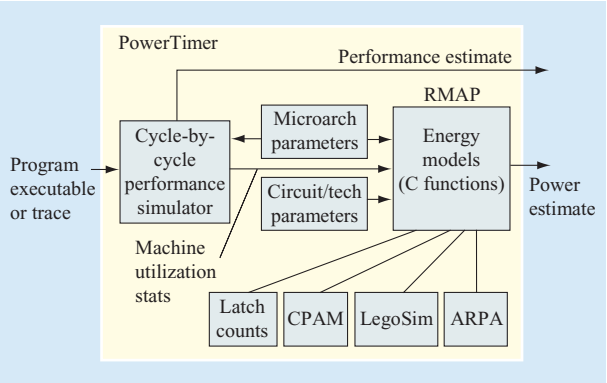


Figure 2

Block diagram of PowerTimer toolset.

The RMAP suite can be used in two different modes: a) integrated and compiled into the simulator code for the functions to be called on a cycle-by-cycle basis, depending on the macros and units that are activated in a given simulation cycle; b) called once, using average unit usage statistics at the end of the simulation run. The integrated mode (a) allows cycle-by-cycle characterization of power, power–performance, and power swing ($di/dt$) metrics in a straightforward manner. The postprocessing mode (b) allows the computation of average power and power–performance data very efficiently, without significant slowing of the simulation.

Our goal of supporting multiple processor development projects naturally steered us to the requirement of designing the RMAP function suite in a manner that allows maximum flexibility and portability. At this time, we are working with customized, processor-specific energy functions for several target processors in development within IBM. However, on the basis of the experience that we gather from these design points, which differ significantly in functionality and performance, we are developing a parameterized, portable RMAP suite for use by other processor development groups within the company. In association with our research MET simulation toolset [4, 5], we also plan to release a generic, nonconfidential version of RMAP for use by our external, university research partners.

The high-level pipeline structure of the parameterized Turandot/MET microprocessor model [5] is shown in **Figure 3**. The modeled microarchitecture is similar in complexity to that of a current-generation PowerPC processor [6, 7]. The model assumes an in-order front end in which, on a given cycle, the instruction fetch unit

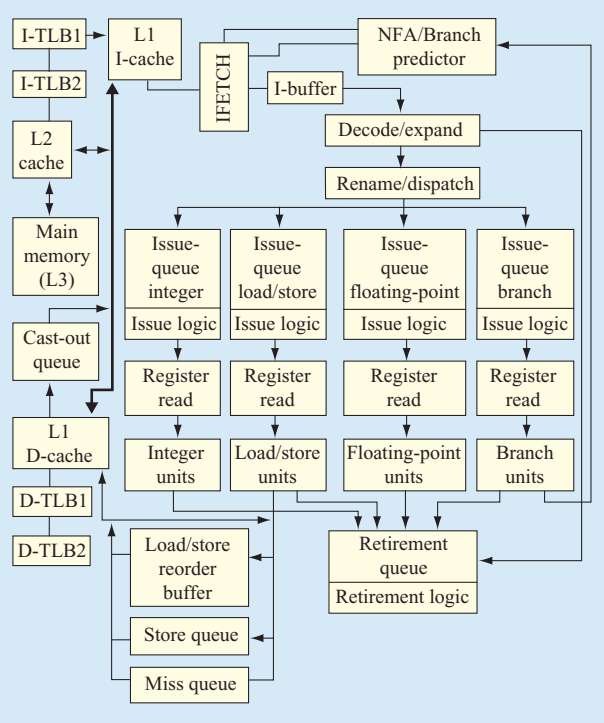


**Figure 3**

High-level pipeline structure of the parameterized Turandot/MET microprocessor model.

(IFETCH) accesses the level-1 instruction cache (L1 I-cache) to bring the next sequential group of instructions into the instruction buffer (I-buffer). The maximum number of instructions that can be fetched per cycle is a parameter that can be adjusted (like most of the other bandwidth parameters in the model). The decode/expand unit is a parameterized multicycle pipeline unit that decodes up to five instructions per cycle to form a basic dispatchable group (DG). Some complex instructions (e.g., a floating-point "store with update") are "cracked" into two or more micro-operations in forming such a DG, which contains a maximum of five (micro)-operations. Since a branch instruction always causes a DG to terminate, a DG may contain from one to five (micro)-operations. After the register renaming and dispatch cycle(s), instructions from a DG are distributed into one of four different issue queues, depending on their operation types (e.g., integer, load/store, floating-point, or branch). Each instruction issue logic is implemented to support "out-of-order" issue of instructions into respective execution pipelines. Up to two instructions can be issued per cycle from each issue queue, after a "register read" pipeline cycle. The model supports out-of-order execution, with in-order retirement logic using a standard retirement queue (reorder buffer) mechanism. Other structures modeled are the various queues and buffers that are part of such out-of-order, superscalar processors. Separate next-fetch-address (NFA) predictor and two-level branch predictor mechanisms are supported. Turandot models an L1–L2 cache hierarchy, with split instruction and data L1 caches and a unified L2 cache. Instruction and data translation lookaside buffers (TLBs) are also modeled, along with the cast-out queue required to support a store-in caching mechanism for the L1 data cache. Main memory (L3) is modeled as an infinite, perfect storage with a constant (albeit parameterized) access latency.

As described in [4], this research simulator was calibrated against a pre-RTL, detailed, latch-accurate processor model (referred to as the R-model in [4]). The R-model is a custom simulator, written in C++ (with mixed VHDL "interconnect code"). There is a one-to-one correspondence of signal names between the R-model and the actual VHDL (RTL) model; however, the R-model is about two orders of magnitude faster than the RTL model and is considerably more flexible. Many microarchitecture parameters can be varied, albeit within restricted ranges. Turandot, on the other hand, is a classical trace/execution-driven simulator, written in C, which is one to two orders of magnitude faster than the R-model. It supports a much greater number and range of parameter values.

In this paper, we report power–performance results using the same version of the R-model which was used in [5]. We first used our developed energy models in conjunction with the R-model: This ensured accurate measurement of the resource utilization statistics within the machine. To circumvent the simulator speed limitations, we used a parallel workstation cluster; also, we postprocessed the performance simulation output and fed the average resource utilization statistics to the energy models to obtain the average power numbers. Looking up the energy models on every cycle during the actual simulation run would have slowed the R-model execution even further. While it would have been possible to obtain instantaneous, cycle-by-cycle energy consumption profiles through such a method, it would not have changed the average power numbers for entire program runs.

Having used the detailed, latch-accurate reference model for our initial energy characterization, we were able to examine the unit- and queue-level power numbers in detail in order to understand, test, and refine the various energy models. Currently, we have reverted to using an energy-model-enabled Turandot model for rapid CPI vs. power tradeoff studies with full benchmark traces. Turandot makes it possible to experiment with a wider range and combination of machine parameters.

## 3. Derivation of the energy models

In this section, we describe the methods used for deriving the RMAP energy function suite in PowerTimer. Several methodological paths are currently used to derive energy functions for use within the RMAP suite (see Figure 2).

The first path, used in very early-stage (concept-phase) modeling, derives energy models on the basis of unit- and pipeline stage-level latch counts estimated by the design team. These latch counts are estimated from a) logic-level bit specifications of individual functions, where available; or b) area and latch-density based projections from prior designs, suitably scaled by technology upgrade parameters. Per-latch-bit power dissipation numbers are easily estimated or measured for the target technology, and clocked latches are known to account for 70–80% of unconstrained logic power in current processors. Hence, a latch-based energy model formulation for the non-array portions is often adequate during concept-phase microarchitecture definition.

The second path begins with detailed, macro-level power simulation data that is available from prior processor projects. This path is useful in formulating models for design macros that tend to be reused in newer designs with relatively small changes. This method of energy function generation is also appropriate during the early stages of the actual implementation phase when schematic-level descriptions of the circuit netlists are available for use in circuit-level characterization and tuning. The low-level power characterization data is generated using a research tool designated as CPAM [3]. A utility script converts the macro-level power data into higher-level, unit-specific energy functions appropriate for use by a microarchitectural simulator. Processor-specific scaling parameters (for size, configuration, circuit style, and technology) are used to configure the derived RMAP functions in an appropriate manner. **Figure 4** depicts the derivation of the energy models in more detail.

The energy models are based on circuit-level power analysis that has been performed on structures used in a recent high-performance IBM PowerPC processor, the IBM POWER4* [7]. The power analysis has been performed at the macro level; generally, multiple macros combine to form one microarchitectural-level structure which we designate as a subunit. For example, the fixed-point issue queue (one subunit) might contain separate macros for storage memory, comparison logic, and control. Power analysis has been performed on each macro to determine its power as a function of the input (i.e., macro input data/control pins) switching factor (SF). The *HoldPower*, or power when no switching is occurring, is also generated. These two pieces of data allow the formation of simple (albeit idealized) linear equations for the power of each macro. In general, for the $i$th of $N$ macros that make up the microarchitectural unit being modeled, the power equation is modeled (from available CPAM data) as the linear equation

$$Power(i) = (C_i \times SF) + HoldPower(i),$$

where $C_i$ is a macro-dependent constant that defines the slope of a straight line, assuming the $x$ and $y$ axes to be $SF$ and $Power(i)$, respectively. The energy model for a subunit is determined by essentially "summing" the linear equations for each macro within that subunit. This results in an overall linear power equation for the whole microarchitectural unit, with unit-level *HoldPower* equal to the $HoldPower(i)$ summed over all macros that make up that unit. The slope of the resulting straight line is currently computed as a simple arithmetic mean of the individual $C_i$s. This is clearly an approximation; such averaging of the slopes should ideally be done by using appropriate *weights* for each component macro. The weight is (relatively) larger if the sensitivity to the input switching factor of that macro is (relatively) greater than the sensitivity to the input switching factor of the overall unit. Determination of the correct weights using empirical circuit-level experiments for large microarchitectural units is very time-consuming; hence, in the initial version of PowerTimer, the CPAM-based energy models assume equal weights for component macros within a given unit. Initial results in our ongoing power–performance validation studies (see Section 6 for a brief methodological description), indicate that this assumption does not have any significant impact on the relative accuracy of any useful microarchitecture-level tradeoff experiment.

We have generated these power models for all microarchitecture-level structures (subunits) modeled in our research simulator [5]. In addition to the models that specify the power characteristics for particular subunits (such as the fixed-point issue queue), we can derive power models for more generalized structures: for example, a generalized issue queue model. These parameterized models are useful for estimating the power cost of additions to the baseline microarchitecture. The

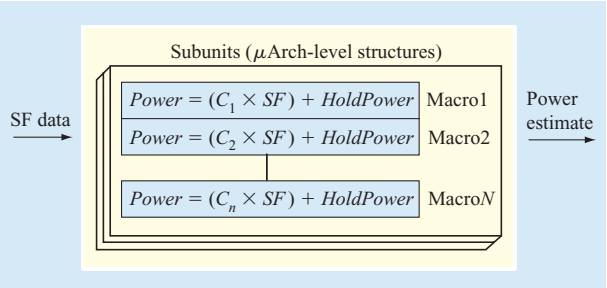


**Figure 4**

PowerTimer energy models.

generalized model is derived by analyzing the power characteristics of similar structures within the baseline microarchitecture. For example, the fixed-point, floating-point, logical-op, and branch-op issue queues have very similar functionality and power characteristics and have been used to derive a generalized issue-queue power model based on parameters such as the number of entries, storage bits, and comparison operations. CMOS-technology-related power scaling is done using a simple $CV^2f$ formulation. Voltage and frequency targets of the new processor design point are known from design specifications, and capacitance scaling is carried out using known dimension-scaling factors (for example, the factor of 0.7 that is usually selected to scale linear dimensions in going from one generation to the next [8]). These scaling factors can change somewhat, depending on the prior and target technology generation and perhaps even the target operating voltage or frequency; we act in consultation with the relevant design team and technology experts in using the best known technological scaling factors for use in PowerTimer.

Since we are interested in determining the power–performance tradeoff for future microarchitectures within a given product family, we must devise a method of scaling the power of microarchitectural structures as the size of these subunits increases. The scaling factor that is used is dependent on the particular structure; for example, the power of a cache array scales differently from that of an issue queue. In addition, as resources increase in size, they necessarily cause other structures to become larger. For example, as the number of rename registers increases, the number of tag bits within each entry of the issue queue increases. Generally, as we increase the number of entries in a structure, there is a proportional increase in the power. For this reason, we use linear scaling as a basis for many of the structures that we consider. In addition, we have performed detailed analysis on the scaling of queue and mapper structures. For these structures, we have determined the average power per storage bit and per comparison operation. As the queues and mappers increase in size, we compute the number of storage bits and comparisons that occur for the larger structures. We also use previously published work on power scaling within cache arrays, which we discuss in Section 4.

The third methodological path of energy model generation in PowerTimer (referring again to Figure 2) begins with newer circuit-simulation experiments to characterize latch types, clock buffers, and macros that are new and therefore not available from prior project data sets. With the anticipated use of clock-gating modes in future processors, the need to characterize the primitive energy characteristics (with and without clock gating) for the various latch types and associated local clock buffers is of paramount importance. This style of clocking and associated control logic is relatively new in IBM high-end processor development projects and therefore requires careful characterization at the circuit level. We have developed a methodology that uses PowerSpice-based detailed circuit-simulation experiments to obtain empirical energy data for primitive building blocks such as latches, clock-buffers, multiplexors, and interconnect (wiring) logic; analytical equations that have high-level organizational and technology parameters as arguments are formulated to model various combinations and sizes of the primitive blocks as they are used to design structures such as queues and buffers.

The fourth path to derive RMAP energy functions (Figure 2) is an independent, analytical formulation methodology. We are currently experimenting with analytical energy equations for regular structures such as SRAM array macros. The Array Power Analysis (ARPA) tool has been developed with the goal of modeling the energy and delay characteristics of IBM-specific SRAM array designs that implement cache macros. This tool is currently under validation, with reference data for specific SRAM designs obtained from prior designers.[1] Prior to the availability of a tuned, calibrated, and validated ARPA model for predictive use in our modeling work, we are relying on designer-provided energy models for customized SRAM array macros.

### *Scaling and fine-tuning the energy models*

An aspect of modeling that is dealt with in scaling macro-level energy data obtained from the analysis of previous designs to form energy models for use in future processors is repipelining [2]. This is necessary to reflect the impact of changes in pipeline depth of execution unit pipes (e.g., the floating-point unit or load-store unit pipe), and the corresponding increase in latch counts. For example, in going from an $m$-stage pipeline to an $n$-stage pipeline ($m < n$), we require the use of a suitable "repipelining" transformation to account for the larger latch counts in the latter design. Such a repipelining function must take into account the logic "shape function" that profiles the bit width variation with levels of logic in a given hardware function.

Another aspect pertains to modeling the variation of power usage of a given resource as a function of its size parameters. For example, we need to know how the power consumption of an array structure (such as a branch history table) or a queue structure (such as an out-of-order issue queue) would vary with the number of entries.

[1] Special thanks are due to Donald Plass and his team at the IBM Server Division in Poughkeepsie, NY, and to Rajiv Joshi at the IBM Thomas J. Watson Research Center in Yorktown Heights, NY.

Using multiple macro data points from previous designs to characterize queues of different units in a prior design, one solution is to extrapolate to obtain the power characteristics of a generic queue structure in some cases. An alternative is to use newer "per-latch," "per-entry," or cross-sectional array data available from more recent circuit simulation experiments conducted by us or by our colleagues. A more generic approach is to resort to analytical models based on the analysis of cross sections and design templates for the design of a variety of queue, register file, and other array-like regular structures.

A third aspect pertains to the use of reasonable switching factors for individual macro logic entities within the modeled processor. In classical trace-driven simulation, data values in registers or data path logic are often ignored; only the cycle-by-cycle timing behavior is modeled. This precludes the determination of accurate data bit vector switching factors during simulation. Thus, although unit usage frequencies are accurately modeled, the data switching factors on a given use of a resource are often not accurately modeled in early-stage simulators. The use of execution-driven mode in the base simulation is one way of factoring the data switching factor information accurately. This mode is also needed to model the power overhead of speculative waste in an accurate manner.

Of course, the use of technology-specific scaling factors to estimate power and area shrink factors for given macros may have the greatest impact on the accuracy of derived RMAP functions. The device models for future deep submicron technologies are still volatile in some cases, and the use of approximate scaling factors in early-stage modeling can lead to significant estimation errors. In this context, particular mention must be made of estimation of various forms of leakage power for future designs. This aspect of predictive extrapolation models is especially error-prone and therefore of concern in our modeling research.

There are clearly several different sources of error that make the estimation of power consumption a difficult task during the initial design phases. However, our goal is to provide a facility for making *relatively accurate* early-stage power and performance *tradeoffs*. For example, we wish to quantify the power savings potential with the use of fine-grain, pipeline-stage-level clock gating. We are less interested in the absolute accuracy of the power projections (with or without clock gating). Also, we are interested in projecting optimal issue widths and pipeline depths in concept-phase analysis under given definitions of power–performance efficiency metrics. For such metrics, the PowerTimer/RMAP methodology allows us to rank and compare different microarchitectural ideas that are proposed in early design stages.

Validation and tuning of the energy models in the context of their use in early-stage power–performance tradeoff analysis is an ongoing task that we expect will be continued for the duration of future processor development projects. Our prior performance model validation methodology (see, e.g., [5]) is being augmented to cover validation of power–performance models. Also, we are designing a set of circuit-level test designs in support of a high-performance processor test chip[2] that should allow us to calibrate our energy models for key structures (such as latches, issue queues, pipelined arithmetic units, and buffers) against measurements made at the RTL level and eventually at the hardware level.

## 4. Using PowerTimer to design power-efficient high-performance processors

In this section, we report experimental results based on the SPEC** CPU benchmark suites [9] and the TPC-C** trace [10]. All workload traces are PowerPC-based. The SPEC CPU95 and SPEC CPU2000 traces were generated using the Aria tracing facility within the MET toolset [4]. The particular SPEC trace repository used in this study was created by using the full reference input set. However, sampling was used to reduce the total trace length to 100 million instructions per benchmark program.[3] A systematic validation study to compare the sampled traces against the full traces was carried out to finalize the choice of exact sampling parameters. The TPC-C trace used is a contiguous (i.e., unsampled) trace collected and validated by the processor performance team at the IBM Systems Group facility in Austin, TX; it is about 180 million instructions long. In the next three subsections we present examples of the use of the PowerTimer toolset. The results show the average value of the CPI and average value of $[(CPI)^3 \times \text{power}]$ for the SPEC95 trace suite and the *tpcc* trace. Each SPEC data point was obtained by averaging across the benchmark suite. The choice of the particular power–performance efficiency metric $(CPI)^3 \times \text{power}$, which corresponds to energy $\times$ $(delay)^2$ (referred to as an $ED^2$ product metric in [11]) is not of great consequence in this discussion. *Qualitatively*, the basic conclusions derived from the results shown do not change if we change the weighting of CPI (i.e., the delay component) in such metrics. The issue of efficiency metrics is an important one that deserves detailed, separate exposition. In other words, the issue of choosing

[2] Designated as LPX, a low-power issue–execute processor test chip under joint development by our group and a research group from the University of Rochester; preliminary paper published at PACS'02 Workshop (at HPCA-8), February 2002.
[3] The sampled *apsi* trace was excluded from our SPEC95 trace repository, because of what we believe is an error in the way the samples were stitched together in this particular case; this error resulted in simulation deadlock at the end of a trace sample. Rather than include results based on a small number of instructions completed, we decided to discard this trace from our study. Since our group does not own this centralized SPEC trace repository, it has been difficult to get the trace problem fixed.

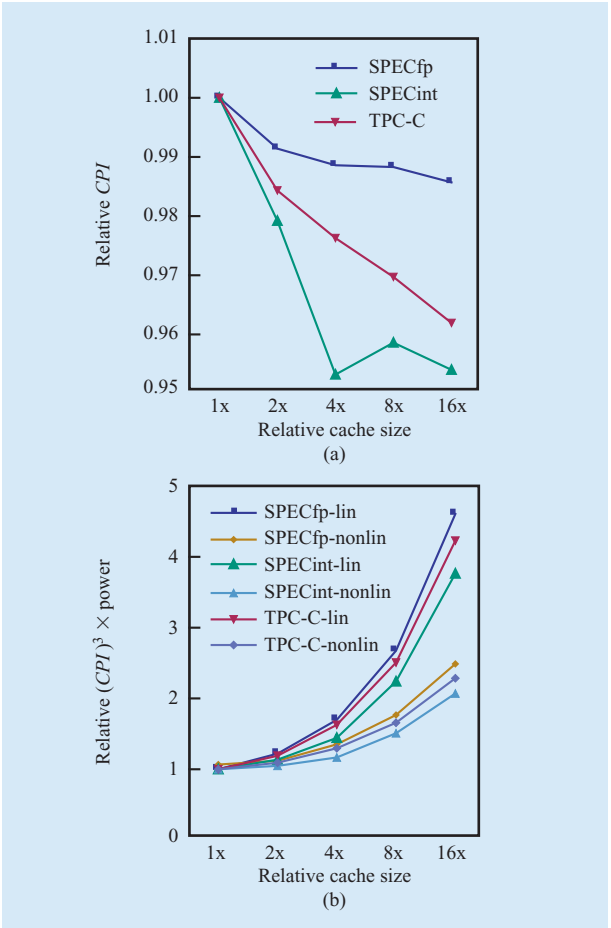


Figure 5

Variation of performance and power–performance with cache size.

the "right" exponent $x$ in an $ED^x$ product metric [or, equivalently, the exponent $y$ in a frequency-and-voltage-independent $(CPI)^y \times$ watt metric or an overall $(MIPS)^y$ per watt (where $y = x + 1$) metric] is crucial in making quantitative arguments about trends, tradeoffs, and optimality conditions in power–performance behavior. The interested reader is referred to other recent publications [12, 13] for a detailed treatment of this topic. In the PowerTimer effort, we have tended to use simple arguments based on the cube-root rule [11, 13] of the basic $CV^2f$ formulation of dynamic power in adopting the $(CPI)^3 \times$ watt and $(MIPS)^3$ per watt efficiency metrics. If the frequency $f$ is assumed to be roughly a linear function of the voltage $V$, and if net performance is assumed to be proportional to $f$, such a performance cubed-power formulation is seen to yield a voltage-invariant way of comparing the power–performance efficiency of two processors.

### *Data cache size and the effect of scaling techniques*

In this subsection we evaluate the relationship among performance, power, and L1 data cache size. We vary the cache size by increasing the number of cache lines while leaving the line size and cache associativity constant. **Figure 5** shows the results of increasing the cache size from the baseline architecture (points labeled 1x on the $x$-axes). Part (a) illustrates the relation between the cache size in the first-level data cache and the *relative* CPI for the workloads that we studied. The CPI value for each cache size is computed as a ratio, relative to the base 1x *CPI* for that workload.[4] Part (b) shows the relation when we consider the metric $(CPI)^3 \times$ power. From the figure, it is clear that the small CPI benefits of increasing the data cache are outweighed by the increases in power dissipation due to larger caches. In part (b) we show the results using two different scaling techniques. In the first technique, we assume that power scales linearly with the cache size. As the number of lines is doubled, the power of the cache is also doubled. The second technique is based on data from [14] which pertains to energy optimizations within multilevel cache architectures. In [14], data is presented for cache power dissipation for conventional caches with sizes ranging from 1 KB to 64 KB.

Using the second scaling technique, labeled *nonlin* in part (b), the cache power is scaled with the ratios presented in [14]. The increase in cache power by doubling cache size using this technique is roughly 1.46x, as opposed to the 2x that is obtained if the simple linear scaling method is used. Obviously, the choice of scaling technique can greatly affect the results. It is clear, however, that with either scaling choice, conventional performance-focused cache organizations do not scale in a power-efficient manner. Note that the curves shown in part (b) assume a fixed circuit/technology generation; they are intended to show the effect of adding more cache to the current design. In the simulation runs, we implicitly assume that the cache access time in cycles remains unchanged as the cache size is varied. This is, of course, an unrealistic assumption; and we could use a cache timing model to estimate the increase in access latency with the increase in cache size. However, this correction would only result in making the CPI decrease less sensitive to cache size increase; as a result, the $(CPI)^3 \times$ power curves would increase more rapidly with cache size while preserving the relative ordering of the various curves shown. Thus, in effect, the experimental results shown in Figure 5 are intended to illustrate the point that increases in cache size to benefit architectural performance at a

[4] Note that the apparent "outlier" point for SPECint** at the 8x relative cache size is nothing more than a very small experimental anomaly; note that the 4x, 8x, and 16x data points are all around 0.955 the relative CPI—i.e., the CPI is essentially flat beyond the 4x cache size. See also the discussion at the end of the section on the number of completion buffers.

given microarchitecture technology design point are invariably at the expense of degraded power–performance efficiency, unless other means, such as banked cache design with partitioned access, are used to curb the power growth.

### *Number of completion buffers*

In the target microarchitecture, the number of completion buffers determines the total number of instructions that can be active within the machine. The completion table is very similar to a reorder buffer in that it tracks instructions as they dispatch, issue, execute, wait for exceptions, and complete. **Figure 6** shows the effects of varying the number of completion buffers on performance and the power–performance metric. From part (a), it is evident that little additional performance is gained by increasing the number of buffers past the current design point (1x). When considering $(CPI)^3 \times$ power in part (b), we see that power efficiency is slightly degraded by increasing the number of entries, owing to a roughly 3% increase in the power dissipation of the core.

In the figure, we note an apparent anomalous behavior (for SPECfp**), where the CPI decreases for the 1.2x point, only to increase again at the 1.4x point. Such behavior is not uncommon in sensitivity experiments for aggressively speculative, superscalar processor models. When the number of completion buffers (or, equivalently, the reorder buffer size) is increased *without* commensurate increases in other resources, such behavior is particularly not unusual. Although increasing the number of instructions in flight generally tends to increase performance (or decrease the CPI), at some point, overaggressive issue of instructions without a commensurate increase in other resources (e.g., branch-predictor tables) can increase the executions of mis-speculated instructions, causing a slight performance decrease (i.e., increase in the CPI). Since our processor model contains numerous other speculative actions besides branch-prediction-based instruction processing, such anomalies in single-parameter sensitivity experiments are quite common in our microarchitectural studies. In sensitivity experiments, therefore, it often makes sense to vary several related parameters in synchrony, as illustrated in the following subsection.

### *Ganged sizing*

Out-of-order superscalar processors of the class considered rely on queues and buffers to efficiently decouple instruction execution in order to increase performance. The depth of the pipeline and the sizes of the resources required to support decoupled execution (queues, rename registers, completion table) combine to determine the performance of the machine. Because of this decoupled execution style, increasing the size of one resource without regard to the other resources may quickly create a performance bottleneck. Thus, in this section we consider the effects of varying multiple parameters rather than just a single one. **Figure 7** shows the effects of varying all of the major queue, buffer, and table sizes within the processor core. This includes issue queues, rename registers, branch-predictor tables, memory disambiguation hardware, and the completion table. For the buffers and queues, the number of entries in each resource is scaled by the values specified in the charts (0.6x, 0.8x, 1.2x, and 1.4x). For the instruction cache, data cache, and branch-prediction tables, the size of the structures is doubled or halved at each data point. From part (a), it can be seen that performance is increased by 5.5% for SPECfp, 9.6% for SPECint, and 11.2% for TPC-C as the size of the resources within the core is increased by 40% (except for

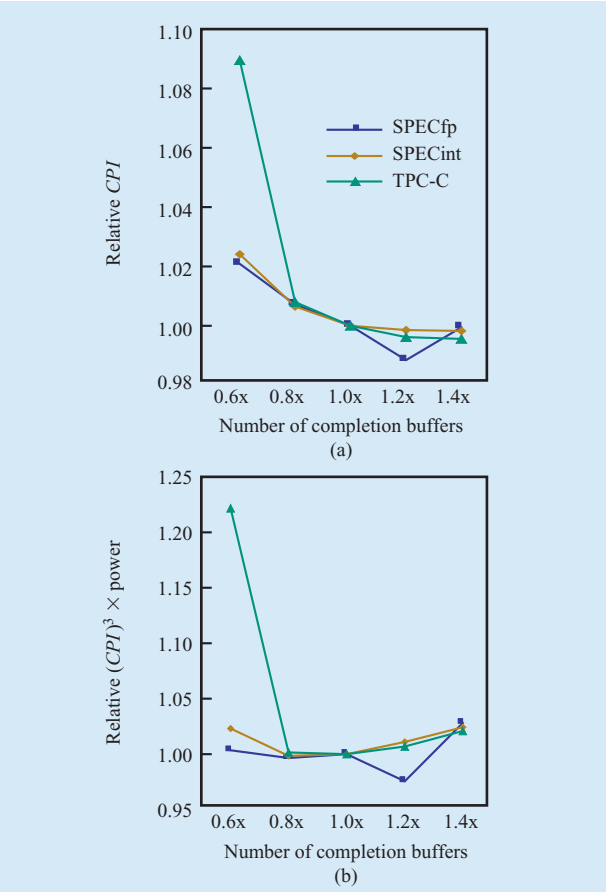


**Figure 6**

Variation of performance and power–performance with number of completion buffers.

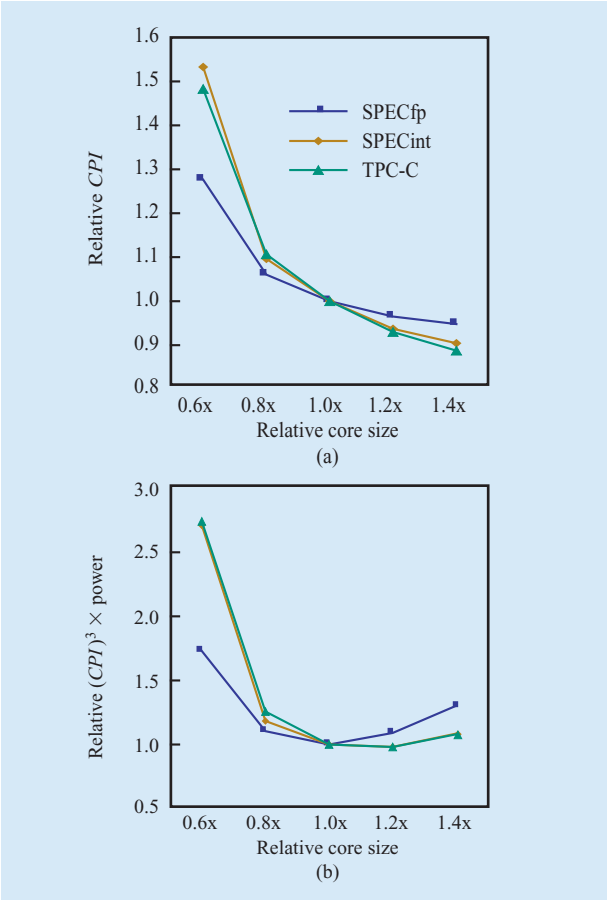


Figure 7

Variation of performance and power–performance with core size (ganged parameters).

the caches, which are 4x larger). The configuration had a power dissipation of 52–55% higher than the baseline core. Part (b) shows that the most power-efficient core microarchitecture is somewhere between the 1x and 1.2x core points.

### *Analysis of pipeline optimization for power and performance*

In this section we demonstrate the usefulness of PowerTimer to explore the issues involved in simultaneously optimizing superscalar pipelines for power and performance. The analyses presented represent an excerpt from [15]. The choice of pipeline depth is one of the fundamental issues confronting the architect/designer during the very-early-stage microarchitecture definition phase of future high-performance, power-efficient processors. Recent studies [16–18] seem to suggest that the possibility remains for further increases in pipeline depth, with performance optima in the range of eight to eleven FO4[5] inverter delays per stage (consisting of six to eight FO4 logic delays and two to three FO4 latch delays) for current out-of-order superscalar design paradigms. However, even in these performance-centric analysis studies, the authors do point out the practical difficulties of design complexity, verification, and power that must be solved in attaining these idealized limits. We now examine the practical, achievable limits when power dissipation constraints are also factored in. It is important that power dissipation be carefully minimized to avoid design points which promise ever-higher performance, yet under normal operating conditions, with commodity packaging and air cooling, deliver only a fraction of the theoretical peak performance.

For this study we have generated power models for all microarchitecture-level structures (subunits) modeled in Turandot, our research simulator [4]. PowerTimer uses microarchitectural activity information from the Turandot model to scale down the unconstrained hold and switching power on a per-cycle basis under a variety of clock-gating assumptions. In the study, we focus on a realistic form of clock gating which considers the applicability of such gating on a per-macro basis to scale down either the hold power or the combined hold and switching power, depending on the microarchitectural event counts. In order to quantify the power–performance efficiency of pipelines of a given FO4 depth, we have extended the PowerTimer methodology for scaling the power dissipation from the power models of our base FO4 design point across a range of FO4 depths.

The power data measured by PowerTimer (for a particular design point) can be expressed as $P_{\text{base}} = C V_{\text{dd}}^2 f(\alpha + \beta) CGF$, where $\alpha$ is the average "true" switching factor in circuits that represent transitions required for the functionality of the circuit, and is thus the switching factor measured by a register-transfer-level (RTL) simulator run in zero-delay mode. In contrast, $\beta$ is the average glitching factor that accounts for spurious transition in circuits due to race conditions. Thus, $\alpha + \beta$ is the total number of transitions actually seen inside circuits. Both $\alpha$ and $\beta$ are averaged over the whole processor over non-gated cycles with appropriate energy weights (the higher the capacitance at a particular node, the higher the corresponding energy weight). *CGF* is the clock-gating factor, defined as the fraction of cycles in which the microarchitectural structures are not clock-gated. The *CGF* is measured from our PowerTimer runs at each FO4 design point, as described above.

Next we analyze how each of these factors will scale with FO4 pipeline depth. We start by defining *FreqScale* as

[5] Fanout-of-four (FO4) delay is defined as the delay of one inverter driving four copies of an equally sized inverter. The amount of logic and latch overhead per pipeline stage is often measured in terms of FO4 delay, which implies that deeper pipelines have fewer FO4s per stage. Hereafter, the term FO4, when used alone, designates "FO4 inverter delays per pipeline stage."

the ratio $FO4_{base}/FO4$, the base FO4 depth divided by the actual FO4 depth (including latch insertion overheads). *FreqScale* is the scaling factor used to account for the clock frequency; it applies to both hold power and switching power.

With fixed logic hardware for given logic functions, the primary increase in chip-load capacitance is the large increase in latch and clock node capacitance. We next define *LatchScale* as

$$LatchScale = LatchRatio \times (FO4_{base}/FO4)^{LatchGrowthFactor}.$$

The term characterizes the hold power dissipation, but does not affect the switching power dissipation. *LatchRatio* is the ratio of the hold power to the switching power. *LatchGrowthFactor* (LGF) characterizes the rate at which the number of latches grows as a function of pipeline depth.

The amount of additional power that is spent in latches and clock in a deeply pipelined design is critically dependent on the logic shape functions of the structures that are being pipelined. Logic shape functions essentially describe the number of latches that would have to be inserted at any given point in a piece of combinatorial logic if it were to be pipelined. The term $(FO4_{base}/FO4)^{LatchGrowthFactor}$ recognizes that the latch and clock power increases as the pipeline depth increases. Logic shape functions being flat (which in many cases is not true) is one reason for the *LatchGrowthFactor* to be 1.0 and for the latch and clock power to change linearly with the pipeline depth. Larger values recognize the fact that for certain hardware structures the logic shape functions are not flat, so the amount of latches needed in the deeply pipelined design increases superlinearly.

The next two factors that must be considered for dynamic power dissipation when migrating to deeper pipelines are $\alpha$ and $\beta$, the chip-wide activity and glitching factors. The "true" switching factor $\alpha$ does not depend on the pipeline depth, since it is determined by the functionality of the circuits. The glitching factor at any net, on the other hand, is determined by the difference in delay of paths from the output of a latch that feeds the circuit to the gate that drives that net. Once a glitch is generated at some net, there is a high probability that this glitch will propagate down the circuit up to the input of the next latch down the pipeline. Furthermore, the greater the distance from the latch output to the inputs of a gate, the higher the probability of the existence of non-equal paths from the output of the latch to the inputs of the gate. Therefore, the average number of spurious transitions grows with the depth of the logic—the greater the FO4 depth, the higher the average glitching factor. Experimental data, collected by running a dynamic circuit-level simulator, PowerMill**, on post-layout-extracted netlists of sample functional units (built specifically for these experiments), shows that the average glitching factor can be modeled as being linearly dependent on the logic depth per pipeline stage, measured in terms of the number of FO4 delays, viz:

$$\beta = \beta_{base}(FO4/FO4_{base}).$$

To factor the effect of the dependence of the glitching factor on the pipeline depth, we introduce the following factor, which applies only to the switching power:

$$GlitchScale = [(1 - LatchRatio)/(1 + \beta_{base}/\alpha)] \times \{1 + [(\beta_{base}/\alpha)(FO4/FO4_{base})]\}.$$

In this formula, $\beta_{base}$ is the actual glitching factor averaged over the baseline microprocessor for the base FO4 design point. Note that $\beta_{base}$ appears in the formula only in the ratio $(\beta_{base}/\alpha)$, which makes a lot of sense, since the glitching factor is found to be roughly proportional to the "true" switching factor in the range from 0 to 0.3.

The following equation expresses the relationship between the power for the base-FO4 case and the scaled power for a new FO4 design point when considering the combination of all of the above factors:

$$P_{FO4} = CGF \times FreqScale \times (LatchScale + GlitchScale) \times P_{base}.$$

In general, CGF decreases with deeper pipelines because the amount of clock gating potential increases with deeper pipelines. The increased clock gating potential is primarily due to the increased number of cycles in which units are in stall conditions; this in turn leads to an increase in the clock gating potential on a per-cycle basis.

For this study we assumed that $LatchRatio = 0.8$, $LatchGrowthFactor = 1.0$, and $(\beta_{base}/\alpha) = 0$ (no glitching) in the PowerTimer. We used Turandot to model a generic, parameterized, out-of-order 8-issue, 5-wide superscalar processor (see Figure 3) with 32-KB instruction and data L1 caches and a 2-MB unified L2 cache. **Table 1** details the latency values for the 19-FO4 base design point of this study. We assume a latch overhead of two FO4 delays and one FO4 delay for the clock skew and jitter overhead. The 19 FO4 latency values are then scaled with the FO4 depth (after accounting for latch and clock skew overhead).

In this study, we report experimental results based on PowerPC traces of a set of 21 SPEC2000 benchmarks: *ammp*, *applu*, *apsi*, *art*, *bzip2*, *crafty*, *equake*, *facerec*, *gap*, *gcc*, *gzip*, *lucas*, *mcf*, *mesa*, *mgrid*, *perl*, *sixtrack*, *swim*, *twolf*, *vpr*, and *wupwise*. We have also used a 172-million instruction trace of a commercial application, TPC-C.

**Figure 8** shows the results for five metrics: *BIPS* (billions of instruction per second), *IPC* (instructions per cycle), *BIPS* per watt, $(BIPS)^2$ per watt, and $(BIPS)^3$ per watt. Optimizing the last three metrics corresponds to optimizing for energy, energy–delay product [19, 20], and $energy \times delay^2$ [11, 12]. We plot each metric relative to

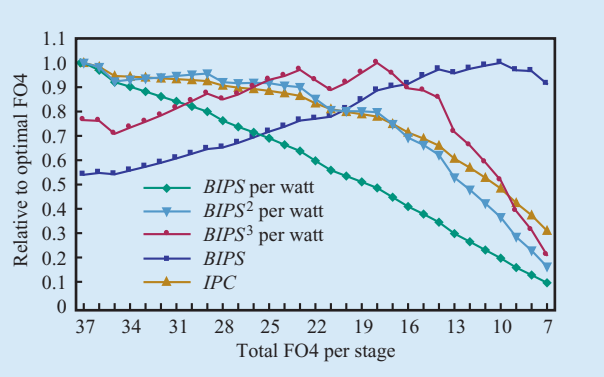


**Figure 8**

Optimal performance and power–performance points (SPEC2000).

the optimal FO4 design point for that metric. The figure shows that the optimal FO4 depth for performance (defined by *BIPS*) is 10 FO4, although pipelines of 8 FO4 to 15 FO4 are within 5% of the optimal. Because of the superlinear increase in power dissipation and sublinear increases in overall performance, the number of *BIPS* per watt always decreases with deeper pipelines. Only $(BIPS)^3$ per watt shows an optimum point that is not at the shallowest pipeline; for this set of experiments the optimal depth is 18 FO4. $(BIPS)^3$ per watt decreases sharply after the optimum; at the performance-optimal pipeline depth of 10 FO4, the $(BIPS)^3$ per watt metric is reduced by 50% over the 18-FO4 depth.

**Figure 9** presents similar results for our TPC-C trace. The optimal number of *BIPS* for TPC-C is relatively flat from 10 to 14 FO4 delays per pipeline stage. Using $(BIPS)^3$ per watt, the optimal pipeline depth shifts to 25–28 FO4 delays per pipeline stage, primarily because *BIPS* decreases less dramatically with shallower pipelines. Power also increases less dramatically for TPC-C with deeper pipes, because the additional amount of clock gating is more pronounced owing to large increases in the number of stall cycles relative to the SPEC2000 suite.

We have also performed a detailed sensitivity analysis of the optimal pipeline depth against key assumptions and design implementation choices [20]. Our analysis shows that there is a range of pipeline depth for which performance increases can be achieved at a modest sacrifice in power–performance efficiency. Pipelining beyond that range leads to a drastic reduction in power–performance efficiency, with little or no further performance improvement. **Figure 10** shows a summary view of the power–performance optimality characteristics for various workload classes.

In this section we have demonstrated the importance of considering power and performance in unison when optimizing the pipeline structure. On the basis of the combination of power and performance modeling performed using PowerTimer, we have shown that a purely performance-driven, power-unaware design may lead to the selection of an overly deep pipelined microprocessor design operating at an inherently power-inefficient design point.

**Table 1** Latencies for 19-FO4 design point.

| *Fetch latencies* | | *Decode latencies* | |
|---|---|---|---|
| *Latency parameters* | *Latency in cycles* | *Latency parameters* | *Latency in cycles* |
| NFA predictor | 1 | Multiple decode | 2 |
| L2 I-cache | 11 | Millicode decode | 2 |
| L3 cache (infinite) | 85 | Expand string | 2 |
| I-TLB miss | 10 | Mispredict cycles | 3 |
| L2 I-TLB miss | 50 | Register read | 1 |
| *Execution pipe latencies* | | *Load/store latencies* | |
| *Latency parameters* | *Latency in cycles* | *Latency parameters* | *Latency in cycles* |
| Fix execute | 1 | L1 D-load | 3 |
| Float execute | 4 | L2 D-load | 9 |
| Branch execute | 1 | L3 (data) | 77 |
| Float divide | 12 | Load float | 2 |
| Integer multiply | 7 | D-TLB miss | 7 |
| Integer divide | 35 | L2 D-TLB miss | 50 |
| Retire delay | 2 | StoreQ forward | 4 |

*Legend*: TLB = translation lookaside buffer. I and D qualifiers stand for "instruction" and "data," respectively. L1, L2, and L3 stand for level-1, level-2, and level-3, respectively.

### *Other uses of PowerTimer in early-stage definition and design*

In this subsection, we illustrate a few other uses of the PowerTimer/RMAP methodology. The following classes of data generated by this toolset have been used by various design teams within IBM:

- *Unit-level utilization data for a given benchmark run*. This data is normally available from performance simulators. It gives the design team an initial indicator of the regions of the chip that are heavily utilized and are therefore likely to be "hot" from a power or power-density viewpoint. In addition, the data identifies opportunities for clock gating in regions or units that are sparsely utilized.
- *Gated-mode power savings for a given run*. This data gives the design team a first-cut estimate of the expected power savings across units and the whole chip under various gating conditions (e.g., under coarse- or fine-grain clock gating or supply-voltage gating). Results based on PowerTimer have shown that the saving in average power dissipation can be expected to be more than 50% of the unconstrained (maximum) power for the vast majority of workloads executing on current-generation superscalar processors.
- *Power-swing (di/dt) characteristics*. Since power consumption can be tracked on a cycle-by-cycle and unit-by-unit basis, it is possible to measure the maximum and average power swing with respect to a given reference power level. This data is useful for determining possible large *di*/*dt* problems resulting from clock gating for both general workloads and specially architected test kernels. Such a study is also useful for assessing the nature and extent of the need to insert decoupling capacitances for control of inductive ($L di/dt$) noise on the power-supply rails.

Because of space limitations, we do not include representative data and analysis to cover the above aspects of the capability and applicability of PowerTimer. However, we discuss below the various modes of clock gating supported by PowerTimer for accurate estimation of workload-dependent power savings under these modes. Since a large fraction of power is dissipated in latches and clocking, deviations in benchmark-to-benchmark power dissipation in high-performance processors is strongly correlated with the frequency of clock-gating events. As comprehensive clock-gating methodologies become a pervasive part of nearly all types of microprocessors designed, one of the most critical parts of the PowerTimer simulation infrastructure is making sure that realistic and accurate clock-gating information is used with the energy models. A common misperception with power–performance simulators is that the accuracy is

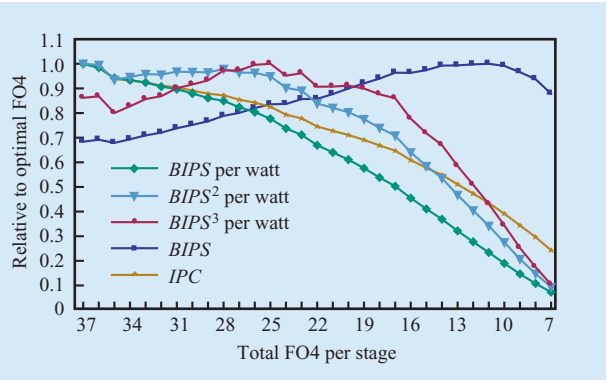


**Figure 9**

Optimal performance and power–performance points (TPC-C).

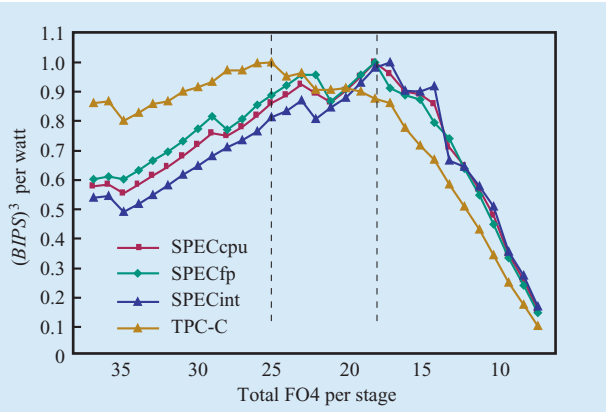


**Figure 10**

Power–performance variation with pipeline depth.

primarily a function of the energy models used. In fact, accurate clock-gating events play an equal role in ensuring realistic power estimates for processors. In some cases, providing early-stage estimates of clock-gating utilization can be just as challenging as developing energy models, because in early-stage modeling it is difficult to know exactly what structures can be clock-gated and under what conditions.

In PowerTimer, clock-gating information is provided entirely by the accompanying microprocessor performance simulator in terms of utilization data for all of the microarchitectural structures. To estimate the power dissipation in the presence of clock gating, it is important to know a) how the power of the structure/unit will react to the clock-gating condition and b) what microarchitectural event or utilization condition should be used for the clock- gating condition. Recall from Section 3 that our

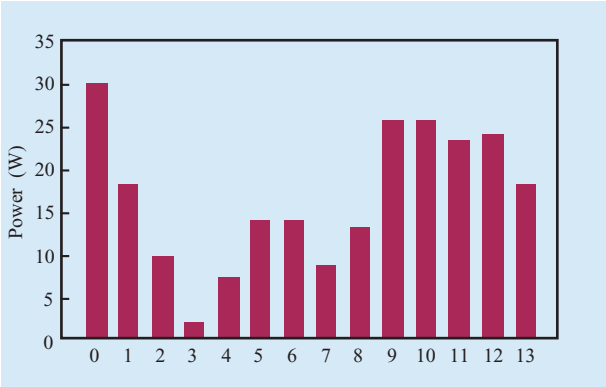


Figure 11

Average power in various clock-gating modes.

power models include two pieces of data for each microarchitectural structure: the unconstrained power due to clocking (local clock buffers and latches) and the unconstrained power due to logic switching (combinatorial logic) scaled to a realistic switching factor. PowerTimer assumes that a structure can have the following four possible power states, depending on the clock-gating conditions.

- $P\_constrained_0$: The structure cannot be clock-gated, either because there is no clock-gating condition that the performance simulator can generate or because the designers determine that the structure cannot be clock-gated.
- $P\_constrained_1$: Both clock and logic power of the structure can be clock-gated in proportion to the frequency of the clock-gating event.
- $P\_constrained_2$: Only the clock power of the structure can be gated.
- $P\_constrained_3$: Only the logic power of the structure can be gated.

The next step is to determine the clock-gating conditions. Since all of our microarchitectural structures can be classified as either a buffer, queue, pipeline stage (with accompanying logic), or array structure, the following four classes of events are sufficient to cover all of the gating conditions:

- $AF_1$: The number of valid entries in a buffer, queue, or pipeline stage.
- $AF_2$: The number of valid entries in a buffer, queue, or pipeline stages minus the number of stalled entries.
- $AF_3$: The number of writes to a buffer, queue, or array structure.
- $AF_4$: The number of writes and reads to a buffer, queue, or array structure.

We combine the power states ($P\_constrained_1$, $P\_constrained_2$, $P\_constrained_3$) with the four gating conditions above to arrive at all 12 possible clock-gating modes ($Mode_1$ through $Mode_{12}$). For example, $Mode_1$ is power state $P\_constrained_1$ combined with the gating condition $AF_1$. In addition, we denote $P\_constrained_0$ (with no clock gating) as $Mode_0$. It is important to note that not all of the clock-gating modes are applicable to all of the structures. Therefore, we derive $Mode_{13}$ to denote "realistic clock gating" as determined by the user by applying one of the 12 clock-gating modes to each individual structure. However, for illustrative purposes, **Figure 11** shows the variation in simulated power (averaged over the SPEC2000 trace suite) as a function of the particular mode of clock gating chosen. These experiments were done using our baseline Turandot-based research PowerTimer model. Although not all of the modes are realistic, these results show a rather large range of variation in average power, depending on the aggressiveness of the clock-gating support assumed.

## 5. Related prior work

From the viewpoint of functional usage, PowerTimer is similar to prior academic research tools such as Wattch [21, 22] and SimplePower [23], or tools developed in other companies, such as the research tool TEM$^2$P$^2$EST [24] or the ALPS simulator [25] used in the Intel** Pentium** 4 design. However, each of these microarchitecture-level power simulation tools uses a different methodology for generating the energy models; and, of course, the base performance simulation models that are in use with the energy models are also different in terms of the architecture (ISA), the family of microarchitectures, and the level of detail.

Wattch is the most widely used research power–performance simulator within the academic community. The base performance simulator in Wattch is the well-known SimpleScalar toolset [26]. The energy models derived for use in Wattch, broadly speaking, comprise two categories: a) analytical equations, modeling specific circuit classes—e.g., SRAM (static random-access memory) and CAM (content-addressable memory) structures, clocking networks, and issue queues; b) technologically scaled power numbers derived from published values for irregular logic structures that are hard to analyze—e.g., functional unit (ALU) blocks. The analytical equations are controlled by technology-specific per-unit-length capacitance values for metal, gate, and diffusion layers and structural geometry parameters of the modeled entity.

As in the case of Wattch, the TEMPEST tool uses SimpleScalar as the base performance model. Again, the energy models are either analytical equations (for regular structures) or empirically derived from available data. The empirically derived models are based on power density and area estimates.

SimplePower uses state-transition-based energy usage models to capture detailed switching-dependent power variations in given logic blocks. Thus, for a given ALU macro, detailed circuit simulation-based energy data is gleaned for many (and ideally all) possible cycle-to-cycle transition patterns on the input pins. These energy values are stored in a lookup table, which is accessed to compute state transition-specific energy usage during execution-driven performance simulation. In this methodology, there is an issue concerning the very large sizes of energy lookup tables; however, the authors of [23] have developed methods for reducing the table sizes through collapsing of equivalent state transition entries.

The Intel architecture-level power simulator (ALPS) is built around a proprietary, Pentium-4-specific microarchitectural simulator. The energy models for microarchitectural blocks are derived from empirical energy data available from prior processors.

PowerTimer, in contrast, uses a phased (hierarchical) suite of energy functions (RMAP) that are refined as the design and simulation model evolves. In the very early stages (concept phase), simple latch-count-based models are used for the logic; and per-port, per-access estimated power numbers are used for array and register file macros. As the design progresses, analytical equations derived from empirical power data for primitive building blocks are used for newer circuit structures within the target structure. When detailed circuit schematics are available, accurate circuit- simulation-based (CPAM) energy data is collected for each macro and abstracted for use in forming energy models for microarchitectural blocks. On occasion, prior-generation CPAM data is also used (after scaling) to form energy models for units that have not yet been characterized in the new design environment.

## 6. Model validation and other future work

The use of early-stage, workload-driven power analysis tools (such as PowerTimer) is relatively new in industrial application, and a number of issues remain open for further research and development. One of the obvious issues is that of model validation. Since current-generation processors are not instrumented to monitor power consumption on a unit-by-unit basis, it is not yet possible to directly validate model-predicted power profiles against measured ones. Pre-silicon validation of function is a mature field (e.g., [27, 28]); testing, calibration, and validation of pre-silicon performance models have been addressed in recent work (e.g., [5, 29]), but it is still an evolving field. The Wattch project at Princeton [21, 22] addresses the issue of validating power models by comparing analytically modeled capacitances against post-layout-extracted ones. Also, the issue of absolute versus relative accuracy in models that use abstractions in selected power or performance submodels within a simulator has been addressed in recent work [22, 30]. We are currently pursuing further research in establishing pre-silicon reference models on expected power bounds for given architectural test cases. This is an extension of prior work that was limited to performance (CPI) bounds [5, 31]. This research should lead us into formalizing a systematic, parameter-driven test case generation methodology—one that should allow a design team to generate a suite of test cases with *a priori* bounds on cycle-count (or CPI), unit utilizations, and power.

The basic idea in this approach is to statically estimate the bounds on CPI and unit-level average utilizations for given architectural test cases. Lower bounds on CPI for loop-oriented test cases can be analytically estimated by assuming infinite queue and buffer sizes and then deducing the performance-limiting bandwidth parameter for the given test case [31]. For example, if a loop is provably memory-bound, a lower bound on cycles per loop iteration is obtained by dividing the total number of load and store instructions in the loop by the number of load-store units, or by the number of cache ports, whichever is smaller. Our power–performance model validation procedure goes one step beyond this by deducing the various queue, cache, and pipeline stage utilization bounds using analytical models. Since unit utilizations can be transformed to power numbers through suitable weighting functions, we effectively have a method of deducing power and performance bounds for given loop test cases. If measured (simulated) power and performance, under appropriate parametric settings of the model, violate the analytically derived bounds, we detect a model defect. We are currently engaged in completing this work, which will be reported separately in a forthcoming publication.

In carrying out this validation work, our emphasis, as stated before, is on *relative* accuracy. We believe that the primary goal of microarchitecture-level models such as PowerTimer is to guide the early-stage-design architects so that the fundamental microarchitectural design parameters, such as pipeline depth, instruction issue width, cache sizes, queue sizes, and basic latency and bandwidth parameters, are chosen correctly from the viewpoint of power–performance balance. In our current pipeline-level energy scaling models (as described in the section on analysis of pipeline optimization for power and performance), we do not consider the circuit-level power–performance optimization choices available to the designer on a per-stage basis. In future work, we may

incorporate such additional design space exploration choices (e.g., see the paper by Zyuban and Strenski in this issue [13]) within the PowerTimer framework.

In other work, we are currently extending the definition of our power models into a hierarchical, modular library format. When completed, this power function library will allow us to construct and use energy models in various stages of design (e.g., in very early stages, we can use simple latch- or area-based energy models and analytical formulations), while in later stages we can build more detailed models based on parameterized, circuit-level characterizations of primitive building blocks targeted for use in a given design.

The base Turandot model is currently being significantly overhauled and updated in order to make it much more user-friendly and modular, with the addition of microarchitectural features such as simultaneous multithreading (SMT) and support for various types of "in-order" issue mechanisms. Correspondingly, the associated power models are also being augmented. We plan to report the updated and fully validated PowerTimer-II modeling and analysis results in future publications.

## 7. Summary

We have described the PowerTimer toolset that is currently in use to facilitate early-stage power–performance analysis and microarchitecture definition of high-end, general-purpose IBM PowerPC processors. PowerTimer uses a variety of sources for power models, such as output from a circuit-level power analysis and extraction tool or analytical models derived in a bottom-up modeling methodology.

Early-stage power–performance modeling is now a critical aspect of future processor design. Future processor cores continue to have aggressive performance goals, but they are now facing tight power budgets and power-density limits. In previous processor generations, such as the IBM POWER3* [6] and POWER4* [7], early-stage definition studies did not require a systematic analysis of power because associated estimated increases in power dissipation were still comfortably below the limits dictated by the packaging/cooling solutions appropriate for high-end PowerPC workstations and servers.

## Acknowledgments

The authors are grateful to several graduate student interns who have assisted in characterizing circuit structures to help formulate or verify architecture-level power sensitivities for specialized components such as queues, multiplexors, latches, and local clock buffers. Special mention must be made of Koushik Das, Alper Buyuktosunoglu, and Tejas Karkhanis in this regard. We gratefully acknowledge the help received from numerous colleagues at the IBM Thomas J. Watson Research Center, including Victor Zyuban, Philip Strenski, Stanley Schuster, and Peter Cook, in strengthening our understanding of circuit-level power modeling and tradeoff issues. Zhigang Hu, a colleague who joined our team after the initial submission of this paper, is currently engaged in upgrading the base Turandot performance model to incorporate many new features. The authors are grateful to Zhigang for his help in the ongoing development of the next-generation PowerTimer-II toolset, which will support multithreading, among many other new features.



## References

1. M. K. Gowan, L. L. Biro, and D. B. Jackson, "Power Considerations in the Design of the Alpha 21264 Microprocessor," *Proceedings of the IEEE/ACM Design Automation Conference*, 1998, pp. 726–731.
2. D. Brooks, J.-D. Wellman, P. Bose, and M. Martonosi, "Power–Performance Modeling and Tradeoff Analysis for a High-End Microprocessor," presented at the Workshop on Power-Aware Computer Systems (PACS'00, held in conjunction with ASPLOS-IX), Cambridge, MA, November 2000; also appeared as *Lecture Notes on Computer Science* (*LCNS*), Vol. 2008, 2001, pp. 126–136.
3. J. S. Neely, H. H. Chen, S. G. Walker, J. Venuto, and T. J. Bucelot, "CPAM: A Common Power Analysis Methodology for High Performance Design," *Proceedings of the 9th Topical Meeting on Electrical Performance of Electronic Packaging*, Scottsdale, AZ, October 2000, pp. 303–306.
4. M. Moudgill, J.-D. Wellman, and J. Moreno, "Environment for PowerPC Microarchitecture Exploration," *IEEE Micro* **19,** No. 3, 15–25 (May/June 1999); see also *http://www.research.ibm.com/MET/*.
5. M. Moudgill, P. Bose, and J. Moreno, "Validation of Turandot, a Fast Processor Model for Microarchitecture Exploration," *Proceedings of the IEEE International Performance, Computing and Communication Conference*, February 1999, pp. 451–457.
6. F. P. O'Connell and S. W. White, "POWER3: The Next Generation of PowerPC Processors," *IBM J. Res. & Dev.* **44,** No. 6, 873–884 (November 2000).
7. J. M. Tendler, J. S. Dodson, J. S. Fields, Jr., H. Le, and B. Sinharoy, "POWER4 System Microarchitecture," *IBM J. Res. & Dev.* **46,** No. 1, 1–116 (January 2002).
8. S. Borkar, "Design Challenges of Technology Scaling," *IEEE Micro* **19,** No. 4, 23–29 (July/August 1999).
9. Standard Performance Evaluation Corporation (SPEC), Warrentown, VA; see *http://www.spec.org/* for details.
10. Benchmark C of the Transaction Processing Recording Council (TPC), San Francisco, CA; see *http://www.tpc.org/* for details.
11. D. M. Brooks, P. Bose, S. E. Schuster, H. Jacobson, P. N. Kudva, A. Buyuktosunoglu, J.-D. Wellman, V. Zyuban, M. Gupta, and P. W. Cook, "Power-Aware Microarchitectures: Design and Modeling Challenges for Next-Generation Microprocessors," *IEEE Micro* **20,** No. 6, 26–44 (November 2000).

12. V. Zyuban and P. Strenski, "Unified Methodology for Resolving Power–Performance Tradeoffs of the Microarchitectural and Circuit Levels," *Proceedings of the International Symposium on Low-Power Electronics and Design*, August 2002, pp. 166–171.
13. V. Zyuban and P. N. Strenski, "Balancing Hardware Intensity in Microprocessor Pipelines," *IBM J. Res. & Dev.* **47,** No. 5/6, 585–598 (this issue, September/November 2003).
14. U. Ko, P. T. Balsara, and A. K. Nanda, "Energy Optimization of Multilevel Cache Architectures for RISC and CISC Processors," *IEEE Trans. VLSI Syst.* **6,** No. 2, 299–308 (June 1998).
15. V. Srinivasan, D. Brooks, M. Gschwind, P. Bose, P. Emma, V. Zyuban, and P. Strenski, "Optimizing Pipelines for Power and Performance," *Proceedings of the International Symposium on Microarchitecture* (*MICRO-35*), November 2002, pp. 333–344.
16. A. Hartstein and T. R. Puzak, "The Optimum Pipeline Depth for a Microprocessor," *Proceedings of the 29th International Symposium on Computer Architecture* (*ISCA-29*), May 2002, pp. 7–13.
17. M. S. Hrishikesh, D. Burger, N. P. Jouppi, S. W. Keckler, K. I. Farkas, and P. Shivakumar, "The Optimal Logic Depth per Pipeline Stage Is 6 to 8 FO4 Inverter Delays," *Proceedings of the 29th International Symposium on Computer Architecture* (*ISCA-29*), May 2002, pp. 14–24.
18. E. Sprangle and D. Carmean, "Increasing Processor Performance by Implementing Deeper Pipelines," *Proceedings of the 29th International Symposium on Computer Architecture* (*ISCA-29*), May 2002, pp. 25–34.
19. R. Gonzalez and M. Horowitz, "Energy Dissipation in General Purpose Microprocessors," *IEEE J. Solid-State Circuits* **31,** No. 9, 1277–1284 (1996).
20. T. Conte, K. Menezes, and S. Sathaye, "A Technique to Determine Power-Efficient, High Performance Super Scalar Processors," *Proceedings of the 28th Hawaii International Conference on System Science*, January 1995, Vol. 1, pp. 324–333.
21. D. Brooks, V. Tiwari, and M. Martonosi, "Wattch: A Framework for Architectural Level Power Analysis and Optimizations," *Proceedings of the International Symposium on Computer Architecture* (*ISCA*), June 2000, pp. 83–94.
22. D. Brooks, "Design and Modeling of Power-Efficient Computer Architectures," Ph.D. dissertation, Princeton University, November 2001.
23. N. Vijaykrishnan, M. Kandemir, M. J. Irwin, H. S. Kim, and W. Ye, "Energy-Driven Integrated Hardware–Software Optimizations Using SimplePower," *Proceedings of the International Symposium on Computer Architecture* (*ISCA*), June 2000, pp. 95–106.
24. A. Dhodapkar, C. Lim, G. Cai, and W. Daasch, "TEM$^2$P$^2$EST: A Thermal Enabled Multi-Model Power/Performance ESTimator," *Digest of Technical Papers*, *Workshop on Power-Aware Computer Systems* (*PACS'00*), Cambridge, MA, November 2000; also appeared as *Lecture Notes on Computer Science* (*LCNS*), Vol. 2008, 2001, pp. 112–125.
25. S. H. Gunther, F. Binns, D. Carmean, and J. C. Hall, "Managing the Impact of Increasing Microprocessor Power Consumption," *Intel Technol. J.,* Q1, 2001; see *http://www.intel.com/technology/itj/q12001/articles/art_4.htm/*.
26. D. Burger and T. Austin, "The SimpleScalar Tool Set, Version 2.0," *Technical Report TR-1342*, University of Wisconsin, June 1997.
27. A. Aharon, A. Bar-David, B. Dorfman, E. Gofman, M. Leibowitz, and V. Schwartzburd, "Verification of the IBM RISC System/6000 by a Dynamic Biased Pseudo-Random Test Program Generator," *IBM Syst. J.* **30,** No. 4, 527–537 (April 1991).
28. M. Kantrowitz and L. M. Noack, "Functional Verification of a Multiple Issue, Pipelined, Super Scalar Alpha Processor—the Alpha 21164 CPU Chip," *Proc. Digital Tech. J.* **7,** No. 1, 136–144 (1995).
29. B. Black and J. Shen, "Calibration of Microprocessor Performance Models," *IEEE Computer* **31,** No. 5, 59–65 (May 1998).
30. D. Brooks, M. Martonosi, and P. Bose, "Abstraction via Separable Components: An Empirical Study of Absolute and Relative Accuracy in Processor Performance Modeling," *Research Report RC-21909*, IBM Thomas J. Watson Research Center, Yorktown Heights, NY, December 2000.
31. P. Bose, "Testing for Function and Performance: Towards an Integrated Processor Validation Methodology," *J. Electron. Testing: Theory & Appl.* **16,** 29–48 (2000).

*Received November 10, 2002; accepted for publication July 3, 2003*

**David Brooks** *Division of Engineering and Applied Sciences, Harvard University, Cambridge, Massachusetts 02138 (dbrooks@eecs.harvard.edu).* Dr. Brooks is an Assistant Professor of Computer Science at Harvard University. He received a B.S. degree from the University of Southern California in 1997, and M.A. and Ph.D. degrees from Princeton University in 1999 and 2001, respectively, all in electrical engineering. While this paper was being written, Dr. Brooks was a Research Staff Member at the IBM Thomas J. Watson Research Center in Yorktown Heights, New York. His research interests include architectural-level power-modeling and power-efficient design of hardware and software for embedded and high-performance computer systems.

**Pradip Bose** *IBM Research Division, Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598 (pbose@us.ibm.com).* Dr. Bose is a Research Staff Member at the Thomas J. Watson Research Center. He currently leads the power-aware microarchitecture project, focusing on the development of methodologies for early-stage power–performance tradeoff analysis and design of power-efficient microprocessors. His current research interests include computer architecture, power–performance modeling and validation, and design automation. Dr. Bose received M.S. and Ph.D. degrees in electrical and computer engineering from the University of Illinois at Urbana–Champaign, and a B.Tech. (Honors) degree in electronics and electrical communication engineering from the Indian Institute of Technology, Kharagpur. At IBM, Dr. Bose has been involved in the earliest research efforts on superscalar RISC machines, which led to the RS/6000 family of workstations developed by IBM in Austin, Texas. He has worked closely with virtually all of the high-end PowerPC development projects since then. Dr. Bose has been active in numerous leading conference committees; he is a Senior Member of the IEEE and Editor-in-Chief of the *IEEE Micro* magazine.

**Vijayalakshmi Srinivasan** *IBM Research Division, Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598 (viji@us.ibm.com).* Dr. Srinivasan joined the IBM Thomas J. Watson Research Center in 2001 as a Research Staff Member. She received a B.S. degree in physics from the University of Madras in 1990, an M.S. degree in computer science and engineering from the Indian Institute of Science in 1994, and a Ph.D. in computer science and engineering from the University of Michigan in 2001. Her research areas include computer architecture and performance analysis. Dr. Srinivasan is currently part of a power-aware microsystems project focusing on developing high-level energy models for various microarchitecture structures that can be used with architecture-level machine simulators to evaluate power–performance tradeoffs.

**Michael K. Gschwind** *IBM Research Division, Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598 (mkg@us.ibm.com).* Dr. Gschwind is a Research Staff Member at the Thomas J. Watson Research Center. At IBM, he has contributed to several generations of binary translation architectures exploiting instruction-level parallelism, the evaluation of future microarchitecture options for current architectures. He was one of the originators of the supercomputer-on-a-chip "Cell" next-generation high-performance system architecture, which is currently being developed in a joint development center in Austin, Texas, by the IBM, Sony, and Toshiba corporations. Dr. Gschwind's current research is focused on the power and performance of high-frequency, low-power, high-performance architectures for media, high-performance, and general-purpose computing applications. Before joining IBM in 1997, he was a faculty member at the Department of Computer Engineering, Technische Universität Wien, Vienna, Austria. Dr. Gschwind received M.S. and Ph.D. degrees in computer science from Technische Universität Wien in 1991 and 1996, respectively. His research interests include compilers, computer architecture, instruction-level parallelism, hardware/software codesign, application-specific processors, and field-programmable gate arrays. He is the author of more than 60 papers, holds several patents on high-performance computer architecture, and has received eight IBM Invention Plateau Awards and several awards for his technical contributions. Dr. Gschwind is a Senior Member of the IEEE and the IEEE Computer Society, and a member of Phi Kappa Phi and the Fulbright Alumni Association.

**Philip G. Emma** *IBM Research Division, Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598 (pemma@us.ibm.com).* Dr. Emma received B.S., M.S., and Ph.D. degrees in electrical engineering from the University of Illinois, joining IBM at the Thomas J. Watson Research Center in 1983. He has worked in the areas of systems, architecture, microarchitecture, circuit design, packaging, and interconnect technology. He holds more than 80 patents in these areas and is an IBM Master Inventor. Dr. Emma currently manages the Systems Technology and Microarchitecture Department at the Thomas J. Watson Research Center; he is a member of the IBM Academy of Technology and a Fellow of the IEEE.

**Michael G. Rosenfield** *IBM Research Division, Austin Research Laboratory, 11501 Burnet Road, Austin, Texas 78758 (mgrosen@us.ibm.com).* Dr. Rosenfield is currently Director of the Austin Research Laboratory, focusing on high-performance VLSI design and tools, system-level power analysis, and new system architectures. He was previously Senior Manager of VLSI Design and Architecture at the IBM Thomas J. Watson Research Center in Yorktown Heights, New York, where he and his team were involved in high-performance microprocessor VLSI design for the IBM Server Group and the IBM Microelectronics Division. Their work pertained to tools, methodologies, and commonality as well as power-aware microarchitecture, circuits/technology codesign, performance analysis, exploratory microarchitectures, and advanced compiler design. Previously, Dr. Rosenfield held management positions at the Research Division in parallel communication architecture and in advanced lithography. In 1993, he was the technical assistant to the Research Vice President of Systems, Technology, and Science. Dr. Rosenfield received Ph.D. and M.S. degrees from the University of California at Berkeley and a B.S. degree in physics from the University of Vermont.