Power-Performance Modeling and Tradeoff Analysis for a High End Microprocessor

David Brooks and Margaret Martonosi
Department of Electrical Engineering
Princeton University
dbrooks,mrm@ee.princeton.edu

John-David Wellman and Pradip Bose
IBM T.J. Watson Research Center
Yorktown Heights, NY
pbose@us.ibm.com

Abstract

We describe a new power-performance modeling toolkit, developed to aid in the evaluation and definition of future power-efficient, PowerPC™ processors. The base performance models in use in this project are: (a) a fast but cycle-accurate, parameterized research simulator and (b) a slower, pre-RTL reference model that models a specific high-end machine in full, latch-accurate detail. Energy characterizations are derived from real, circuit-level power simulation data. These are then combined to form higher-level energy models that are driven by microarchitecture-level parameters of interest. The overall methodology allows us to conduct power-performance tradeoff studies in defining the follow-on design points within a given product family. We present a few experimental results to illustrate the kinds of tradeoffs one can study using this tool.

1 Introduction

Power dissipation limits constitute a key new constraint in the design of high-end microprocessors. These limits are becoming important for two main reasons:

Chip-level cost/performance evolution:

At the chip-level, there has been an ever-increasing growth in complexity, clock frequency and die size. This is driven by advances in semiconductor technology and the quest to keep up with Moore's Law from a performance viewpoint. Power consumed by the processor must be dissipated by the use of appropriate packaging and cooling solutions. These latter solutions get more sophisticated and costly as power increases. As single-threaded uniprocessor IPC performance slowed due to fundamental ILP limits, the added complexities needed to meet net performance growth targets caused power (and cost) budget overruns. The impact on verification cost and time-to-market also started to affect the actual cost-performance trends. Multithreading and multiprocessing at the chip-level have evolved in an attempt to correct the cost-performance imbalance. That is, through architectural paradigm shifts, the hope is to boost the net performance significantly, but with relatively small power, complexity and die-size increments. However, packaging technology limits within the "air-cooling" regime, have forced designers to look for power-saving opportunities at all levels of high-end design, irrespective of the microarchitectural paradigm adopted.

System-level, volumetric cost/performance evolution:

At the very high-end, one can argue that the system cost (with all the "box", memory and peripheral chip/board and switch costs) is so much higher than a single processor chip cost that worrying about power/cost issues at the single chip level is irrelevant. However, this is not a correct perspective. Within the form factor limits of a server "box", if one is forced to increase the volume occupied by cooling hardware (e.g. through the use of refrigeration or liquid cooling), then that takes away from the number of processors which could have been included. Also, there are some basic upper limits on the amount of current drawn by a server box to meet the overall processing and cooling needs. Thus, the volumetric cost/performance growth constraints translate into a "per-chip" power limit for a particular system product targeted for a given market.

Thus, early-stage microarchitecture definition and trade-off analysis studies must now (more than ever

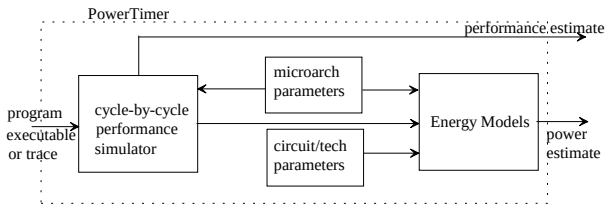

Figure 1. Block Diagram of PowerTimer.

before) try to factor in considerations of power and design complexity. Recently, there have been papers from academia and industry [1, 3, 10, 11, 12, 13] that address the issue of modeling and design of power- and complexity-aware microarchitectures. In this paper, we report ongoing work in this area within IBM Research. The power-performance modeling methodology described in the prior work of Brooks et al. [1] is adapted for use within the modeling framework of a real, server-class processor development project. The key new contributions in this power-performance modeling tool are:

- Energy models that are derived from real, circuit-level power simulation data, but are then driven by microarchitecture-level parameters of interest. These higher-level abstractions are suitable for conducting power-performance tradeoff studies to define the follow-on design points within a given product family. Technology parameters and scaling equations are additional inputs to the model.
- A web-based graphical user interface, which allows one to quickly characterize the fundamental tradeoffs between performance growth and power-related cost, based on prior, one-time simulation data collected in a spreadsheet database.

Using this new modeling toolkit, we evaluate a current generation, high-end PowerPC processor design point from the viewpoint of power-performance efficiency. As part of this evaluation, we examine the sensitivity of such efficiency metrics with respect to individual (and combinations of) microarchitecture-level parameters: cache size and geometry parameters, queue/buffer sizes, number of ports to various storage resources, various other bandwidth parameters, etc.

2 *PowerTimer*: an Energy-Aware Performance Simulation Toolkit

Figure 1 shows the high-level block diagram of PowerTimer, our energy-model-enabled performance simulator. The basic methodology is similar to earlier models, like Wattch [1].

The energy models are derived from circuit-level power simulation data, collected on a detailed, macro-by-macro basis. These models are controlled by two sets of parameters: (a) technology/circuit parameters, which allow appropriate scaling from one CMOS generation to the next; and (b) microarchitecture-level parameters: various queue/buffer sizes, pipe latencies and bandwidth values. These latter parameters also drive the base performance simulator in the usual manner. The energy models can be used in two different modes. First, the performance simulator can be used standalone, to produce detailed CPI and resource utilization statistics. These can then be processed through the energy models to generate average, unit-wise power numbers. Second, the energy models can be embedded in the actual simulation code, so that they are "looked up" as needed on a cycle-by-cycle basis. This mode allows one to view the cycle-by-cycle energy characteristics in more detail; but the average statistics at the end of the run would obviously be the same as in the first mode.

2.1 Energy Model Construction

In the Wattch simulator [1], and in other similar toolkits [12, 13], analytical capacitance models were developed for various high-level block-types, such as RAMs, CAMs and other array structures, latches, buses, caches, and ALUs. While some of the characterizing parameters are gross length and width values which a logic-level designer or microarchitect could relate to, others were at a much lower (circuit or physical design) level. In our current (PowerPC-specific) work, the goal is to form unit-specific energy models controlled by parameters familiar to a high-level designer or microarchitect. Thus, for example, once a characterizing equation has been formed for one of the issue queues, one is able to play "what-if" games in PowerTimer, by simply varying the queue size as normally done in microarchitectural performance simulation.

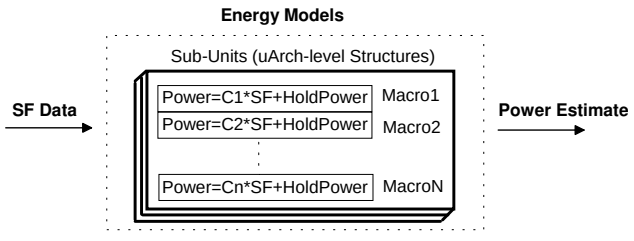

Figure 2. PowerTimer Energy Models.

Figure 2 below 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 in a current, high performance PowerPC processor. The power analysis has been performed at the macro level; generally, multiple macros combine to form one micro-architectural level structure which we will call a sub-unit. For example, the fixed-point issue queue (one sub-unit) might contain separate macros for storage memory, comparison logic, and control. Power analysis has been performed on each macro to determine the macro's power as a function of the input switching factor. The *hold power*, or power when no switching is occurring, is also generated. These two pieces of data allow us to form simple linear equations for each macro's power. The energy model for a sub-unit is determined by summing the linear equations for each macro within that sub-unit. We have generated these power models for all microarchitecture-level structures (sub-units) modeled in our research simulator [8, 9]. In addition to the models that specify the power characteristics for particular sub-units (such as the fixed-point issue queue), we can derive power models for more generalized structures; for example, a generalized issue queue model. These generalized models are useful for estimating the power cost of additions to the baseline microarchitecture. The generalized model is derived by analyzing the power characteristics of structures within the baseline microarchitecture. For example, the fixed-point, floating-point, logical-op, and branch-op queues have very similar functionality and power characteristics and the energy analysis for these queue structures has been used to derive a generalized issue-queue power model based on parameters such as the number of entries, storage bits, and comparison operations.

Since we are interested in determining power-performance tradeoff analysis for future microarchitectures within a particular product family, we must determine a method of scaling the power of microarchitectural structures as the size of these sub-units increases. The scaling factor depends on the particular structure; for example, the power of a cache array will scale differently than 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 queues increases. Generally, as we increase the number of entries in a structure, there will be 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 3.3.

2.2 Web-Based Interface and Power-Performance Metrics

In order to thoroughly explore the modeled design space, we selected 19 workloads (8 SPECint95, 10 SPECfp95, and TPC-C) each of which was evaluated for over 75 hardware configurations. Analyzing this amount of data is difficult and a GUI makes the results of our analysis useful to our colleagues within IBM Research, as well as designers within the IBM Product groups. We developed a web-based back-end analysis tool which allows the user to select the benchmarks of interest and the microarchitectural parameter(s) to vary as well as the technology parameters such as frequency, voltage, and feature size.

The tool also allows the selection of various power-savings features such as the style of conditional clocking within the microarchitecture. Finally, the tool provides the choice of five power-performance metrics: Average CPI, average power dissipation, $CPI * power$, $(CPI)^2 * power$, and $(CPI)^3 * power$. The latter three metrics correspond to energy, energy-delay product [5, 2], and $energy * delay^2$. Since power is proportional to the square of supply voltage (Vdd) multiplied by clock frequency, and clock frequency is proportional to Vdd, power is proportional to the cube of Vdd. Thus delay cubed multiplied by power provides a voltage-invariant power-performance characterization metric which we feel is most appropriate for server-class microprocessors. In the remainder of the paper we will present our power-performance results as $(CPI)^3 * power$.

3 Power-Performance Evaluation Examples

In this section, we first provide a high-level description of the processor model assumed in our simulation toolkit. Then, we present some example experimental results with analysis and discussion. The results were obtained using our current version of PowerTimer, which works with pre-silicon performance models used in defining future PowerPC structures.

3.1 Base Microarchitecture Model

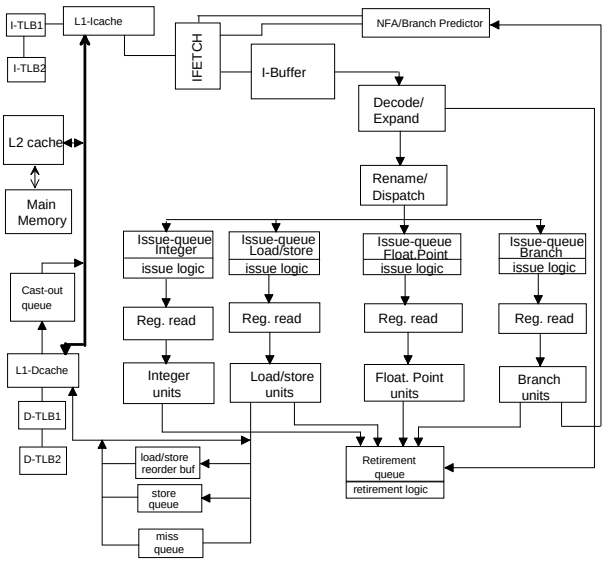

Figure 3. Processor Organization Modeled by the Turandot Simulator.

For the purposes of this paper, we assume a generic, parameterized, out-of-order superscalar processor model adopted in a research simulator called Turandot [8, 9]. The overall pipeline structure (as reported in [8]), is repeated here in Figure 3. The modeled microarchitecture is similar in complexity to a current generation microprocessor (e.g. [4, 7]). As described in [8], this research simulator was calibrated against a pre-RTL, detailed, latch-accurate processor model (referred to as R-model in [8]). The R-model is a custom simulator, written in C++ (with mixed VHDL "interconnect code"). There is a 1-to-1 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 1-2 orders of magnitude faster than 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 R-model that was used in [8]. That is, we first used our 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 post-processed the performance simulation output and fed the average resource utilization statistics to the energy models to get the average power numbers. This is faster than the alternative of looking up the energy models on every cycle. While it would have been possible to get 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 look at 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 fast CPI vs. Power tradeoff studies with full benchmark traces. Turandot allows us to experiment with a wider range and combination of machine parameters. In future publications and talks based on PowerTimer, we plan to report these results in detail.

3.2 Workloads Used in the Study

In this paper, we report experimental results based on the SPEC95 benchmark suite and a commercial TPC-C trace. All workload traces are PowerPC-based. The SPEC95 traces were generated using the tracing facility called *Aria* within the MET toolkit [9]. 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. A systematic validation study to compare the sampled traces against the full traces was done, in finalizing 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 IBM Austin. It is about 180 million instructions long.

In the following three sections we present examples of the use of the PowerTimer simulation infrastructure. The results show the average CPI and average $(CPI)^3 * power$ for the traces described above. Each SPEC data point was obtained by averaging across the benchmark suite. Note, however, that we have excluded *apsi* from the SPECfp results due to a problem with these simulation runs.

3.3 Data Cache Size and the Effect of Scaling Techniques

In this section we evaluate the relationship between performance, power, and L1 data cache size. We vary

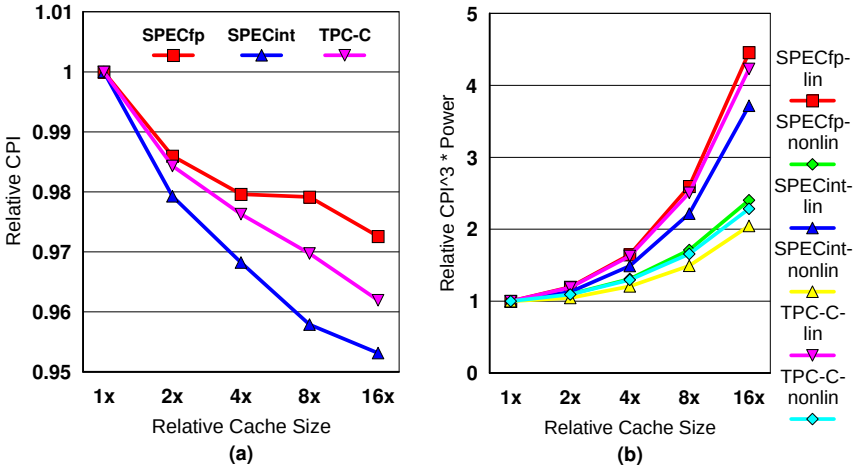

Figure 4. Variation of Performance and Power-Performance with Cache Size.

the cache size by increasing the number of cache lines per set while leaving the linesize and cache associativity constant. Figure 4a and 4b show the results of increasing the cache size from the baseline architecture (points labeled 1x on the x-axes). Figure 4a 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. Figure 4b shows the relation when we consider the metric $(CPI)^3 * power$. From Figure 4a, 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 Figure 4b, we show the results with two different scaling techniques. The first technique assumes 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 scaling technique is based on data from [6] which studied energy optimizations within multi-level cache architectures. In [6], data is presented for cache power dissipation for conventional caches with sizes ranging from 1KB to 64KB.

In the second scaling technique, which we call "nonlin" in Figure 4b, the cache power is scaled with the ratios presented in [6]. The increase in cache power by doubling cache size using this technique is roughly 1.46x, as opposed to the 2x with the simple linear scaling method. Obviously the choice of scaling technique can greatly impact the results. It is clear, however, that with either scaling choice, conventional performance-focused cache organizations will not scale in a power-efficient manner. (Note that the curves shown in Figure 4b assume a fixed circuit/technology generation; they are intended to show the effect of adding more cache to the current design.)

3.4 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 re-order buffer in that it tracks instructions as they dispatch, issue, execute, wait for exceptions, and complete. Figures 5a and 5b show the effects of varying the number of completion buffers on performance and the power-performance metric. From Figure 5a, it is evident that little additional performance is gained by increasing the number of buffers past the current design point (1x). When considering $(CPI)^3 * power$ in Figure 5b, we see that power-efficiency is slightly degraded by increasing the number of entries due to a roughly 3% increase in the core's power dissipation.

3.5 Ganged Sizing

Out-of-order superscalar processors of the class considered, rely on queues and buffers to efficiently decouple instruction execution 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 in the machine may quickly create a performance bottleneck. Thus, in this section we consider the effects of varying multiple parameters rather than just a single parameter.

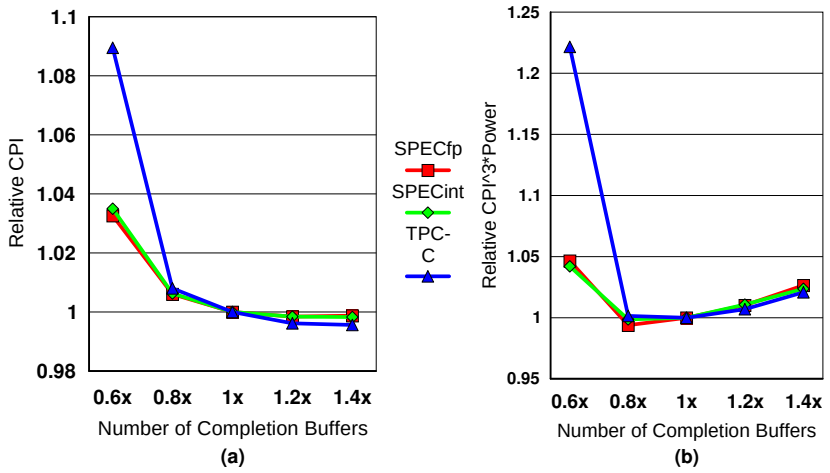

Figure 5. Variation of Performance and Power-Performance with Number of Completion Buffers.

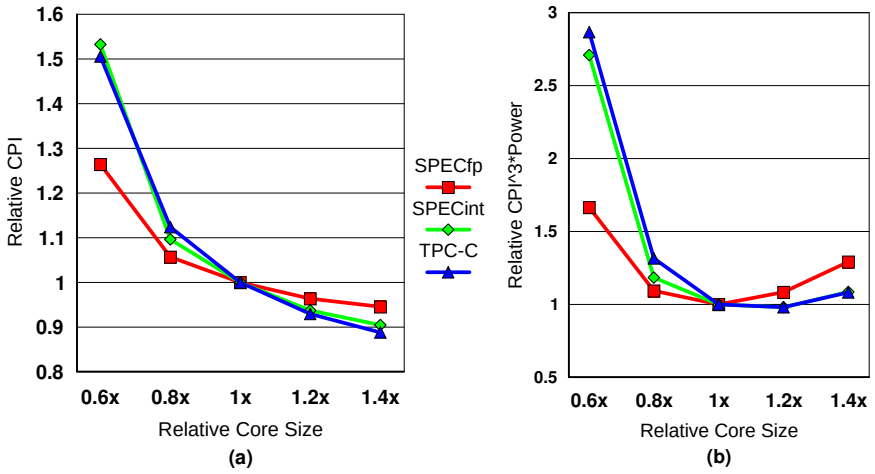

Figure 6. Variation of Performance and Power-Performance with Core Size (ganged parms).

Figure 6a and 6b show the effects of varying all of the resource 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 are doubled or halved at each data point. From Figure 6a, we can see 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 the caches which are 4x larger). The configuration had a power dissipation of 52%-55% higher than the baseline core. Figure 6b, shows that the most power efficient core microarchitecture is somewhere between the 1x and 1.2x cores.

4 Conclusion

We have described PowerTimer: a research power-performance simulator designed to help with the definition and evaluation of follow-on products within the high-end PowerPC microprocessor family. Based on this model, we have evaluated power and performance tradeoffs using SPEC95 workloads and a TPC-C trace. We have presented a few selected experimental results from our analysis repository to illustrate the kinds of tradeoffs that one may be able to study using this toolkit. A web-based interface allows users to view specific power-performance tradeoff curves of their choice.

This allows users to evaluate the worth and wisdom of making specific microarchitecture-level enhancements to an existing design point. The tool allows one to evaluate whether a certain aspect of the design is inherently power-efficient or not. For example, in an initial, voltage-invariant "technology remap" scenario, we may like to know whether simply increasing the cache sizes, without perturbing the core engine would buy us enough performance to counterbalance any power increase.

PowerTimer allows one to experiment with a large number of design parameters and there are multiple choices available in terms of selecting a power-performance efficiency metric. We have presented just a few examples in this paper. For example, one can study the effectiveness of various flavors of conditional clocking to see how the sensitivity curves are affected. Also, the use of technology scaling parameters, allows the user to explore the future design space in a realistic manner.

5 Acknowledgment

The authors would like to thank the other members of the power-aware microarchitecture team at IBM Research who provided valuable feedback during group discussions dealing with this particular work. Special thanks are due to Scott Neely and Steve Schmidt for help with the circuit-level power simulation tools and data.

Finally, Martonosi and Brooks were also supported in part by an NSF ITR grant, a donation from Intel, and an IBM University Partnership award. In addition, David Brooks is supported by an NSF Graduate Research Fellowship and a Princeton University Gordon Wu Fellowship.

References

[1] 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.

[2] T. Conte, K. Menezes, and S. Sathaye. A technique to determine power-efficient, high performance superscalar processors. In *Proceedings of the 28th Hawaii Int'l Conference on System Science*, 1995.

[3] A. Dhodapkar, C. Lim, and G. Cai. TEM^{2}P^{2}EST: A Thermal Enabled Multi-Model Power/Performance ESTimator. In *Power Aware Computing Systems Workshop at ASPLOS-IX*, Nov. 2000.

[4] K. Diefendorff. Power4 focuses on memory bandwidth. *Microprocessor Report*, pages 11–17, Oct. 6, 1999.

[5] R. Gonzalez and M. Horowitz. Energy Dissipation in General Purpose Microprocessors. *IEEE Journal of Solid-State Circuits*, 31(9):1277–84, 1996.

[6] U. Ko, P. Balsara, and A. Nanda. Energy optimization of multilevel cache architectures for RISC and CISC processors. 6(2):299–308, June 1998.

[7] C. Moore. The Power4 System Microarchitecture. *Microprocessor Forum*, Oct 2000.

[8] M. Moudgill, P. Bose, and J. Moreno. Validation of Turandot, a fast processor model for microarchitecture exploration. In *Proceedings of the IEEE International Performance, Computing, and Communications Conference (IPCCC)*, pages 451–457, Feb. 1999.

[9] M. Moudgill, J. Wellman, and J. Moreno. Environment for PowerPC microarchitecture exploration. 19(3):9–14, May/June 1999.

[10] Papers presented at: 1998 ISCA Workshop on Power-Driven Microarchitecture: http://www.cs.colorado.edu/ grunwald/LowPowerWorkshop/agenda.html, 1998.

[11] Papers presented at: 2000 ISCA Workshop on Complexity-Effective Design: http://www.ece.rochester.edu/ albonesi/ced00.html, 2000.

[12] N. Vijaykrishnan, M. Kandemir, M. Irwin, H. Kim, and W. Ye. Energy-driven integrated hardware-software optimizations using simplepower. In *Proc. of the 27th Int'l Symp. on Computer Architecture*, June 2000.

[13] V. Zyuban. *Inherently lower-power high performance superscalar architectures.* PhD thesis, University of Notre Dame, January 2000.