

A Comparison of Two Architectural Power Models

Soraya Ghiasi and Dirk Grunwald
University of Colorado
Department of Computer Science
Boulder, CO 80309
{ghiasi, grunwald}@cs.colorado.edu

University of Colorado

Abstract. Reducing power, on both a per cycle basis and as the total energy used over the lifetime of an application, has become more important as small and embedded devices become increasingly available. A variety of techniques are available to reduce power, but it is difficult to quantify the benefits of these techniques early in the system design phase when processor architecture is being defined. Accurate tools that allow for exploration of the design space during this phase are crucial. This paper describes our experience with two such tools, the Cai-Lim power model and Wattch, which have been made available to the computer architecture community over the past year.

We focus on how the models are constructed, the granularity of activity revealed by the models, the ability to understand *why* particular power results are obtained and the accuracy of the models. We raise concerns about detailed simulations where the power model, the simulator model and the desired architecture to be simulated differ and the validity of data obtained in such situations.

Keywords: Power Analysis Tools, Performance Comparison, Validation, Architectural-definition stage

1 Introduction

It is increasingly important to consider power use at every phase of processor design. Traditionally, power estimates and analysis were performed after a design was laid out, typically using SPICE analysis. As power demands have increased, CAD tools include power analysis earlier in the design cycle. For example, the Synopsys toolsuite contains tools to estimate power demands using information from standard cells and activity profiles from system components. The Synopsys PowerCompiler automatically adds clock gating, sizes transistors and suggests some logic transforms that may reduce power.

Recently, the computer architecture community has attempted to model power during the architectural design phase. As in any CAD problem, different levels of abstraction, accuracy and effort are appropriate for different stages of the design process. During the architectural design phase, power analysis tools should assist in floor-planning architectural designs intended to save power and coarse-grain estimates of power. At the synthesis or layout phase, power analysis tools should assist in transistor sizing, clock gating and the like. As a design nears completion, power analysis tools must provide accurate information which may affect packaging selection or system specification. As designs progress from initial to final stages, the tools should produce increasingly accurate information; that increased accuracy requires additional time and expense. Early in the design phase, designers need tools that quickly reflect power savings in reliable terms but with possibly less accuracy.

We think that architecture definition stage power analysis tools will be used in one of two styles. In the first style, an architect may have an idea for reducing average power that could be easily added to an existing design. The designer would like to quickly determine if the new mechanism would actually save power by modifying an existing design; if it appears promising, the full design can be synthesized and analyzed in more detail. Tools designed to modify existing designs may be reasonably accurate because they can exploit detailed functional and circuit models from extant designs. The full processor design may not be greatly impacted by the modifications, resulting in less error.

Alternatively, the designer may be starting a new processor design, possibly using a new process technology and microarchitecture. Power estimation tools for this design style face a more difficult challenge because aspects of the processor design that affect a large portion of the power budget (such as clocking policies and clock networks) may not be finalized. Thus, the power analysis tool must be generalizable and extensible to insure that it is useful.

Most architectural-level power estimation tools use *activity based power estimation*. This paper compares two activity based power models that have been available to the architecture community for approximately one year. We compare these models using two experiments. In one experiment, we try to determine if a processor should use in-order or out-of-order execution semantics, a question typically asked much earlier in the design cycle. In the second experiment, we modify a processor design that is very similar to the design used to develop the power model.

Our results were discouraging. Using available information about the performance bounds of the power models, we found the two extant power models disagree on the efficacy of the design choices in each experiment and do not always produce statistically significant results.

In Section 2, we examine the power models currently available. Our experiments and experimental criteria are described in Section 3. We present our results in Section 4, followed by our conclusions and suggested directions for future work in Section 5.

2 Available Power Models

Developing an accurate model is a time consuming task that requires detailed low level knowledge of circuit design and infrastructure that many researchers do not generally have. Two tools, the Cai-Lim power model and Wattach, are currently available to researchers in the community who would like to use power models during the architecture definition stage.

Each of these are fundamentally similar and rely on *activity based power models* to estimate power demands. A transistor or gate-level power model uses a set of input vectors, such as a specific use of an adder or cache logic, to model power demands of a given design. Gate level models keep detailed information about the state of each register, latch or bitline in the final design. This is time-consuming, because the architecture is simulated in great detail, but is also accurate because of that same level of detail – the detailed power models may accurately record the effects of precharging, short circuit loss and so on. Statistical or activity based gate or RTL designs are simpler. They work by assigning an “average power cost” for a 0,0, 0-1, 1-0 or 1-1 transition on individual device structures. These costs are determined by detailed models of individual circuit elements. The RTL model is then used to execute test vectors or applications and the occurrence of each action (*e.g. 0-1 transition*) is recorded and multiplied by the pre-computed activity cost. The model is less time-consuming because only the register state must be maintained, not the full analog circuit characteristics.

The models we examine are statistical or activity functional power models. In these power models, the power demands of an entire functional unit may be measured over many different test vectors. The resulting power measurement constitutes an “average” power measurement for that functional unit and each use of the functional element in the microarchitectural simulator is assumed to use the average power. For example, a integer ALU can perform many functions over many inputs. In a statistical functional energy power model, the model developer would use a set of representative (or randomly generated) inputs to determine the cost or power demand of the “average integer operation”, and then use that value each time the integer ALU is accessed. These models typically assume some cost for inactive power, but often assume that inactive power is simply 10% of active power. These models are much faster than the more detailed models because even less state and computation must be performed for each simulated execution cycle – the energy model boils down to a set of “activity counters” for specific functional components of the microarchitecture and estimated average power costs. Some of these quantities can be scaled for different component sizes. For example, a memory unit can be broken into smaller components and a “power density” and area can be recorded for a each smaller component; as the size of that structure is varied, the area is used to scale the total power cost for accessing that unit. This is particularly useful for memory-intensive or array structures within a processor.

Clearly, the accuracy of these models depends greatly on the accuracy of the estimate power models of each functional component and the detail with which component accesses are modeled. This can limit their applicability – if the microarchitectural model can not be or is inaccurately represented using the available components, the model produces meaningless results. The accuracy of the models is also affected by how and in what situation they are applied. We have found that it is necessary to understand the underlying architecture of both the simulation model and the power model to exercise the appropriate caution in applying the power model to modified architectures.

Both of the models we examine are based on the SimpleScalar toolset[3], a toolset commonly used to model microarchitectures in educational and some research environments. The basic microarchitecture sim-

ulator in SimpleScalar (`sim-outorder`) models an out-of-order microprocessor with a Register Update Unit [13], variable instruction fetch width, configurable memory sizes and variable branch predictor and fetch processing organization. The simulator does not model any specific extant processor, such as the Intel PentiumPro or Compaq 21264, but can be quickly configured to model processors with different resource constraints.

We now present an overview of each tool including what is provided, as well as information on granularity, access to details, flexibility and validation.

2.1 Cai-Lim Power Model

The Cai-Lim power model[5] is an activity sensitive power model built upon the SimpleScalar 2.0 out-of-order simulator. It partitions the SimpleScalar architecture into 17 hardware structures which are further subdivided into a total of 32 blocks. Each block is then further partitioned into power density and area for both active and inactive contributions from dynamic, static, PLA, clock, and memory sections of the block. Area estimates are based on publicly available designs with additional area allocated for clocking, interconnects, and power supply. SPICE simulations were used to find the active power density of typical designs based on Taiwan Semiconductor Manufacturing Corporation's 0.25 μ m process files. Inactive power densities were estimated as 10% of the active power densities. Power density numbers are used as constants in conjunction with the activity counters to model power consumption.

The Cai-Lim model is designed to provide a "structure that any microprocessor design can use with their own data"[4]. The values for power density and areas provided with the currently available version of the power model does not match a specific architecture but is based on common structures available today such as a 6T SRAM cell in a .25 μ m process.

The Cai-Lim model tracks how a hardware structure is used by breaking it down into different types of accesses and then counting each time that type of access occurs during a cycle. For example, the level 1 data cache is comprised of 3 blocks - logic, cache and taglines - which are be used whenever there is an access, writeback, replacement, or invalidation to the cache. This structural breakdown and its associated information provide an opportunity for detailed modeling and an ability to track reductions in dynamic activity.

All values for power densities and areas have been pre-computed and included as part of the source code. The lack of direct access to the models used to generate these values makes it difficult to scale units appropriately, even within the same process and architectural family. For example, the original RUU modeled contains 16 entries. If the number of entries is to be changed, the area should be scaled appropriately because it makes up a larger portion of the overall design. Unfortunately, linearly scaled is not necessarily the same as appropriately scaled. The lack of directly accessible details on structural scaling factors limits the Cai-Lim power model's ability to be used to directly compute relative contributions to power from different blocks. The model is difficult to extend without examining the original process files to determine how to incorporate new hardware structures. It is also difficult to perform scaling of components based on changes in size of the component, not changes in technology.

Even with this shortcoming, the Cai-Lim model remains quite flexible, but of unknown accuracy. In general terms, a well designed architecture definition stage model is expected to be accurate to about 25%[5]. The Cai-Lim model currently uses the same power density for all blocks which may lead to larger inaccuracies than expected.

The Cai-Lim model has been validated against the SPICE models used to generate the elements and power densities. Its ability to model a real architecture is unknown. The Cai-Lim model is not intended to provide absolute numbers for any design, but can provide relative numbers between designs. However, without an analysis of the per component contribution to the overall error in final results, it is difficult to assess whether a given change produces meaningful results.

TEM²P²EST is a new version of the Cai-Lim model[6], but was not available for evaluation in time. The Cai-Lim model has also been adapted for use in SMTSIM[12].

2.2 Wattach

Wattach[2] is a collection of power models. The first is an "all components always on" model and the remaining 3 models are activity sensitive with varying degrees of conditional clocking enabled. The conditional clocking options range from full power for a component consumed during any cycle in which it is accessed

and zero otherwise to linearly scaled power based on component usage when accessed and 10% of base power when not accessed. Wattch is built upon the SimpleScalar 3.0 out-of-order simulator that has been extended conceptually from a 5 stage pipeline to an 8 stage pipeline. The additional stages are inserted only for the global power calculation and have no other effect on the system. Wattch developed basic components - array structures, content-addressable memories, combinational logic and wires, and clocking. These components are then used to build a parameterized model of major hardware structures. It again does not model a particular architecture but its use of an H-tree clock distribution and separate accounting of clock power resembles an Alpha. Rather than using a power density and area based model, capacitances are calculated from wire delays[10, 11] and then used to generate a per cycle cost for a structure. These costs may then be scaled by usage indicated by activity counters.

Wattch is designed to provide a framework upon which future work can be built. Wattch provides an infrastructure for comparisons between different power reduction techniques and provides a parameterized model that can quantify power reductions. It also exposes the underlying details of its power model to the users. Wattch claims an accuracy within 10% of layout-level power tools. Wattch's published results indicate an average accuracy of +/-13% when comparing relative power against known relative powers for implemented architectures (Pentium Pro and Alpha 21264). A per component analysis of Wattch's accuracy is not available in currently published works.

Wattch tracks how a hardware structure is used in a manner similar to that employed by the Cai-Lim model but does not currently support the same level of granularity. In addition, some of the structures are based on an all or nothing power calculation while others, such as branch prediction, use a specified cost for up to $2 * someLimitingFactor$. Counters for different types of accesses are employed, but some details are left out. For example, Wattch tracks accesses to the level 1 data cache, but does not currently track invalidations, replacements, or writebacks. It does take into account logic, array (cache in the Cai-Lim model) and tag contributions to the total power of the cache.

Wattch provides greater access to the underlying details of the models than the Cai-Lim model provides. Researchers are provided with information in the code about how the power numbers are derived and structural scaling factors are calculated by Wattch during an initialization phase of the program. Technology scaling factors are included for processes ranging from $.10\mu m$ to $.80\mu m$. Wattch uses this information as well as the SimpleScalar configuration specified to initialize power values for different aspects of the hardware structures. For example, the power calculations for the instruction window are dependent on the process size as well as the RUU size.

Wattch provides some of the flexibility needed by researchers but has shortcomings that can limit its usefulness. The lack of granularity in access counting limits Wattch's ability to identify activity reduction power savings. The underlying hardware structure breakdown provides users with the opportunity to build customized components to study specialized hardware, but with unknown accuracy. Wattch also does not provide information about the contributions to total error by the various components, leaving researchers with little way to judge the validity of designs that involve components of sizes that differ from published implementations. A detailed correlation study would reduce this last concern.

3 Methodology

We have attempted to place Wattch and the Cai-Lim power model on as similar a footing as possible. Wattch consists of 4 power models. We have selected the Wattch conditional clocking power model (model cc3 in Wattch's source code) that most closely matches the active and inactive contributor model used by the Cai-Lim model. Both models calculate the active circuit power and then set the inactive circuit power to 10% of the active circuit power.

3.1 Experiments

Our experiments, described below, were performed using a modified SimpleScalar 3.0 out-of-order simulator that contained both Wattch and the Cai Lim model. To ensure the combined models produced independent results, we also performed a separate data independence experiment to confirm that an out-of-order simulator containing only Wattch or only the Cai-Lim model produced the same results as our combined out-of-order simulator. We found this to be true for data runs in all our power experiments. We ran the SPEC95INT benchmarks under our combined Wattch/Cai-Lim simulator using the configurations shown in Table 1.

Pipeline Simulation Configurations		
Parameters	4-wide	8-wide
Machine Width	4-wide fetch, 4-wide issue, 4-wide commit	8-wide fetch, 8-wide issue, 8-wide commit
Window Size	64 entry register update unit, 32 entry load/store queue	256 entry register update unit, 128 entry load/store queue
Branch Misprediction	min. recovery latency 6 cycles	
L1 Icache	64kB, 32 byte lines, 2-way set-associative, 2 cycles hit latency	
L1 Dcache	64kB, 32 byte lines, 2-way set-associative, 2 cycles hit latency	
L2 Cache Combined	512kB, 64 byte lines, direct mapped, 6 cycles hit latency	
Memory	128 bit wide, 26 cycles access latency	
BTB	1024 entry, 4-way set-associative, 32 entry return address stack	
TLB	64 entry (I), 128 entry (D), fully associative, 30 cycle miss latency	
Functional Units and Latency (total/issue)	4 Int ALU (1/1), 1 Int Mult (3/1) / Div(20/19), 2 Load/Store (2/1), 4 FP Add (2/1), 1 FP Mult (4/1) / Div (12/12) / Sqrt (24/24)	8 Int ALU (1/1), 2 Int Mult (3/1) / Div(20/19), 4 Load/Store (2/1), 8 FP Add (2/1), 2 FP Mult (4/1) / Div (12/12) / Sqrt (24/24)

Table 1. Pipeline parameters for our base pipeline simulations.

Low Power Mode for IPC Matching. In prior work, we explored the possibility of exploiting naturally occurring variations in the IPC of a program to reduce power consumption[8]. We found that our MPEG benchmarks were able to spend a significant amount of time in a dynamically pipeline gated fetch mode, in an in-order issue mode, or a combination of these two modes depending on the characters of the time-varying IPC. To continue this work, we undertook a limit study to explore the effects of running applications under dynamic pipeline gating, in in-order issue mode, or in the base case for the next smaller 2^n machine width (half-width) for the duration of the applications run. A single set of results consists of a base case (8-wide or 4-wide), pipeline gated run, an in-order issue run, and a half-width base case (4-wide or 2-wide). The lower power modes we investigated involve no explicit conditional clocking on unused sections of circuit blocks. Instead, the power savings observed stem from reduced activity throughout the pipeline and the conditional clocking provided by the power models.

Dynamic Instruction Queue Resizing. Folegnani *et al.* introduced a separate low power optimization [7] that extended the design of the current Compaq Alpha 21264 architecture. They found that the ready instruction queue was often underutilized and that the extent of this underutilization varied over an application's lifetime. A special tag is added to instructions issued indicating if they were issued from the last segment of the current instruction queue. When no instructions are issued from the last segment for some number of cycles, the segment is powered off, reducing the power consumed by the queue as well as reducing the power needed for writebacks and selections. We performed similar simulations using the basic 8-wide and 4-wide configurations from Table 1. Power reductions observed under dynamic instruction queue resizing are due to explicitly shutting down a portion of the instruction queue. Power consumption may change in other blocks as well because the distribution of the workload is changed by applying dynamic instruction queue resizing.

3.2 Evaluation Criteria

Both models are designed to demonstrate relative power savings between designs. We have selected our criteria on this basis. Each application run is normalized against the power results of the appropriate base case so that relative power reductions can be compared. We examine per cycle power and energy where appropriate. Wattch claims an accuracy of within 10% of the results provided by layout-level power tools. We use this 10% estimate of error provided by Wattch to evaluate whether or not the results from Wattch cc3 model and the Cai-Lim model are statistically different from each other as well as statistically different from the base case. We believe the actual error in the models may be larger, implying that the results from additional experiments will fall below the threshold of statistical significance. This has implications for comparisons between Wattch and the Cai-Lim model within a single data set as well as for comparisons between applied power reduction techniques and base case without the technique under either model.

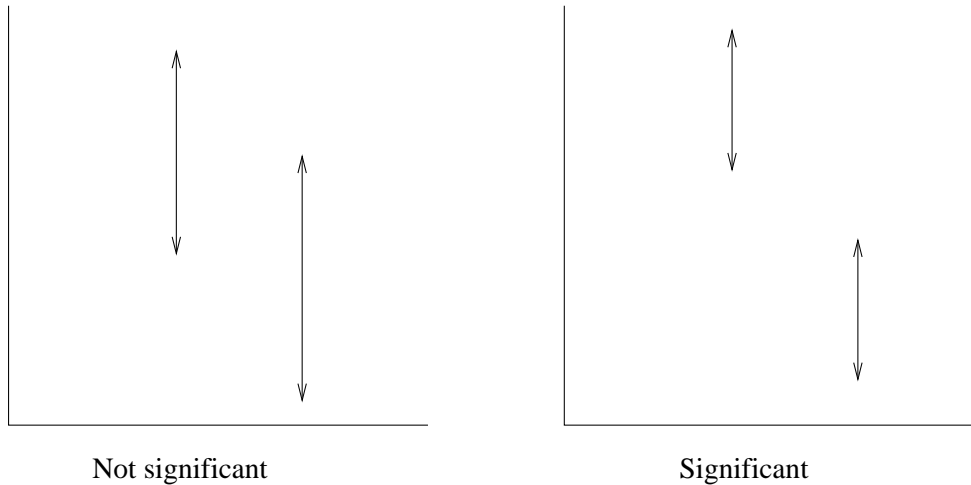


Fig. 1. The overlapping confidence intervals shown on the left fail the visual test for statistical significance. The non-overlapping intervals on the right indicate a statistically significant result.

3.3 Statistical Significance

The analysis of errors under either power model should take into account the initial random errors in calculations and then the propagation of these errors throughout the remainder of the power model. We have not done this in detail for the power models we investigated, but we hope it will be done for power models in wide-spread use.

We use Wattach's published error estimation as the sample variance and use a sample size of one to calculate the 90% confidence level. Conceptually, we are most interested in whether a run differs from its base case. If the resulting confidence interval includes the base value, we cannot say the new result is statistically different from the base case result. Applying this technique leads to a range of approximately $\pm 16\%$. Normally, a t-test is applied in situations where the confidence intervals overlap with each other, but do not include the base case result. A t-test is not possible with a sample size of one, so the visual test for significance was applied instead. An example of a visual test for significance is shown in Figure 1.

Throughout the remainder of the paper, when we discuss statistical significance we are using the 90% confidence level to determine significance. Additional information on confidence intervals can be found in [1, 9, 14]. Increasing the confidence level will increase the range that must be considered when evaluating the results.

4 Results

For each experiment, we briefly discuss our expected results. We then present our actual results including the measure of statistical significance developed in Section 3. Finally, we analyze why the results differ from each other.

4.1 Low Power Mode for IPC Matching

Figure 2 shows a subset of the results for this experiment. These benchmarks were selected because they represent the outliers in the 4-wide case. Other benchmark results fall between these. All runs shown are normalized against a base case where no power reduction techniques have been applied. We are primarily interested in per cycle power because our intent is to discover appropriate lower power modes that can be applied dynamically in response to natural variations in the IPC over the lifetime of the program. We also examine the energy to see if it is significantly different than the base case and if it would lead us to draw different conclusions if energy reduction were our primary goal.

Our expectation was that the three techniques applied would each result in a power reduction with in-order issue producing the greatest relative reduction in power. It was unclear if energy would also be reduced because of its dependence on the number of total cycles executed. Finally we were primarily interested in which technique(s) would provide us with the largest relative power reduction.

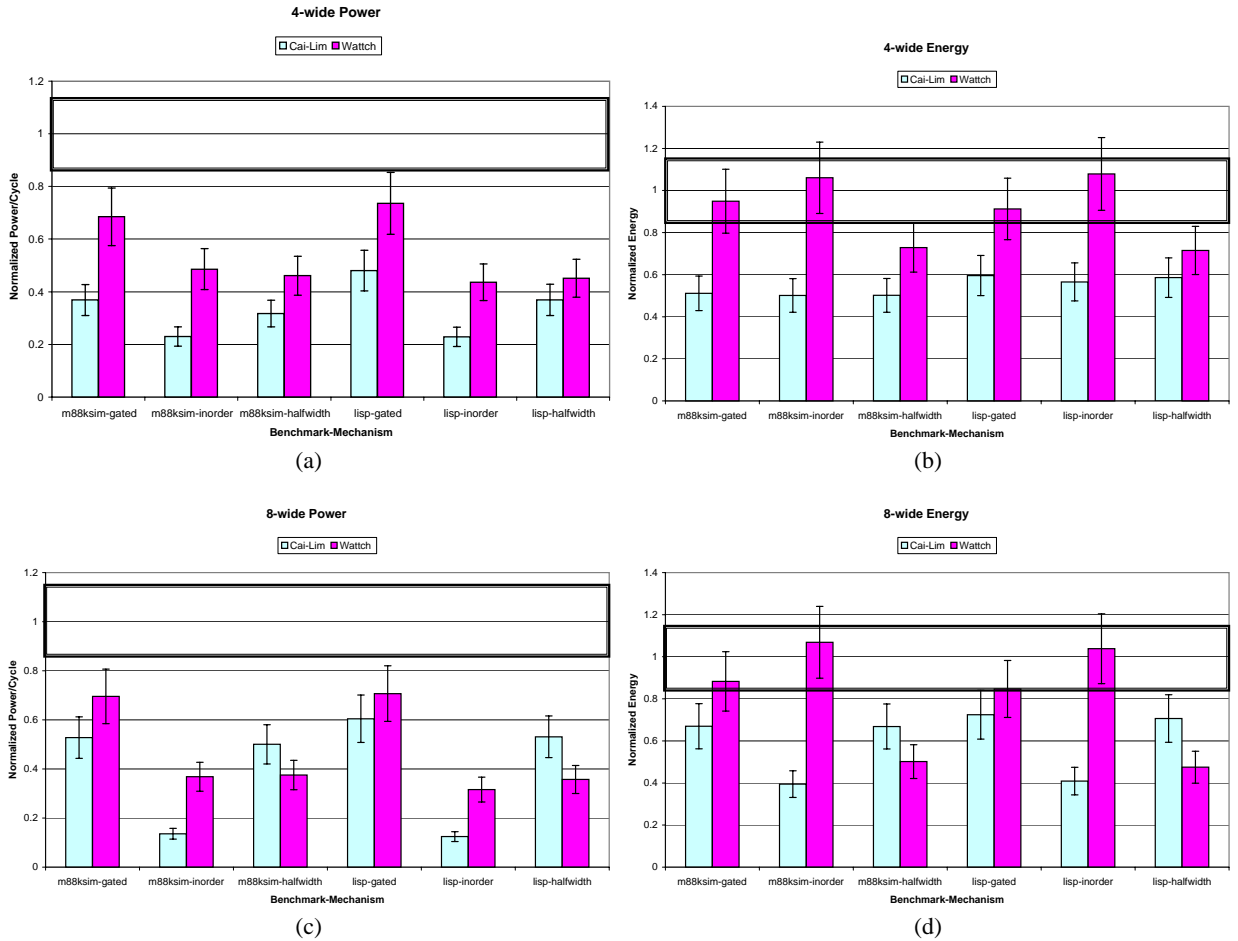


Fig. 2. Experiment 1 - Possible IPC Matching Techniques on 4 and 8-wide microarchitectures. All results are normalized against the appropriate base case. Error estimation is based upon the 10% margin of error reported by Watch at the 90% confidence level. Error bars for the base cases are indicated by the box centered around 1.

Results. Applying the pipeline gating technique for the entire duration of the benchmarks runs indicates a statistically significant reduction in relative per cycle power under both 4-wide and 8-wide microarchitectures (see Figure 2 (a) and (c)). However, the results for the Cai-Lim model and Watch are also significantly different from each other in our 4-wide microarchitecture results. The Cai-Lim model consistently produces lower relative per cycle power results.

In-order issue produces a further power reduction under the Cai-Lim power model and Watch for both 4-wide and 8-wide microarchitecture. In all Watch results, the reduction is significantly larger than that provided by pipeline gating. The results under the Cai-Lim model included both statistically significant (Figure 2(a) and (b), lisp-gated and lisp-inorder) and insignificant (Figure 2(a) and (b), m88ksim-gated and m88ksim-inorder) values. The Cai-Lim model and Watch are also significantly different from each other.

The results from the half-width case highlight differences between the models. Under the Cai-Lim model, this case is indistinguishable from the pipeline gating case (Figure 2 (a) and (c), m88ksim-gated and m88ksim-halfwidth, and lisp-gated and lisp-halfwidth). Under the Watch model, it is instead indistinguishable from the in-order case (Figure 2 (a) and (c), m88ksim-inorder and m88ksim-halfwidth, and lisp-inorder and lisp-halfwidth). m88ksim-halfwidth and lisp-halfwidth show the Cai-Lim model and Watch are also not statistically significant from each other in the half-width case.

The two power models would lead to different choices if they were used to select an appropriate lower power mode to exploit periods of low IPC activity. The Cai-Lim model indicates in-order issue is the best choice for per cycle relative power reduction for both 4-wide and 8-wide designs. Watch fails to distinguish between in-order issue and half-width allowing the designer to arbitrarily choose between these two techniques.

We also explore the affect of these techniques on energy. The effects of small differences are magnified because of the increase in the runtime of the program. In all cases, the Cai-Lim model indicates an overall reduction in energy. The Wattach model has more variable results, including a number of cases where the energy is not statistically different from that of the base case.

Based on our energy results, we would reach different conclusions depending upon which model we were relying upon. For example, Figure 2(b) on a 4-wide machine, the Cai-Lim model indicates that all three techniques are equally valid and produce comparable energy reductions. Under Wattach, we find that the best energy reduction technique is actually that of running at the next smaller machine width.

We can draw similarly opposing conclusions based upon the 8-wide results (see Figure 2(d)). The Cai-Lim model clearly favors the use of in-order issue for energy reduction, while Wattach predicts in-order issue will have the worst relative energy effects and will, in fact, increase energy.

Analysis. The Cai-Lim model produces larger power reductions than Wattach in all cases except for the half-width 8-wide microarchitecture (a 4-wide machine). Although not all of these results lie outside the range required for statistical significance, they do indicate an overall trend. The Cai-Lim model appears to be better able to distinguish differences in dynamic activity than the Wattach model. The results are partially a result of the underlying microarchitectural model and partially a result of the types of activities recorded.

For example, the Wattach model determines whether or not any instructions have been fetched this cycle by counting the number of icache and branch predictor accesses. The resulting icache power is based on the notion of accessed or not accessed. This models an architecture where only a single line is fetched unless the pipeline stalls. After a line is fetched, as many instructions as possible are used from it. However, the out-of-order simulator used in this case does not model this architecture so there is a disparity between the power model’s architecture and the architectural simulator’s architecture. In addition, Wattach provides no mechanism for power consumption accounting during fetch if the fetch engine is able to fetch multiple cache lines per pipeline cycle.

The Cai-Lim model instead distinguishes between accesses to next PC prediction and writes to the dispatch queue. The number of accesses and writes are factored into separate circuit blocks. This model allows multiple fetches during a cycle to be accurately accounted for and shows power reductions when the pipeline gating technique is applied partially because the writes to the dispatch queue have been halved. This more closely models the underlying simulation architecture which can make repeated icache searches during fetch and an architecture in which the fetch engine runs faster than the rest of the pipeline.

The accounting and breakdown done by each model also has implications on the power consumed by the register file. For example, on a 4-wide architecture, the Cai-Lim model indicates that half as much power is consumed by the register file under pipeline gating when it is compared to the base case. Wattach indicates a much smaller reduction. The Cai-Lim model produces results that indicate larger per cycle power and overall energy savings for the load-store queue (LSQ), the register file, and the Register Update Unit (RUU) than Wattach does.

Overall, the Cai-Lim model has a finer granularity for determining activity within units. For example, Wattach employs a single notion of instruction cache (icache) access, but the Cai-Lim model tracks instruction lookaside buffer (itlb) accesses, replacements, invalidations, and writebacks in addition to instruction tag (itag) and icache contributions that are dependent upon level-1 icache (i1l) accesses, replacements, invalidations, and writebacks. In addition, each unit is broken down into its dynamic, static, clock, memory, and PLA contributors allowing the model to be finally tuned by adjusting its power densities and areas to match specific target architecture. Unfortunately, this granularity is not fully utilized in the current implementation of the Cai-Lim model because the same power density is used throughout for each block leading to potentially erroneous results. Additionally, the lack of access to the details needed for determining the area used by components whose sizes differ from the Cai-Lim base model means that these were not adjusted to reflect their new size.

4.2 Dynamic Instruction Queue Resizing

Figure 3 shows the results for this experiment. All runs shown are normalized against a base case where no power reduction techniques have been applied. We are primarily interested in energy because our intent is to discover if dynamic instruction queue resizing is an appropriate technique for exploiting periods during an application’s lifetime where newly arrived instructions are not being issued. We are interested in whether or

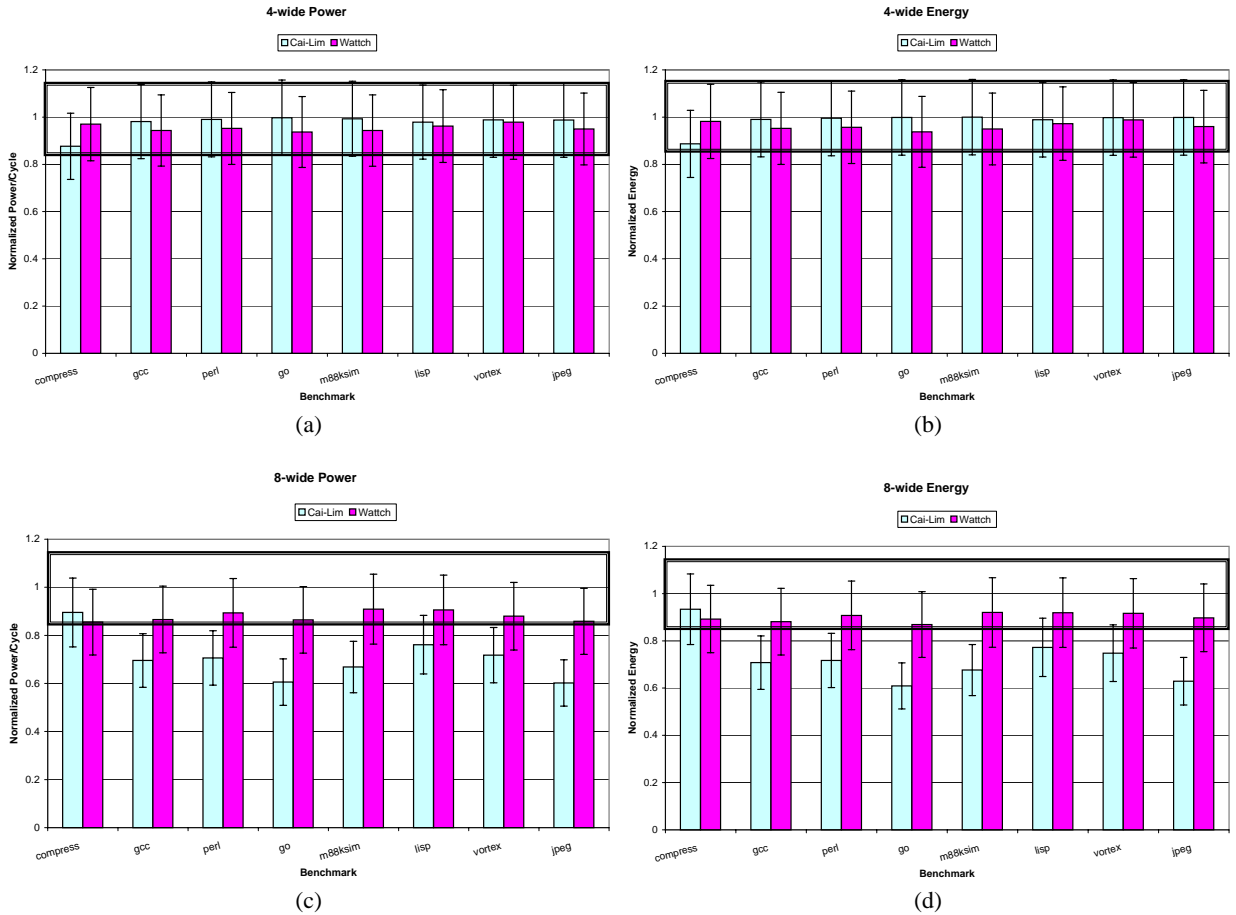


Fig. 3. Experiment 2 - Dynamic Instruction Queue Resizing on 4 and 8-wide Alpha-like architectures. All results are normalized against the appropriate base case. Error estimation is based upon the 10% margin of error reported by Watch at the 90% confidence level. Error bars for the base cases are indicated by the box centered around 1.

not the models are able to reflect the fact that the average instruction queue size under dynamic instruction queue resizing is often considerably less than its maximum size and how well conditional clocking can be applied to this block in the power models.

We expected both models to show a relative power reduction due to the scaling of the instruction queue. We also expected an energy reduction because the overall effect on the runtime of the application is less than 10%. We are more concerned with energy results in this case because we are interested in the overall effect of applying this technique.

Results. Under a 4-wide architecture, we observed only a single case where the energy reduction was statistically significant using our minimum estimate of the error in the power models (Figure 3 (a) Cai-Lim compress). Realistically, the error is larger and none of the results are statistically different from the baseline. The Watch and Cai-Lim models also are not distinguishable from each other. Our results are much more conservative than Folegnani *et al.* because we used a smaller instruction queue than they did.

The 8-wide architecture indicates a statistically significant energy reduction for all benchmarks except compress when examined with the Cai-Lim model. Once again, the Watch model produced energy reductions that are not statistically different from the baseline. In this case, the Cai-Lim and Watch models are statistically different from one another for the benchmarks that show the largest power reductions (go, m8ksim, and jpeg). Cai-Lim and Watch are not statistically distinguishable for the remaining benchmarks.

Under the Cai-Lim model we would conclude that dynamic instruction queue resizing is a worthwhile technique to pursue on our 8-wide architecture. We would not draw similar conclusions from either power model on the 4-wide architecture or from Watch on the 8-wide architecture.

We have included the average per cycle power results as well as an indication that dynamic instruction queue resizing did not increase the run time of the applications inordinately. This can be seen by examining the normalized average per cycle power results and comparing them to the normalized energy results. Dynamic instruction queue resizing increases the average runtime of an application by a few percent.

Analysis. There are a number of factors that contribute to the resulting differences. One difference in our results generated by the two power models can primarily be attributed to the different clock representation methods employed by each model. The Watch model keeps track of the clock power as a separate component, while the Cai-Lim model calculates the clock power on a per block basis using the amount of area of a block devoted to clock power distribution. This allows the Cai-Lim model to scale the clock contribution of array structures more easily. Watch precalculates the clock contribution and then simply scales it for each cycle by the comparing the current cycle’s total power to the maximum total power. In order to accurately examine the effects of array structure scaling, it would be necessary to either recalculate the clock contribution every time an array structure such as the instruction queue is resized or to precompute all the possible clock values based on all possible array structure sizes during power model initialization. Neither of these solutions was applied in this case.

Another contributor to the observed differences can be found in the base numbers themselves. The RUU includes the reorder buffer behavior. The rename registers cannot be gated off because they may still be referenced. The power savings then comes from not dispatching new instructions into the “disabled” portion of the RUU and not broadcasting values to it. Under the Cai-Lim model, these form a larger contribution to the total power than they do under Watch.

4.3 Which Model to Use?

It is unclear which model provides a more accurate picture of the experiments. The unknown accuracy of the Cai-Lim model could mean that the results it produced are wildly inaccurate. Its use of a single power density for all components and lack of accessible structural scaling factors also limit its applicability. The less finely grained counters as well as the accounting methods used during fetch and decode under Watch may not account for the types of activity reduction the low power modes are expected to produce. The unknown contributions to Watch’s total error that occur under structural scaling suggests that caution should be used when applying it outside existing processor designs.

Further analysis should be done before using either power model in situations where the target architecture differs significantly from that used by the power model. For example, both power models are based upon the use of an RUU. The inaccuracies that may be introduced by attempting to model a reservation station-based design may blur any possible power savings.

5 Conclusions and Future Work

Although architectural level power modeling tools are important as we continue to explore new low power optimization techniques, we find that neither of the two freely available tools are sufficiently accurate or complete to allow wide-ranging experiments. We also urge extreme caution when analyzing results produced by either of the current solutions. Our results indicate that even at a 90% confidence level, the power models often fail to produce results that are statistically distinguishable from the base case. Unfortunately, we also observed situations in which the models produce results which are statistically different from each other for the same power reduction technique. We find that the current accuracy of the models will need to be improved upon to detect anything but the largest power savings.

We have found it is necessary to keep in mind both the underlying architectural model used in the power model as well as the model of the simulator. In situations where the two models differ, such as fetch and decode under Watch and SimpleScalar, one or both models may need to be modified to more closely match the desired behavior.

It is hoped that as these models are revised and enhanced these shortcomings will be overcome, making them more generally useful and accessible to the research community. In particular, we would like to see enhanced versions that combine the flexibility of Watch with the granularity of Cai-Lim. In addition, to produce results which we can confidently state are significant, it will be necessary for new models to more thoroughly analyze the error in individual components as well as the overall error.

References

1. A. O. Allen. *Probability, Statistics, and Queueing Theory with Computer Science Applications*. Academic Press.
2. D. Brooks, V. Tiwari, and M. Martonosi. Wattch: A Framework for Architectural-Level Power Analysis and Optimization. In *Proceedings of the 27th International Symposium on Computer Architecture*, pages 83–94, Vancouver, Canada, June 2000.
3. D.C. Burger and T.M. Austin. The SimpleScalar Tool Set, Version 2.0, 1997.
4. G. Cai. Personal communication, September 2000.
5. G. Cai and C. H. Lim. Architectural level power/performance optimization and dynamic power estimation. Cool Chips Tutorial colocated with MICRO32, November 1999.
6. A. Dhodapkar, C. H. Lim, and G. Cai. *TEM²P²EST*: A Thermal Enabled Multi-Model Power/Performance ESTimator. In *Workshop on Power-Aware Computer Systems*, Boston, MA, Nov 2000.
7. D. Folegnani and A. Gonzalez. Reducing Power Consumption of the Issue Logic. In *Workshop on Complexity Effective Design*, Vancouver, Canada, June 2000.
8. S. Ghiasi, J. Casmira, and D. Grunwald. Using IPC Variation in Workloads with Externally Specified Rates to Reduce Power Consumption. In *Workshop on Complexity Effective Design*, Vancouver, Canada, June 2000.
9. R. Jain. *The Art of Computer Systems Performance Analysis*. Wiley Press.
10. S. Palacharla, N. P. Jouppi, and J. E. Smith. Quantifying the complexity of superscalar processors. Technical Report CS-TR-96-13-28, University of Wisconsin, November 1996.
11. S. Palacharla, N. P. Jouppi, and J. E. Smith. Complexity-effective superscalar processors. In *Proceedings of the 24th International Symposium on Computer Architecture*, Denver, CO, June 1997.
12. J. S. Seng, D. M. Tullsen, and G. Z. N. Cai. Power sensitive multithreaded architecture. In *Proceedings of the 2000 International Conference on Computer Design*, 2000.
13. G. S. Sohi and S. Vajapeyam. Instruction issue logic for high-performance interruptable pipelined processors. In *Proceedings of the 14th Annual Symposium on Computer Architecture*, pages 27–34, 1987.
14. J. Taylor. *An Introduction to Error Analysis*. University Science Books.