Power-Performance Simulation: Design and Validation Strategies

David Brooks†, Pradip Bose‡, Margaret Martonosi††

† Dept. of Computer Science, Harvard University ‡ IBM T.J. Watson Research Center †† Dept. of Electrical Engineering, Princeton University

***Abstract*— Microprocessor research and development increasingly relies on detailed simulations to make design choices. As such, the structure, speed, and accuracy of microarchitectural simulators is of critical importance to the field. This paper describes our experiences in building two simulators, using related but distinct approaches.**

One of the most important attributes of a simulator is its ability to accurately convey design trends as different aspects of the microarchitecture are varied. In this work, we break down accuracy—a broad term— into two sub-types: *relative* and *absolute* accuracy. We then discuss typical abstraction errors in power-performance simulators and show when they do (or do not) affect the design rule choices a user of those simulator might make. By performing this validation study using the Wattch and PowerTimer simulators, the work addresses validation issues both broadly and in the specific case of a fairly widely-used simulator.

I. Introduction

Because the computer systems we build today are so complex, they are difficult to reason about; as a result, detailed simulations have become essential both for designing real systems and for evaluating research ideas in our field. Despite the crucial importance of simulators to the field of architecture, strategies for how best to structure and validate them are only infrequently discussed. This paper first briefly describes the anatomy of two related but distinct simulators: Wattch [2] and PowerTimer [3].

Wattch was one of the first tools to link a traditional architectural performance simulator with energy models. This link is accomplished by sending both static information describing the simulated microarchitecture and dynamic information about the run-time characteristics of applications to the energy models. PowerTimer [3] represents an IBM effort to apply the Wattch methodology to industrial performance simulators with energy models developed from circuits built for a high-performance commercial microprocessor.

The major difference between PowerTimer and Wattch is in the formation of energy models. PowerTimer's energy models are formed from empirical data collected from an existing, commercial microprocessor. These models are then technology-scaled to provide energy numbers for other design points. This style of abstraction was chosen for PowerTimer because it is particularly 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.

In Wattch, low-level analytical capacitance equations are generated for major nodes within common hardware structures. Thus Wattch takes a top-down approach using analytically derived capacitance equations from known structures of specific modules. PowerTimer is more of a bottom-up approach which uses existing, low-level circuit macros to generate higher-level energy models for microarchitectural units.

As systems grow more complex, simulator validation becomes increasingly crucial. Validation not only checks for bugs, but also helps to quantify a model's accuracy and applicability in different parts of the design space; users need to know how accuracy is affected by model abstractions used to provide superior simulation speed, to improve design space flexibility, or to speed model construction.

In this paper we discuss model accuracy within the context of Wattch and PowerTimer, two power-performance simulators. We show how different types of possible modeling abstractions and errors can affect the design choices one uses a simulator to evaluate. Our results distinguish cases when *absolute accuracy* is required, versus cases when the easier-to-achieve standard of *relative accuracy* is sufficient. This analysis can help power-performance simulator writers and users focus on areas that improve the likelihood of good design choices.

II. Simulator Design and Energy Modeling Strategies

In the Wattch simulator [2], and in other similar toolkits [8], [10], 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 can relate to, others are at a much lower (circuit or physical design) level. In the PowerTimer 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.

In PowerTimer, 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.

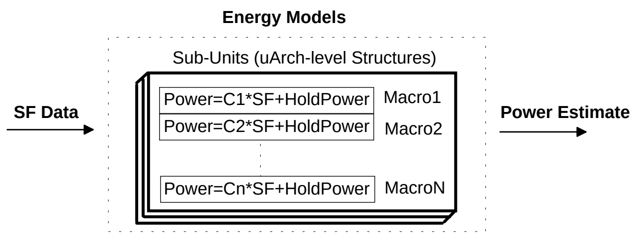

Fig. 1. PowerTimer Energy Models.

Figure 1 depicts the derivation of PowerTimer's 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. This analysis has been performed at the macro-level and in general multiple macros will combine to form a microarchitectural level structures corresponding to units within our performance model. For example, the fixed-point issue queue 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 microarchitectural structure is determined by summing the linear equations for each macro within that structure. We have generated power models from over 400 macro-level power equations corresponding to over 60 microarchitectural structures in our research simulator [5], [6].

In addition to the models that specify the power characteristics for particular super-macro (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 structures 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.

III. Types of Modeling Error

At the highest level, a model or simulator has metrics that the simulation is intended to produce. These may be aggregate numbers like execution cycles or total energy requirements for a program run. Or the numbers may be more fine-grained, such as distributions of the number of instructions ready to issue each cycle, the maximum and minimum instantaneous power, etc. We use the term absolute accuracy to refer to a simulator's ability—for a particular metric—to closely track the value measured by the 'real system' or a better model for that same metric. Relative accuracy, on the other hand, reflects that a simulator produces a range of results that properly reflect relative changes with design parameters, even if the absolute value of the result may not be perfect.

Achieving relative accuracy is much easier than achieving absolute accuracy, especially during the early-stages of the design process. This is because relative accuracy can be maintained despite errors in low-level technology parameters, incorrect assumptions about circuit-design styles, clocking network design methodologies, etc. Absolute accuracy is degraded due to all of these conditions.

A simulator with good relative accuracy provides quite a bit of useful information to an architect. For example, design trade-off studies with the goal of choosing architectural parameters to achieve an optimal power-performance efficiency can easily be performed. This is not to say that absolute accuracy is not important at all. For example, determining the true maximum wattage of a particular chip requires good absolute modeling accuracy. In contrast, relative accuracy can help designers deduce the design point that will produce the maximum wattage, but may not predict the actual wattage with sufficiently small error. In some cases, however, good relative accuracy combined with bounding techniques can help CPU designers with problems requiring some degree of absolute accuracy.

Previous work in power-performance model validation has mainly looked at validating models against more detailed information derived from lower-level tools. Comparing low-level capacitance values is one precise means of validating a power simulator. This method of validation has shown the models to be accurate within 10%, which is similar to what has been reported by the CACTI authors for analytical delay models [9] and later for analytical power models [7]. Amrutur and Horowitz have also studied analytical power and delay models for SRAMs [1].

IV. Robustness of Relative Accuracy

While simulator error is never a good thing, it is important to understand how different types of error influence (or not) the design process. Understanding the effects of different types of error gives guidance for how to interpret simulator results. These results give some insight into the robustness of the relative accuracy of the power models and demonstrate the extent to which a design tradeoff study can withstand error in the low level power models.

When performing a design tradeoff study, a methodology must first be established for deciding when to choose a particular design point over another design point. When viewing design tradeoff curves visually, we often would like to choose the "knee" of the curve so as to pick the point that is close to optimal without pushing too deeply into region of diminishing marginal returns. While this process is intuitive to designers, this paper quantifies it in order to be able to characterize whether a chosen design point is or is not acceptable. In particular, we propose the *acceptable range window* as a method to quantify the selection of design points from raw power/performance data.

The experiments in this paper quantify the amount of acceptable error within a power/performance simulator tolerated before different design points are chosen. The acceptable range window forms a group of points which meet the criteria for selection. Generally, we choose the lowest cost point within the acceptable range window for implementation.

Two different definitions of the acceptable range window are considered:

- 1) +/-R1% of target metric at optimal choice (*range1*). This chooses candidate design points that are within a percentage of the target metric.
- 2) +/-R2% of (worst_choice - optimal_choice) for this design study (*range2*). This is geared to be more selective in cases when the design tradeoff curve is so flat that many choices might satisfy *range1*. This definition adjusts for the fact that when optimal and non-optimal options are close together, the range of acceptable designs may be narrower.

In this paper we use Wattch as the baseline simulator to perform three representative studies and we consider acceptable range windows of defintion 1) with R1 equal to 5%. While all of the types of error that we consider disturb the absolute accuracy of the simulator, this study quantifies the effect on the relative accuracy of the simulator by investigating two design tradeoff study scenarios. These design tradeoff studies investigate energy-delay product for the number of RUU-entries and the size of the L1 Instruction Cache. We have also done additional experiments both with Wattch for the L1 Data Cache as well another power simulator, PowerTimer for both definitions of the acceptable range window [4]. This paper we show plots for the *vortex* application which tended to have the most interesting results (the most deviations). In the end of each subsection, we summarize the results for 5 of other the SPECint95 applications – *compress*, *gcc*, *go*, *ijpeg*, and *m88ksim*.

For each design tradeoff experiment, we check for overlap between the acceptable range windows of the baseline simulator and the modified simulator. Agreement implies that relative accuracy was maintained, and the correct design choice would be selected despite simulator error. We define a *critical deviation* if it would cause the designer to choose a point other than the least cost design point in the baseline acceptable range window – we will highlight these cases when they occur.

Example 1: Error in an Independent Unit: Consider first a simple scenario in which a designer is using a simulator to make a sizing choice about one of the hardware units, say the register file or the L1 caches. This experiment considers what happens if the designer uses a simulator which has error in the power estimate for a unit that is totally independent of the units for which design decisions are being made. For example, error in the ALU power model or the global clock power, is mostly independent of the power model for the RUU or the L1 caches. While the absolute accuracy of the model suffers quite a bit under these conditions, the relative accuracy of the model for a particular design study is typically less severely affected.

Figures 2 and 3 show two graphs each for the *vortex* application while varying the number of RUU entries and the I-Cache size. In each of the graphs there are five curves showing the power and energy-delay product trends while varying the microarchitectural parameters. The five lines labeled -.2x through .2x refer to the amount of error (additional power dissipation) inserted into the model. The amount of power added/subtracted is equal to the ratio given multiplied by the total chip power of the baseline case with an 80-entry RUU, and 64KB D- and I-Caches.

The first graph in each figure shows the power dissipation while varying both conditions. Since the additional power dissipation added in this experiment is independent of the RUU or cache power models, it does not affect the relative accuracy of this curve and only shifts the curves up and down by the corresponding amounts.

The second graph in each figure shows the energy-delay product while varying the microarchitectural parameters and the amount of error. The energy-delay product factors in the IPC, performance, for the various microarchitectural choices. Because of this, the energy-delay product curves are skewed by the IPCs of the various design points. Each of the energy-delay product figures also has several highlighted (circled) data points. These points represent designs that fit within the acceptable range window (with the *range1* criteria) for each curve. If the acceptable range window for the base case (without artificial error) matches the curves where artificial error exists, the we can say that relative accuracy was preserved. For example, in Figure 2 the same instruction cache would be chosen (64KB) even with the .2x and -.2x error conditions. On the other hand, an RUU of 16 entries would fit into the acceptable range window with -.1x and -.2x error conditions, whereas only a 32-entry RUU is in the acceptable range window with the base simulator.

For the other 5 SPECint95 applications that we considered, only *ijpeg* experienced a critical deviation in the acceptable range window. This occured for the RUU design study with -.2x error. When considering SPECint95 as an aggregate, there was no deviation from the acceptable range window with any of the error conditions that we considered.

Example 2: Error in Bitline Capacitance: A second major class of experimental inaccuracy in power models is error that occurs in a model that is used within many microarchitectural structures. For example, in Wattch, array structures such as the L1 instruction and data caches, the L2 cache, and the branch predictor tables are all modeled as instantiations of a single 'cache' power model. Error in the cache power model affects the power estimates for all of these units. In this example, we consider a specific scenario in which we have misestimated bitline capacitance. Since bitline capacitance estimates are used within the array structure models for caches and register files, an error in bitline capacitance affects all three of the microarchitectural parameters under study, as well as several independent structures.

Figures 4 and 5 show the power and energy-delay product for the *vortex* application while varying the number of RUU entries and I-Cache size. The five curves shown are similar to the ones in the previous section, but each of these curves shows a different ratio for the bitline capacitance scaling that was used. Again, significant deviations are difficult to see from these curves even with 0.6x and 1.4x scaling of the bitline capacitance estimates. In fact, the acceptable range windows were identical for both the RUU and I-cache design studies under all error conditions and there were no critical deviations for any of the SPECint95 applications.

Example 3: Error in Dependent Unit Scaling Factors: As a third example, we consider the effect of an error solely within the unit focused on by design study. Such an error might arise if, for example, we modeled a different circuit-design style than was actually used for that particular structure, or if an incorrect sub-banking scheme was assumed, etc. These experiments explore this error by scaling the power estimate for the individual structures (RUU and L1 Caches) by 1x through 2x.

Figures 6 and 7 show the power and energy-delay product for the *vortex* application while varying the number of RUU entries and I-Cache size. Each of the five curves again shows the energy-delay product as that particular unit's power estimate scales by 1x through 2x. This type of error clearly affects the design tradeoff study. As the amount of scaling increases, instead of just shifting the results, the curves begin to separate as more scaling is applied. The acceptable range windows begin to differ more for both the RUU and I-Cache design studies. For the RUU, the acceptable range window is between 16 and 64 entries, whereas with -.2x error it is between 32 and 80 entries and with .2x error it is between 16 and 48 entries.

The acceptable range windows highlight how and when this error disturbs the design tradeoff study. Even with 25% error, ie, a 1.25x scaling factor, there is very little change in the acceptable range windows for the three design tradeoff studies with these applications. However, *vortex* with the 1.5x and larger scaling factors resulted in critical deviation by choosing a design point with a smaller number of RUU-entries. There were no other critical deviations when considering the remainder of the SPECint95 applications.

Overall, the errors in this third example—those that specifically involve the unit under study— more likely to change the design choices made. This is because the additional scaling on the microarchitectural structure, in the absence of the scaling in other independent units, causes the structure in the tradeoff experiment to become a larger share of the overall chip power dissipation 'pie'.

Example 4: Error in Scaling Factors in PowerTimer: One of the potential sources of inaccuracy in PowerTimer is the scaling factors used to estimate the power changes while varying resource sizes. For many structures, power increases proportionally to the number of entries in a structure. For example, if the number of entries in an issue queue doubles, the power consumption doubles.

Figure 8 shows the power dissipation and energy-delay product while varying the size of the core by scaling all of the issue queues, renamers, and other major microarchitectural structures in the core while leaving the caches constant. This figure shows the baseline core, labeled "Core-1x", and larger and smaller cores named by the specified scaling factor for the scaled structures. Within the chart, we show scaling factors for the energy models rangings from 1.2x and 1.6x (sub-linear power scaling), 2x (linear scaling), and 2.4x and 2.8x (super-linear power scaling). These scaling factors are the amount that the power is increased while doubling the size of a structure – we extrapolate these scaling factors for the smaller amount of scaling performed here.

Because this experiment changes the size of many structures, it is likely that this experiment will be especially susceptible to varying scaling ratios. When considering the choice of core size with SPECint95 in aggregate, with sub-linear scaling of 1.2x, the larger Core-1.2x would be chosen as EDP optimal. At 1.6x scaling, the acceptable range window covers Core-0.8x through Core-1.2x, so Core-0.8x is the minimum cost choice. With 2x-2.8x scaling ratios, the baseline core is always chosen.

V. Conclusion

This paper describes the highlights of Wattch and PowerTimer, two power-performance modeling infrastructures. Compared to Wattch, PowerTimer uses a very similar methodology for power estimation, although its energy models are based on existing circuits for an industrial microprocessor. PowerTimer's models are best suited for exploring microarchitectural tradeoff decisions building off of this core microarchitecture.

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. 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.

This paper has considered the relative versus absolute accuracy of the architecture simulators we use. In particular, working with the Wattch power simulator, we have investigated some likely primary sources of error and demonstrated how design tradeoff studies can tolerate some error because *relative* inaccuracy need not affect the design point chosen.

When performing a design tradeoff study, it is most important to provide accurate power models for the unit under consideration in the study. Error in independent units does not affect the study, and errors that can affect multiple units could also have small disturbances because relative accuracy is maintained. However, errors that affect only the unit under study can lead to errors in the relative accuracy of the power model and incorrect design choices in some cases.

Both Wattch and PowerTimer are available for external use, and we encourage other researchers to assist in the further development and validation of these toolkits. Wattch is available at http://www.ee.princeton.edu/ dbrooks/wattch-form.html and PowerTimer is available by contacting Pradip Bose at IBM, pbose@us.ibm.com.

References

[1] B. Amrutur and M. Horowitz. Speed and power scaling of SRAM's. *IEEE Journal of Solid-State Circuits*, 35(2):175–185, 2000.

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

[3] D. Brooks, J.-D. Wellman, P. Bose, and M. Martonosi. Power-Performance Modeling and Tradeoff Analysis for a High-End Microprocessor. In *Power Aware Computing Systems Workshop at ASPLOS-IX*, Nov. 2000.
[4] D. M. Brooks. *Design and Modeling of Power-Efficient Computer Architectures*. PhD thesis, Princeton Univ., Nov. 2001.
[5] 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.
[6] M. Moudgill, J. Wellman, and J. Moreno. Environment for PowerPC microarchitecture exploration. *IEEE Micro*, 19(3):9–14, May/June 1999.
[7] G. Reinman and N. Jouppi. CACTI 2.0. In *WRL Research Report*, 1999.
[8] N. Vijaykrishnan, M. Kandemir, M. Irwin, H. Kim, and W. Ye. Energy-driven integrated hardware-software optimizations using simplepower. In *Proceedings of the 27th Annual International Symposium on Computer Architecture*, June 2000.
[9] S. Wilton and N. Jouppi. An Enhanced Access and Cycle Time Model for On-chip Caches. In *WRL Research Report 93/5, DEC Western Research Laboratory*, 1994.
[10] V. Zyuban. *Inherently Lower Power High Performance Superscalar Architectures*. PhD thesis, University of Notre Dame, March 2000.

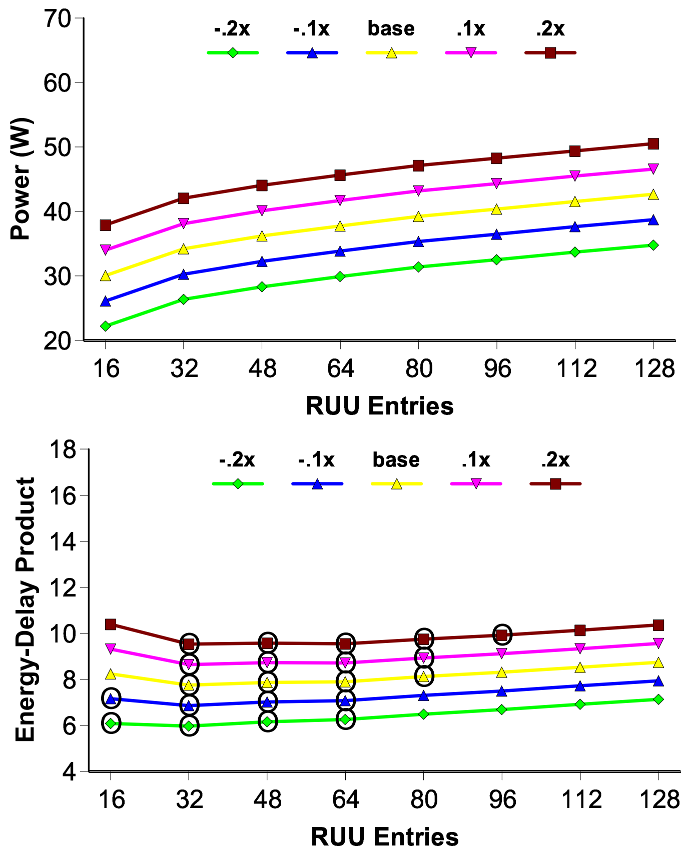

Fig. 2. Power and EDP for *vortex* varying indep. unit and RUU entries.

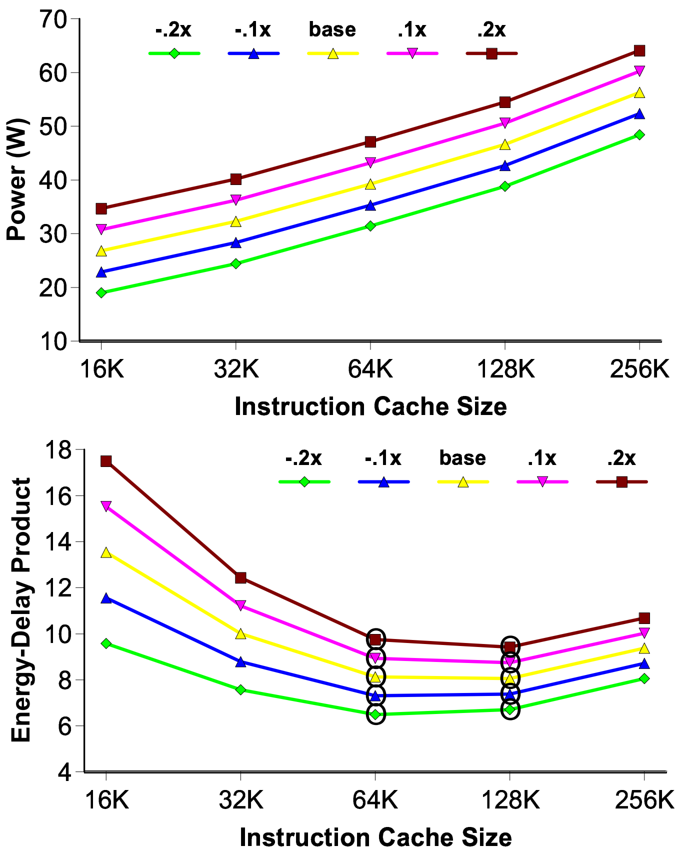

Fig. 3. Power and EDP for *vortex* varying indep. unit and I-Cache size.

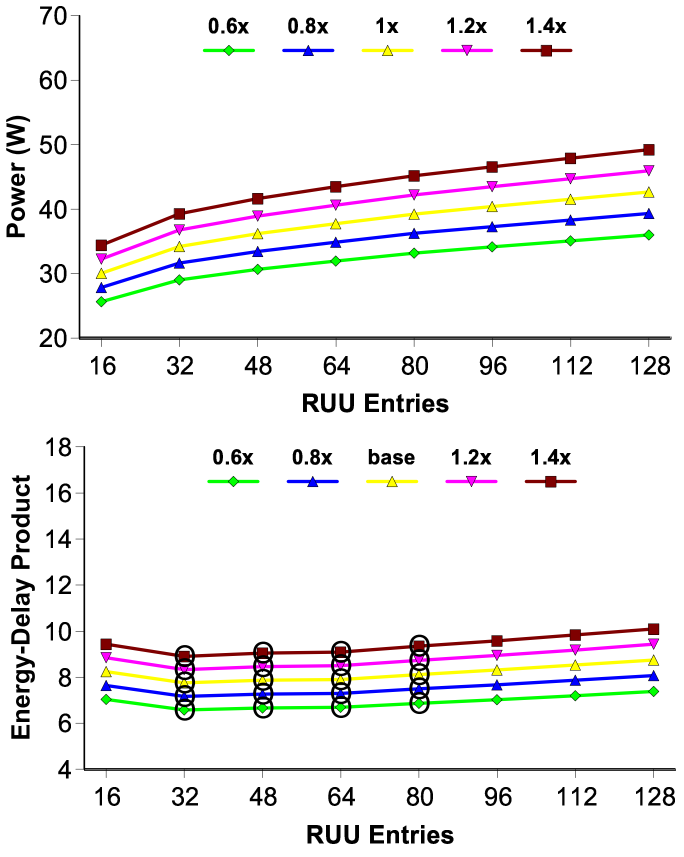

Fig. 4. Power and EDP for *vortex* varying bitline error and RUU entries.

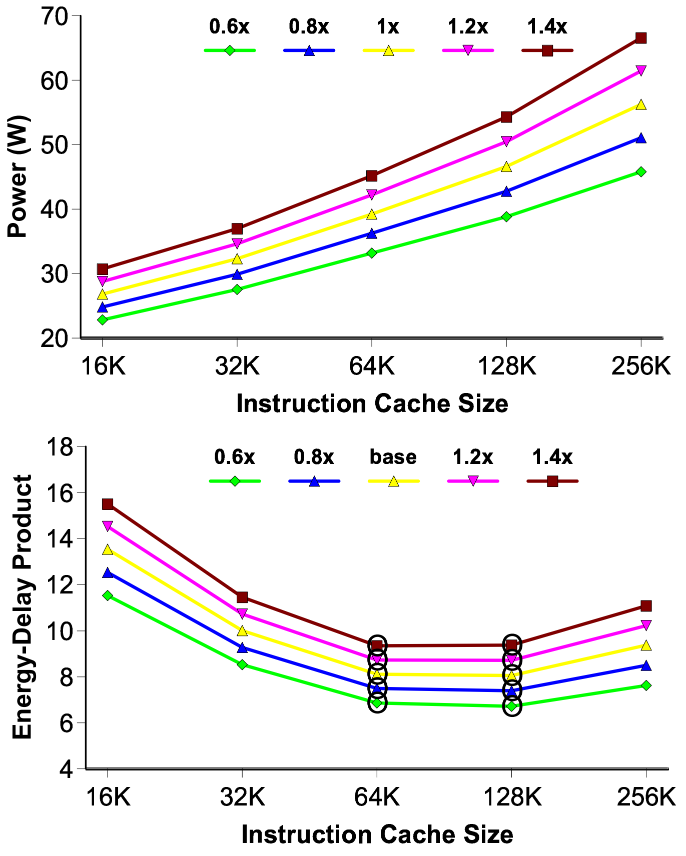

Fig. 5. Power and EDP for *vortex* varying bitline error and I-Cache size.

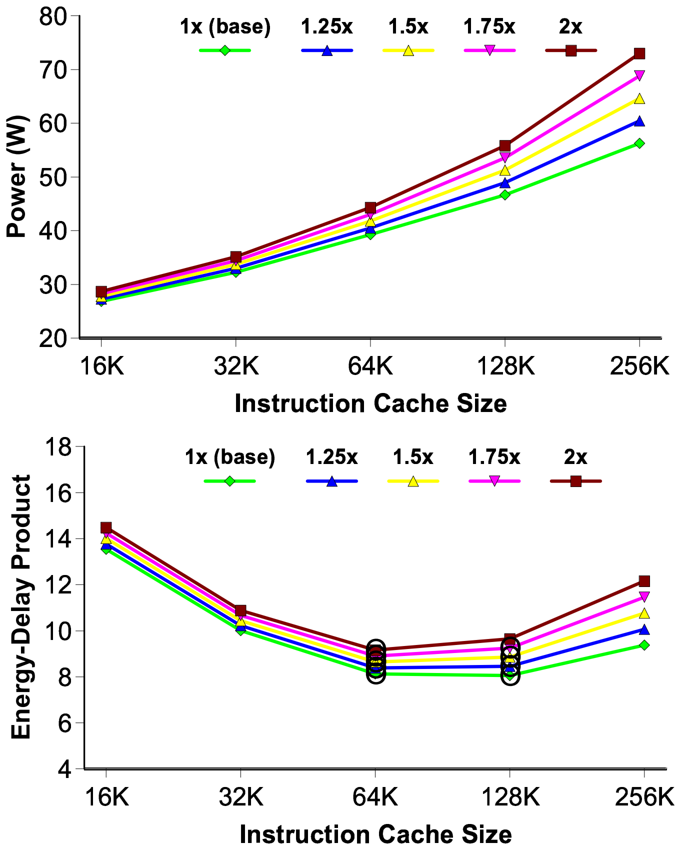

Fig. 7. Power and EDP for *vortex* varying ICache-sf and I-cache size.

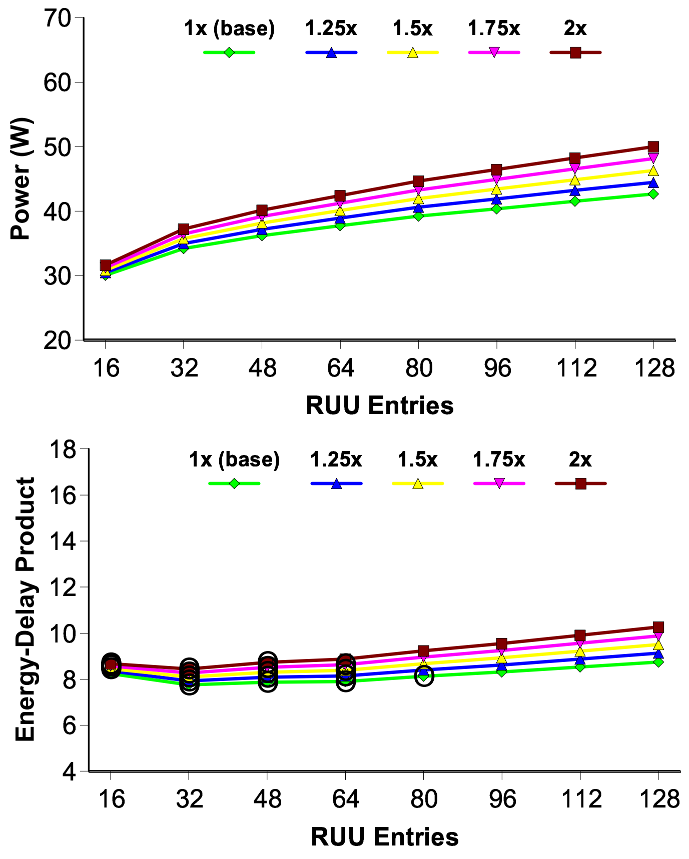

Fig. 6. Power and EDP for *vortex* varying RUU-scale-factor and RUU entries.

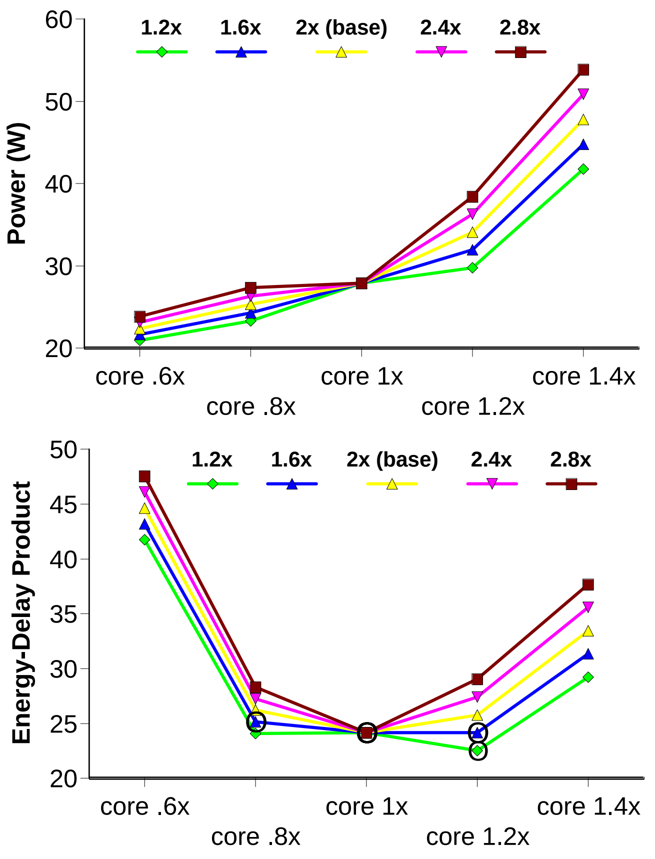

Fig. 8. Power and EDP for *SPECint* varying Core Parameters.