

HLSpower: Hybrid Statistical Modeling of the Superscalar Power-Performance Design Space

Ravishankar Rao¹, Mark H. Oskin², and Frederic T. Chong¹

¹ University of California at Davis

² University of Washington

Abstract. As power densities increase and mobile applications become pervasive, power-aware microprocessor design has become a critical issue. We present HLSpower, a unique tool for power-aware design space exploration of superscalar processors. HLSpower is based upon HLS [OCF00], a tool which used a novel blend of statistical modeling and symbolic execution to accelerate performance modeling more than 100-1000X over conventional cycle-based simulators.

In this paper, we extend the HLS methodology to model energy efficiency of superscalars. We validate our results against the Wáttch [BTM00] cycle-based power simulator. While minor second order power effects continue to require detailed cycle-by-cycle simulation, HLSpower is useful for large-scale exploration of the significant power-performance design space. For example, we can show that the instruction cache hit rate and pipeline depth interact with power efficiency in a non-trivial way as they are varied over significant ranges. In particular, we note that, while the IPC of a superscalar increases monotonically with both optimizations, the energy efficiency does not. We highlight the design capabilities by focusing on these non-monotonic contour graphs to demonstrate how HLSpower can help build intuition in power-aware design.

1 Introduction

The dramatic gains in microprocessor performance over the past decade have come at the cost of commensurate increases in power density. This increasing power density threatens to end our winning streak with Moore’s Law. Furthermore, power consumption is already a critical issue for mobile platforms, cost-conscious consumer products, and high-end server farms. Battery life is the driving factor in mobile platforms. Consumer products, such as set-top boxes, would like to avoid the costly and noisy addition of a cooling fan. High-end servers farms would like to avoid the construction of special building spaces to support the weight and electrical needs of high-powered cooling units.

We clearly entered the era of power-aware design. Several circuit-level design techniques have been employed to reduce power, including clock-gating, voltage scaling, dynamic clock regulation and asynchronous logic [CYVA96] [MDG93]. However, researchers have only recently begun to study the application and

architectural impact on power consumption. While these studies have been productive, they have relied upon the ingenuity of their inventors and brute-force technique of cycle-level simulation [MKG98] [PLL⁺00] [pac].

In the authors previous work [OCF00] a new approach to performance simulation was introduced. This approach relied upon the combination of statistical models and traditional cycle-by-cycle simulation to achieve rapid but accurate performance estimates. In this paper we extend this simulation technology to model power as well as performance.

To demonstrate the power of this new simulation technology we have undertaken a number of studies of architectural trade-offs. While most reveal only the intuitively obvious result that power-efficiency closely tracks instruction execution efficiency there are some unique exceptions. This paper will present four of these trade-off studies that do not follow the general power-performance trend.

In the next section we present an overview of current architectural power research and simulation technology, followed in Section 3 by a description of HLS, a statistical superscalar performance simulator. Next in Section 4 we describe the extensions to HLS to create HLSpower, a statistical power and performance modeling tool. In Section 5 we validate HLSpower against Wattach, and demonstrate a couple of non-obvious design trade-offs and their affect on power. Finally we conclude in Section 6.

2 Background

As the importance of power-aware microprocessor design has grown, so has the research in this area. Several prior work has focussed on specific parts or issues of the processor. For instance [BK198] and [KG97] have focused on caches, while power benefits of value-based clock gating in integer ALUs has been studied here [BM99].

Tools such as PowerMill [H⁺95] have been developed in the CAD community to perform circuit-level power estimation.

Early design Stage Power and performance simulator(ESP) [SNT94] does architecture-level estimation specifically for CMOS RISC processors. Given an object code, ESP executes it, and at each clock cycle, determines the active components, and sums up the current for them. The value of these currents is precomputed for different hardware structures.

SimplePower [YVK100] is a cycle-accurate RT level energy estimation tool. It uses transition sensitive energy models for energy estimation, coupled with analytical energy models for modelling memory system and on-chip buses. However, SimplePower currently models an in-order five stage pipelined datapath, assuming a perfect cache.

Cai-Lim [CL99] and the Wattach [BTM00] are two popular cycle-level simulator. These simulators provide power estimation through cycle-level simulation of a microprocessor. These simulators provide a significant improvement in speed over circuit-level simulation, at the expense of accuracy. However, being cycle-accurate, exploration of large design spaces is impractical, for benchmark ap-

plications. In the next section, we describe our statistical modeling techniques which allow performance estimation at significant speedups over cycle-level simulation. Our goal is to apply these techniques to power estimation, enabling the same design space explorations for power-aware design as we have previously achieved for performance-oriented design.

3 HLS

HLS is a hybrid simulator which uses statistical profiles of applications to model instruction and data streams. HLS takes as input a statistical profile of an application, dynamically generates a code base from the profile, and symbolically executes this statistical code on a superscalar microprocessor core. The use of statistical profiles greatly enhances flexibility and speed of simulation. For example, we can smoothly vary dynamic instruction distance or value predictability. This flexibility is only possible with a synthetic, rather than actual, code stream. Furthermore, HLS executes a statistical sample of instructions rather than an entire program, which dramatically decreases simulation time and enables a broader design space exploration which is not practical with conventional simulators. In this section, we provide a brief overview of the HLS approach. Details are available in [OCF00].

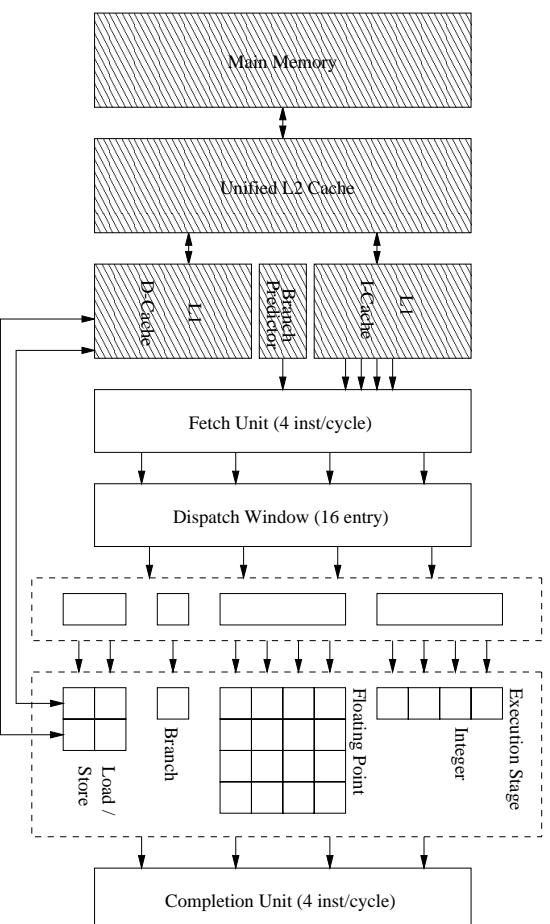


Fig. 1. Simulated Architecture

The key to the HLS approach lies in its mixture of statistical models and structural simulation. This mixture can be seen in Figure 1, where components

of the simulator which use statistical models are shaded in gray. HLS does not simulate the precise order of instructions or memory accesses in a particular program. Rather, it uses a statistical profile of an application to generate a synthetic instruction stream. Cache behavior is also modeled with a statistical distribution.

Once the instruction stream is generated, HLS symbolically issues and executes instructions much as a conventional simulator does. The structural model of the processor closely follows that of the SimpleScalar tool set [BA97], a widely used processor simulator. This structure, however, is general and configurable enough to allow us to model and validate against a MIPS R10K processor [OCF00].

The overall system consists of a superscalar microprocessor, split L1 caches, a unified L2 cache, and a main memory. The processor supports out-of-order issue, dispatch and completion. It has five major pipeline stages: instruction fetch, dispatch, schedule, execute, and complete. The similarity to SimpleScalar is not a coincidence: the SimpleScalar tools are used to gather the statistical profile needed by HLS.

Benchmark	Simple- scale IPC	HLS IPC	HLS IPC σ	Error
perl	1.27	1.32	0.05	4.2%
compress	1.18	1.25	0.06	5.5%
gcc	0.92	0.96	0.03	3.9%
go	0.94	1.01	0.04	6.8%
jpeg	1.67	1.73	0.06	3.9%
h	1.62	1.50	0.06	7.2%
m8ksinn	1.16	1.14	0.03	1.5%
vortex	0.87	0.83	0.03	5.1%

Table 1. Correlation between SimpleScalar and HLS Simulators for SPECint95 (ref input)

Table 1 lists the experimental results from executing the SPECint95 benchmarks on both the reference inputs in SimpleScalar versus the statistical simulator. Across the board, we note that the error (the difference between the two simulation techniques) is not more 7.2%. This is for benchmarks with significantly different cache behaviors and code profiles.

4 HLSpower

HLS can be extended to measure both performance and power consumption. This extension into HLSpower appears straightforward. By adding activation counters to the structural components of HLS, power can be estimated as with

conventional cycle-level simulators. Several components of HLS, however, are statistical rather than structural. In particular, the branch prediction table and the caches are modeled by their statistical hit rate and accuracy, not their structure. This modeling gave us both speed and flexibility in HLS, but is problematic in power modeling.

Fortunately, we note that the accuracy and hit rate of the branch table and the caches are independent of most other parameters in HLS. We can use the SimpleScalar cycle-level simulator to produce a mapping of statistical parameter to actual structure. Once this mapping is created, it will apply to a wide range of design spaces, preserving the advantages of statistical simulation inherent in HLS.

For instance, we first obtained a plot of the cache hit rate and its size. In order to vary the cache hit rate in our statistical model, we change the cache configuration in our power models. Similarly, in order to change the branch prediction accuracy, a mapping of its size and the prediction rate was obtained. From this plot, to vary the prediction accuracy in our statistical model, we change the size in the power model.

4.1 Validation Results

We apply HLSpower towards modeling the Wattach simulator, a previously published cycle-by-cycle power simulator based upon SimpleScalar. HLSpower is fully programmable in terms of queue sizes and inter-pipeline stage bandwidth; however, the baseline architecture was chosen to match the baseline SimpleScalar architecture. The various configuration parameters are summarized in Table 2. Table 3 compares HLSpower to Wattach. We can see that HLSpower can model the processor core within 16% of Wattach and that total power (including caches and branch tables) is within 22%. While these tolerances are not useful for detailed microarchitectural evaluation, we shall see in the next section that HLSpower predicts design space trends which correlate well with Wattach, yet are orders-of-magnitude faster to obtain.

As another level of comparison, we present a comparison of our models with published results of a high-end processor. We compare our model with Wattach and Alpha 21264[GBJ98]. Table 4, gives the results of relative power breakdowns of different hardware structures. As seen from the table, our results track Alpha and Wattach closely. The results presented are the average of different benchmarks.

5 Results

In this section, we give two examples of contour plots that can be generated using HLSpower to facilitate power-aware design space exploration. In these contour plots, we vary two design parameters, one on each axis, and plot average power efficiency. We compare plots generated with HLSpower and ones generated with Wattach for Perl, to provide additional validation of our model.

Parameter	Value
Instruction fetch bandwidth	4 inst.
Instruction dispatch bandwidth	4 inst.
Dispatch window size	16 inst.
Integer functional units	4
Floating point functional units	4
Load/Store functional units	2
Branch units	1
Pipeline stages (integer)	1
Pipeline stages (floating point)	4
Pipeline stages (load/store)	2
Pipeline stages (branch)	1
L1 I-cache access time (hit)	1 cycle
L1 D-cache access time (hit)	1 cycle
L2 cache access time (hit)	6 cycles
Main memory access time (latency+transfer)	34 cycles
Fetch unit stall penalty for branch mis-predict	3 cycles
Fetch unit stall penalty for value mis-predict	3 cycles

Table 2. Simulated Architecture configuration

Benchmark	Wattch		HLS-Power		Error	
	total	core only	total	core only	total	core only
perl	1.0	0.71	0.90	0.67	9.76%	4.4%
compress	1.32	0.91	1.12	0.87	15.15%	4.93%
gcc	0.92	0.66	0.76	0.64	17.39%	3.05%
jpeg	1.29	0.94	1.10	0.83	14.5%	11.21%
li	1.17	0.81	1.12	0.82	4.27%	-1.61%
m88ksim	1.13	0.82	1.02	0.77	9.74%	5.84%
vortex	0.96	0.67	0.75	0.56	22.2%	16.49%
go	0.85	0.62	0.77	0.59	9.41%	5.61%

Table 3. Correlation between Wattch and HLS power for SPECint95 (normalized to Perl).

Hardware Structure	Alpha 21264	Wattch	HLSpower
Caches	16.1%	15.3%	15.7%
Out-of-Order Issue Logic	19.3%	20.6%	23.3%
Memory Management Unit	8.6%	11.7%	9.2%
Integer Exec. Unit	10.8%	11.0%	6.5%
Floating Point Exec. Unit	10.8%	11.0%	6.7%
Total Clock Power	34.4%	30.4%	38.7%

Table 4. Comparison between HLSpower, Wattch and Reported breakdowns for Alpha 21264

Power efficiency, in all our graphs has been normalized to the baseline architecture. Thus an efficiency greater than one, only indicates that it is more than that of the baseline architecture.

Although exact measurements of time were not performed, a simple analysis indicates the considerable speedup HLSpower achieves over Watrch. The performance simulator HLS achieves about 100-1000x speedup over SimpleScalar. Watrch has an overhead of power calculations over SimpleScalar, and HLSpower has about the same overhead over HLS. Thus it is fairly straightforward to see that HLSpower is about 100-1000x faster than Watrch.

5.1 I-Cache vs. Branch Prediction

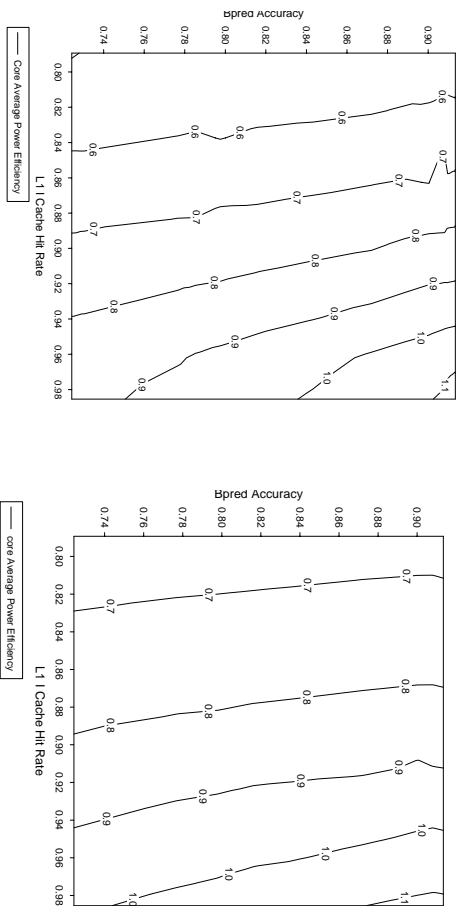


Fig. 2. HLS-Power (left) versus Watrch (right) (processor-core only)

Our first set of graphs compare I-cache hit rate with branch prediction accuracy. A contour plot of the power efficiency, as the two parameters are varied, is generated. We plot the core power efficiency and the total power efficiency separately. The cache hit rate, and the branch prediction accuracy were varied by changing their sizes in the power models, as obtained from the mapping (explained earlier). From Figure 2, as we might expect, power efficiency in the processor core increases as either the hit rate or the branch prediction accuracy increases. We can further see that the trends predicted by HLSpower are qualitatively similar to those found with Watrch.

We can see in Figure 3, however, that total processor efficiency decreases as I-cache hit rate increases, especially if branch prediction accuracy is not high.

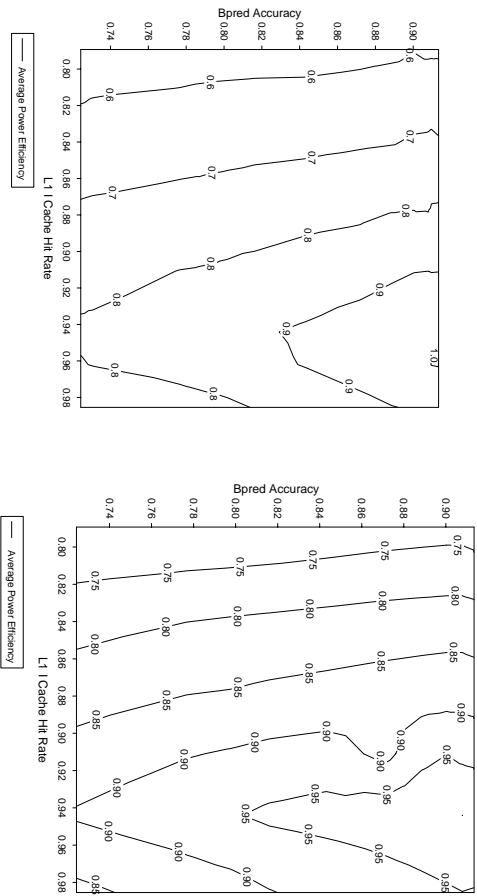


Fig. 3. HLS-Power (left) versus Wattach (right) (caches included)

This non-monotonic behavior may seem initially surprising, but is actually a fairly intuitive tradeoff. If the branch predictor is not performing well, then the power used in the I-cache to fetch unneeded instructions actually decreases power efficiency. Once again, we can see that the Wattach results agree qualitatively with HLSpower.

5.2 I-Cache vs. Pipelining

Our next set of graphs investigates a less surprising result. We compare I-cache hit rate with pipeline depth. Again, we plot the processor core and the total power separately. In Figure 4, we can once again see non-monotonicity if we consider total power efficiency. The effect comes from the fact that as I-cache hit rate increases, the size and power of the cache must increase to obtain that hit rate. For small pipelines, this effect is more pronounced. As the number of pipeline stages increase, more instructions are need and a higher hit rate is more worthwhile.

6 Conclusion

Our goal has been to provide a proof-of-concept that the HLS statistical approach can be applied to power simulation for design space exploration. Our results indicate that statistical power estimation is, indeed, feasible and that interesting insights can be quantified using contour plots with large variation of

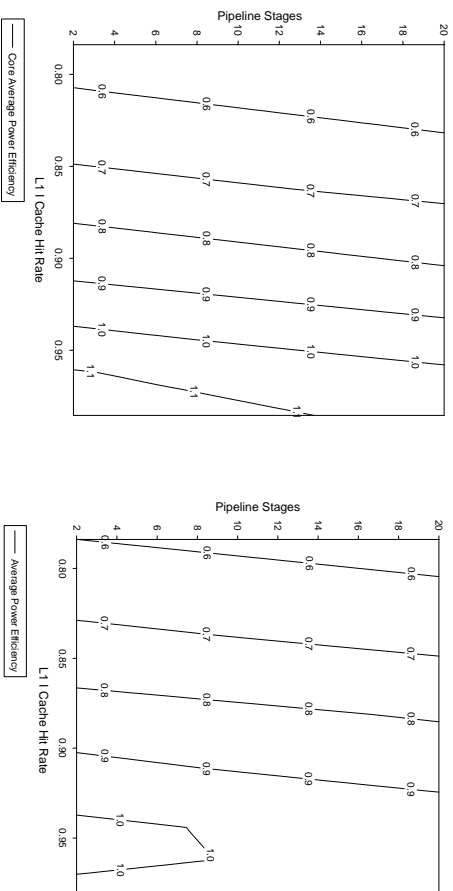


Fig. 4. HLSpower pipeline stages vs. I-cache hit rate – processor core (left) and total power (right)

parameters and previously difficult numbers of data points. We note that cycle-level power simulation continues to improve [DLCDD01], and these models can be incorporated into HLSpower in the future.

References

- [BA97] D. Burger and T. Austin. The SimpleScalar tool set, v2.0. *Comp Arch News*, 25(3), June 1997.
- [BK198] R.I. Bahar, T. Kellner, and M. Irwin. Power and performance tradeoffs using various caching strategies. In *Proceedings of the International Symposium on Low-Power Electronics and Design*, 1998.
- [BM99] D. Brooks and M. Martonosi. Dynamically exploiting narrow width operands to improve processor power and performance. In *Proceedings of the 5th International Symposium on High-Performance Computer Architecture*, January 1999.
- [BTM00] David Brooks, Vivek Tiwari, and Margaret Martonosi. Watrch: A framework for architectural-level power analysis and optimizations. In *Proceedings of the 27th Annual International Symposium on Computer Architecture*, pages 83–94, Vancouver, British Columbia, June 12–14, 2000. IEEE Computer Society and ACM SIGARCH.
- [CL99] G. Cai and C. H. Lim. Architectural level power/performance optimization and dynamic power estimation. Cool Chips Tutorial colocated with MICRO32, November 1999.
- [CYVA96] Anantha Chandrakasan, Isabel Yang, Carlin Vieri, and Dimitri Antoniadis. Design considerations and tools for low-voltage digital system design. In

- 33rd Design Automation Conference (DAC'96)*, pages 113–118, New York, June 1996. Association for Computing Machinery.
- [DLCD01] Ashutosh Dhodapkar, Chee How Lim, George Cai, and W. Robert Dasch. TEM²P²EST: A thermal enabled multi-model power/performance ESTimator. *Lecture Notes in Computer Science*, 2008:112–125, 2001.
- [GBJ98] M. Gowan, L. Brio, and D. Jackson. Power considerations in the design of the alpha 21264 microprocessor. In *35th Design Automation Conference*, 1998.
- [H⁺95] C. X. Huang et al. The design and implementation of PowerMill. In *Proc. Int. Workshop Low-Power Design*, pages 105–110, April 1995.
- [KG97] M. B. Kamble and K. Ghose. Analytical energy dissipation models for low power caches. In *Proceedings of the International Symposium on Low-Power Electronics and Design*, 1997.
- [MDG93] J. Monteiro, S. Devadas, and A. Ghosh. Retiming sequential circuits for low power. In Michael Lightner, editor, *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pages 398–402, Santa Clara, CA, November 1993. IEEE Computer Society Press.
- [MKG98] S. Manne, A. Klausner, and D. Grunwald. Pipeline gating: Speculation control for energy reduction. In *Proceedings of the 25th Annual International Symposium on Computer Architecture (ISCA-98)*, volume 26,3 of *ACM Computer Architecture News*, pages 132–141, New York, June 27–July 1 1998. ACM Press.
- [OCF00] Mark Oskin, Frederic T. Chong, and Matthew Farrens. HLS: Combining statistical and symbolic simulation to guide microprocessor designs. In *27th Annual International Symposium on Computer Architecture*, pages 71–82, 2000.
- [pac] proceedings of the second workshop on power-aware computer systems. To appear.
- [PLL⁺00] Gi-Ho Park, Kil-Whan Lee, Jang-Soo Lee, Jung-Hoon Lee, Tack-Don Han, Shin-Dug Kim, Moon-Key Lee, Yong-Chun Kim, Seh-Woong Jeong, Hyung-Lae Roh, and Kwang-Yup Lee. Cooperative cache system: A low-power cache structure for embedded processor. In Anonymous, editor, *Cool Chips III: An International Symposium on Low-Power and High-Speed Chips, Kikai-Shinko-Kaikou, Tokyo, Japan April 24–25, 2000*, 2000.
- [SNT94] T. Sato, M. Nagamatsu, and H. Tago. Power and performance simulator: Esp and its application for 100 mips/w class risc design. In *Low Power Electronics, 1994. Digest of Technical Papers*, pages 46–47, 1994.
- [YVKI00] W. Ye, N. Vijaykrishnan, M. Kandamir, and M. J. Irwin. The design and use of simplepower: a cycle-accurate energy estimation tool. In *Design Automation Conference 2000*, pages 340–345, 2000.