An Integrated Performance and Power Model For Superscalar Processor Designs

Yongxin Zhu
School of Computing
National University of Singapore
Singapore 117543
Tel: 65-68741178
Fax: 65-67794580
Email: zhuyx@comp.nus.edu.sg

Weng-Fai Wong
School of Computing
National University of Singapore
Singapore 117543
Tel: 65-68746902
Fax: 65-67794580
Email: wongwf@comp.nus.edu.sg

Ştefan Andrei
Singapore-MIT Alliance
National University of Singapore
Singapore 117543
Tel: 65-68741178
Fax: 65-67794580
Email: andrei@comp.nus.edu.sg

Abstract— On current superscalar processors, performance and power issues cannot be decoupled for designers. Extensive simulations are usually required to meet both power and performance constraints. This paper describes an integrated performance and power analytical model. The model's performance and power results are in good agreement with detailed simulations, previous models and physically measured results. For designers, the model enables quick and flexible explorations into a subset of even entire huge parameter space of more than 15 workload and architectural parameters plus leakage power, feature sizes, clock and voltage.

I. Introduction

Following a recent vigorous growth of hand-held and portable devices, power efficient processors have stepped into the center of the stage of the EDA industry and research. The performance of embedded processors is constrained by stringent power budgets in terms of maximum power consumption and battery life. Therefore, tradeoffs between power and performance have received plenty of attention. Among these efforts, extensive simulations [4], [14] are the most common approach.

To fasten the design process, some unified analytical modelling approaches were proposed to address both the performance and power issues. Brooks et. al. [3] introduced a measured metric called *power-performance efficiency* to quantify the effectiveness of a processor servicing a task within a limit on its power consumption. Conte et. al. [6] separated architectural and technology components of dynamic power, and used a near-optimal search to tailor a processor design to different benchmarks. While models of Conte and Srinivasan et. al. [9] covered high level statistics and optimization of pipeline stages, many details of processor components were skipped.

The above unified methods did not address another important power issue, *static power dissipation*, assuming the static power is less than the dynamic power by one or more order of magnitude. With the downward scaling of technology nodes, the static power has grown to the same order of magnitude as the dynamic power [7], and it must be considered in current designs. The major part of the static power is *leakage power* [1].

II. Our Contributions and Related Work

To arrive at a more realistic system-wide view of the power-performance trade-off, we start with a good performance model of superscalar processors [15]. That performance model was validated against actual performances on in-order-issue SPARC processors and showed an average error of 5.1%. In that model, nearly all major processor components were modelled.

In this paper, we extend that performance model [15] by considering out-of-order issue processors. For the out-of-order-issue SimpleScalar simulated processor [2], an average error of 5.9% between the predicted performance and the simulated one was achieved over the SPEC2000 benchmarks.

To study the dynamic power, we link the performance metrics to the dynamic capacitance of each processor components, thereby deriving the power consumption for each component and finally the whole processor. The validation using SimWattch [4] is on average within 10.9% accuracy. Our model's average result agrees with the measured one reported by Synopsys Power Compiler with a power library from Virginia Tech [10]. Our average result also agrees with that of the Berkeley Advanced Chip Performance Calculator (BACPAC) [12].

We also consider the static power in our integrated model. Khouri and Jha [7] summarized the ratios of the static power over the total power. We use these ratios to accounts for the static power. Our model's projected maximum total power of UltraSPARC-III processor [11] agrees with the published one.

Our model enables distinct views of factors involved in co-optimization of power and performance. It allows us to quantify the impact by any individual factor. In terms of efficiency, it takes us just two hours to explore a design space containing 487 million design points. With a slight change in the problem formulation, it is also possible to use the design space exploration to yield the configuration with the lowest energy consumption needed to complete a computation task within a certain performance constraint.

We present our performance model in Section III, our power model in Section IV, validation results in Section V. We handle the concrete co-optimization tradeoffs in Section VI. This is followed by a conclusion.

III. Performance Model

To model a generic superscalar processor, we used a *network* of *multiple-class multiple-resource* (MCMR) systems. Each stage of the pipelines contributes to the final results of the processor. The lowest throughput of all the pipeline stages is the bottleneck of the entire processor and determines the maximum possible throughput of the processor. We shall now recall the main results of the performance model [15], then describe our extensions by considering out-of-order issue processors.

The throughput of the processor Θ is the minimum of the service rates of *decoder unit* (μ_{dec}), *central window* (μ_{win}), and *retirement unit* (μ_{ret}):

$$\Theta = \min\{\mu_{dec}, \mu_{ret}, \mu_{win}\} \ . \tag{1}$$

μ_{dec} denotes the average decoding rate without overflow in the central window. Let W_{dec} denote the decoding width; $\overline{I}_{br}$ be the average number of (non-branch) instructions between two branch instructions; T_{br} be the misprediction penalty time; $p_{ins,miss}$ be the instruction cache miss ratio; $t_{ins,pen}$ be the instruction cache miss penalty time; and $p_{br,prtd}$ be the probability of a correct branch prediction. If $\overline{I}_{br} < W_{dec}$,

$$\mu_{dec} = \frac{C_1}{C_2 + C_3 \times t_{ins,pen} \times p_{ins,miss}} \ , \tag{2}$$

where C_1, C_2, and C_3 are linear functions of $\overline{I}_{br}$, T_{br}, W_{dec}, and $p_{br,prdt}$. The rest cases can be looked up in [15].

μ_{ret} denotes the retirement rate under an in-order retirement policy. Its parameters are as follows. W_{ret} is the the maximum number of instructions that can be retired per cycle. $\overline{D}$ is the average dependence distance (inclusive of one of the instruction in the dependence) between two instructions that have a data dependence relation. For $\overline{D} < W_{ret}$, μ_{ret} is given below, the rest cases can be referred in [15]:

$$\mu_{ret} = (2 \times \overline{D})/(1 + T_{dep}) \ , \tag{3}$$

where T_{dep}, the average time for an antecedent instruction to pass through the functional units, is:

$$T_{dep} = [\sum_{i}^{type} (t_i \times S_i)] \times (1 + \overline{P}_{dep}) \ , \tag{4}$$

where $type \in \{ieu, fpu, lsu, br\}$ is the set of types of functional units, namely the integer execution unit, the floating point unit, the load store unit and the branch unit. $S_i \in [0, 1]$ is the fraction of the total number of instructions that is executed on functional unit i, and t_i is the average service time of each functional unit of type i. Typically, $t_{ieu} \in \{1, 2\}$, $t_{fpu} \in \{3, ..., 6\}$, $t_{lsu} = p_{d,prtd} + t_{dat,pen} \times (1 - p_{d,prtd})$ and $t_{br} = p_{i,prtd} + t_{ins,pen} \times (1 - p_{i,prtd})$. The parameters $p_{d,prtd}$, $p_{i,prtd} \in [0, 1]$ represent the probabilities of the data cache prediction and the instruction cache prediction, respectively.

For μ_{win}, We extend previous studies to out-of-order issue(not considered in [15]) processors with multiple instruction types or classes(not considered in [8]).

On out-of-order-issue processors, any independent and ready instruction in the instruction window may be dispatched to an available functional unit. Hence, μ_{win} is the total sum of service rates of functional units μ_t where t is an instruction type, that is $\mu_{win} = \mu_{ieu} + \mu_{fpu} + \mu_{lsu} + \mu_{br}$.

Given $\rho_{k,t}(Z_{win})$ as the probability that k instructions of type t are issued from the window of size Z_{win} and F_t representing the number of functional units of type t. then:

$$\mu_{win} = \sum_{t}^{type} \sum_{k=1}^{F_t} (\rho_{k,t}(Z_{win}) \times k) \ , \tag{5}$$

and $\rho_{k,t}(Z_{win}) = \mathcal{P}_{k,t}(Z_{win}) \times \phi_{pipe,t}(k)$, where $\mathcal{P}_{k,t}(Z_{win})$ is the probability that k independent instructions are extracted from Z_{win} instructions and $\phi_{pipe,t}(k)$ is the probability that at least k pipeline units of type t are available.

$$\begin{aligned} \text{So, } \mathcal{P}_{k,t}(Z_{win}) = {} & \mathcal{P}_{k-1,t}(Z_{win} - 1) \times p_t^{(Z_{win}-1)} + \\ & + \mathcal{P}_{k,t}(Z_{win} - 1) \times (1 - p_t^{Z_{win}-1}) \ , \end{aligned} \tag{6}$$

and the initial cases are $\mathcal{P}_{1,t}(1) = 1$ and $\mathcal{P}_{i,t}(j) = 0$, $\forall\ i > j$. The variable $p_t = p \times S_t$ represents the probability of an independent instruction of type t, and p is the overall probability of instruction independence. In practice, $k \in \{1, ..., 8\}$ and $Z_{win} \in \{2, ..., 20\}$.

$$\phi_{pipe,t}(k) = \sum_{j=k}^{F_t} \binom{F_t}{j} q_t^j (1 - q_t)^{F_t - j} \ , \tag{7}$$

where $q_t = q \times S_t$ is the probability that an instruction of type t will be issued to a functional unit of type t, q is the overall probability of functional units availability for a instruction ready to issue. The notation $\binom{F_t}{j}$ stands for $\frac{F_t!}{j!(F_t - j)!}$, where $j! = 2 \times 3 \times \ ... \ \times j$.

IV. Power Model

The power consumption of a resource consists of a dynamic and a static component, i.e., $\pi_{tot,res} = \pi_{static,res} + \pi_{dyn,res}$. The static portion is given by $\pi_{static,res} = I_{static,res} \times V_{dd}$. The leakage current $I_{static,res}$ is an exponential function of threshold voltage V_t (in mV) by Sylvester and Keutzer [13]:

$$I_{static,res} = 10 \times \omega \times 10^{-V_t/95} \ , \tag{8}$$

where ω is the device width in micro meter.

For any technology node, the static power takes a usually stable portion of the total power. Khouri and Jha [7] summarized the ratios of the static power over the total power based on 6 different circuits. We use the averaged ratios in TABLE I.

For the dynamic power component, we model dynamic power as a traditional function of *dynamic capacitance* ($\mathcal{C}_{res}$), the *supply voltage* (V_{dd}) and the *clock frequency* (Ω):

$$\pi_{dyn,res} = \mathcal{C}_{res} \times V_{dd}^2 \times \Omega \ . \tag{9}$$

The accesses to each resource are obtained from the simulators. With total dynamic capacitance and number of accesses of a resource, we can obtain the *dynamic capacitance per access to the resource* ($\mathcal{C}_{a,res}$) for each benchmark. This enables us to establish a link between the performance model and the power model. We also need the *average number of accesses to the resource per request (instruction)*, denoted by $N_{a,req,res}$. Then the *number of accesses a resource services each cycle* $N_{a,res}$ can be obtained as: $N_{a,res} = \mu_{res} \times N_{a,req,res}$.

TABLE I. The Proportions of Leakage Power in Total Power.

Tech.	stat. pwr. /tot. pwr. without leakage opti.	stat. pwr./tot. pwr. with leakage opti.	V_{dd}
0.35μm	9.8%	6.6%	3.3
0.18μm	22.6%	11.7%	1.8
0.13μm	43.4%	26.9%	1.5
0.10μm	48.1%	25.5%	1.2
0.07μm	56.2%	25.1%	0.9

The link to power is expressed as the *dynamic capacitance per resource per cycle*, $\mathcal{C}_{res,cycle} = N_{a,res} \times \mathcal{C}_{a,res}$. We assume $\mathcal{C}_{res}$ to be equal to $\mathcal{C}_{res,cycle}$, so $\mathcal{C}_{res} = \mu_{res} \times N_{a,req,res} \times \mathcal{C}_{a,res}$. With the total dynamic capacitance per resource or $\mathcal{C}_{res}$, we can obtain the power consumption in (9). Note that μ_{res} is obtained in the performance model. The total power of a processor is the sum of the power consumption by each resource/component. There are two new service rates of instruction cache and data cache, μ_{icache} (the amount of decoder unit's output plus L1 instruction cache miss ratio) and μ_{dcache} (the load/store unit's output plus L1 data cache miss ratio). More precisely, $\mu_{icache} = (1 + p_{ins,miss}) \times \mu_{dec}$ and $\mu_{dcache} = (1 + p_{dat,miss}) \times \mu_{lsu}$. The analysis of μ_{dec} and μ_{lsu} is found in Section II. The dynamic power costs of all resources form the total dynamic power:

$$\pi_{dyn,tot} = \pi_{win} + \pi_{ret} + \pi_{dec} + \pi_{ieu} + \pi_{fpu} + \pi_{lsu} + \pi_{br} + \\ + \pi_{icache} + \pi_{dcache} \ . \quad (10)$$

V. Validation of the Model

The performance model's parameters are obtained from simulation traces of benchmarks. These parameters characterize the benchmark. Except for the two miss ratios for the instruction and data caches that can be looked up in [5], all the rest ones are independent of the architectural features of the processor being modelled and so can be obtained in a single run. The inputs to the performance model are given in TABLE II.

This extended performance model for out-of-order issue processors is validated with SimpleScalar out-of-order issue processor. We use five benchmarks from the SPEC2000 suite, namely `256.bzip2`, `183.equake`, `181.mcf`, `177.mesa` and `175.vpr`. Although we have only taken the level one cache into consideration in validating the current performance model, a small average error of 5.9% was still obtained.

To validate our power model, we use a Sim-Wattch simulator customized with parameters from SimplePower. The simulator configuration is based on the 0.25μm process technology for a processor running at 2.5 volt with a clock frequency of 600MHz. For each component of the processor, the capacitance is obtained by either using the same empirical formulas used by Sim-Wattch or by means of summing up the bit stream changes in SimplePower. The relevant primitives are listed in TABLE III.

Our architectural analysis yields values of $N_{a,req,res}$ for different resources: $N_{a,req,win} = 6$, $N_{a,req,regfile} = 2$ and

TABLE II. Benchmark Characts. for the Performance Model.

Bench.	bzip2	equake	mcf	mesa	vpr
S_{ieu}	45.7%	26.3 %	39.4 %	42.2 %	43.6 %
S_{fpu}	0.0%	15.3%	0.0 %	7.0 %	5.6 %
S_{br}	15.9%	6.1%	2.7 %	1.0 %	1.0 %
S_{lsu}	28.5%	41.5%	48.2%	53.4 %	53.4 %
$\overline{D}$	1.996	1.955	2.016	1.873	1.911
$\overline{I}_{br}$	7.26	6.69	3.65	4.18	4.74
$\overline{P}_{dep}$	0.562	0.504	0.620	0.425	0.589
T_{dep}	1.972	2.403	2.085	2.248	2.187
p	0.438	0.4962	0.3802	0.5755	0.5178
q	0.991	0.975	0.995	0.95	0.983
$p_{ins,miss}$	0.0110	0.0343	0.0038	0.0296	0.0067
$p_{dat,miss}$	0.0227	0.0552	0.1589	0.0221	0.0820

TABLE III. Capacitance (in 10^{-10} farad) Primitives.

Bench.	bzip2	equake	mcf	mesa	vpr
$\mathcal{C}_{a,win}$	0.631	0.898	1.004	0.762	0.769
$\mathcal{C}_{a,regfile}$	2.665	3.806	4.527	3.330	3.590
$\mathcal{C}_{a,dec}$	0.421	0.603	0.614	0.485	0.501
$\mathcal{C}_{a,ieu}$	16.32	24.09	26.18	19.56	20.33
$\mathcal{C}_{a,fpu}$	16.32	24.09	26.18	19.56	20.33
$\mathcal{C}_{a,lsu}$	2.527	3.981	4.087	3.912	3.035
$\mathcal{C}_{a,br}$	38.90	53.14	37.39	28.84	43.83
$\mathcal{C}_{a,icache}$	2.751	3.911	3.846	3.152	3.194
$\mathcal{C}_{a,dcache}$	17.09	27.02	27.72	27.35	24.33

$N_{a,req,dec} = N_{a,req,ieu} = N_{a,req,fpu} = N_{a,req,lsu} = N_{a,req,br} = N_{a,req,icache} = N_{a,req,dcache} = 1$. We assume the service rate of register file equals retirement unit, i.e. $\mu_{ret} = \mu_{regfile}$.

The capacitance primitives in TABLE III and service rates are used in (9) and (10) to obtain the individual power for each resource and summed up to the total power. TABLE IV lists the analytical and simulated results of power consumption. On average, there is a relative error of 10.9% between simulated results and analytical ones. The analytical results are usually under-estimated as our power model does not include all the resources in the simulator e.g. the result bus and the L2 cache.

We further validate with other power models. The BACPAC [12] calculator shows that the typical power consumption is 24.03 watts for a 5-million-transistor processor running at 600MHz and V_{dd} of 2.5V. The power consumption is close to the averaged analytical power of 27.38 watts in TABLE IV. Using the same V_{dd}, Ω and a 0.25μm technology based power library [10], the Synopsys Power Compiler also reports a total power of 32.1 watts for a RISC processor design in the scale.

Validation via Maximum Total Power: The UltraSPARC-III processor [11] is a 4-way superscalar processor manufactured with a 0.13μm process and a V_{dd} of 1.5 volts. The published maximum total power for the 1.2 Ghz version is 50 watts.

In TABLE IV, the maximum dynamic power π_{dyn} reported by our model is 31.10 watts for the benchmark `equake` with Ω of 600Mhz and V_{dd} of 2.5 volts. With the Ω and V_{dd} of UltraSPARC-III, the $\pi_{dyn,tot}$ is adjusted as $\frac{31.10}{600\times10^6\times2.5^2} \times 1.2 \times 10^9 \times 1.5^2 = 22.392$ watts. According to TABLE I, the leakage power for the 0.13μm process is 43.4% of the total power, the maximum total power computed by our model is $22.392/(1 - 0.434) \approx 40$ watts. This is a further evidence of the accuracy of our model.

VI. Applications of the Model

We now show by examples how the model can be used to explore the design space for the co-optimized solution. We will also discuss the impacts of individual factors.

Impact of Leakage Power: As the feature size decreases as technology scales, the leakage power takes a significant portion of the total power budget. Using our model, we study the impact of leakage power on the maximum clock frequencies and

TABLE IV. Simulated & Analytical Average Power (in watts) and Relative Error (absolute values).

Power	bzip2	equake	mcf	mesa	vpr	Avg.
π_{dyn}(Sim.)	23.36	32.85	30.17	31.95	33.79	30.42
π_{dyn}(Model)	21.05	31.10	31.08	26.36	27.30	27.38
Rel. Err.(%)	9.51	5.34	3.00	17.5	19.2	10.91

TABLE V. Maximum Clock Frequency & Power Under a Constraint of 25 Watts.

	Max. Clk. & Power without Leakage Optimization		
Tech.	Clk.(Ghz)	Dyn.Power(watts)	Leak.(watts)
0.35μm	0.4	19.03065611	2.45
0.18μm	1.3	18.40154351	5.65
0.13μm	1.4	13.76183809	10.85
0.10μm	2.0	12.58225197	12.025
0.07μm	3.0	10.6162751	14.05
	Max. Clk. & Power with Leakage Optimization		
Tech.	Clk.(Mhz)	Dyn.Power(watts)	Leak.(watts)
0.35μm	0.4	19.03065611	1.65
0.18μm	1.5	21.2325502	2.925
0.13μm	1.8	17.69379183	6.725
0.10μm	2.9	18.24426536	6.325
0.07μm	5.2	18.40154351	6.275

dynamic power consumptions in TABLE V. We obtain these metrics by varying Ω and fixing the rest parameters.

TABLE V shows that the leakage power without optimization grows consistently along with the technology downsizing. For the technology node of 0.07μm, the leakage power overtakes the dynamic power as the dominant power factor. This trend hinders the increase of clock frequencies which ranges from 400 Mhz for 0.35μm technology to 3 Ghz for 0.07μm technology.

A Co-optimization Case - Impact of L1 Instruction Cache: To obtain the configuration with the least energy for a task whose number of instructions is n_i, we look for the minimal total energy to finish the task. Let $\pi_{u,x}$ be the upper power limit for the x-th optimization case, the constraint $\pi_{dyn,tot} \leq \pi_{u,x} \leq \pi_U$ should hold when seeking for the maximum performance $\Theta \times \Omega$. If such a case x exists, then the time to execute the application is $n_i/(\Theta \times \Omega)$. Consequently, this yields the minimal total energy, at the x-th case where the power is $\pi_{dyn,tot}$:

$$E_x = n_i \times \pi_{dyn,tot}/(\Theta \times \Omega). \quad (11)$$

As an example, the benchmark`183.equake` has 1.3691×10^9 instructions with the test input. We set dynamic power limits $\pi_{u,x}$ to be in $\{1, \ldots, 33\}$ and Ω=600MHz. Fixing the rest parameters, we explore the number of lines in a directly mapped L1 instruction cache whose line width is 32 words. The miss ratios are obtained through simulation runs. It is also possible to use analytical cache models to obtain estimates. According to (11), the total energy is minimized to 47.64 joules with a L1 instruction cache of 128 lines.

Functional Units and Window Size: The number of functional units, the pipeline length and the window size are inter-related in the co-optimization of the power and performance.

To separate their impacts, we do separate rounds of co-optimization under a dynamic power constraint of 33 watts by enumerating one of the three factor and keeping the rest fixed at the default values. Our calculation shows the optimization of the pipeline length under the power constraint may significantly improve the power-performance efficiency. This conclusion is in agreement with the results of Srinivasan, et. al. [9].

Exhaustive Design Space Exploration: Though exhaustive exploration is not usually feasible, we take it to give an idea of the efficiency of our approach. We measured the time for our co-optimization model implemented in Java to exhaustively search the design space. It took $7,413$ seconds on a 1.4GHz Intel Xeon PC to explore a space of $487M$ cases consisting of $\{W_{dec}, T_{br}, t_{ins,pen}, t_{ieu}, t_{fpu}, t_{dat,pen}, F_{ieu}, F_{fpu}, F_{lsu}, F_{br}$, and $Z_{win}\}$ under a power limit π_U of 40 watts with other parameters fixed. The exploration speed is 65.7K cases per second.

VII. Conclusion

We proposed a unified analytical model for both power and performance. Extensive validations indicate the accuracy of the model. Using our model, we studied the impact of leakage power on the performance improvements for different technology nodes. We also proposed an approach to co-optimize power and performance and find the optimal total energy. The implementation of our model achieved a fast exploration speed of 65.7K cases/sec into a space containing $15+$ parameters.

Because of its completeness, flexibility and efficiency, our model should be a useful tool for designers to make power-aware decisions at early stages of design.

References

[1] F. A. Aloul, S. Hassoun, K. A. Sakallah, and D. Blaauw. Robust sat-based search algorithm for leakage power reduction. In *Proc. of the 12th Int'l Workshop on Integrated Circuit Design, Power and Timing Modeling, Optimization and Simulation*, pages 167–177. Springer-Verlag, 2002.

[2] T. Austin, D. Burger, G. Sohi, M. Franklin, S. Breach, and K. Skadron. The simplescalar architectural research tool set. In *Available online, http://www.cs.wisc.edu/ mscalar/simplescalar.html*, 1998.

[3] D. Brooks, P. Bose, S. Schuster, H. Jacobson, P. Kudva, A. Buyuktosunoglu, J. Wellman, V. Zyuban, M. Gupta, and P. Cook. Power-aware microachitecture: Design and modeling challenges for next-generation microprocessors. *Micro, IEEE*, 20(6):26–44, November-December 2000.

[4] D. Brooks, V. Tiwari, and M. Martonosi. Wattch: A framework for architectural-level power analysis and optimizations. In *Proc. of 27th Ann. Int'l Symp. Computer Architecture (ISCA)*, pages 83–94. IEEE Computer Society Process, Los Alamitos, California, USA, 2000.

[5] J. F. Cantin and M. D. Hill. Cache performance for spec cpu2000 benchmarks, version 3.0. In *Available online, http://www.cs.wisc.edu/multifacet/misc/spec2000cache-data/*, 2004.

[6] T. Conte, K. Menezes, S. Sathaye, and M. Toburen. System-level power consumption modeling and tradeoff analysis techniques for superscalar processor design. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 8(2):129–137, 2000.

[7] K. Khouri and N. Jha. Leakage power analysis and reduction during behavioral synthesis. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 10(6):876–885, December 2002.

[8] Y. Pyun, C. Park, and S. Choi. The effect of instruction window on the performance of superscalar processors. *IEICE Transactions on Fundamentals of Electronics Communications and Computer Sciences*, E81A(6):1036–1044, June 1998.

[9] V. Srinivasan, D. Brooks, M. Gschwind, P. Bose, V. Zyuban, P. N. Strenski, and P. G. E. Sylvester. Optimizing pipelines for power and performance. In *Proc. of MICRO-35*, pages 333–344, November 2002.

[10] J. Sulstyo and D. S. Ha. A new characterization method for delay and power dissipation of standard library cells. *VLSI Design*, 15(3):667–678, 2002.

[11] Sun. Ultrasparc III Cu processor. In *Available online, http://www.sun.com/processors/UltraSPARC-III/DS00101USIIICu0902.pdf*. Sun Microsystems, 2001.

[12] D. Sylvester, W. Jiang, and K. Keutzer. Bacpac - Berkeley Advanced Chip Performance Calculator. In *Available online, http://www.eecs.umich.edu/ dennis/bacpac/*, 1998.

[13] D. Sylvester and K. Keutzer. Getting to the bottom of deep submicron. In *Proc. of Int'l Conference on CAD*, pages 203–211, November 1998.

[14] N. Vijaykrishnan, M. Kandemir, M. Irwin, H. Kim, and W. Ye. Energy-driven integrated hardware-software optimizations using simplepower. In *Proc. of the ISCA-27*, pages 95–106, June 2000.

[15] Y. Zhu and W. Wong. Sensitivity analysis of a superscalar processor model. In *Proc. of the 7th Asia-Pacific Computer Systems Architectures Conference, Melbourne, Australia*, pages 109–118, January 2002.