The Shape of the Processor Design Space and its Implications for Early Stage Explorations

STIJN EYERMAN LIEVEN EECKHOUT KOEN DE BOSSCHERE
Department of Electronics and Information Systems (ELIS)
Ghent University
Sint-Pietersnieuwstraat 41, B-9000 Gent
BELGIUM
{seyerman,leeckhou,kdb}@elis.UGent.be

Abstract: - Designing a microprocessor involves determining the optimal microarchitecture for a given objective function and a given set of constraints. This paper studies the shape of the design space of superscalar out-of-order processors under different objective functions and constraints. We show that local optima exist whose objective function values are significantly worse than for the global optimum, in several cases more than 20% off. We subsequently consider the implications of this observation for early design stage exploration studies. Four design space search algorithms (random descent, steepest descent, one-parameter-at-a-time and simulated annealing) are evaluated according to their ability to avoid local optima and their overall simulation time. We conclude that one-parameter-at-a-time achieves a good balance between both criteria. In addition, we study the usefulness of fast simulation techniques for early design stage exploration. A case study with statistical simulation shows that significant simulation speedups are achieved while incurring little inaccuracy (a few percent) on the optimal design point.

Keywords: - design space exploration, microprocessor optimization, power/performance tradeoffs

1 Introduction

Designing a new microprocessor is a complex process as the processor design space is huge and the various design parameters and constraints interact with each other. These design issues typically concern performance, cycle time, power consumption, chip area, reliability, security, verifiability, etc. The task for a designer is to optimize the microarchitecture such that a given objective function is optimized. The objective function can take many forms depending on the target domain of the microprocessor under design. For example, when designing a server microprocessor, performance typically needs to be optimized while keeping the power consumption within a given power budget. For an embedded microprocessor on the other hand, the objective criterion typically involves energy- and/or cost-efficiency, or minimal energy/cost/area for a given performance target.

Obviously, the time-to-market for a newly designed processor should be kept as short as possible. Indeed, a computer company wants its microprocessor to outperform competitors at the time the new microprocessor gets introduced to the market. Although a short time-to-market is important for general-purpose microprocessors, it is even more important, not to say of vital importance, for embedded microprocessors. For example, designing a customized application-specific embedded microprocessor should be done as fast as possible to shorten the time-to-market in the rapidly evolving embedded market. To meet the short time-to-market both in the general-purpose and the application-specific domain, researchers have proposed various (semi-)automated design space exploration techniques, see for example [1, 4, 6, 8, 9, 11, 15, 16, 19].

In this paper, we study the shape of the design space of superscalar out-of-order processors and what the implications are for early design stage exploration techniques. To this end, we first investigate whether local optima exist in the processor design space and how this is affected per application and objective function. We show that local minima exist and that depending on the objective function the performance in the local optima can be significantly lower than in the global optimum, more than 20% in several cases. To the best of our knowledge there is no prior work available in the literature that characterizes the design space in terms of local and global optima for superscalar out-of-order microprocessors. We subsequently study the impli-

benchmark	input	simpoint
bzip2	program	10
crafty	ref	1
eon	rushmeier	19
gcc	166	100
gzip	graphic	4
perlbmk	makerand	2
twolf	ref	32
vortex	lendian2	58

Table 1. The SPEC CINT2000 benchmarks used in this paper, their inputs and their early single simulation points.

cations of these observations for early design stage explorations. We identify two main issues related to early design stage methods: the search algorithm for exploring the design space and the fast simulation techniques to drive the search algorithm. Obviously, under the existence of local optima, there is a potential pitfall that the search algorithm will yield a local optimum instead of the global optimum. We evaluate four automated design space search algorithms and study their performance in terms of (i) total simulation time, (ii) ability for avoiding local minima and (iii) inefficiency of the identified optimum. Next to detailed simulation, we also consider a case study in which we use a fast simulation technique, namely statistical simulation, to drive the design space exploration.

This paper is organized as follows. Section 2 presents our experimental setup. In section 3 we study the processor design space in search of local minima under different objective functions. Section 4 investigates the implications for early design stage exploration. Finally, we conclude in section 5.

2 Experimental setup

We use SimpleScalar/Alpha v3.0 as our simulation framework [3]. In order to estimate dynamic power consumption, we use Wattch v1.02 [2] which quantifies on-chip power consumption at the architectural level. We assumed a 0.18 μm-technology and the most aggressive clock gating mechanism (`cc3`): a unit that is unused consumes 10% of its max power, and a unit that is used for a fraction *x* only consumes a fraction *x* of its max power.

The chip area estimates are obtained using a tool developed by Steinhaus et al. [17].[1] This tool takes as input the SimpleScalar configuration file and computes an estimate for the chip area of the microarchitecture in terms of λ^2 with λ being half the minimum feature size of the chip technology.

The 8 SPECint2000 benchmarks that we use are given in Table 1 along with their inputs and their single 100M-instruction simulation points as determined by Early SimPoint [14].[2]

3 The shape of the processor design space

In this section, we want to verify whether local optima exist in the processor design space, and if they do, how many local optima exist, how inefficient they are compared to the global optimum, and how this is affected by the objective functions and the design constraints. Before studying the shape of the processor design space, we first define a local and global minimum. A *local minimum* is a design point that has a lower objective function value than all its neighbours (along all dimensions). A *global minimum* is the local minimum with the lowest objective function value. Obviously, if the design space only has a global optimum and no local optima, the potential risk from local optima is nihil making the design process much simpler. If on the other hand, local optima exist, the design process will need special care to avoid the design search process from identifying a local optimum as the optimal design point.

3.1 Optimization criteria

As mentioned earlier, there are a large number of potential objective functions when designing a microprocessor with different boundary conditions. Obviously, the shape of the design space can change wildly when the objective function is changed. Different objective functions result in different optima and possibly also in a different number of local minima. In this paper we consider three example objective criteria with a varying degree of complexity.

The first objective function is *minimal energy-delay product (EDP)*, i.e. the purpose is to identify the design point in a given processor design space with minimal EDP. EDP is an energy-efficiency metric that is often used in the context of general-purpose processors. It is defined as follows: $EDP = EPI \cdot CPI = EPC \cdot CPI^2$.

Our second objective function is *minimal energy under the constraint of near-optimal performance*. The purpose of this objective function is to identify the design point with a CPI that is within 2% of the optimal CPI while minimizing the energy consumption. This objective criterion gives a higher weight to performance than the 'minimal EDP' objective function because it requires near-optimal performance. Minimal EDP on the other hand, is willing to sacrifice $x\%$ performance if the energy consumption decreases by more than $x\%$.

Our third objective function is *minimal energy-delay-square product ED^2P under the constraint of a maxi-*

[1] `http://www.informatik.uni-augsburg.de/lehrstuehle/info3/research/complexity`

[2] `http://www.cs.ucsd.edu/~calder/simpoint`

parameter	dependent parameters	values			
window size	RUU size	16	32	48	64
	LSQ size	8	16	24	32
processor width	issue/decode/reorder width	2	4	6	8
	int ALUs	2	4	6	8
	int multiplies	1	1	2	2
	memory ports	1	2	3	4
	fp ALUs	1	1	2	2
	fp multiplies	1	1	2	2
branch predictor	bimodal predictor	1K	2K	4K	8K
	gshare predictor	1K	2K	4K	8K
	meta predictor	1K	2K	4K	8K
	BTB	64	128	256	512
L1 I-cache	size	8KB	16KB	32KB	64KB
	associativity	1	1	2	2
	latency	1	1	2	2
L1 D-cache	size	8KB	16KB	32KB	64KB
	associativity	1	1	2	2
	latency	1	1	2	2
L2 cache	size	256KB	512KB	1MB	
	associativity	4	4	8	
	latency	16	18	20	

Table 2. Processor design space.

mum chip area. This objective function minimizes the energy-delay-square product ED^{2}P which is defined as $ED^2P = EPI \cdot CPI^2 = EPC \cdot CPI^3$ under the constraint that the chip area does not exceed a given threshold. ED^{2}P gives a larger emphasis on performance than EDP. ED^{2}P is also voltage-independent and frequency-independent (to a first degree). In this paper, we measure chip area in terms of λ^2 with λ being half the minimum feature size. This metric makes it easy to compare and project the chip area of designs in future technologies which makes it particularly useful during early design stage explorations.

3.2 Existence of local optima

In this paper, we consider the design space of a superscalar out-of-order processor in which we vary six important microarchitectural parameters: window size, processor width, branch predictor, L1 I-cache, the L1 D-cache and unified L2 cache. Table 2 shows how the various parameters (and several dependent variables) are varied. The number of design points equals 3,072 in total. All these design points were evaluated for all benchmarks through detailed processor simulation. These data allow us to identify the global minimum, i.e. the processor configuration with the minimum objective function value. This is done for the three objective functions from the previous section, see Table 3. As expected, we observe that the optimal configuration depends on the objective function as well as on the benchmark.

Table 3 also shows the number of local minima per benchmark for all three objective functions along with their inefficiencies. We define the *inefficiency of a local optimum* as the relative difference of the objective function's value in the local and global optimum. For example, for objective function F, this is $\frac{F_{local}-F_{global}}{F_{global}}$ with F_{local} and F_{global} the objective function in the local and global minimum, respectively. A first important observation that we make from Table 3 is that local minima indeed exist and that the number of local minima varies for different objective functions and different benchmarks. For the 'minimal EDP' objective criterion most benchmarks do not show local minima; only two benchmarks have a single local minimum. The number of local minima generally increases for more complex objective functions, see for example the 'minimal energy for near-optimal performance' and the 'minimal ED^{2}P for a given chip area' objective criteria. For the latter objective function, we observe up to 6 or 7 local minima for particular benchmarks. We also observe from Table 3 that the inefficiencies can be significant, in many cases the inefficiencies in the local minima are more than 20% (up to 230% for crafty).

4 Design space exploration techniques

We now study how the shape of the processor design space impacts design space exploration. Automated design space exploration is particularly useful when designing customized embedded microprocessors. In this section, we consider four processor design space search algorithms: random descent, steepest descent, one-parameter-at-a-time and simulated annealing. This choice is motivated by the fact that (i) random descent is a very simple approach, (ii) steepest descent is a well known approach from other research domains, (iii) one-parameter-at-a-time was previously used by Fornaciari *et al.* [6] and (iv) simulated annealing, used for example by Conte *et al.* [4], tries to avoid local optima.

We start by presenting four design space search algorithms. We subsequently evaluate their performance using detailed processor simulation; this is to study the intrinsic performance of the search algorithms. Third, we study design space exploration using fast simulation techniques.

4.1 Design space search algorithms

Random descent is the simplest algorithm. It starts from a given design point. The idea is then to randomly choose a dimension at each iteration and randomly increment or decrement the microarchitectural parameter along that dimension. If the objective function improves, we accept this new design point; otherwise, we reject it. The algorithm finishes when the objective function can no longer be improved. Obviously, in order to reduce the number of simulations, we keep track of the design points that were evaluated previously so that we do not need to recompute them.

Steepest descent is similar to random descent except that in each iteration the dimension is determined along which the biggest improvement in the objective function is observed. The algorithm then takes a step

minimal EDP								
	optimal configuration						local minima	
benchmark	window	width	bpred	L1 I$	L1 D$	L2 $	#	inefficiency
bzip2	48	6	2K	8KB	16KB	1MB	0	n/a
crafty	64	8	8K	64KB	64KB	1MB	0	n/a
eon	64	8	8K	64KB	32KB	256KB	0	n/a
gcc	48	8	1K	8KB	32KB	256KB	0	n/a
gzip	48	8	1K	32KB	16KB	256KB	1	20.4%
perlbmk	32	6	1K	32KB	32KB	1MB	1	14.1%
twolf	48	4	4K	32KB	32KB	1MB	0	n/a
vortex	64	8	8K	64KB	32KB	1MB	0	n/a
minimal energy for near-optimal performance								
	optimal configuration						local minima	
benchmark	window	width	bpred	L1 I$	L1 D$	L2 $	#	inefficiency
bzip2	64	8	8K	8KB	32KB	1MB	1	6.4%
crafty	64	8	8K	64KB	64KB	1MB	0	n/a
eon	64	8	8K	64KB	32KB	256KB	0	n/a
gcc	64	8	1K	8KB	32KB	256KB	0	n/a
gzip	64	8	1K	32KB	64KB	256KB	0	n/a
perlbmk	48	6	2K	32KB	32KB	1MB	2	4.9%, 7.2%
twolf	64	8	8K	64KB	64KB	1MB	2	4.4%, 7.1%
vortex	64	8	8K	64KB	32KB	1MB	1	8.1%
minimal ED^2P for a given chip area (15 billion λ^2)								
	optimal configuration						local minima	
benchmark	window	width	bpred	L1 I$	L1 D$	L2 $	#	inefficiency
bzip2	64	2	4K	8KB	8KB	256KB	0	n/a
crafty	32	2	2K	32KB	32KB	256KB	6	3.0%, 3.4%, 88.1%, 88.6%, 89.1%, 231.8%
eon	48	2	2K	32KB	16KB	256KB	1	48.3%
gcc	64	2	4K	8KB	8KB	256KB	0	n/a
gzip	32	2	2K	32KB	8KB	256KB	1	5.9%
perlbmk	32	2	2K	32KB	32KB	256KB	3	2.8%, 18.8%, 19.8%
twolf	64	2	1K	32KB	32KB	256KB	7	3.5%, 4.7%, 7.7%, 8.5%, 9.4%, 11.2%, 13.2%
vortex	48	2	2K	32KB	8KB	256KB	4	2.7%, 27%, 27.4%, 51.7%

Table 3. Optimal configurations and the number of local minima along with their inefficiency for the three objective functions.

in that direction to its new design point. The algorithm finishes when the objective function can no longer be improved. We expect steepest descent to take fewer iterations than random descent since steepest descent, as its name suggests, chooses the steepest slope at each iteration. However, at each iteration the objective function needs to be calculated for each of the neighboring design points in order to find the steepest slope. In an n-dimensional space, this requires $2 \cdot n$ simulations per iteration.

One-parameter-at-a-time assumes a fixed ordering of dimensions along which to optimize. In the first iteration of the algorithm, the objective function is optimized along the first dimension while keeping the other dimensions unchanged. In the second iteration, the objective function is optimized along the second dimension, etc. When all dimensions are optimized, the algorithm starts over again until the objective function can no longer be improved. We studied the performance of this algorithm for different fixed orderings and found that the ordering of the parameters does influence the performance for individual benchmarks, but we found no particular ordering that is best for all benchmarks.

Simulated annealing is the most complicated search algorithm that (if properly tuned) can avoid getting stuck in local optima. Similar to random descent, simulated annealing chooses a random direction around design point X. If the objective function improves along that dimension, the new design point Y is accepted. If not, the new design point is accepted with a probability $e^{-\frac{F_Y - F_X}{T}}$. This probability is a function of the objective functions in the current and the new design point, as well as of the 'temperature' T that decreases as the algorithm continues. The basic idea of simulated annealing is to accept worsenings of the objective function in the beginning of the algorithm so that a broad area in the design space can be explored. As the algorithm continues, the temperature is decreased so that the worsenings that get accepted also decrease. However, the accepted worsenings are large enough to jump over local optima.

4.2 Evaluation using detailed simulation

Figure 1 compares the performance of the four search algorithms using detailed processor simulation. We quantify the number of simulations, the probability for the search algorithm to yield a local optimum, and the average inefficiency of the search algorithm (measured by comparing the objective function's value of

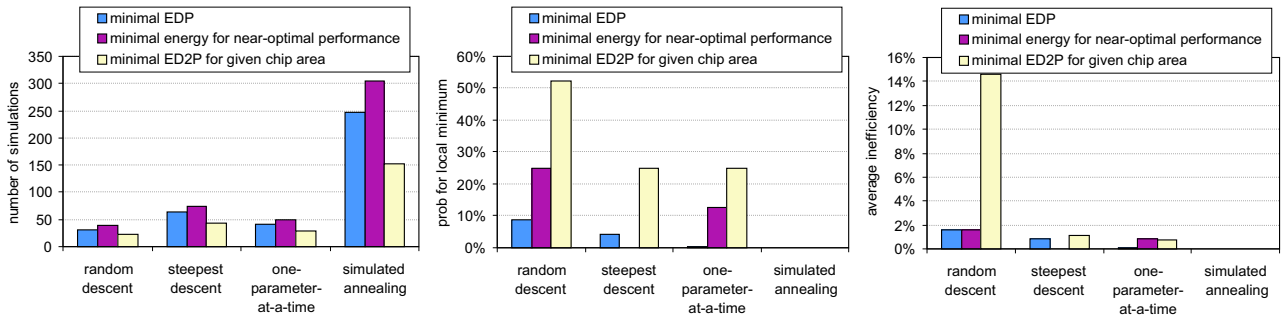

Figure 1. The performance of the four search algorithms: the number of simulations (left), probability to identify a local minimum (middle) and average inefficiency (right).

the global optimum versus the objective function's value of the optimum as identified by the search algorithm). This was done by running the search algorithms 100 times starting from randomly chosen initial design points. The results shown in Figure 1 are average numbers over all benchmarks. We observe that simulated annealing results in significantly more simulations than the other search algorithms by a factor of 3.5 to 7.9. The reason is that simulated annealing accepts worsenings in order to explore the design space across mountains. Steepest descent on its turn also needs more simulations than random descent and one-parameter-at-a-time. This is due to the fact that steepest descent evaluates all adjacent design points in order to identify the steepest slope to go into.

A second important observation that is to be made from Figure 1 (middle graph) is that all search algorithms, except for simulated annealing, have a non-zero probability of yielding a local optimum. Depending on the objective function and the search algorithm, this probability can be significant, up to 52% ('minimal ED^2P for a given chip area' and random descent). Not surprisingly, the probability for yielding a local optimum increases for 'minimal EDP', 'minimal energy for near-optimal performance' and 'minimal ED^2P for a given chip area', respectively. Indeed, the latter objective function showed the largest number of local minima, and their inefficiencies were also the highest, see Table 3. These non-zero probabilities for yielding a local minimum result in inefficiencies on the global minimum, see graph on the right in Figure 1. For the 'minimal ED^2P for a given chip area' objective function and the random descent search algorithm, the inefficiency can be up to 14%.

As such, we conclude that random descent requires the smallest number of simulations, however the probability for yielding a local minimum is significant, ranging from 9% up to 52% depending on the objective function. Steepest descent and one-parameter-at-a-time require more simulations but the probability for a local minimum can also be up to 25%. The inefficiencies for these search algorithms are small however, smaller

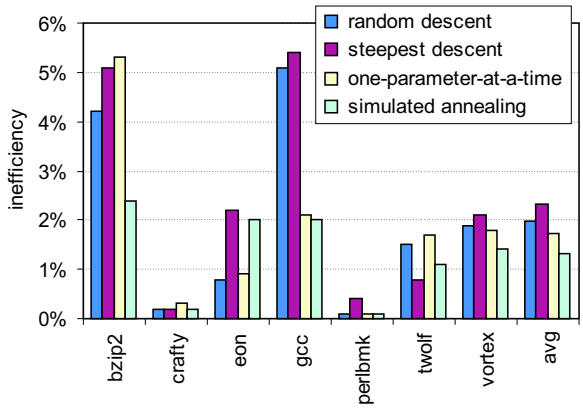

Figure 2. Design space exploration using statistical simulation in conjunction with search algorithm for the 'minimal EDP' objective criterion.

than 1.2%. Simulated annealing finds the global optimum, however at the cost of significantly more simulations. One-parameter-at-a-time seems to make a good balance between the number of simulations (less than steepest descent and simulated annealing) and accuracy (the inefficiency being less than 0.9%).

4.3 Evaluation using fast simulation

Until now, we considered the performance of the search algorithms when used in conjunction with detailed processor simulation. However, given the time-consuming behavior of detailed simulation it is to be expected that fast simulation techniques can significantly reduce the total simulation time during design space exploration. The purpose of this section is to study design space exploration through fast simulation. We use statistical simulation as our example fast simulation technique. Statistical simulation [5, 12, 13] collects a number of program statistics from a program execution. These statistics comprise information concerning the instruction mix, the inter-operation dependencies, the basic blocks, the cache miss behavior and the branch misprediction behavior. Based on these statistics a synthetic trace is generated that is subse-

quently fed into a statistical simulator. The important advantage of statistical simulation is that the synthetic trace is several orders of magnitude shorter than a real program execution making statistical simulation a very fast simulation technique—compare the the 100K to 1M instructions in a synthetic trace versus the billions of instructions in a real program execution (even compared to the 100M simulation points used in this paper, a synthetic trace is very short). Figure 2 shows the inefficiency for the four search algorithms in conjunction with statistical simulation. The inefficiency of statistical simulation is around 2% on average for the 'minimal EDP' objective function. We observed similar results for the other objective functions, although with slightly higher inefficiencies (around 4%).

5 Conclusion

This paper studied the shape of the processor design space under different objective functions and constraints. We have shown that local optima indeed exist and that their inefficiencies can be significant, in several cases more than 20% compared to the global optimum. We subsequently investigated the implications of this observation for early design stage exploration techniques. We evaluated four search algorithms according to their overall simulation time and their ability for avoiding local optima. Of the four algorithms investigated we conclude that one-parameter-at-a-time makes a good balance between overall simulation time and performance of the identified optimal design point. In addition, we considered statistical simulation as an example fast simulation technique and conclude that statistical simulation is accurate enough to be useful for early design stage explorations. The inefficiency of the identified optimum is small (only a few percent) while yielding significant simulation speedups.

We feel this paper only scratches the surface. In future work we will study (i) how the shape of the processor design space is affected by other objective functions, (ii) other design space exploration techniques (e.g. genetic algorithms and tabu search) that are potentially faster and more accurate than those studied in this paper, (iii) the applicability of analytical modeling [7, 10] and other fast simulation techniques such as statistical sampling [18] for design space exploration.

References

[1] S. G. Abraham and S. A. Mahlke. Automatic and efficient evaluation of memory hierarchies for embedded systems. In *MICRO-32*, pages 114–125, Nov. 1999.

[2] D. Brooks, V. Tiwari, and M. Martonosi. Wattch: A framework for architectural-level power analysis and optimizations. In *ISCA-27*, pages 83–94, June 2000.

[3] D. C. Burger and T. M. Austin. The SimpleScalar Tool Set. Computer Architecture News, 1997. See also `http://www.simplescalar.com` for more information.

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

[5] L. Eeckhout, R. H. Bell Jr., B. Stougie, K. De Bosschere, and L. K. John. Control flow modeling in statistical simulation for accurate and efficient processor design studies. In *ISCA-31*, pages 350–361, June 2004.

[6] W. Fornaciari, D. Sciuto, C. Silvano, and V. Zaccaria. A design framework to efficiently explore energy-delay tradeofss. In *CODES'01*, pages 260–265, Apr. 2001.

[7] A. Ghosh and T. Givargis. Analytical design space exploration of caches for embedded systems. In *DATE'03*, Mar. 2003.

[8] T. Givargis, F. Vahid, and J. Henkel. System-level exploration for pareto-optimal configurations in parameterized systems-on-a-chip. In *ICCAD'01*, Nov. 2001.

[9] G. J. Hekstra, P. B. G. D. La Hei, and F. W. Sijstermans. TriMedia CPU64 design space exploration. In *ICCD'99*, Oct. 2001.

[10] T. S. Karkhanis and J. E. Smith. A first-order superscalar processor model. In *ISCA-31*, June 2004.

[11] J. Kin, C. Lee, W. H. Mangione-Smith, and M. Potkonjak. Power efficient mediaprocessors: Design space exploration. In *DAC'99*, June 1999.

[12] S. Nussbaum and J. E. Smith. Modeling superscalar processors via statistical simulation. In *PACT-2001*, pages 15–24, Sept. 2001.

[13] M. Oskin, F. T. Chong, and M. Farrens. HLS: Combining statistical and symbolic simulation to guide microprocessor design. In *ISCA-27*, pages 71–82, June 2000.

[14] E. Perelman, G. Hamerly, and B. Calder. Picking statistically valid and early simulation points. In *PACT-2003*, pages 244–256, Sept. 2003.

[15] T. Sherwood, M. Oskin, and B. Calder. Balancing design options with Sherpa. In *CASES'04*, Oct. 2004.

[16] G. Snider. Spacewalker: Automated design space exploration for embedded computer systems. Technical Report HPL-2001-220, HP Laboratories Palo Alto, Sept. 2001.

[17] M. Steinhaus, R. Kolla, J. L. Larriba-Pey, T. Ungerer, and M. Valero. Transistor count and chip-space estimation of SimpleScalar-based microprocessor models. In *WCED 2001 in conjunction with ISCA-28*, June 2001.

[18] R. E. Wunderlich, T. F. Wenish, B. Falsafi, and J. C. Hoe. SMARTS: Accelerating microarchitecture simulation via rigorous statistical sampling. In *ISCA-30*, June 2003.

[19] J. J. Yi, D. L. Lilja, and D. M. Hawkins. A statistically rigorous approach for improving simulation methodology. In *HPCA-9*, Feb. 2003.