A version of this will appear in ISPASS-2001 November 4-6, 2001 (www.ispass.org)

How to Compare the Performance of Two SMT Microarchitectures

Yiannakis Sazeides
Department of Computer Science
University of Cyprus
yanos@ucy.ac.cy

Toni Juan
Computer Architecture Department
Universitat Politecnica de Catalunya
antonioj@ac.upc.es

Abstract

In this paper we discuss methods and metrics for comparing the performance of two simultaneous multithreading microarchitectures. We identify conditions under which the instructions-per-cycle metric may be misleading for comparing two simultaneous multithreading microarchitectures for the same amount of work. Part of the problem is isolated to the definition of what is same work. When simulating a mix of independent programs under the same initial conditions on two different simultaneous multithreading microarchitectures there are two approaches to ensure the work of the two runs is same: **constant-work-per-thread** *or* **variable-work-per-thread**. *For both approaches the total number of instructions in the run is constant, however, for the first, the instructions from each thread is also constant, whereas for the second is not.*
We claim that:
(a) when simulating two microarchitectures with the constant-work-per-thread approach, the instructions-per-cycle is sufficient to compare them to establish the microarchitecture with the best performance,
(b) when variable-work-per-thread approach is used the instruction-per-cycle may be inadequate for comparing performance. We attribute this to the inability of the instructions-per-cycle metric to account for differences in the load-balance of the two runs.
A new performance metric, **SMT-speedup**, *is proposed that enables accurate comparison of the performance of two simultaneous multithreading microarchitectures for runs with different load-balance. The new metric considers the load-balance in terms of the size and performance of each thread.*

In light of the insight gain in this paper we contend that a simultaneous multithreading microarchitecture may need to trade-off throughput and load-balance to achieve the best performance.

1 Introduction

Microarchitectural simulation is an essential tool for the design of high performance processors because it provides an inexpensive and effective mean to designers and researchers to explore the design space. With simulation the performance potential of a new mechanism can be established without actually implementing it and a safe decision can be made as to whether the mechanism should be incorporated in a processor.

Typical microarchitectural simulation involves several steps. Prior to simulation the workload that will be simulated is selected. For each benchmark and data set in the workload a representative region(s) is chosen. The same amount of work is then simulated on two microarchitectures, without and with the new mechanism, and performance metrics from each run are recorded. The various metrics are then summarized and analyzed to establish potential benefits/detriments of the mechanism.

When investigating mechanisms aimed to improve performance one of the most illuminating metrics is throughput or Instructions per Cycle(IPC). The IPC for a given simulation run is the length of the region over the latency - number of cycles to execute the region. Suppose that a *base* microarchitecture has IPC *x* for a region and a *new(modified)* microarchitecture has for the same region IPC *y*, one of the following will then be true:
$\mathbf{x} < \mathbf{y}$, the new microarchitecture has higher performance and hence the new mechanism is increasing performance,
$\mathbf{x} > \mathbf{y}$, the base is better indicating the mechanism is detrimental to performance,
x = y, the new mechanism is not influencing performance.
The speedup $\frac{y}{x}$ is a deciding factor as to whether to consider the mechanism further.

The above methodology is widely accepted and used by designers and researchers to evaluate the performance potential of mechanisms for a given microarchitecture. In this paper we argue that although this methodology may be dependable for single-threaded processors may not always be for simultaneous multithreading(SMT) processors[1, 2, 3]. SMT is a processor microarchitecture distinguished by its ability to exploit inter-thread and intra-thread parallelism in the same execution cycle using a unified resource pool.

In particular, this work will show that when comparing the IPC of simulations with a mix of independent programs, the potential of a a new mechanism, and hence of a SMT microarchitecture, may not be assessed correctly. Because SMT mechanisms aim typically to adjust the load-balance - the contribution from each thread - to achieve better performance and the IPC metric ignores load-balance and only considers the total number of instructions from all threads. To overcome this we propose a SMT performance metric (*SMT-speedup*) that considers load-balancing. The metric takes into account the contribution from each thread in terms of its size and single-thread performance. Central to the problem of comparing SMT performance lies the definition of what is same work for two SMT simulations. Two different definitions of same work are presented and their ramifications are discussed.

1.1 Outline

The paper is organized as follows. In Section 2 we will illustrate why may be problematic to compare two SMT microarchitectures based on IPC. A new performance metric that accounts for the workload composition is presented in Section 3. Section 3 includes a discussion about how and when to use the proposed metric. Related work is discussed in Section 4 and the work concludes with Section 5.

2 Problem

Lets assume a *base* SMT microarchitecture and that a new mechanism (e.g. new fetch thread-chooser) is considered for it. Henceforth we assume that simulations on the *base* and *new* microarchitecture have the same initial conditions, equal clock, the mechanism under consideration has no influence on the single-thread performance, and the regions simulated are *representative* of the programs behavior (we revisit the issue of representative behavior in Section 3.2).

To evaluate the new mechanism we perform simulation using a mix of two independent programs, *B* and *R*. The simulations are for the same number of instructions. Note that the choice for two threads is for illustration and that the issues discussed here can be generalized to any number of threads.

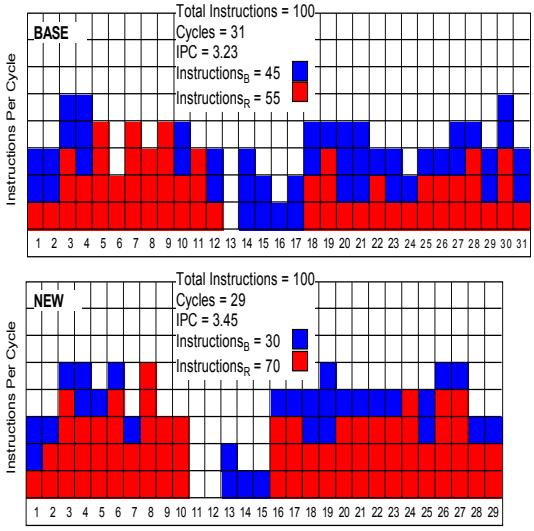

Figure 1: IPC Cycle Profile for two SMT runs

Fig. 1 is used to illustrate the problem that may arise when comparing the performance of two SMT microarchitectures based on throughput. The two SMT microarchitectures, *base* and *new*, are simulated for the same amount of work, 100 instructions, from the two threads. The IPCs of the two runs are 3.23 and 3.45 respectively. The IPC of the *new* microarchitecture is (3.45/3.23 - 1) = 7% faster than the *base*. However, the contribution of thread *R*(*B*) in the first run is 55(45) instructions whereas in the second run is 70(30): *the total load is the same, but the load-balance in the two runs is different.* We repeat that the initial conditions for the two runs are the same, the different load-balance is due to the new mechanism under evaluation.

We argue that is unclear for the above and analogous scenarios which microarchitecture is better performing since (a) the load-balance between the two runs is not invariant, and (b) the IPC metric only considers the total number of instructions and ignores the contribution from each thread.

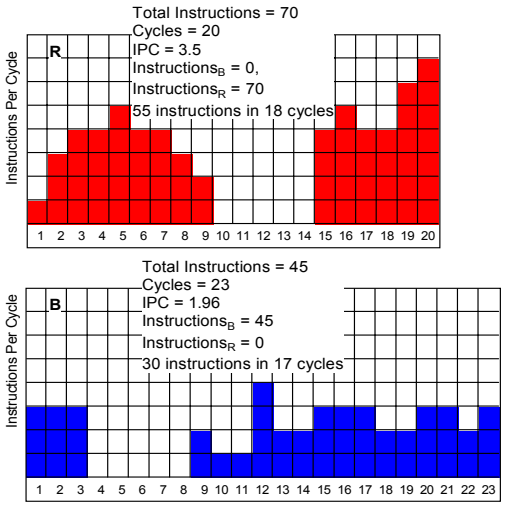

Figure 2: IPC Cycle Profile for Single-Thread Runs

In Fig. 2 we show a possible cycle by cycle execution profile for each of the threads *R* and *B* when executing in single-thread mode for the same work as in Fig. 1. We can conclude that if we were to execute back-to-back the work from the two threads - 55 instructions of *R*, followed by 45 instructions from *B* - the total latency will be 18+23=41 cycles. Whereas if were to execute 70 instructions from *R* followed by 30 from *B* the total latency will be 20+17=37 cycles. If we compare the latency of the multithread SMT runs in Fig. 1 with the corresponding total latency of single thread runs for the same workload - both in size and composition - we can observe: that the *base* configuration is ((18+23)/31 - 1)= 32% faster whereas the modified is ((20+17)/29 - 1) = 28% faster. *This relation is opposite of the IPC relation*! So which SMT microarchitecture is best performing?

We contend that the base *microarchitecture is the best performing provided all threads are having the same execution priority*. This is the case because while the *new* microarchitecture favored the thread with high IPC (thread *R*) and achieved overall higher IPC, it compromised the load-balance at the expense of the slow thread (thread *B*). In contrast, the *base* microarchitecture offered a balanced execution from both threads with only slightly worse throughput. *Effectively, a SMT microarchitecture may need to trade-off throughput and load-balance to achieve the best performance.*

The above suggest that SMT performance metrics that overlook load-balance can be misleading. We believe the problem with comparing SMT performance based on IPC/throughput was recognized before [2]: *"Given our measurement methodology, it is possible that the throughput increases could be overstated if a fetch policy simply favors those threads with most inherent ILP or the best cache behavior, thus achieving improvements that would not be seen in practice."* Consequently, **we make a case for a performance metric that considers the contribution from each thread** for comparing the performance of simulations with

different load-balance. We underline that this is a SMT processor specific issue because is the first microarchitecture with the distinctive ability of simultaneous execution from multiple threads in the same cycle.

3 Proposed Metric

In this section we define a speedup metric suitable for measuring SMT performance and then we discuss how to use it for comparing the performance of two SMT microarchitectures. The Section includes a discussion about the use and limitations of the proposed metric.

3.1 SMT-Speedup

Given a SMT microarchitecture, if we perform a simulation using a mix of *T* threads for *I* instructions and the latency of the run is *L* cycles, we define its *SMT-speedup* to be the ratio

$$\frac{L_1+...+L_T}{L}$$

where L_j is the latency to execute I_j instructions from thread *j* in single-thread mode. Note that

$I = I_1 + ... + I_T$, and if we define a weight $W_j = \frac{I_j}{I}$, then $I = I\,W_1 + ... + I\,W_T$.

Given that $IPC = \frac{I}{L}$ and $IPC_j = \frac{I_j}{L_j}$, then we can express *SMT-speedup* as follows:

$$IPCx(\frac{W_1}{IPC_1}+...+\frac{W_T}{IPC_T})$$

The *SMT-speedup* gives the benefit obtained by executing a workload in simultaneous multithreading mode as compared to the total latency of the same workload (in size and composition) in single-thread mode. The higher the speedup the better the performance. The proposed metric considers the load-balance by taking into account (a) the relative contribution of each thread (weights W_j), and (b) the threads single-thread performance (single-thread IPC_j). The first term is the IPC of the multithreaded run (*SMT-IPC*), whereas the second is the inverse weighted harmonic IPC for all single-thread runs (*effective-single-thread IPC*). Note that based on our assumptions the above metric has value 1 when there is only one thread in the workload.

Two important implications of the above definition is that, depending on the load-balance, different IPCs can correspond to the same *SMT-speedup* and a single IPC may correspond to different *SMT-speedups*. This supports our claim that the IPC may be a misleading metric for comparing SMT performance.

3.2 Comparing SMT-Speedups/Load Composition

When desired to compare the performance of a new mechanism for an SMT microarchitecture then (a) we should simulate two microarchitectures with and without the mechanism for the same amount of work and initial conditions(same thread mix, same starting instruction count etc), (b) compute their respective *SMT-speedups*, and (c) compare them to determine the microarchitecture with the highest *SMT-speedup* (best performance). In effect, we claim that the performance comparison of two SMT microarchitectures should not be through a direct comparison, but rather through an indirect comparison that determines the SMT microarchitecture with the best performance over single thread performance.

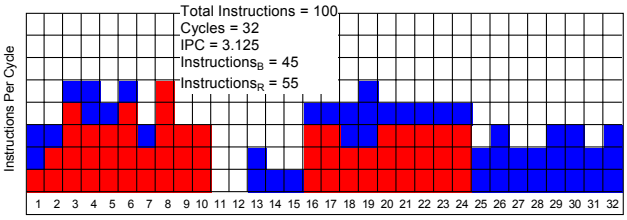

Figure 3: IPC Cycle Profile for SMT Run with Constrain in Thread Size

The proposed metric is not useful to evaluate a mechanism that affects single-thread performance because it may influence both the SMT-IPC and effective-single-thread-IPC. Thus, a SMT microarchitecture with higher(lower) IPC may have lower(higher) *SMT-speedup* which makes *SMT-speedup* comparisons ineffective.

Recall that although the total number of instructions in two SMT simulations can be the same, the load-balance will very likely be different(Section 2).
This difference in load-balance raises two questions that correspond to two simulation approaches:
(a) can we ensure a comparison of two SMT microarchitectures with identical workloads?
(b) if the workloads are different can it lead to misleading assessment of results?
Next we discuss these two approaches.

Identical Workload (Constant-work-per-thread)

In general, we believe is unrealistic to expect identical load-balance in two simulation runs with same initial conditions when the two simulated SMT microarchitectures employ different mechanism(s).

Instead of relying on mechanism properties to ensure identical workload for two simulation runs, we can introduce a simulation constrain to enforce it. Specifically, given the load-balance of a reference run, stop fetching instructions from a thread in another run when the threads instruction count in the reference run is reached. We refer to this as *constant-work-per-thread* approach, since a thread will contribute a constant number of instructions in both runs. Since the workload composition of the two runs is identical then is meaningful to compare directly the IPCs (note that the corresponding *SMT-speedup*s have the same effective-single-thread-IPC).

To explicate see Fig. 3 where we use the *base* run in Fig. 1 as the reference for simulating the *new* microarchitecture. When thread *R* in cycle 24 reached its threshold (55 instructions) no more instructions were fetched from it. And the simulation continued until the remaining thread, *B*, reached its threshold of 45 instructions in cycle 32. The IPC of the run in Fig. 3 is 3.125. Since the load-balance is identical we can compare the IPCs (3.23 vs 3.125) and determine that the *base* is best performing. As seen in Fig. 3 the load is not well distributed and thus the worse performance.

We believe the constant-work-per-thread approach is "fair" because always compares performance for identical work (note that running all threads to completion is a special case of the constant-work-per-thread approach). Possibly the only caveat of this approach is that may not be representative of the actual SMT execution behavior. For instance, one could argue that for typical execution always the maximum number of threads will be running. Therefore, this

kind of approach will be meaningful when the % of time all threads are running is dominant.

Different Workload (Variable-work-per-thread)

The discussion in Section 2 assumed simulation with same total work but non-identical workloads - *variable-work-per-thread approach* - and concluded that the IPC metric may be insufficient for comparing SMT performance. The proposed *SMT-speedup* metric was shown to account for load-balance difference.

One may argue that comparing *SMT-speedups* with *variable-work-per-thread approach* is as meaningless as comparing IPCs (see Section 2) and hence can lead to erroneous interpretations. *We believe* this is not the case as long as the instructions simulated from each thread in a mix is representative of the benchmarks single-thread behavior.

Particularly, given a multithread simulation run with a mix of T independent programs, and I_j is the instruction count from a thread j, then for each thread its I_j should be equal to the length of the representative region that was selected for the benchmarks single-thread behavior.

The equality criterion may be difficult to satisfy but with some relaxation should be feasible most of the time. For example, when the instruction counts between a representative region length and the simulated differ by a *small* amount, one may consider the instructions simulated to be representative. In such a case may be useful also to consider other parameters in the equality criterion. A good indication that what is simulated is representative will be if the effective-single-thread-IPC of the simulation run is equal to the effective-single-thread-IPC of the representative regions.

If not all threads are exercised or the regions simulated are non-representative then either the mechanism under consideration is problematic and/or the simulation length should be modified. If the representative behavior criterion can not be satisfied, then remains an open issue as to how to determine the best performing SMT microarchitecture with variable-work-per-thread.

To the question whether both constant-work-per-thread and variable-work-per-thread simulation approaches are needed, we answer that empirical investigation of their behavior is needed to quantify their difference and similarities. This represents an important subject for future work.

4 Related Work

SMT is a processor microarchitecture distinguished by its ability to exploit inter-thread and intra-thread parallelism in the same execution cycle using a unified resource pool[1, 2, 3].

A discussion on how to compare and summarize rates and execution time can be found in [4]. The metric used in previous SMT work for comparing the performance was the IPC/throughput[1, 2]. Typically the IPCs of two microarchitectures were compared for the same amount of work but is unclear whether the load-balance between runs was forced to be identical or not[1, 2]. The problematic of comparing SMT performance based on IPC/throughput was mentioned before [2]. However, no specific cause was suggested or solution was proposed to overcome the problem.

The issues discussed in this paper are unique to SMT processors since no previous microarchitecture has the ability for simultaneous issue of instructions across different threads in the same cycle.

An orthogonal issue with important implications to this work is the definition of regions that are representative of a benchmark's behavior[5]. This is critical, as described in Section 3.2, for establishing whether the simulated region is representative of the benchmarks behavior.

To the best of our knowledge this is the first work that:
(1) identified conditions for which may be problematic to compare SMT microarchitectures based on IPC,
(2) argued that SMT processors may need to trade-off throughput and load-balance to achieve best performance, and
(3) proposed a SMT performance metric that considers load-balance.

The relevance and utility of this work is stressed by the implementation, until a late development phase, of the Alpha EV8 SMT processor[6] and the recent disclosure by Intel that SMT(hyper-threading) technology will be used in its future processors[7].

5 Conclusions

In this paper we demonstrated that comparing the IPC of two SMT microarchitectures may be not always dependable method for establishing the best performing microarchitecture. The problem is due (a) to the inability of the IPC metric to take into account the load-balance in a run with a mix of independent programs, and (b) the definition of same work. We proposed a new performance metric, *SMT-speedup*, that considers the load-balance in terms of the size and performance of each thread. We presented two approaches for simulating same work for SMT: constant-work-per-thread and variable-work-per-thread and rationalize why the IPC is more appropriate to compare the performance for the first and the *SMT-speedup* for the second.

In light of the views in this paper we declare that for the simulation of a simultaneous multithreading microarchitecture: (a) with constant-work-per-thread approach, better performance means higher throughput, and (b) with variable-work-per-thread approach, better performance means a trade-off between load-balance and throughput.

We anticipate the proposed metric to prove useful to designers and researchers for establishing with more accuracy the performance potential of SMT processors. For future work we plan to use the *SMT-speedup* metric to evaluate previously proposed SMT mechanisms that were selected(rejected) based on their IPC and variable-work-per-thread approach. We are also planning to investigate methods for selecting representative regions for SMT microarchitectural simulation. Finally, we plan to investigate and compare the empirical behavior of the constant-work-per-thread and variable-work-per-thread approaches.

6 Acknowledgments

This work has been partially supported by the Spanish Ministry of Education under grant CYCIT TIC98-0511. The authors while with Compaq in 1999 had useful discussions on the subject of measuring SMT performance with George Chrysos, Venkata Krishnan, André Seznec and Tryggve Fossum. We also thank Francisco Jesus Sanchez for his comments on an earlier draft of the paper.

References

[1] D. M. Tullsen, S. J. Eggers, and H. M. Levy, "Simultaneous multithreading: Maximizing on-chip parallelism," in *Proceedings of the 22nd Annual International Symposium on Computer Architecture*, pp. 392–403, June 1995.

[2] D. M. Tullsen, S. J. Eggers, J. S. Emer, H. M. Levy, J. L. Lo, and R. L. Stamm, "Exploiting choice: Instruction fetch and issue on an implementable simultaneous multithreading processor," in *Proceedings of the 23rd Annual International Symposium on Computer Architecture*, pp. 191–202, May 1996.

[3] S. Eggers, J. Emer, H. Levy, J. Lo, R. Stamm, and D. Tullsen, "Simultaneous Multithreading: A Platform for Next-generation Processors," *Computer*, pp. 12–18, September 1997.

[4] J. E. Smith, "Characterizing Computer Performance with a Single Number," *Communications ACM*, vol. 31, pp. 1202–1206, October 1988.

[5] K. Skadron, P. Ahuja, M. Martonosi, and D. Clark, "Selecting a Single, Representative Sample for Accurate Simulation of SPECint Benchmarks," in *Tech Report TR-595-99, Princeton Dept. of Computer Science*, Jan. 1999.

[6] K. Diefendorff, "Compaq Chooses SMT for Alpha," *Microprocessor Report*, December 1999.

[7] K. Krewell, "Intel Embraces Multithreading," *Microprocessor Report*, September 2001.