The Effect of the Speculation Depth on the Performance of Superscalar Architectures

Eliseu M. C. Filho
Edil S. T. Fernandes
Department of Systems and Computer Engineering
COPPE/Federal University of Rio de Janeiro
P.O. Box 68511 21945-970 Rio de Janeiro,RJ Brazil
e-mail: {eliseu,edil}@cos.ufrj.br

Abstract

Speculative execution is a key concept to increase the performance of superscalar processors. Given accurate branch prediction mechanisms, the efficiency of speculative execution is mainly determined by the speculation depth. In this work, we evaluate the pressure of speculative execution on the resource requirements of a typical superscalar architecture.

1 Introduction

Speculative execution improves processor performance by anticipating the execution of instructions beyond the boundaries of basic blocks. On the other hand, there is a cost imposed by the cancellation of instructions along mispredicted paths. Current dynamic branch prediction mechanisms [1] yield an average accuracy as high as 85%, and there are new predictors that can achieve a near 100% accuracy [2]. Given these accuracy ratios, the cost of mispredictions are significantly reduced, and the performance gain becomes mainly determined by the ability to anticipate the execution of instructions across unresolved branches. This ability is characterized by the *speculation depth* — the number of unresolved branches beyond which the instruction dispatch mechanism stalls. Current superscalar machines exhibit different values for the speculation depth: it is two branches in the PowerPC 604 [3], four branches in the PowerPC 620 [4] and in the R10000 [5] and 16 branches in the SPARC64 [6].

The speculation depth has implications on the architecture's configuration. The resources in the execution unit should be dimensioned to accommodate the flow of dispatched instructions resulting from a certain value of the speculation depth. This is an important factor that should be taken into account to obtain a well-balanced architecture. This work quantifies the demands on the machine's resources for different speculations depths. Section 2 of this paper describes the superscalar model considered in the study. Section 3 shows the experiments to assess the resource requirements. The paper concludes in Section 4 with some remarks.

2 Superscalar Model and Experimental Framework

The superscalar model here adopted is representative of existing superscalar microarchitectures. The model is organized as a pipeline with seven stages: *fetch*,

predict, *decode*, *dispatch*, *issue*, *execute* and *writeback*. The fetch stage transfers instructions from the cache memory into a fetch buffer. The predict stage transfers instructions from this buffer into an instruction queue (i-queue), and predicts the result of conditional branches by using a branch target buffer [1]. The dispatch stage sends decoded instructions, in order, from the i-queue to *reservation stations*. The maximum number of instructions that can be dispatched on a cycle will be referred to as the *dispatch width*. The issue stage schedules instructions dynamically using a scheme based on the Tomasulo algorithm [7]. Finally, the writeback stage sends the results of completed instructions to the register file and to the reservation stations. A reorder buffer and a future register file [8] are employed to support speculative execution. Branch instructions are executed, in order, by a specific branch unit. In the case of a misprediction, the branch unit stalls the dispatch stage, recovers the correct architectural state and then resumes instruction dispatch.

In this work we have used a trace-driven simulator to reproduce the operation of our superscalar model. This simulator accepts programs compiled for the SPARC Version 7 architecture [9]. The traces were generated by a simulator of a four-stage pipelined scalar implementation of the SPARC architecture. The instruction latencies adopted are similar to those in the PowerPC 604. We have used eight integer and six floating-point applications from the SPEC92 and SPEC95 benchmarks as the test programs. The traces used in the experiments cover the execution of 20 million of each test program.

3 Effect on the Resource Requirements

In architectures like the one considered here, the number of reservation stations and the size of the reorder buffer determine the maximum dynamic instruction window size. In order to avoid excessive dispatch stalls, these two architectural parameters should be properly adjusted to the flow of instructions resulting from a certain combination of the speculation depth and dispatch width. In another work [10], we present this balance for various machine configurations. Due to the lack of space, here we present only the results concerning the number of reservation stations, for a machine configuration which dispatches four instructions per cycle, with two integer units and one memory unit.

Instruction dispatch can be stalled due to: 1) all reservation stations are occupied, 2) the reorder buffer is full, 3) the speculation depth is insufficient, 4) a mispredicted branch was found 5) the instruction queue is empty. Dispatch stalls due to an insufficient speculation depth occur whenever the number of pending branches is equal to the speculation depth and a new branch instruction arrives for dispatch. We have measured the individual contribution of each one of these stall components.

In Figure 1 we show the occurrence of dispatch stalls when the number of reservation stations per integer unit (*integer reservation stations*) is changed and the number of reservation stations attached to the memory unit (*memory reservation stations*) is fixed at a large value. Dispatch stalls are expressed as

the percentage of *zero-dispatch cycles* within the total cycle count. Each stacked bar gives the percentage of zero-dispatch cycles for a certain number of integer reservation stations. Each group of stacked bars corresponds to a different value of the speculation depth. The percentages are the geometric mean of the individual values presented by the integer test programs.

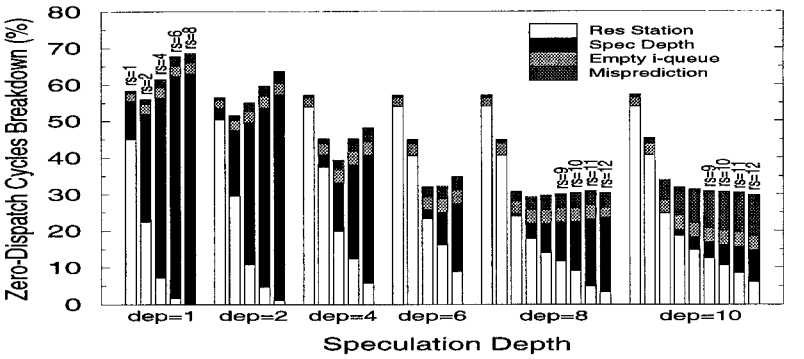

Fig. 1. Varying the number of integer reservation stations

For speculation depths less than or equal to four branches, we observe a high number of dispatch stalls, ranging from 40% to 70% of the total cycle count. The contribution from branch mispredictions does not exceed 4% of the total cycle count, while dispatch stalls caused by an empty i-queue accounts for at most 4%, even for deep speculations.

There is an inverse relationship between the stall components related to the speculation depth and to the number of integer reservation stations. For example, with a four-branch speculation depth, dispatch stalls caused by the lack of integer reservation stations drops from 54% to only 6% when the number of reservation stations is increased. On the other hand, stalls due to an insufficient speculation depth increases from 0.1% to 30%, keeping the total number of stalls still high. This happens because the inclusion of more integer reservation stations increases the number of instructions being dispatched per cycle, including branch instructions. As more branches are dispatched within a time interval, the speculation depth is more frequently reached.

To compensate this effect, one would simply increase the speculation depth. However, with a deeper speculation, more instructions would be dispatched within the same time interval, and the number of dispatch stalls caused by the lack of integer stations would increase again. The balance between the speculation depth and the number of integer stations can not be obtained by simply fixing a value for one of these parameters and then choosing the appropriate value for the other one. Both parameters should be dimensioned together, until the number of dispatch stalls is minimized. Figure 1 indicates that the number

of stalls is minimized for a speculation depth of 8 branches and 6 integer reservation stations. Another minimum was obtained with a speculation depth of 10 branches, but in this case 12 integer reservation stations were required.

Observe that a deep speculation does not always contribute to the reduction of stalls. This comes from the fact that the average dynamic instruction window size increases with the speculation depth, allowing more instructions to be ahead of the mispredicted branch within the window. This increases the number of cycles during which dispatch remains blocked, waiting until the branch reaches the head of the reorder buffer. From a certain value of the speculation depth, this effect starts dominating, avoiding any significant reduction in the number of zero-dispatch cycles. For all the configurations examined, this happens with speculation depths equal or greater than 10 branches.

In order to evaluate the demand of memory reservation stations, we have attributed values for the speculation depth and the number of integer reservation stations that minimize the zero-dispatch cycles. These values were taken from the previous experiments. Memory reservation stations were then added until the corresponding stall component becomes comparable to the components related to the speculation depth and to the number of integer reservation stations. The results are shown in Figure 2. Stalls caused by an insufficient number of memory reservation stations are above 10% of the total cycles when there are less than 14 of these stations. In [10] we have also evaluated machines that can perform two memory accesses simultaneously. In this case, six memory reservation stations are sufficient to keep dispatch stalls to a ratio below 10%.

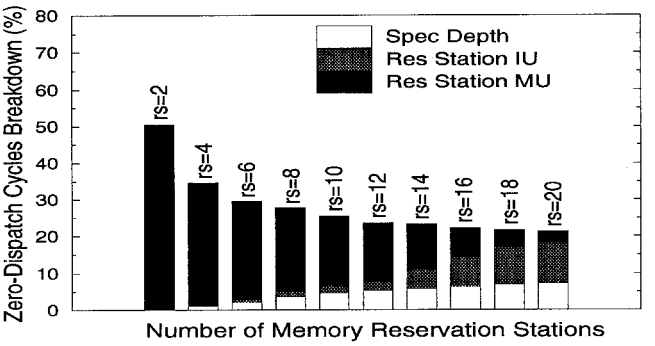

Fig. 2. Varying the number of memory reservation stations.

4 Conclusions

The cost of speculating across a limited number of branches can be much higher than the cost related to speculations along wrong paths. Dispatch stalls involved

with state recovery does not exceed 12% of the total cycles, while in a configuration with a two-branch depth (like the PowerPC 604), the percentage of dispatch stalls caused by an insufficient speculation depth is as high as 30%. Nevertheless, there is a specific value of the speculation depth from which the number of stalls related to state recovery starts dominating, limiting the performance. In the majority of the configurations investigated, this threshold occurred for a 10-branch speculation depth.

The results presented here demonstrate that the potential benefit of increasing the speculation depth can be lost if the resources are not properly balanced. In [10], we have also assessed the role of the speculation depth on the cycle count of the speculative model, relative to a similar but non-speculative model. For integer programs, we have observed an average reduction of 18% in the cycle count for a two-branch speculation depth (as in the PowerPC 604), and of 40% for a speculation depth of four branches (as in the PowerPC 620 and R10000). An average reduction of 48% (and of almost 60% for some integer benchmarks) was obtained with a speculation depth of eight branches.

5 References

[1] Lee, J. K. F., A. J. Smith, *Branch Prediction Strategies and the Branch Target Buffer Design*, IEEE Computer Vol. 17 N. 1, September 1980, pp. 261-294.

[2] Yeh, T.-Y., Y. Patt, *Two-Level Adaptive Training Branch Prediction*, Proc. of the 24th Annual International Symposium on Microarchitecture, 1991, pp. 51-61.

[3] Song, S. P., M. Denman, J. Chang, *The PowerPC 604 RISC Microprocessor*, IEEE Micro Vol. 14 N. 5, October 1994, pp. 8-17.

[4] Diep, T. A., C. Nelson, J. P. Shen, *Performance Evaluation of the PowerPC 620 Microarchitecture*, Proc. of the 22th International Symposium on Computer Architecture, 1995, pp. 163-175.

[5] MIPS Inc., *The R10000 Microprocessor User's Manual*, 1995.

[6] Williams, T., N. Patkar, G. Shen, *SPARC64: A 64-b 64-Active-Instruction Out-of-Order-Execution MCM Processor*, IEEE Journal of Solid State Circuits Vol. 30 N. 11, November 1995, pp. 1215-1226.

[7] Tomasulo, R. M., *An Efficient Algorithm for Exploiting Multiple Arithmetic Units*, IBM Journal of Research and Development Vol. 11 N. 1, January 1967, pp. 25-33.

[8] Smith, J. E., A. R. Pleszkun, *Implementing Precise Interrupts in Pipelined Processors*, IEEE Transactions on Computers Vol. 37 N. 55, May 1988, pp. 562-573.

[9] Sun Microsystems, *The SPARC Architecture Manual, Version 7*, Mountain View, CA, 1987.

[10] *The Effect of the Speculation Depth on the Performance of Superscalar Architectures*, Tech Rep ES415/96, Department of Systems and Computer Engineering, COPPE/UFRJ.