Errata on "Measuring Experimental Error in Microprocessor Simulation"

Rajagopalan Desikan*, Doug Burger†, Stephen W. Keckler†, Llorenc Cruz‡, Fernando Latorre‡, Antonio González‡ and Mateo Valero‡

Computer Architecture and Technology Laboratory
†Department of Computer Sciences
*Department of Elec. & Computer Engineering
The University of Texas at Austin

‡Department d'Arquitectura de Computadors
Universitat Politècnica de Catalunya
Barcelona, Spain

1 History and Summary

This short paper serves to correct the errors contained in the paper entitled "Measuring Experimental Error in Microprocessor Simulation," presented at the 2001 International Symposium on Computer Architecture (ISCA-28) [2]. That paper contained a study of a validated microarchitectural simulator called `sim-alpha`, and included a case study that compared results obtained from a configuration of `sim-alpha` to those reported by Cruz *et al.* [1]. The comparison showed a disparity in the results between the two studies, which was invalid due to an error in the way the operand bypass network of `sim-alpha` was modified.

In this paper, we present a revised comparison that models the bypass network in `sim-alpha` consistently with the work of Cruz *et al.* (henceforth called the reference study). After explaining the bypassing issues in detail, we show that the new comparison between the revised results and the reference study shows similar trends, thus validating the accuracy of the reference study.

In addition, we (the authors of the `sim-alpha` study) would like to state explicitly that we regret any professional harm caused to the authors of the reference study (Cruz, González, Valero, and Topham). The original intent was to show that different simulators modeling similar targets could cause different conclusions to be drawn, thus emphasizing the need for consistency across simulators and research efforts. Since the comparison with the reference study was made with a (modified) validated simulator, and the two sets of results showed a different trend (because of the error in modeling the bypass), it is natural to infer that the reference study was incorrect. We thoughtlessly published the comparison without contacting the authors of the reference study beforehand. We apologize to them both for implying an error in their (correct) methodology and for the discourteous way in which it was presented.

2 Partial bypassing

The bypass networks from the output of functional units to earlier stages in the pipeline are essential for achieving high ILP. Compared to current-generation machines, future machines with deeper pipelines, wider issue cores, and larger wire/gate delay ratios, have been shown to have unscalable bypass networks [4]. That trend has led some researchers to consider partitioned architectures [3], and others to consider partial bypass networks.

To illustrate different bypass configurations, Figure 1 shows a number of different bypass configurations for a single-issue pipeline, modeled after the Alpha 21264. The stages are Fetch, Slot, Map, Issue, Register Read, Execute, and Writeback. An arc from the end of one stage to the beginning of another indicates a bypass path. Results are forwarded along that path, between the two stages connected by the arc. A full-bypass pipeline is one in which the value just computed in the execute stage can be forwarded to any previous stage between the issue and execute stages.

Figure 1a shows a pipeline with a single-cycle register file access. Since the pipeline supports full bypassing (from the execute "E" stage to the register read "R" stage), instructions B and C, which are both dependent on instruction A, can issue in each of the two cycles following the issue of instruction A.

Figure 1b shows a pipeline, with full bypass, for which the register file takes two cycles to access. Completed instructions may be bypassed to any of the intermediate latches, allowing consumer instructions B and C to again issue in the two cycles succeeding producer instruction A. We will refer to this organization as `FB`, for "full bypass".

Figures 1c and d show two different partial bypassing pipelines, in which the set of bypass paths is incomplete, restricting the cycles during which consumer instructions may be issued while a producer instruction is in flight. Both pipelines incur two-cycle register read delays, and each pipeline supports bypassing to one of those two stages. Figure 1c shows the partial bypass scheme measured by Cruz *et al.* [1]. The bypass path goes to stage R1, but not stage R2. Thus, a consumer instruction B cannot issue in the cycle immediately following the issue of the producer instruction A, as shown by the grey-shaded pipeline bubble. This scheme has the advantage that the issue control logic and scheduler are simple–once an instruction is woken up, it will always be possible to issue it. We call this scheme `PB-R1`.

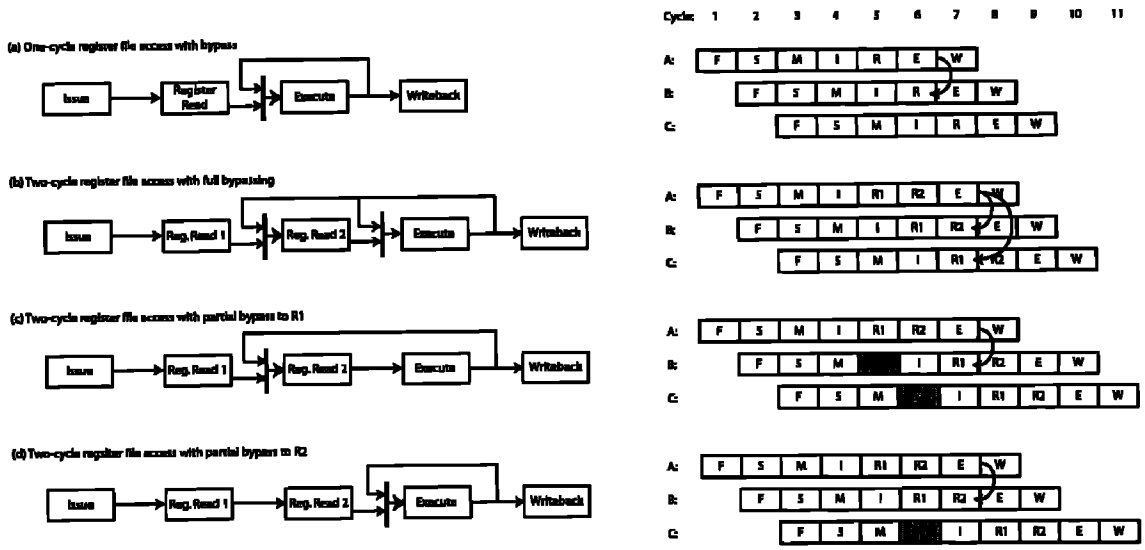

Figure 1: Different bypass implementations

In Figure 1d, we show another possible partial bypass configuration, called PB-R2. In this configuration, computed results may be bypassed to the last stage of the register read, but not the first. This version of partial bypassing will offer higher performance (as dependent, single-cycle instructions may be issued back-to-back), at the cost of increased scheduler complexity. Schedulers are simpler to design if an instruction is always ready to issue once it is first ready to issue. Scheduler complexity is increased if it has to support an instruction being ready to issue, then not ready, then ready again as in PB-R2. The consumer instruction B is shown to issue in the cycle following the issue of its parent instruction, but then in the next cycle, a one-cycle bubble must be inserted into the pipeline before C can be issued, as the result of A cannot be forwarded to stage R1.

Finally, we note that organization (d) is not feasible if the bypass network itself consumes the bulk of a clock cycle. If the bypass network takes a cycle, then the most aggressive bypassing organization possible would resemble organization (c) in terms of pipeline bubbles and issue restrictions.

3 Revised results

Figure 2 summarizes the set of experiments originally published in the sim-alpha paper [2]. The numbers presented herein are higher than those reported previously [2], due to fixes intended to increase the simulator accuracy, but they do not differ qualitatively. For brevity, we show only three harmonic means, instead of individual benchmarks. The bars represent the means of the five integer benchmarks, the five floating point benchmarks, and all ten benchmarks used, respectively.

We implemented a partial bypass scheme, on an 8-wide issue configuration, and compared the performance trends across a one-cycle register file delay, a two-cycle register file delay with full bypassing, and a two-cycle register file delay with partial bypassing. In the sim-alpha paper, a significant discrepancy in the trends and the differential in the absolute performance of the two simulators was noted. The disparity was not explained, allowing the reader to conclude that it was due to either subtle differences in the organization of the simulated microarchitectures, or inaccurate modeling in one of the two simulators.

The disparity was due to neither factor; it is attributable to the fact that the implemented bypass scheme was PB-R2, instead of PB-R1. The authors of the sim-alpha paper incorrectly interpreted the bypass explanation in paragraph 3, section 2 of [1] to mean the last stage in the register file access, instead of the last stage furthest from the execution units from which the result is forwarded. Thus, in the original sim-alpha implementation, dependent instructions can be issued back-to-back, whereas in the Cruz *et al.* implementation, a minimum one-cycle bubble always exists between a producer instruction and its consumer.

Figure 3 shows the same comparison as in the sim-alpha paper, but with the PB-R1 scheme implemented, and a correct comparison. We note that, with the new comparison, we see similar drops in IPC when the bypassing is restricted to one stage in a two-cycle register file access. This correspondence between the two studies is shown in Figure 4, in which the IPCs of each study have been normalized to their respective base cases of single-cycle register file access. This figure shows that the relative drop due to the restricted bypass is almost exactly identical in the two studies for the integer bench-

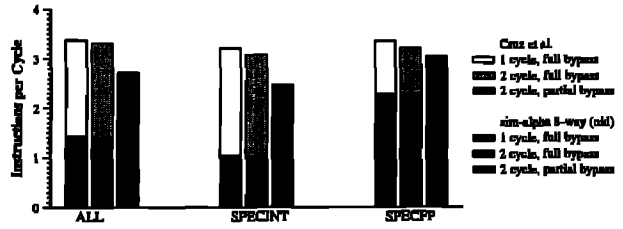

Figure 2: Revision of the original partial bypassing comparison described on p. 267 of the ISCA-28 proceedings [2].

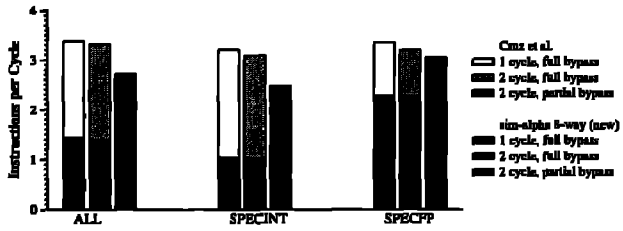

Figure 3: Correct partial bypass comparison.
Figure 3: Correct partial bypass comparison.

marks. The relative drop in floating-point benchmark performance is significantly larger for the `sim-alpha` variant than for the reference study.

4 Discussion

Both simulators in the original comparison simulated what their designers intended them to simulate. However, the partial bypass comparison presented was invalid due to the error in modeling the wrong bypassing configuration in `sim-alpha`. We rectified the comparison by comparing 2-cycle register file accesses with full bypass networks to partial bypassing in which operands are forwarded only to the first stage of the register file access. After the revision, the two simulators both showed identical performance drops for the SPECINT95 benchmarks, and large slowdowns for the SPECFP95 benchmarks.

The motivation for striving for more consistent simulation results across different simulators is not to obtain identical results in terms of absolute performance, but to confirm that added optimizations result in similar performance trends. We saw such a matching performance trend across two simulators in this revised comparison. We have seen other such correspondences; as reported in the original study [2], the `sim-outorder` simulator from the SimpleScalar tool suite shows some similar performance trends to `sim-alpha`. For example, the relative performance drops in the two simulators are consistent when the three-cycle L1 D-cache latency is reduced to one cycle.

A simulator that models all of the complexities in an actual microarchitecture can play a useful role, by providing a lower-level point of comparison to measure trends. However, these

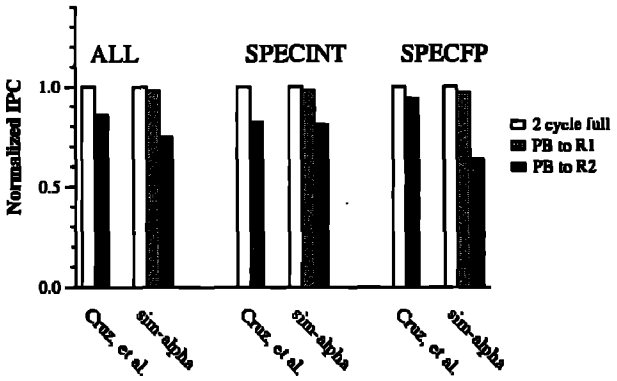

Figure 4: Normalized partial bypass comparison.

complexities of a real implementation may interfere with the evaluation of new, higher-level ideas, particularly because the actual implementations are so carefully tuned. Higher-level simulators such as `sim-outorder` and the reference simulator of Cruz *et al.*, are necessary in addition to validated simulators, as particular implementation choices may affect results as equally as fundamental implementation constraints at the low level. When both levels show similar trends, as in this study, the researcher may be confident that the results gained are not the artifact of an idiosyncrasy of one particular simulator.

Acknowledgments

We thank Mark Hill for his feedback on a draft of this paper.

References

[1] José-Lorenzo Cruz, Antonio González, Mateo Valero, and Nigel P. Topham. Multiple-banked register file architectures. In *Proceedings of the 27th Annual International Symposium on Computer Architecture*, pages 316–325, June 2000.

[2] Rajagopalan Desikan, Doug Burger, and Stephen W. Keckler. Measuring experimental error in microprocessor simulation. In *Proceedings of the 28th Annual International Symposium on Computer Architecture*, June 2001.

[3] R. Kessler, E. McLellan, and D. Webb. The alpha 21264 microprocessor architecture. In *Proceedings of International Conference on Computer Design*, pages 90–105, October 1998.

[4] Subbarao Palacharla, Norman P. Jouppi, and J. E. Smith. Complexity-effective superscalar processors. In *Proceedings of the 24th Annual International Symposium on Computer Architecture*, pages 206–218, June 1997.