RETROSPECTIVE: A First-Order Superscalar Processor Model

Tejas Karkhanis[1] James E. Smith[2]

[1]*Alphabet, Inc.* [2]*University of Wisconsin-Madison (Emeritus)*

I. BACKGROUND

At the time of the 31st ISCA, out-of-order superscalar processors with speculative execution had become a mainstay of high performance processor design. Arriving at an optimal design entailed deciding on benchmarks, coding a cycle accurate trace-driven and/or execution-driven simulator, and performing relatively detailed experiments. Insights on how to optimize the design were extracted post-hoc. In industry, design exploration would often start with a "2X design" configuration of an existing familiar design, where all structures would be scaled by 2X. The design process with only detailed cycle accurate simulation, while necessary, would be time consuming and would not provide insight into the design choice.

II. DEVELOPING THE IDEA

Our hypothesis was that mechanistic models based on the first principles of governing processor dynamics can (1) provide deeper insights into inner workings of an out-of-order superscalar processor, and (2) complement detailed simulation and provide a more efficient design optimization process. Initially, we investigated the inner workings of data cache misses[6] using SimpleScalar[7]. There had been prior attempts at modeling processor core performance, however prior models were computationally and conceptually complex[8, 9, 10] or modeled only a specific part of the processor[11,12].

We started by taking a physics-like experimental approach to determine whether the principle of superposition could be applied to decompose the modeling problem. We conducted a series of simulation experiments where we first removed all miss-events to model ideal execution and then simulated realistic miss-events one at a time. Summing the cycles per instruction (CPI) increments from the individual runs gave results very close to the CPI when all miss-events were simulated. This result was encouraging and allowed us to focus on understanding ideal behavior and the different miss-events independently.

Ideal performance is determined only by the true dependences with the upper limit set by the issue width. We observed that under ideal conditions the critical dependence chain of the instructions in the Reorder Buffer (ROB) determines performance. Previous efforts at understanding performance such as those of Riseman and Foster[11] and Taha and Willis[12], inspired us to look at modeling ideal performance by considering the average critical path length given a ROB-sized set of consecutive instructions.

Our experiments demonstrated that the miss-events were separable for analysis. Different miss-events, however, had different characteristics and needed to be modeled appropriately.

For front-end instruction cache misses, an insight was that the miss penalty is independent of the front-end pipeline; it depends largely on the miss delay.

Front-end branch mispredictions had to be handled differently than instruction cache misses. The transient of draining and refilling the execution backend led to the insight that the branch misprediction penalty is not necessarily equal to the pipeline depth alone; it can be much more than the pipeline depth unlike a common assumption at the time.

The backend miss-event model, exemplified by data cache misses, derived from our previous work on understanding the inner workings of a data cache miss[6]. In that work, we were able to mathematically model the effect of the number of inflight load misses on performance.

III. WHAT WE LEARNT AND WHAT FOLLOWED

With the mechanistic model we were able to explain:

- Why branch predictor accuracy should scale superlinearly as a function of increased back-end capacity, and therefore demonstrate that the "2x design" method is too simplistic.
- How to reduce the instruction cache miss penalty by selectively increasing instruction fetch bandwidth after a cache miss.
- Why the data cache miss penalty decreases significantly as the number of inflight data cache misses increases.

After paper publication, mechanistic modeling experienced an uptick in interest. We received queries from other research groups regarding better versions of the model, applications of the model, and requests to include the modeling process in graduate school courses.

As a next step, we developed an optimization process based on the model insights. By 2005 we were able to arrive at a one-shot model for determining key parameters of a processor configuration based on the issue width and

program characteristics. With this optimization method, we were able to provide application specific configurations for out-of-order superscalar processors. Furthermore, we were able to derive the ratio of issue buffer size to reorder buffer size, which tracked the ratio for actual implementations[13].

Working with Lieven Eeckhout's group in Ghent University, insights from the model were applied to the design of a cost-efficient Performance Counter Unit [1,2,3]. A related "top-down" method is implemented in out-of-order superscalar processors, for example in Intel's processors [4]. "Top-down Analysis" is also supported and implemented in the Linux distributions [5].

IV. ACKNOWLEDGEMENTS

We would like to thank the SRC contract 2000-HJ782, NSF grants CCR-9900610, CCR-0311361, EIA0071924, IBM, and Intel for making this and follow-on work possible.

REFERENCES

[1] S. Eyerman, L. Eeckhout, T. Karkhanis and J. E. Smith, "A performance counter architecture for computing accurate CPI components", ACM SIGPLAN Notices, vol. 41, no. 11, pp. 175-184, 2006.

[2] S. Eyerman, L. Eeckhout, T. Karkhanis and J. E. Smith, "A mechanistic performance model for superscalar out-of-order processors", ACM Transactions on Computer Systems, vol. 27, no. 2, pp. 1-37, 2009.

[3] S. Eyerman, L. Eeckhout, T. Karkhanis and J. E. Smith, "A top-down approach to architecting CPI component performance counters", Micro IEEE, vol. 27, no. 1, pp. 84-93, 2007.

[4] Ahmad Yasin, "A Top-Down method for performance analysis and counters architecture", IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), 2014.

[5] "Top-Down Analysis", https://perf.wiki.kernel.org/index.php/Top-Down_Analysis

[6] T. Karkhanis and J. E. Smith, "A day in the life of a data cache miss", WMPI-2, 2002.

[7] Burger, Doug, and Todd M. Austin. "The SimpleScalar tool set, version 2.0." *ACM SIGARCH computer architecture news* 25.3 (1997): 13-25.

[8] D. J. Ofelt, "Efficient Performance Prediction for Modern Microprocessors," Stanford University PhD Thesis, 1999.

[9] D. B. Noonburg and J. P. Shen, "Theoretical Modeling of Superscalar Processor Performance," *International Symposium on Microarchitecture*, pp. 52-62, 1994.

[10] P. G. Emma and E. S. Davidson, "Characterization of Branch and Data Dependencies on Programs for Evaluating Pipeline Performance," *IEEE Transactions on Computers*, Vol. 36, pp. 859-875, 1987.

[11] E. Riseman and C. Foster, "The Inhibition of Potential Parallelism by Conditional Jumps," *IEEE Transactions on Computers*, vol. C-21, pp. 1405-1411, 1972.

[12] T. Taha and D. S. Wills, "An Instruction Throughput Model of Superscalar Processors," *International Workshop on Rapid Systems Prototyping*, 2003, pp. 156-163.

[13] T. S. Karkhanis. Automated Design of Application-specific Superscalar Processors (pp. 133). PhD thesis, University of Wisconsin, 2006.