Correlation Between State and Control Signals in Out-of-Order Issue Logic

Kenneth Barr Kenneth Conley Serhii Zhak
Advanced VLSI Computer Architecture -- MIT 6.893 Fall 2000
{kbarr, conley, zhak}@mit.edu

Abstract

Current out-of-order control logic is optimized for performance and does not take advantage of energy-saving techniques. Key out-of-order logic structures, including register renaming logic and superscalar issue logic, will be examined for correlation between state and asserted control signals. Logical structures will also be modified to increase the correlation between these signals. By exploiting correlation, predictive or memoization techniques could be implemented in the issue logic to eliminate redundant work and increase energy efficiency. Time permitting, the benefit of these techniques will be analyzed.

1 Introduction

2 Related Work

In dynamically scheduled superscalar processors, the size (and thus delay and power consumption) of the issue logic depends quadratically on the product of instruction issue width and instruction window size. This delay is mainly due to the associative search required by the issue logic [3].

A recent paper [6] modeled the power consumption of an out-of-order superscalar processor at the level of functional blocks. It found that the renaming table, instruction queue, and reorder buffer, all necessary for extracting instruction level parallelism, are responsible for an average of 53% of the total power consumed by a processor. In addition, each of these three units considered individually consumes more energy than any other part of the processor (e.g., cache, branch predictor, functional units, and I/O). To address the power consumption of the instruction queue, the authors proposed a dynamically resized queue which saved 57% of the power consumed by the queue and reduced the total power consumption of the processor by about 15%.

Hiraki [7] proposed a decoded instruction buffer that memoized control signals for execution with power savings of 40% and Wang [14] proposed a new encoding scheme for storing control signals in microprogrammed control units and claimed 4.8%-16.5% reduction in switching activity.

Our study, motivated by such measurements and proposals, will investigate the possibility of bypassing broadcast and select mechanisms in the issue/renaming logic. Since it is typically designed with performance in mind [5], this logic may consume lots of unnecessary power.

Potential for modified renaming schemes has been shown in [13], but we hope to suggest a scheme that works entirely dynamically. Jourdan [8] proposed physical register reuse in a value-identity detection scheme to improve performance, but we hope to reuse *mappings* of logical registers to physical registers in order to reduce energy. We also plan to consider the potential of hardware memoization in the control unit. This technique has been shown to both reduce power and improve performance of various microprocessor structures [4] [11].

Research has also been done at the circuit level for saving power in control logic [9], but these results are not applicable to our research.

3 Methodology and Results

The Simplescalar microarchitectural simulator [1] has been extended to provide detailed per-instruction statistics that explore trends in register renaming. In addition a Java parser was written to condense Simplescalar pipetrace output for ease of analysis. This condensed output may be sorted and filtered with supporting scripts as appropriate.

We have begun our research with an attempt to quantify the upper bounds of correlation between processor state and asserted control signals. As we propose modifications to baseline architecture, it will become necessary to consider the effect of mispeculating control signals.

3.1 Pipetracing

We define a consistent trace as one in which a given instruction spends the same amount of time in each

stage of the pipeline every time it is dynamically executed. Initial analysis of pipetraces show consistency to be disappointingly low (44-72%, with most around 50%).

One source of variation is cold versus warm memory hits. Cold accesses to memory seem to contribute exactly two bad traces to most of the instructions -- a phenomena we have yet to explain. Surprisingly, filtering out cache misses, TLB misses, and branch mispredicts, gives only a 1% improvement in general.

The low consistency may be due to the fact that a lot of the benchmarks have a bimodal distribution, i.e. there are two instruction paths/timings that are dominating dynamic execution. These are so equally balanced in some cases that it's uncanny. We have built a compiler to target the simulator, and hope to determine the cause of this behavior using handwritten test cases as opposed to the pre-compiled Spec benchmarks. If the dual nature of instructions is coincidental, perhaps we can restructure the machine or control logic to force a single trace character for these bimodal instructions so that control logic does not have to make decisions when it sees them.

3.2 Renaming

The Simplescalar RUU scheme [11] ties renaming to all aspects of out-of-order execution. Unfortunately, this makes it difficult to quantify the effects of register renaming or to see how renaming effects other aspects of out-of-order control.

We hope to add a decoupled, parameterized renaming structure or modify the existing Simplescalar scheme. Tests with varying rename structures and policies may reveal helpful trends in asserted logic. Intuition suggests that fewer choices of physical registers would lead to higher correlation at the expense of performance. Perhaps performance would take less of a hit if alternative schemes for determining a mapping were implemented.

Currently we extend Simplescalar to manage a "free list" of physical registers. Also, we have added extensive per-instruction renaming statistics including minimum, maximum, and average length of time a register stays renamed; number of dynamic instances of an instruction; and the number of times an register is renamed to its initial renaming.

Initial results were compiled by examining these statistics after running 500K instructions of a Perl benchmark on a default Simplescalar configuration (with 16 RUU_stations). Figure 1 shows that every instruction needs a physical register for at least 3 cycles, but 80% relinquish their mapping within 7 cycles. This gives hope that a scheme could quickly reassign a mapping. We plan to compare this with average CPI to see if instructions can wait this long to have their registers renamed.

The "remap rate" is defined as the number of times a destination register is mapped to its initial mapping divided by the number of times that instruction is repeated. Figure 2 shows the discouraging results, but there is some good news: we consider a "remap" has occurred only when mappings match the first mapping. Perhaps a two-bit-counter-style or confidence mechanism could enhance this so a system would assign a mapping based on the *majority* of previous mappings or only guess if confidence for a remap is high.

4 Conclusions

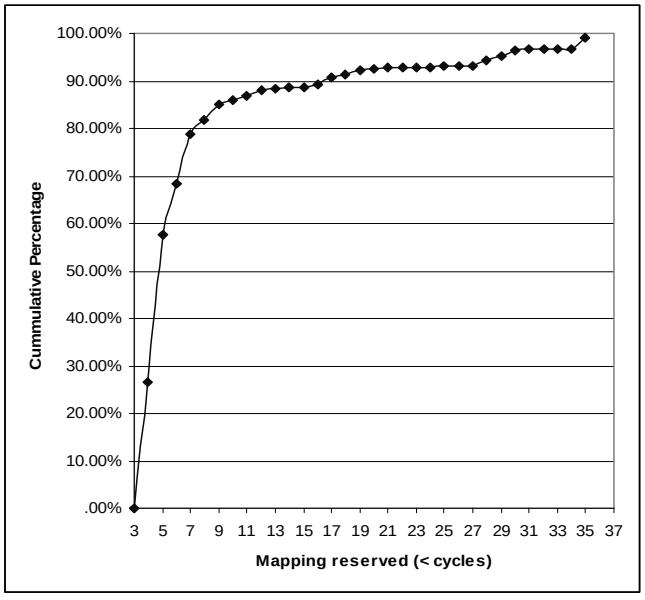

Figure 1

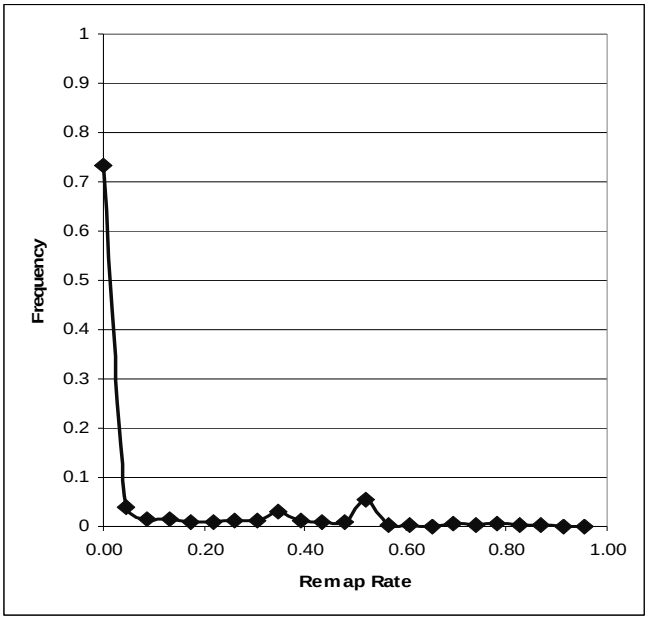

Figure 2

References

[1] D. Burger and T. Austin. The SimpleScalar tool set, version 2.0. *Technical Report CS-TR-97-1342, University of Wisconsin, Madison*. June 1997.

[2] G. Cai. Architectural Level Power/Performance Optimization and Dynamic Power Estimation. In *Proceedings of the CoolChips tutorial, 32nd Annual International Syumposium on Microarchitecture*, 1999.

[3] R. Canal and A. Gonzalez. A Low-Complexity Issue Logic. In *Proceedings of the 14th International Conference on Supercomputing (ICS-00)*, May 8-11, 2000.

[4] D. Citron, D. Feitelson and L. Rudolph. Accelerating Multi-Media Processing by Implementing Memoing in Multiplication and Division Units. *ASPLOS VIII*, October 1998.

[5] J. Farrell and T. Fischer. Issue Logic for a 600-MHz Out-of-Order Execution Microprocessor. *IEEE Journal of Solid State Circuits*. May 1998.

[6] D. Folegnani and A. Gonzalez. Reducing Power Consumption of the Issue Logic. *ISCA-2000 workshop?*

[7] M. Hiraki, R. Bajwa, H. Kojima, D. Gorny, K. Nitta, A. Shridhar, K. Sasaki, and K. Seki. Stage-Skip Pipeline: A Low Power Processor Architecture Using a Decoded Instruction Buffer. *ISLPED*, 1996.

[8] S. Jourdan, R. Ronen, M. Bekerman, B. Shomar, and A. Yoaz. A novel renaming scheme to exploit value temporal locality through physical register reuse and unification. In *Proceedings of the 31st annual ACM/IEEE international symposium on Microarchitecture*, 1998.

[9] U. Ko, A. Hill, and P. Balsara. Design Techniques for High-Performance, Energy-Efficient Control Logic. *ISLPED*, 1996.

[10] P. Landman and J. Rabaey. Activity-Sensitive Architectural Power Analysis for the Control Path. *ISLPED*, 1996.

[11] A. Ma, M. Zhang, and K. Asanović. Way Memoization to Reduce Fetch Energy in Instruction Caches. *MIT*, 2000.

[12] G. Sohi. Instruction Issue Logic for High-Performance, Interruptible, Multiple Functional Unit, Pipelined Computers. *IEEE Transactions on Computer*s, March 1990.

[13] E. Sprangle and Y. Patt. Facilitating superscalar processing via a combined static/dynamic register renaming scheme. In *Proceedings of the 27th annual international symposium on microarchitecture*, 1994.

[14] C. Wang and K. Roy. An Activity-Driven Encoding Scheme for Power Optimization in Microprogrammed Control Unit. *IEEE Transactions on VLSI Systems, Vol 7, No. 1*, March 1999.

Code

All tables, figures, and performance results included in this paper were generated by code produced by the authors. The renaming code, extensions to the Simplescalar simulator, lives on CAG LCS machines in ~kbarr/6.893/ken. The pipetrace parser is in ~conley/893.