

BTB Access Filtering: A Low Energy and High Performance Design

Shuai Wang, Jie Hu, and Sotirios G. Ziavras
Department of Electrical and Computer Engineering
New Jersey Institute of Technology
Newark, NJ 07102
{sw63, jhu, ziavras}@njit.edu

Abstract

Powerful branch predictors along with a large branch target buffer (BTB) are employed in superscalar processors for instruction-level parallelism exploitation. However, the large BTB not only dominates the predictor energy consumption, but also becomes a major roadblock in achieving faster clock frequencies at deep sub-micron technologies. In this paper, we propose a filtering scheme to reduce the accesses to the BTB to achieve a significant dynamic energy reduction in the BTB while maintaining the performance. Our experimental evaluation using the SPEC2000 benchmark suite shows that our BTB Access Filtering (BAF) design achieves a 88.5% dynamic energy reduction over a default 2K-entry 2-way BTB at the cost of a negligible 0.1% performance loss, on the average across all benchmarks. We also studied the leakage behavior and its control in our BAF design. The results show that by applying a drowsy strategy, we can achieve a very effective leakage control in the BTB, a 83% leakage reduction at a marginal 0.3% performance overhead. For high performance design, our BAF can also improve BTB's performance scalability at new technologies. In deeply-pipelined designs, BAF design yields a 2.7% (and 8.1%) performance improvement over a conventional 2-cycle (and 3-cycle) BTB, with its energy efficiency fully exploited.

1 Introduction

Modern high-performance superscalar processor design is mainly driven by techniques exploiting high instruction-level parallelism (ILP) and faster clock frequencies with continuously advancing CMOS technology. Besides out-of-order issue/execution and register renaming, speculative execution is another major form of ILP exploitation. With correct branch predictions, the processor not only eliminates pipeline stalls due to control hazards, but also enables the datapath front-end to supply sufficient instructions for ILP exploitation at later pipeline stages. However, a mispredicted branch requires flushing from the datapath pipeline all instructions fetched along the speculated path and refilling the pipeline with new instructions from the resolved

target address. Therefore, the branch misprediction penalty is recognized as a major performance limiter in speculative superscalar processors. A highly accurate branch predictor is of critical importance to the design of ILP processors, which has been the focus of tremendous research efforts [21][24][14][12]. As the direction predictor is getting more and more sophisticated, a large BTB [18] is usually adopted to supply target addresses for predict-taken branches, leading to non-trivial energy consumption in branch predictors [5][16]. Consequently, energy optimization in BTB is becoming an indispensable component in the design of energy-aware processors at deep sub-micron technologies.

As the logic depth of the pipeline stage keeps reducing [9] at deeply-pipelined designs for higher clock frequencies, operations in many conventional large monolithic structures such as the issue queue and register file, can no longer be completed within a single pipeline stage, eventually leading to reduced performance and significantly increased design complexity. Therefore, plenty of research has been devoted to exploring complexity-effective issue queue and register file designs, e.g., [15][6][4][23][17][22] among many others. On the other hand, research on branch prediction has traditionally focused on direction prediction [21][24][14][12], etc. There is limited work on the energy optimization in branch predictors [10][16][11][5][19], and very few on the complexity and performance scalability of predictor designs. Moreover, once the major datapath components adopt such complexity-effective designs, the branch predictor, especially a large BTB, may eventually become a performance hindrance due to its monolithic structure. It is then important to explore new scalable BTB designs for high-performance processors designed at new technology generations.

Based on the study of the distribution of the dynamic branch direction (taken/not-taken), we propose a filtering scheme controlled by the branch direction predictor to reduce the accesses to the BTB. In order to maintain the performance, we introduce an additional small BTB structure into our filter, which further reduces the accesses to the BTB. For leakage control, the drowsy scheme becomes more effective in our BAF design, because the access filtering makes the BTB inactive during most periods of the time. Furthermore, our BAF design aims to provide a low-complexity scalable design supporting higher clock fre-

quencies at new technology generations. Although a multi-level BTB was initially proposed and evaluated in [18] for performance improvement, it was not considered as a practical implementation option at that technology generation. In our BAF microarchitecture, the large BTB is rarely accessed because of the filtering effect. Therefore, the performance degradation caused by the long access latency in the large BTB at new technology generations can be amortized.

In [16], the prediction probe detector (PPD) detects the non-branch instructions in each cache line to avoid unnecessary accesses to the branch predictor and BTB. However, for completely energy saving, the PPD needs to be accessed before the branch predictor which may increase the pipeline latency. Moreover, since each PPD entry is corresponding to a cache line in the instruction cache, it will increase the energy consumption and design complexity especially with a set-associative instruction cache. Different from PPD, our BAF targets at the removal of the unnecessary accesses (for all predicted not-taken branches) to the BTB for energy saving. Further, the BAF design only incurs a negligible performance loss and slight changes in hardware implementation. An adaptive scheme to resize the BTB according to its on-demand utilization was proposed in [11] to reduce energy consumption. This adaptive scheme requires a profiling phase to collect utilization information and relies on the software to apply the resizing. In contrast, our BAF is a pure hardware implementation that requires no changes to the compiler or the application program, which makes BAF transparent to the software layer. Our experimental results show that the BAF design can achieve 88.5% dynamic energy reduction with only 0.1% performance loss.

The rest of the paper is organized as follows. The next section presents the experimental evaluation framework for this work. In Section 3, we conduct a detailed study on the energy-performance tradeoffs in conventional BTB designs. We propose our BAF design in Section 4. The experimental results and analysis are presented in Section 5. Finally, Section 6 concludes this work.

2 Experimental Setup

We derive our simulators from SimpleScalar V3.0 [3] to model a contemporary high-performance microprocessor similar to Alpha 21364. Table 1 gives the detailed configuration of the simulated microprocessor. To evaluate the energy efficiency and performance scalability of our BAF design, a modified version of the Wattch power model [2] is used for power profiling (at 70nm technology) during the simulation.

For experimental evaluation, we use SPEC CPU2000 benchmark suite compiled for the Alpha instruction set architecture using “-arch ev6 -non_shared” option with “peak” tuning. We use the reference input sets for this study. Each benchmark is first fast-forwarded to its early single simulation point (*gap* and *ammp* use the standard single simulation point instead of the very large early single simulation point) specified by SimPoint [20]. We use the last 100 million instructions during the fast-forwarding phase to warm-up the caches if the number of skipped instructions

Table 1. Parameters of simulated processors.

Processor Core	
Datapath Width	4 inst. per cycle
Int/FP Issue Queue	20/15 entries
Load/Store Queue	64 entries
Active list (ACL)	80 entries
Int/FP Register File	80/72 registers
Function Units	4 IALU, 2 IMULT/IDIV 2 FALU, 1 FMULT/FDIV/FSQRT 2 MemPorts
Branch Predictor	
Branch Predictor	Alpha 21264 tournament predictor 32-entry RAS
BTB	2048-entry 2-way
Memory Hierarchy	
L1 I/DCache	64KB, 2 ways, 64B blocks, 2 cycles
L2 UCache	4MB, 8 ways, 128B blocks, 12 cycles
Memory	225 cycles first chunk, 12 cycles rest
TLB	Fully-assoc., 128 entries

is more than 100 million. Then, we simulate the next 100 million instructions in detail.

3 Energy-Performance Tradeoffs in BTBs

Due to the critical importance of speculative execution in ILP exploitation, modern high-performance superscalar processors usually adopt large size BTB designs, e.g. Intel has increased the 256-entry BTB in Pentium processors to a 4096-entry BTB in Pentium 4 processors [8]. To better understand the impact of BTB on the processor’s performance and energy behavior, we first perform an experimental study by varying the BTB configuration. The results presented in Figure 1 (a) shows that shrinking a default 2048-entry 2-way BTB (2K-2W) to a direct-mapped 32-entry BTB (32-DM) increases the BTB miss rate from 8.8% (3.0%) to 28.8% (7.8%), on the average for integer (floating-point) benchmarks. For integer benchmarks, halving the BTB size from the default one only causes a minor performance loss of 0.16%. Further reducing the BTB size, the performance loss due to increased BTB miss rate starts getting noticeable. A small 128-entry direct-mapped BTB incurs a 3.1% performance loss over the default one and the smallest BTB (32-entry DM) causes a significant performance loss of 8.0% as shown in Figure 1 (b). Figure 1 (b) also shows that floating-point benchmarks are much more insensitive to the BTB size. On the other hand, Figure 1 (c) indicates that smaller BTBs substantially reduce the BTB energy consumption for both integer and floating-point benchmarks. The overall impact on the processor energy is given in Figure 1 (d). As we can see, integer benchmarks achieve the lowest energy consumption with a 128-DM BTB (which will be selected as the filter buffer in our following study). Further reducing the BTB size increases the overall processor energy because of the considerable performance loss. Due to their performance insensitivity to

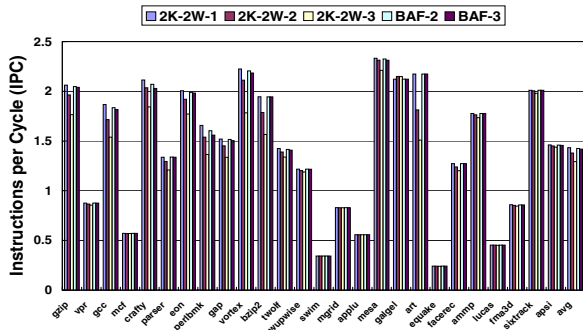


Figure 7. The performance scalability of the BAF.

Our results presented in Figure 7 show that 1) 2-cycle (2K-2W-2) and 3-cycle (2K-2W-3) conventional BTBs incur significant performance losses of 4.4% and 12.0% for integer benchmarks compared to an ideal 1-cycle BTB (2K-2W-1); 2) BAF with a 2-cycle (BAF-2) or 3-cycle BTB (BAF-3) only lose 0.8% or 1.6% performance compared to the ideal BTB, an average for integer benchmarks; and 3) for floating-point benchmarks, BAF introduces an even more negligible performance loss of 0.04% (0.1%), compared to the 1.6% (3.6%) performance loss in the conventional BTBs when the access latency is increased to 2 (3) cycles. These results further confirm that BAF can be a performance-/complexity-effective design for next generation processors.

6 Conclusions

In this work, we focus on the branch target buffer (BTB) design that dominates the energy consumption of the branch prediction unit. Based on our detailed study on the performance and energy tradeoffs of conventional BTBs and the observed filtering effect of the branch direction predictor, we proposed a BTB Access Filtering (BAF) design to significantly reduce the dynamic energy consumption in the BTB at a minimum performance overhead. The experimental results show that 1) our BAF can achieve 88.5% dynamic energy reduction at a performance loss of only 0.1%, and 2) BAF also effectively addresses the leakage energy issue as well as the performance-scalability issue in deeply pipelined datapath designs at deep submicron technologies. The successful design of our BAF has also confirmed us that an effective filtering scheme exploiting microarchitecture-level characteristics can be of great value in harnessing the power/energy issues in major processor components raised by new generation high-performance processor designs.

References

[1] E. Borch, E. Tune, S. Manne, and J. Emer. Loose loops sink chips. In *Proc. of HPCA-8*, pages 270–281, February 2002.
 [2] D. Brooks, V. Tiwari, and M. Martonosi. Watch: a framework for architectural-level power analysis and optimizations. In *Proc. International Symposium on High-Performance Computer Architecture*, 2000.

[3] D. Burger and T. M. Austin. The simplescalar tool set, version 2.0. Technical Report 1342, Computer Sciences Department, University of Wisconsin, 1997.
 [4] R. Canal and A. Gonzalez. Reducing the complexity of the issue logic. In *Proceedings of 2001 International Conferences on Supercomputing*, June 2001.
 [5] Y.-J. Chang. Lazy btb: reduce btb energy consumption using dynamic profiling. In *ASP-DAC '06: Proceedings of the 2006 conference on Asia South Pacific design automation*, pages 917–922, 2006.
 [6] D. Ernst, A. Hamel, and T. Austin. Cyclone: a broadcast-free dynamic instruction scheduler selective replay. In *Proceedings of the 30th Annual International Symposium on Computer Architecture*, June 2003.
 [7] K. Flautner, N. Kim, S. Martin, D. Blaauw, and T. Mudge. Drowsy caches: Simple techniques for reducing leakage power. In *Proc. the 29th International Symposium on Computer Architecture*, Anchorage, AK, May 2002.
 [8] G. Hinton, D. Sager, M. Upton, D. Boggs, D. Carmean, A. Kyker, and P. Roussel. The microarchitecture of the pentium 4 processor. *Intel Technical Journal*, Q1 2001 Issue, Feb. 2001.
 [9] M. S. Hrishikesh, D. Burger, S. W. Keckler, P. Shivakumar, N. P. Jouppi, and K. I. Farkas. The optimal logic depth per pipeline stage is 6 to 8 fo4 inverter delays. In *Proceedings of the 29th Annual International Symposium on Computer Architecture*, May 2002.
 [10] Z. Hu, P. Juang, K. Skadron, D. Clark, and M. Martonosi. Applying decay strategies to branch predictors for leakage energy savings. In *Proc. 2002 Int'l Conf. Computer Design*, pages 442–445, Sep. 2002.
 [11] M. C. Huang, D. Chaver, L. Pinuel, M. Prieto, and F. Tirado. Customizing the branch predictor to reduce complexity and energy consumption. *IEEE Micro*, 23(5):12–25, Sept/Oct 2003.
 [12] D. A. Jimnez and C. Lin. Dynamic branch prediction with perceptrons. In *HPCA '01: Proceedings of the 7th International Symposium on High-Performance Computer Architecture*, page 197, 2001.
 [13] S. Kaxiras, Z. Hu, and M. Martonosi. Cache decay: Exploiting generational behavior to reduce cache leakage power. In *Proc. the Int'l Symposium on Computer Architecture*, 2001.
 [14] S. McFarling. Combining branch predictors. Technical report, WRL Technical Note TN-36, 1993.
 [15] S. Palacharla, N. P. Jouppi, and J. Smith. Complexity-effective superscalar processors. In *Proceedings of the 24th Annual International Symposium on Computer Architecture*, pages 206–218, June 1997.
 [16] D. Parikh, K. Skadron, Y. Zhang, and M. Stan. Power-aware branch prediction: Characterization and design. *IEEE Trans. Comput.*, 53(2):168–186, 2004.
 [17] I. Park, M. Powell, and T. Vijaykumar. Reducing register ports for higher speed and lower energy. In *Proceedings of the International Symposium on Microarchitecture*, Dec. 2002.
 [18] C. Perleberg and A. Smith. Branch target buffer design and optimization. *IEEE Transactions on Computers*, 42(4):396–412, Apr. 1993.
 [19] P. Petrov and A. Orailoglu. Low-power branch target buffer for application-specific embedded processors. In *DSD '03: Proceedings of the Euromicro Symposium on Digital Systems Design*, page 158, 2003.
 [20] T. Sherwood et al. Automatically characterizing large scale program behavior. In *Proc. of ASPLOS X*, October 2002.
 [21] J. E. Smith. A study of branch prediction strategies. In *ISCA '81: Proceedings of the 8th annual symposium on Computer Architecture*, pages 135–148, 1981.
 [22] J. Tseng and K. Asanovic. Banked multiported register files for high-frequency superscalar microprocessors. In *30th International Symposium on Computer Architecture (ISCA-30)*, San Diego, CA, June 2003.
 [23] S. Wallace and N. Bagherzadeh. A scalable register file architecture for dynamically scheduled processors. In *Proceedings of the 1996 Conference on Parallel Architectures and Compilation Techniques*, page 179, 1996.
 [24] T.-Y. Yeh and Y. N. Patt. Alternative implementations of two-level adaptive branch predictions. In *19th Annual International Symposium of Computer Architecture*, pages 124–134, Gold Coast, Australia, May 1992.