RETROSPECTIVE:

Improving Direct-Mapped Cache Performance by the Addition of a Small Fully-Associative Cache and Prefetch Buffers

Norman P. Jouppi

Western Research Laboratory
Digital Equipment Corporation, Palo Alto, CA 94301
jouppi@pa.dec.com

Motivation

The work in this paper was initially motivated by the BIPS project at Digital Equipment Corporation's Western Research Lab. This project started in 1988 with the goal of building a processor that could execute over one billion instructions per second [1]. As an early part of the design process, the performance of various possible system configurations were simulated. We found that for many configurations most of the potential performance was being lost in the memory system. This led to an effort to create a more efficient memory system given the expected resources (i.e., transistor counts, pin bandwidth, etc.) that would be available.

Just prior to this work, Mark Hill had been investigating tradeoffs involving cache set-associativity at the University of California at Berkeley. Mark classified cache misses into the now famous three C's: conflict, capacity, and compulsory misses [2]. Mark also showed that direct-mapped caches could give better overall system performance than set-associative caches in some situations because they had a faster access time.

Many advances in science are the result of new methods of measuring things. In this case, Mark Hill's measuring of the causes of cache misses led to my investigations of ways to reduce each type of cache miss. The miss caches and victim caches presented in the paper were proposed to reduce conflict misses, assuming a direct-mapped cache was used for its smaller access time. Stream buffers were proposed as a method for reducing primarily compulsory and capacity misses. The net result was a memory system with higher performance for a given transistor count.

Elaboration

Immediately after the 1990 ISCA paper I wrote a follow-up paper and submitted it to ASPLOS-III. This paper showed four things. First, it presented an enhancement to stream buffers which could supply prefetch data from any position in the stream buffer, resulting better utilization of data in the stream buffer and higher stream buffer hit rates. Second, it demonstrated that stream buffers were more effective than any of the hardware prefetching techniques that had been discussed in Smith's cache survey [7]. Third, it showed that stream buffers were more effective in reducing cache misses than victim caches. Fourth, and perhaps most interesting, it presented the effective cache size increase resulting from adding multiple stream buffers to a baseline cache design. In many situations, the combination of baseline cache and stream buffers could give miss rates equivalent to those of caches many times larger than the baseline design. Since stream buffers can eliminate compulsory misses, for some larger caches there was no cache size that had miss rates as low as the baseline cache plus stream buffers. Just as adding stream buffers can make a cache appear larger, adding victim caches can effectively provide fractional amounts of cache associativity (e.g., a miss rate equal to a 1.2-way set associative cache). This paper was not accepted to the conference and I never revised it or sent it anywhere else. As part of writing this retrospective I've turned it into WRL tech note TN-53 and put it on the web (see http://www.research.digital.com/wrl/techreports/pubslist.html).

One point of confusion for the reviewers of the ASPLOS submission was something that has come up many times since. In the original ISCA paper, it was not directly and explicitly stated in the text of the paper that data values in the stream buffer or victim cache are not available for use by the processor in the same cycle that data accessed from the cache would be available. However, all the block diagrams in the paper show that the victim caches and stream buffers are only connected to the memory refill side of the caches, and the text states that a cache line can be transferred from the stream buffer or victim cache to the primary cache in the cycle following the cache miss. Since the appearance of the ISCA paper, many people have assumed that a large multiplexor exists in the cache access path of the processor, and that this large multiplexor can select between the result of the cache probe and the contents of a stream buffer or victim cache all within the cache access time. This is certainly not the case, as can be seen from the diagrams in the paper. If there was such a large multiplexor, the access of the primary cache (direct-mapped in these examples) would become much slower. By placing a multiplexor on the cache refill path it is moved out of the critical cache access path at the cost of another cycle of delay. This cycle of delay is much shorter than a full cache miss penalty, so misses served from a stream buffer or victim cache still result in a significant win.

Evolution

Stream buffers and miss or victim caches have appeared in a number of systems, although their use is by no means widespread. The most compelling application remains prefetching instructions with instruction stream buffers; this is because of the highly sequential nature of instruction miss streams. Instruction prefetch buffers have appeared in a number of microprocessor designs.

One data-side application was a combined 2KB miss cache and stream buffer which was placed on the HP PA7100 microprocessor [3] while the primary caches remained off-chip. In this system they wanted large off-chip caches for good performance on large application programs. Off-chip caches are difficult to make set-associative because of limited microprocessor pin counts, and there was not enough space on the die for large caches, so they placed a combined prefetch buffer and miss cache on-chip instead. Because the combined miss cache and prefetch buffer was on chip, it can be accessed in parallel with the off-chip cache.

The Cray T3D and T3E also used stream buffers on the data side, as a replacement for a secondary cache. Because many real numeric applications use non-unit strides, support for non-unit stride prediction and allocation filters to reduce the memory bandwidth requirements were studied by Palacharla and Kessler [4]. As a result of this study allocation filters were implemented in the Cray T3E.

More recently, stream buffers have been studied in the context of more modern processor designs. Farkas et. al [5] found that stream buffers were useful in statically scheduled processors with non-blocking loads and speculative execution. Although dynamically scheduled processors are better at tolerating unpredictable memory latency than statically scheduled processors, they too can benefit from stream buffers as Farkas et. al. showed in [6].

In [6] improved per-load stride prediction and a variation of stream buffers called incremental stream buffers were also proposed. Incremental stream buffers do not attempt to fill the whole stream buffer on a miss, but rather extend the number of lines they try to prefetch if earlier prefetches are used. This reduces the amount of bandwidth wasted in cases where prefetching far beyond the initial miss is not useful. However, it is much better for short streams than methods that require several subsequent misses before allocating a stream buffer. Moreover, for long streams the additional startup overhead is insignificant.

Futures

Although miss caches and victim caches are typically more popular ideas than stream buffers, stream buffers are a more important and lasting contribution. Even with the system configurations of the initial paper, stream buffers made a larger contribution to system performance. As caches get larger, the percentage of misses which are due to conflicts goes down while the percentage due to compulsory misses go up. This further increases the importance of stream buffers at the expense of miss and victim caches. With technology scaling, the latency ratio between off-chip and on-chip cache access increases. This makes sacrificing some cache hit speed for increased cache hit rates (i.e., implementing true cache set-associativity) even more worthwhile. Finally, more recent system innovations such as dynamic scheduling which reduce temporal and spatial locality of reference streams require set-associative caches for good performance.

As technology integration increases, I believe stream buffers will still be important for obtaining the best system performance for ever-limited cache resources. And I believe there is still room for further advances in hardware prefetching.

Acknowledgments

Keith Farkas provided helpful comments on a draft of this retrospective as well as insightful work on stream buffer enhancements.

References

[1] Norman P. Jouppi, et. al., "A 300MHz 115W 32b Bipolar ECL Microprocessor," in the *IEEE Journal of Solid-State Circuits*, November 1993.

[2] Mark D. Hill, *Aspects of Cache Memory and Instruction Buffer Performance*, Ph.D. Thesis, University of California Berkeley, 1987.

[3] Ehsan Rashid, et. al., "A CMOS RISC CPU with On-Chip Parallel Cache", in the *Proceedings of the 1994 International Solid-State Circuits Conference*, pages 210-211.

[4] Subbarao Palacharla and Richard Kessler, "Evaluating Stream Buffers as a Secondary Cache Replacement", in the *Proceedings of the 21st International Symposium on Computer Architecture*, pages 24-33, April 1994.

[5] Keith Farkas, Norman P. Jouppi, and Paul Chow, "How Useful are Non-blocking Loads, Stream Buffers, and Speculative Execution in Multiple Issue Processors?" in the *Proceedings of the 1st Conference on High-Performance Computer Architecture*, January, 1995.

[6] Keith Farkas, Paul Chow, Norman P. Jouppi, and Zvonko Vranesic, "Memory-System Design Considerations for Dynamically-Scheduled Processors" in the *Proceedings of the 24th Annual International Symposium on Computer Architecture*, June 1997.

[7] A. J. Smith, "Cache Memories", *ACM Computing Surveys*, vol. 14, no. 3, pp. 473-530, 1982