Reducing Energy and Delay Using Efficient Victim Caches*

Gokhan Memik
Department of Electrical Engineering
UCLA
memik@ee.ucla.edu

Glenn Reinman
Computer Science Department
UCLA
reinman@cs.ucla.edu

William H. Mangione-Smith
Department of Electrical Engineering
UCLA
billms@ee.ucla.edu

ABSTRACT
In this paper, we investigate methods for improving the hit rates in the first level of memory hierarchy. Particularly, we propose victim cache structures to reduce the number of accesses to more power consuming structures such as level 2 caches. We compare the proposed victim cache techniques to increasing the associativity or the size of the level 1 data cache and show that the enhanced victim cache technique yield better energy-delay and energy-delay-area products. We also propose techniques that predict the hit/miss behavior of the victim cache accesses and bypass the victim cache when a miss can be determined quickly. We report simulation results obtained from SimpleScalar/ARM modeling a representative Network Processor architecture. The simulations show that the victim cache is able to reduce the energy consumption by as much as 17.6% (8.6% on average) while reducing the execution time by as much as 8.4% (3.7% on average) for a set of representative applications.

Categories and Subject Descriptors
B.3.2 [**Hardware**]: Memory Structures – *Design Styles*. C.3 [**Computer Systems Organization**]: Special-Purpose and Application-Based Systems – *Embedded Systems*.

General Terms
Design, Measurement, Performance.

Keywords
Victim caches, Miss Detection, Network Processors.

1. INTRODUCTION
In this paper, we present hardware techniques to reduce microprocessor energy consumption. The proposed techniques improve the efficiency of first level memory hierarchy by utilizing victim caches. Therefore, accesses to the larger and more power consuming structures are prevented, which reduces the execution time of applications and the power consumption.

The proposed techniques can be applied to various multi-processor and single core systems. However, we show the effects of our techniques for a generic chip-multiprocessor representing Network Processors (NPUs). Power consumption has already become a first rate design criteria for high-performance processors. Heat dissipation affects the performance and reliability of processors. The reasons for selecting NPUs as our target platform are twofold. First, NPUs have high power consumption with "power/area" ratios reaching or even surpassing that of desktop processors. Hence, heat dissipation is likely to become a very important problem for next generation NPUs. Second, most NPUs follow the single-chip multiprocessor design methodology [9]. The architectural details vary widely among different NPUs. However, most NPUs employ multiple execution cores and these cores are usually connected by a global system bus. The system bus usually consumes a significant portion of the overall energy, because the capacitive load on the core's input/output drivers is usually much larger than that on the internal nodes [14]. Such processor configurations can benefit immensely from the proposed techniques. As the number of cores in the processor is increased, the need for reducing the bus load also increases. Therefore, the complexity of utilizing the proposed techniques is clearly justified in processors with several cores.

For chip-multiprocessors, the area of the local caches is one of the most important design criteria. Therefore, many NPUs use direct-mapped caches. Although direct-mapped caches have several advantages, they may perform poorly due to conflict misses. Several techniques have been proposed by prior work to improve the performance of the direct-mapped caches [5]. Arguably, the most influential of these techniques is the victim cache proposed by Jouppi [7]. Victim caches may be an attractive alternative solution for multi-core processor compared to increasing the size and/or associativity of the local caches because of their area efficiency. In this paper we present different victim cache enhancements to improve the level 1 data cache efficiency.

This paper is organized as follows. In the following section, we summarize the related work, where we also present the victim cache technique. Section 3 shows the prediction mechanisms. In Section 4, we present the simulation results. Section 5 concludes the paper with a summary.

2. RELATED WORK
Several techniques have been proposed to reduce the energy consumption of high-performance processors [1, 4, 8]. Some of these techniques use small energy-efficient structures to capture a portion of the working set, thereby filtering the accesses to larger structures. Others concentrate on restructuring the caches [1]. Victim caches are also studied as a means to reduce the energy consumption in high-performance processors [3].

In the context of multiple processor systems, Moshovos et al. [13] propose filtering techniques for snoop accesses in the SMP servers. We use filtering of the accesses originating from the core rather than the bus. Some of the techniques presented in this paper were used to detect misses to data caches in a processor with multiple cache levels [11]. In this paper, we introduce new techniques and enhance some of the previously proposed techniques to be utilized to predict misses in victim caches.

Victim caches were proposed by Jouppi [7] to reduce the conflict misses in caches with low degrees of associativity. In the original proposal, the victim cache is placed between the primary cache

ISLPED'03, August 25-27, 2003, Seoul, Korea.

* This work is in part supported by the NSF Grant ECS-0225379.

and the next level of memory hierarchy. On a primary cache miss, the victim cache is accessed. If the data is found in the victim cache, it is promoted to the primary cache and the replaced block is placed into the victim cache. If the data is not found in the victim cache, the level 2 cache is accessed and the arriving block is placed directly into the primary cache. Meanwhile, the block evicted from the primary cache (replaced block) is placed into the victim cache. Bahar et al. [2] propose concurrent access of the primary cache and the victim cache. In our work, this configuration is called *parallel victim cache (PVC)*, whereas the original configuration is called *serial victim cache (SVC)*.

Figure 1 shows two execution cores with different victim cache configurations. Execution core 1 implements PVC where the victim cache and the level 1 cache are accessed concurrently. Execution core 2 implements SVC, where the victim cache is probed only after level 1 cache misses. It also shows the location of our new bypass function, which is used to predict the miss accesses to the victim cache and to eliminate the accesses that are determined to miss.

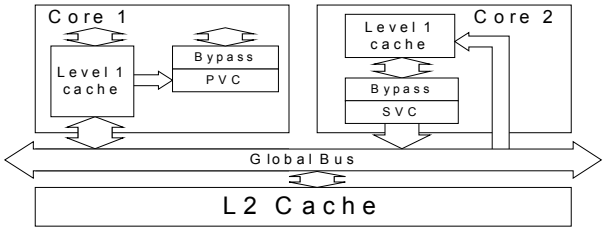

Figure 1. Overview of the victim cache configurations.

3. BYPASS PREDICTION MECHANISMS

We have presented techniques for detecting cache misses [11]. In this paper, we enhance some of these techniques for victim caches. In addition, we present new techniques. There are two categories of prediction techniques: inclusive and exclusive. Inclusive techniques store information about the existing blocks in the victim cache. Exclusive techniques store information about the blocks that are not in the victim cache. The delay and area requirements of each design are presented in Section 3.5. *In all the techniques, the decision to bypass the victim cache access is indicated with a high prediction output.*

3.1 HighLow-Bits Technique

The HighLow-Bits technique identifies the accesses that are going to miss by examining the bit locations that are high (or low) in the address. Specifically, we store the bit locations of the tags that are high (or low) in all the stored blocks. When an access comes to the predictor, we check whether any of the tag bit is set in the new tag and is not set in the stored positions. If so, we know that the access is going to miss at the victim cache. For example, if the victim cache has two blocks with tags `0110` and `1100`, then all the accesses that have the form `x0x1` will miss in the cache. Therefore, by checking the high and low bits, we can quickly eliminate some of the accesses.

The algorithm is implemented using two registers. The first register stores the negation of the OR value of all the tag values in the victim cache. The second register stores the AND value of the tag values. Then, the accessed address is checked against the first register and the negation of the accessed address bits is checked against the second register. If any of the corresponding bits in the registers and the checked address is high, a miss is detected.

3.2 Sum Technique

Sum technique generates a hash value for every tag value. Then, instead of comparing the entire tag, these hash values are compared to the hash value of the accessed address to quickly determine cache misses. Figure 2 presents the specific hash function utilized. The result of the function is sum value that is modified in the for loop. If none of the hash outputs for the cache entries matches the hash output for the accessed address, we can conclude that the access will miss in the cache.

```
for (i = 0; i < SUM_WIDTH; i++) {
        if (tag & 0x1)
                sum += (i + 1);
        tag = tag >> 1;           }
```

Figure 2. Sum hash function

To implement the Sum technique, we utilize an array of flip-flops. Particularly, for every possible outcome, we implement a D flip-flop. These flip-flops store the sum value calculated using the hash function in Figure 2. To check the hit/miss outcome for an access, we again find the corresponding sum value and check whether the value in the corresponding D flip-flop is set. If it is not set, we can determine that the access will miss.

3.3 Table Technique

The final inclusive prediction technique is the Table technique. The Table method stores the least significant N bits of the tag values in the victim cache. If the least significant N bits of the tag of the access do not match any one of the stored values, then the access can be bypassed.

The values are stored in an array of size 2^N bits and the least significant N bits of the tags are used as an address to access the array. The locations corresponding to the victim cache tag values are set to 0 and the remaining locations are set to 1. During an access, the least significant N bits of the tag are used to address this table. The value stored at the corresponding location is used as the prediction (0 means the access should be completed, 1 means the access can be bypassed). In Section 4, we present results for a configuration (8x1) that uses a single table of size 2^8 (N is set to 8).

3.4 Exclusive Predictor Technique

Exclusive predictors store information about the blocks that are not in the victim cache. When a block is accessed in the victim cache and the access results in a miss, the address is stored in an exclusive prediction table. For the PVC, if a block is in the primary cache, accesses to the block will result in victim cache misses. Hence, after the first access, the address is stored in the exclusive prediction table and the consecutive misses to the same block can be captured and prevented.

When a block is placed into the victim cache, the exclusive prediction table should be traversed and if the block address is in the table, it should be removed from it, so that following accesses will not be marked as a miss. In Section 4, we report results for a 32 entry exclusive prediction table.

3.5 Discussion

The prediction techniques discussed in this section are accurate for cache miss predictions. In other words, if the prediction is a miss, then the block certainly does not exist in the victim cache. However, if the prediction is a hit then the access might still miss in the victim cache. The miss predictions should be reliable because the cost of predicting an access will miss when the data is actually in the victim cache is very high, whereas the cost of a hit misprediction is relatively less.

We have measured the delay, energy, and area requirements of each technique. The results are summarized in Table 1. For the HighLow-Bits and Sum techniques, we have implemented RTL level descriptions of the circuits and measured the properties using the Synopsys Design Compiler. For the Table and Exclusive prediction techniques, the CACTI tool is used to find the optimal cache configuration and HSpice is used to simulate the delay. In all calculations, the tag addresses are 32-bit wide.

Table 1. Properties of bypassing mechanisms. The '*gates*' for area in HighLow-Bits and Sum techniques corresponds to an inverter.

Technique	Delay [ns]	Energy [nJ]	Area
HighLow-Bits	0.1984	0.0044	118 gates
Sum	0.3118	0.0125	367 gates
Table	0.3967	0.0231	0.0676 mm^2
Exclusive Pred.	0.4468	0.0220	0.0469 mm^2

4. EXPERIMENTS

We present results for a single core processor much like StrongARM SA-110. The SimpleScalar/ARM is used in the experiments. The necessary modifications to the simulators have been implemented to measure the effects of the victim cache and various bypassing methods.

4.1 Simulation Parameters

The processor is modeled after the StrongARM SA-110 with in-order execution and an issue width of 2. We have selected the cache configurations to model execution cores in NPUs. The base processor does not use victim caches and has 4 KB, direct-mapped L1 data and instruction caches and a 128 KB, 4-way set-associative unified L2 cache. The latency for all L1 caches is set to 1 cycle, and the L2 cache latency is set to 12 cycles. We simulate the applications in the NetBench suite [10].

The victim caches are 8-entry fully associative with 32-byte block size. We report the improvement over the base processor without any victim cache for the serial victim cache (**SVC**), parallel victim cache (**PVC**), the base processor with a 2-way set-associative level 1 data cache (**2-way**), and the base processor with twice the size of the original level 1 data cache (**8 KB**).

4.2 Performance Impact of Victim Caches

First, we investigate the performance impact of victim caches. The results are summarized in Table 2. Overall, increasing the associativity to 2 gives the best results reducing the number of execution cycles by 5.1%. The PVC reduces the execution cycles by 3.6% on average. For most applications, we see that the improvement achieved by PVC is close to the 2-way configuration. For applications such as nat and drr-l, on the other hand, 2-way significantly outperforms the PVC. The reason for this lies in the number of cache misses for these applications. Since they have relatively large number of cache misses, the victim cache is polluted.

4.3 Bypass Prediction Mechanisms

The success of a bypass prediction mechanism is measured in *coverage*. Coverage is the fraction of the victim cache miss accesses that are avoided (bypassed) using the mechanism. The results for the bypass mechanisms for the PVC are summarized in Table 2. In Table 2, we report the coverage for four techniques: HighLow-Bits, Sum (using two arrays, both with SUM_WIDTH set to 10), Table (using a table of 2^8 bits), and Exclusive (using an exclusive prediction table of 32 entries) techniques. On average, the sum technique reduces the number of miss accesses by 98.2%. The HighLow-Bits technique prevents 59.2% of the miss accesses on average, whereas the Table technique prevents the miss accesses by 81.5%. The exclusive prediction mechanisms are also successful in filtering access; on average 90.0% of the miss accesses are prevented by it.

Table 2. Performance improvement for simulated configurations, coverage for bypassing techniques, and reduction in L2 accesses in the embedded processor. The coverages for four bypassing techniques: HighLow-Bits (HighL.), Sum, Table, and Exclusive Predictor (Exclu.), and reduction in the L2 accesses (L2).

Appls.	2-way [%]	8KB [%]	PVC [%]	High L.[%]	Sum [%]	Table [%]	Exclu. [%]	L2 [%]
crc	1.6	0.8	1.4	12.1	96.7	97.3	96.2	56.8
dh	0.0	0.0	0.0	95.4	98.9	95.6	99.9	0.0
drr	7.9	10.4	5.2	62.1	99.2	88.5	91.1	15.4
drr-l	10.4	0.3	6.1	84.2	99.7	95.9	88.6	5.8
ipchains	1.2	0.6	1.2	59.1	95.8	69.9	88.4	10.5
md5	4.5	1.8	4.4	31.0	95.1	83.7	97.7	27.8
nat	11.4	0.2	6.5	83.3	99.5	93.8	88.6	5.4
nat-l	8.2	4.6	4.9	67.5	99.3	91.8	91.9	9.7
rou	1.1	2.3	0.7	81.1	99.5	97.1	88.6	6.1
rou-l	9.3	4.1	5.8	61.5	99.5	95.8	91.6	16.6
snort-l	3.9	5.0	3.6	25.9	97.9	46.7	93.6	19.0
snort-n	1.6	2.0	1.4	50.0	98.4	50.8	91.9	9.1
ssl-m	2.9	5.3	5.1	55.2	99.4	56.1	71.3	23.2
ssl-s	3.8	0.0	2.5	38.8	95.3	47.4	93.0	12.3
ssl-w	2.7	5.4	2.9	53.6	97.2	79.2	92.1	12.0
tl	5.3	0.6	4.0	87.7	99.7	96.8	88.1	5.5
tl-l	13.5	7.8	8.4	76.4	99.7	98.5	90.8	15.7
url	3.0	2.1	1.7	40.2	96.2	82.4	75.9	19.9
average	5.1	3.0	3.7	59.2	98.2	81.5	90.0	15.0

4.4 Energy Measurements

In this section, we report energy improvements for several configurations of the proposed techniques. *We report the improvement in overall energy consumption.* To find the total energy consumption of the processor, we use the energy distribution of the StrongARM components presented by Montanaro et al. [12]. We use the CACTI tool to find the energy consumption of the simulated caches. The energy consumption of the bus is estimated with the model established by Zhang and Irwin [16]. The total energy is the sum of the energy consumed by the level 1 caches (data, instruction and victim), the reminder of the cores, level 2 cache and the system bus.

Figure 3 presents the results for the overall energy reduction. The energy reduction is mostly due to the reduced number of L2 and bus accesses. For each simulated application, the fraction of L2 accesses prevented using the victim cache is presented in the right-most column of Table 2. Overall, the PVC with the Sum bypassing technique is the most efficient in terms of energy consumption, reducing the overall energy consumption by 8.6%. Note that, although the HighLow-Bits mechanism has lower coverage than the Table method, the energy consumption is less with the HighLow-Bits mechanism, because this method consumes the least power among the techniques. Using the energy consumption and the execution time, we first calculate the energy-delay product. The PVC using the Sum technique for bypassing reduces the energy-delay product by 11.8% on average, whereas the 2-way and 8K configurations reduce the energy-delay product by 8.6% and 5.9%, respectively. Finally, we calculate the energy-delay-area product. The area is of the most important design

criteria for chip multiprocessors, hence energy-delay-area product is a good measure for comparing performance of different techniques. According to Montanaro et al. [12], approximately 30% of the area in StrongARM is dedicated to the data caches. We calculate the increase in the total cache size for the different configurations using CACTI. On average, the PVC with the simulated Sum technique reduces the energy-delay-area product by 2.4%, whereas the 2-way and 8 KB configurations increase the energy-delay-area product by 5.4% and 16.8%, respectively.

5. CONCLUSION

Thermal dissipation is becoming one of the most important bottlenecks for high-performance processors. In this paper, we have presented victim cache techniques to improve the efficiency of first level memory hierarchy. By increasing the efficiency of lowest level of hierarchy, the accesses to the global structures that consume large amounts of power are reduced. Although the proposed techniques are most useful in bus-based multi-core systems (such as NPUs), they can be utilized in almost all types of programmable systems. We have also presented techniques to filter victim cache accesses that will result in a victim cache miss. If the victim cache and the primary cache are accessed in parallel, the hit ratio on the victim cache becomes significantly lower. By predicting the victim cache misses, the energy consumption during the misses is prevented. We have presented four different techniques of varying complexity. The simulation results reveal that the techniques are quite successful in filtering misses, reducing the number of misses by as much as 99.9%. These bypassing techniques along with the victim cache reduces the energy consumption of a representative embedded processor by as much as 17.6% and 8.6% on average. This on average corresponds to a reduction of 11.8% in energy-delay product and 2.4% reduction in energy-delay-area product.

REFERENCES

1. Albonesi, D. H. Selective cache ways. In Proceedings of *Int. Symposium of Microarchitecture*, Nov. 1999, Haifa / Israel.

2. Bahar, R. I., B. Calder, and D. Grunwald. A Comparison of Software Code Reordering and Victim Buffers. In Proceedings of *3rd Workshop on Interaction Between Compilers and Computer Architecture*, Oct. 1998.

3. Bahar, R. I., G. Albera, and S. Manne. Power and Performance Tradeoffs using Various Caching Strategies. In Proceedings of *International Symposium on Low Power Electronics and Design*, 1998.

4. Bellas, N., I. Hajj, C. Polychronopoulos, and G. Stamoulis. Architectural and compiler support for energy reduction in the memory hierarchy of high performance processors. In Proceedings of *Intl. Symposium on Low Power Electronics and Design*, 1998.

5. Hennessy, J. L. and D. A. Patterson, *Computer Architecture: A Quantitative Approach*. 1990, San Mateo, CA: Morgan Kaufmann.

6. Intel Corp. SA-110 Microprocessor Technical Reference Manual.

7. Jouppi, N. P. Improving Direct-Mapped Cache Performance by the Addition of a Small Fully-Associative Cache Prefetch Buffers. In Proceedings of *25 Years {ISCA}: Retrospectives and Reprints*, 1998.

8. Kin, J., M. Gupta, and W. H. Mangione-Smith. The Filter Cache: an energy efficient memory structure. In Proceedings of *Intl. Symposium on Microarchitecture*, Dec. 1997, Research Triangle Park / NC.

9. Mangione-Smith, W. H. and G. Memik, *Network Processing: Applications, Architectures and Examples.*, in *Tutorial at International Symposium on Microarchitecture,* Austin / TX. Dec. 2001.

10. Memik, G., W. H. Mangione-Smith, and W. Hu. NetBench: A Benchmarking Suite for Network Processors. In Proceedings of *International Conference on Computer-Aided Design (ICCAD)*, pp. 39-42, Nov. 2001, San Jose / CA.

11. Memik, G., G. Reinman, and W. H. Mangione-Smith. Just Say No: Benefits of Early Cache Miss Determination. In Proc. of *High Performance Computer Architecture*, Feb. 2003, Anaheim/CA.

12. Montanaro, J., *et al.*, *A 160-MHz, 32-b, 0.5-W CMOS RISC microprocessor*. IEEE Journal of Solid-State Circuits, 1996. **31**(11): p. 1703-14.

13. Moshovos, A., G. Memik, B. Falsafi, and A. Choudhary. JETTY: Snoop filtering for reduced power in SMP servers. In Proceedings of *International Symposium on High Performance Computer Architecture (HPCA-7)*, Jan 2001, Toulouse / France.

14. Panda, P. R. and N. D. Dutt. Reducing Address Bus Transitions for Low Power Memory Mapping. In Proceedings of *EDTC-96: IEEE European Design and Test Conference*, pp. 63-67, March 1996.

16. Zhang, Y. and M. J. Irwin. Energy-Delay Analysis for On-Chip Interconnect at System Level. In Proceedings of *IEEE Computer Society Workshop on VLSI*, 1999.

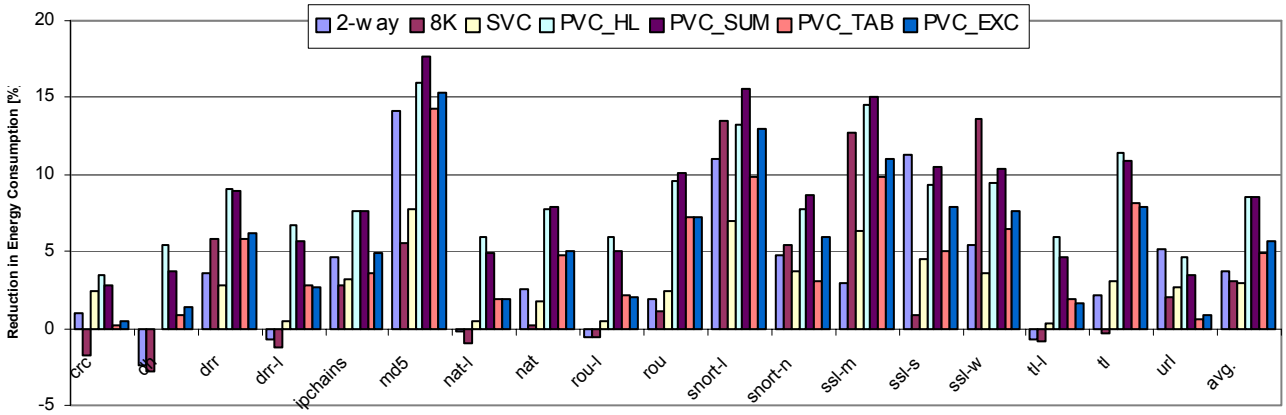

Figure 3. Reduction in Energy Consumption for the embedded processor. (SVC: serial victim cache, PVC: parallel victim cache, HL: HighLow-Bits, SUM: Sum bypassing technique, TAB: Table bypassing technique, EXC: Exclusive prediction technique).