An energy-efficient data cache with byte-repeat pattern encoding

Seungmin Jung, Hyotaek Shim^{a)}, and Seungryoul Maeng

Computer Science Department, KAIST,

335 Gwahangno, Yuseong-gu, Daejeon 305–701, Republic of Korea

a) htsim@camars.kaist.ac.kr

Abstract: The on-chip cache is a significant source of the energy consumption of today's processors. Several data compression techniques including *Frequent Value Caches* are proposed to reduce the energy consumption in the data cache memories. However, the preceding approach has some problems, such as the monitoring time to find the frequent values dedicated for each program and the additional registers to store the frequent values. By studying the behavior of MiBench and MediaBench programs, we observed that many data values stored in the data cache have a few word patterns in which one byte is repeated and/or the rest of bytes are all zeros. These values can be represented with one byte and the pattern type bits. We propose a new energy-efficient data cache, the *Byte-Repeat Pattern Cache*, which employs this encoding scheme.

Keywords: Frequent Value Cache, Byte-Repeat Pattern Cache, data cache memory, low power consumption

Classification: Storage technology

References

- [1] K. Ghose and M. B. Kamble, "Energy Efficient Cache Organizations for Superscalar Processors," *Power Driven Microarchitecture Workshop of 25th International Symposium on Computer Architecture (ISCA)*, Barcelona, Spain, June 1998.
- [2] J. Montanaro, R. T. Witek, K. Anne, et al., "A 160-MHz, 32-b, 0.5-W CMOS RISC microprocessor," *Digital Technical Journal*, vol. 9, no. 1, pp. 49–62, 1997.
- [3] C. Aliagas, C. Molina, M. Garcia, A. Gonzalez, and J. Turbella, "Value Compression to Reduce Power in Data Caches," *Proc. of the 9th International Conference on Parallel and Distributed Computing (Euro-Par)*, Klagenfurt, Austria, pp. 616–622, Aug. 2003.
- [4] Y. Zhang, J. Yang, and R. Gupta, "Frequent Value Locality and Value-Centric Data Cache Design," *Proc. of the 9th International Conference on Architectural Support for Programming Languages and Operating Systems (ACM ASPLOS)*, Cambridge, MA, USA, pp. 150–159, Nov. 2000.
- [5] J. Yang and R. Gupta, "Energy Efficient Frequent Value Data Cache Design," *Proc. of the 35th International Symposium on Microarchitecture (IEEE/ACM MICRO)*, Istanbul, Turkey, pp. 197–207, Nov. 2002.

- [6] C. Zhang, J. Yang, and Frank Vahid, “Low Static-Power Frequent-Value Data Caches,” *Proc. of the 7th Conference on Design, automation and test in Europe (IEEE DATE)*, Paris, France, p. 10214, Feb. 2004.
- [7] M. R. Guthaus, J. S. Ringenberg, D. Ernst, T. M. Austin, T. Mudge, and R. B. Brown, “MiBench: A free, commercially representative embedded benchmark suite,” *Proc. of the 4th International Workshop on Workload Characterization (IEEE WWC)*, Austin, TX, USA, pp. 3–14, Dec. 2001.
- [8] C. Lee, M. Potkonjak, and W. H. Mangione-Smith, “MediaBench: A Tool for Evaluating and Synthesizing Multimedia and Communications Systems,” *Proc. of the 30th International Symposium on Microarchitecture (IEEE/ACM MICRO)*, Research Triangle Park, NC, USA, pp. 330–335, Dec. 1997.
- [9] K. Ghose and M. B. Kamble, “Reducing Power In Superscalar Processor Caches Using Subbanking, Multiple Line Buffers And Bit-Line Segmentation,” *Proc. of the International Symposium on Low Power Electronics and Design (ACM ISLPED)*, San Diego, CA, USA, pp. 70–75, Aug. 1999.
- [10] Sim-Panalyzer 2.0.3, [Online] <http://www.eecs.umich.edu/~panalyzer/>

1 Introduction

Some researchers have reported that the on-chip cache may be responsible for 25%–50% of the total power consumed by a processor [1]. For example, in case of StrongARM-110, 43% of the system power is consumed by the embedded cache [2]. To reduce the energy consumption in the data cache, some researchers have proposed data value compression techniques based on the observation that many values stored in the data cache can be represented with a small number of bits instead of the full bits [3, 4, 5, 6]. Aliagas et al. proposed the *Pattern Cache* predicated on the observation that significant bits of data value are relatively few; the data value can be encoded without storing non-significant bits [3]. The *Frequent Value Cache* (FVC) employs the new value locality that a few values very frequently appear in memory locations [4].

In this paper, we present a new energy-efficient data cache, which is a *Byte-Repeat Pattern Cache* (BRPC) with a new data encoding technique. By studying the behavior of *MiBench* [7] and *MediaBench* [8] programs, we observed that many data values stored in the data cache have a few byte-repeat word patterns in which one byte is repeated several times and/or the rest of bytes are all zeros. These values can be encoded with one byte and a byte-repeat pattern type.

2 Frequent Value Cache

Zhang et al. observed that a few values very frequently appear in memory locations and are therefore involved in a large portion of memory accesses [4]. By studying the behavior of six benchmark programs from the SPECint95 benchmark, they observed that ten distinct *Frequent Values* (FVs) occupy over 50% of all memory locations and the access rate to FVs is 50% on an average.

Exploiting this frequent value phenomenon, Yang et al. [5] proposed the FVC that stores FVs in a compact encoded form. In this approach, the FVs are stored in the *FV register file* that has the same number of the FVs, which typically ranges from 4 to 128. The encoded form is represented in $\log_2 n$ bits, which range from 2 bits for 4 FVs to 7 bits for 128 FVs. The cache data array is divided into a *low-bit array* that contains the $\log_2 n$ index bits and a *high-bit array* that contains the remaining bits. One additional flag bit, which is added to each word in a cache line, is used to distinguish between a FV and a non-FV. The FVs of a program are captured by a *FV finder* for only first 5% of the memory accesses. The FVs obtained by using the limited monitoring are used by the FVC for the remainder of the program execution.

When written into the cache, a word is first encoded through a *FV encoder* to determine whether the word is a FV. If the word is a FV, the $\log_2 n$ index bits generated by the FV encoder are stored in the only low-bit array, and the flag bit is set. Otherwise, the non-frequent word is stored in the low-bit and the high-bit array, and the flag bit is cleared. When a word is read out from the cache, the low-bit array or both arrays are read out according to the flag bit. If the word is a FV, only low-bit array is used as an index to retrieve FV from the FV register file. In this case, dynamic power consumed by cache activity is reduced because the access to the high-bit array is avoided. If the word is a non-FV, both arrays must be read out.

Zhang [6] improved Yang's approach through adding static power reduction technique, exploiting the widely existing FVs in the data cache. In this approach, the FVC turns off the high-bit array unused when the word unit contains a FV, thereby reducing static power consumption.

3 Limitation of the FVC

First, additional power consumption is caused by the FV finder, the FV encoder, and the FV register file. The runtime monitoring to find FVs consumes a large amount of power because the FV finder and the FV encoder consist of a power-hungry *Content-Addressable Memory* (CAM).

Second, the ideal FVs of a given program can be captured only by monitoring a data cache until the program execution finishes. In this case, the power consumption is significantly increased by the monitoring. The preceding study [6], therefore, runs the FV finder for the first 5% of memory accesses, but the partial runtime monitoring makes it difficult to select the appropriate FVs. Accordingly, the hit ratio of the FVs can be decreased, thereby weakening the performance of the power reduction technique.

Third, deploying the fixed monitoring time for each program is only apt for an *Application Specific Instruction-Set Processor* (ASIP) and not for *General Purpose Processor* (GPP) because we should find the FVs dedicated for each program, and because it is very difficult to determine the monitoring time for finding an appropriate set of FVs. In addition, on each context switch, the FV register file needs to be stored and restored, which is unsuitable for GPP.

4 Byte-Repeat Pattern Cache

In this paper, we focus on developing a new encoding technique suitable not only for ASIP but also for GPP. Because we don't use FVs dedicated for each program, the FV register file and the runtime monitoring to find FVs are needless. Our technique can be used readily, regardless of the kind of the programs.

We analyzed the word pattern in the data cache while executing the programs of MiBench and MediaBench and observed that many data have *Byte-Repeat Patterns* (BRPs) as shown in Fig. 1. First, the *Repeated Byte* (RB) word is comprised of the 4 same bytes, such as “0x81818181.” Second, the *Lower Repeated Byte* (LRB) word has the 2 same bytes in its lower part, such as “0x00008181” and the remaining is set to zero. Third, the *Upper Repeated Byte* (URB) word is like “0x81810000.” Fourth, the *Least Byte* (LB) is like “0x00000081.” Finally, the *Most Byte* (MB) looks like “0x81000000.”

Fig. 1 shows the distribution ratio of each pattern and the FVs according to the number of the selected FVs in the data cache. The RB, which mainly consists of “0x00” or “0xFF,” and the LB patterns are the most frequent ones in the BRPs. The results show that 44% of the cached words can be represented with the RB and the LB patterns. This ratio is larger than that of the ideal case of the 16 FVs. In addition, the FV set selected by the partial runtime monitoring cannot be matched with the ideal FV set.

Fig. 2 illustrates the overall architecture of the BRPC. In the BRPC, a data word is comprised of one-byte low-bit and three-byte high-bit array, and an additional pattern type bits for each word is added to distinguish between

Fig. 1. Distribution of frequent values and byte-repeat patterns

Fig. 2. Architectural design overview

the BRP and the non-BRP word. The size of the pattern type is two bits. “00” and “01” mean the LB and the RB, respectively. “11” means that the word is non-BRP, and “10” is not used.

If the word to be written is identified as the BRP by *BRPC checker*, the BRP of the word is stored in the only low-bit array, and the high-bit array is turned off to reduce the static power consumption. Otherwise, the whole word is stored into the low-bit and the high-bit array. If the word to be read is identified as the BRP by checking its pattern type, *BRPC generator* reproduces the whole word by using the BRP stored in the low-bit array and the pattern type bits. Otherwise, both the low-bit and the high-bit array should be read out.

5 Experimental Results

With respect to the amount of power consumption, we compared the proposed BRPC with Zhang’s FVC that outperforms Yang’s FVC. Zhang’s FVC and the BRPC use the subbanking structure [9]. For measuring the power consumption in a data cache, we exploited *Sim-Panalyzer-2.0.3* [10] for power modeling. The size of the data cache is 16 KB. Set-associativity is 8-way and cache line size is 32 Byte. Embedded system benchmark such as MiBench and MediaBench were used.

Memory cell constructing a register file or a cache memory is comprised of six transistors. The FV finder/encoder consists of CAM that has from nine to ten transistors. To simplify comparison and modeling, the FV finder and the encoder were simulated with the SRAM register file. Even though the

Fig. 3. Total power reduction of a data cache

power consumption of CAM memory cells and combinational logic overhead were excluded from Zhang's FVC, the proposed BRPC still outperformed well.

As the number of register files increases in Zhang's FVC, the power consumption also increases due to the power-hungry register files. Consequently, in Zhang's FVC, the power reduction is the largest under the 16 FV set configuration. Fig. 3 illustrates that BRPC reduced the dynamic and static power consumption by 15% of the data cache, and Zhang's FVC reduced 10% under 16 FV set configuration. The main reason of the above results is that the hit ratio of the BRPC is higher than that of Zhang's FVC, which are 34% and 20% respectively. Particularly, our results show that the power consumption is more reduced with the media benchmark because it includes many BRP accesses.

6 Conclusions

We proposed the *Byte-Repeat Pattern Cache* designed for reducing the power consumption of a data cache memory, which is predicated on the observation that many cached words have the Byte-Repeat Patterns. Our approach is applicable not only to ASIP but also to GPP because it doesn't need to find the frequent values dedicated for each program. Therefore, the runtime monitoring and the frequent value register file in the existing FVC are elim-

inated, thereby reducing the overhead such as the encoding and the context switch. We achieved the 15% power reduction, which is 1.5 times as much as the FVC, in the data cache memories due to the higher hit ratio and the reduced additional logic overhead.

Acknowledgments

This research was supported by the MKE (Ministry of Knowledge Economy), Korea, under the ITRC (Information Technology Research Center) support program supervised by the IITA (Institute for Information Technology Advancement). (IITA-2008-C1090-0801-0045)