Way Prediction Set-Associative Data Cache for Low Power Digital Signal Processors

Leiou Wang, Donghui Wang
the Key Laboratory of Information Technology for Autonomous Underwater Vehicles,Chinese Academy of Sciences
Institute of Acoustics, Chinese Academy of Science, Beijing, China
Email: wangleiou@mail.ioa.ac.cn

***Abstract*—In digital signal processors, set-associative caches achieve low miss rates for typical applications but result in significant power consumption. Set-associative caches decrease access time by probing all the data ways in parallel with the tag lookup, although the output of only the matching way is used. The power spent access the other ways is wasted. Eliminating the power consumption by performing the data lookup following the tag comparison definitely increase cache access time, and is unacceptable for high performance level 1 caches. In this paper we propose a way prediction method to reducing set-associative data cache dynamic power while maintaining high performance. This method predicts the matching way and only probes the predicted way, achieving power savings. We evaluate the effectiveness of this method in reducing level 1 data cache power consumption and the simulation results show that this proposed technique achieves a power reduction of up to 18.97% and 14.70% on average, respectively, with negligible area overheads.**

***Keywords*—Cache; low power; way prediction; replacement scheme.**

I. INTRODUCTION

The speed gap between processors and memories has steadily increased. Furthermore, many processor technologies such as pipeline, Very Long Instruction Word (VLIW) and multi-issue policies significantly reduce the Clock cycle Per Instruction (CPI), whereas the reduction of average memory access time is limited. Therefore, to solve the access latency gap between the processor and the main memory access delay, most modern Digital Signal Processors (DSP) employ cache memories.

For embedded DSP, cache memories are relative large compared to simple cores, thus caches consume a significant amount of the total power dissipation [1], [2]. At the same time, as embedded systems and mobile devices become more popular, the low power consideration has become an important design constraint as well as the high performance requirement.

To achieve low miss rates for typical applications, modern DSP utilizes set-associative caches. Set-associative caches probe the tag and data arrays in parallel, and then select the data from the matching ways, which is determined by the tag comparison. At the time of reading the tag and data arrays, the matching way is not known. Consequently, conventional set-associative caches read all the ways but select only one of the ways, resulting in wasted power consumption. As shown in Fig. 1, a two-way set-associative cache discards one of the two ways on every access, wasting nearly 50% of the power consumption. For example, the way 1 tag memory does not include the tag of Instruction 1 (23'h040100), so Instruction 1 accessing way 1 is unnecessary. Likewise, Instruction 2 accessing way 0 is also unnecessary.

This paper proposes a way prediction approach to reduce set-associative data caches power consumption. Unlike their general-purpose counterparts, embedded systems typically execute a well-defined application program, which results in a higher predictability of the execution and memory access patterns. This fact offers additional opportunities for optimizing the power dissipation of the system. Based on this characteristic, the way prediction approach records necessary information and predicts the matching way prior to the cache access, instead of waiting on the tag array to provide the way number. The approach can effectively reduce power consumption because only the predicted way is accessed.

The rest of this paper is organized as follows. In section II, the related work of low power cache techniques will be reviewed. The proposed method will be presented in Section III. Finally, simulation results and conclusions are included in section IV and section V, respectively.

II. RELATED WORK

This section will review similar works in low power cache design.

First, Megalingam et al. [3] has proposed a phased cache and the cache access process which are divided into two phases. In the first phase all the tags are examined in parallel. In the second phase, if there is a hit, then a data access is only performed for the hit way. Min et al. [4] has showed a phased tag cache, this method also divide the cache access process into two phases. But only a small part of the tag is compared in the first phase. The remaining bits of the tag are compared in the second phase to verify whether the result is valid or not. However, both of these two techniques may seriously result in performance loss.

Yang et al. [5] has demonstrated that a small number of distinct values tend to occur very frequently in memory. The cache data array is partitioned so that one array contains low bits and the other contains the remaining high bits. Frequent values are stored in the low bit array while non-frequent values use the space in both data arrays. Ye et al. [6] has found that many values rarely need full bit width and has presented a variable bit-line data cache. This cache data array is divided into several sub-arrays to adapt each data pattern. But these

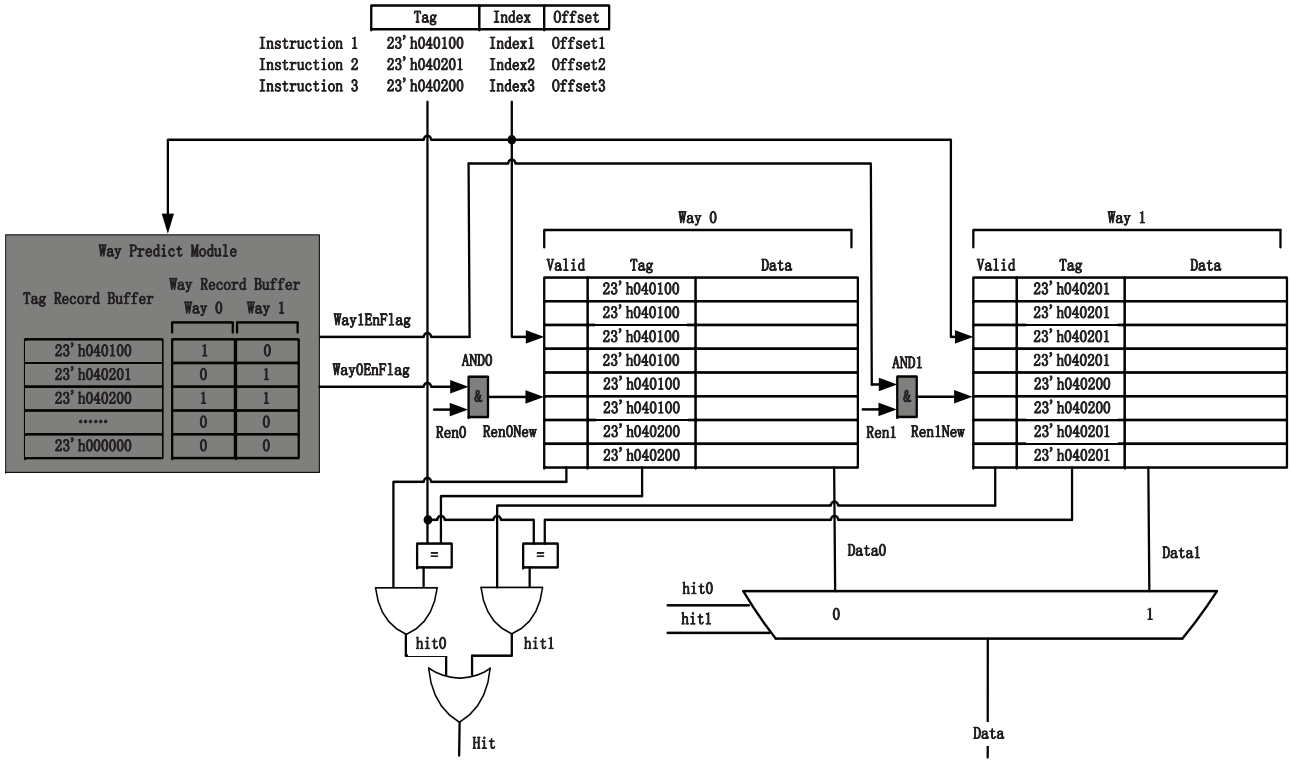

Fig. 1. Way prediction cache architecture.

two techniques require significant micro-architectural change and the control logic is quite complicated.

In [7] Chang et al. has observed most bit values read from the cache are 0s. So the authors introduce a dynamic zero sensitivity scheme that reduces average cache power consumption in reading a 0. And this method is mainly applied to circuit-level.

In [8] Witchel et al. has proposed a direct addressed cache that is a hardware-software design for a power efficient microprocessor data cache. Direct addressing allows software to access cache data without a hardware cache tag check. However, this technique requires the compiler to support and may cause code incompatibility.

Tag overflow buffering [9] provides power efficient cache memory architecture and it exploits the reduced number of tag bits. The main idea of this scheme is based on moving a large number of tag bits from the cache memory into an external register, called a tag overflow buffer, which identifies a current memory locality. The tag overflow buffer in this scheme plays the role of a one entry level 0 cache and it detects the locality of application programs. In [10], [11] authors have proposed other versions for further reducing tag bits. However, these approaches mainly focus on instruction caches.

III. Way Prediction Cache Architecture

As mentioned before, the existing low power cache approaches usually need to greatly change the micro-architecture or require the compiler to support. However, this paper describes a structure-level full-hardware implementation in the following discussion. Surprisingly, this method is quite simple and relies on recording tag information.

A. Way Prediction Cache Architecture

Fig. 1 shows a block diagram of the way prediction cache architecture. The shaded box represents the block which implements the way prediction function, called a Way Predict Module (WPM). In this architecture, the tag need to save into a separate register array, called a Tag Record Buffer (TRB). To predict the access way number, a Way Record Buffer (WRB) is also added. The WRB records the saving way number of corresponding tag in the TRB. For a two-way set-associative cache, the width of the WRB is two bits.

If the TRB does not include a instruction tag value, the tag should be saved into the TRB. At the same time, the corresponding bit in the WRB should be set to 1. If the TRB already has this tag value and the saving way number of this tag differs from the corresponding bit in the WRB, the WRB should be updated. For example, tag 23'h040200 is saved in way 0 and way 1 of the cache, so way 0 bit and way 1 bit of the WRB are both set to 1.

When an instruction needs to look up the set-associative data cache, the TRB should be probed. If there is a hit in the TRB, and the corresponding bits are 2'b10 in the WRB. It is indicate that this tag only exists in way 0 of this cache, so the output signal Way0EnFlag and Way1EnFlag are 1 and 0, respectively. That is to say, this instruction only needs to access the way 0 tag and data memory of the data cache. If the TRB includes the tag of this instruction and the corresponding bits are 2'b11

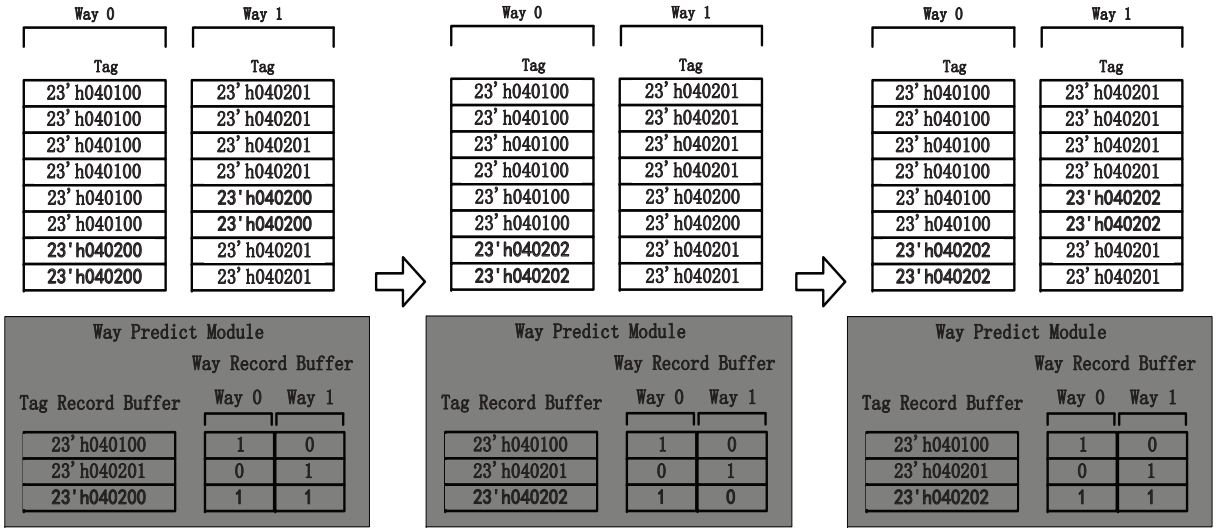

Fig. 2. A replacement example.

in the WRB, or the TRB does not include the tag of this instruction, the output signal Way0EnFlag and Way1EnFlag are both 1. In other words, the two ways of this set-associative data cache all need to be accessed. The functional operations of this architecture are summarized in Table I.

TABLE I
SUMMARY OF WAY PREDICTION CACHE OPERATIONS

TRB	WRB		Description
	Way 0	Way 1	
Hit	1	0	only access way 0
Hit	0	1	only access way 1
Hit	1	1	access way 0 and way 1
Miss	-	-	access way 0 and way 1

B. Replacement Scheme

In order to reduce the entries of the TRB and the WRB, the Replacement Scheme (RS) should be considered.

Since generally 10% of the loop programs occupy 90% of the execution time [12]. And the loop programs tend to save the new tags in a certain pattern. A typical replacement example is shown in Fig. 2. The tag 23'h040200 is replaced with the new tag 23'h040202 in the TRB.

So when there is a new tag need to be saved into tag memory, the replacement scheme operates as follows.

1) : If the TRB does not include this new tag, and the corresponding bits of an old tag are 2'b11 in the WRB. Then this old tag will be replaced with the new tag, and the WRB also needs to be updated.

2) : If the TRB does not include this new tag, and any tag corresponding bits are not 2'b11 in the WRB. Then the new tag should be saved into a new entry, and there is no replacement operation occurs.

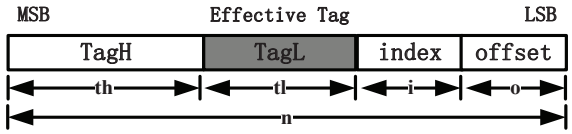

Fig. 3. Address decomposition for tag architecture.

C. TRB Optimization

In this subsection, we further discuss the TRB optimization.

Fig. 3 shows the address decomposition of tag architecture, and n is the total number of entire address bits. In this address model, the tag field is further divided into th and tl. The index and offset field is the same as the conventional cache.

In the address decomposition of tag architecture, the th field in the entire address plays the role of locality detection. If the embedded application program exhibits memory access locality [9] [10], then most tag bits of successive processor requests will be the same, except for a few differences in the tl. In this case, the width of the TRB should be optimized to tl bits.

IV. EXPERIMENTAL RESULTS

In this section, Simulations had been conducted to verify this way prediction method.

A. Simulation Environment

The low power way prediction method of set-associative data cache was applied to SuperV DSP, which implemented with the four-issue VLIW and eight-stage pipeline architecture, operating at 500MHz. Since a 5 bits tag is enough to provide the same level of full bits tag performance [9] [10] [11], our cache parameters are a two-way set-associative, 1KB, 4B line data cache.

The data cache power consumption was evaluated on gate-level with the power analysis tool based on the circuit switching activity and capacitance of different components. The gate-level netlist was generated by the synthesis tool with the technology library of 90nm CMOS process.

In order to evaluate the proposed method, some classic DSP benchmarks [13] and the Powerstone benchmarks [14] were utilized.

B. Simulation Results

The first simulation aims to determine the proper number of the TRB and the WRB entries for our proposal. As the number of the TRB and the WRB entries increase, we can record more tag information of application programs. However, the TRB and the WRB also have power consumption and definitely increase area overhead. Therefore, the replacement scheme was utilized in this way prediction method.

Table II shows the number of the TRB and the WRB entries. From the simulation results, we can find that the replacement scheme can effectively reduce the entries for various application programs. Table II also provides that the optimum width of TRB was 10 bits.

TABLE II
NUMBER OF TRB/WRB ENTRIES AND WIDTH OF TRB

Applications	Number of Entries		Width of TRB(bits)	
	Without RS	With RS	Tag	Effective Tag
autocor	**5**	3	23	10
bexp	**4**	2	23	10
blit	**17**	3	23	10
crc	3	3	23	10
dotprod	**4**	2	23	10
dotp_sqr	**4**	3	23	10
engine	2	2	23	10
firlms2	**4**	3	23	10
matmul	2	2	23	10
maxidx	3	3	23	10
maxval	3	3	23	10
minval	3	3	23	10
vector_max	**9**	2	23	10
vector_mul	3	3	23	10
vector_sum	**34**	3	23	10

Embedded systems execute a few specific predefined application programs. Therefore, system architectures should be determined for the specific predefined embedded application programs in the system development time. In this paper, we decide that the proper number of the TRB and the WRB entries was three, and the width of the TRB was 10 bits. And we hope that some other system architects can choose their own candidate based on our results.

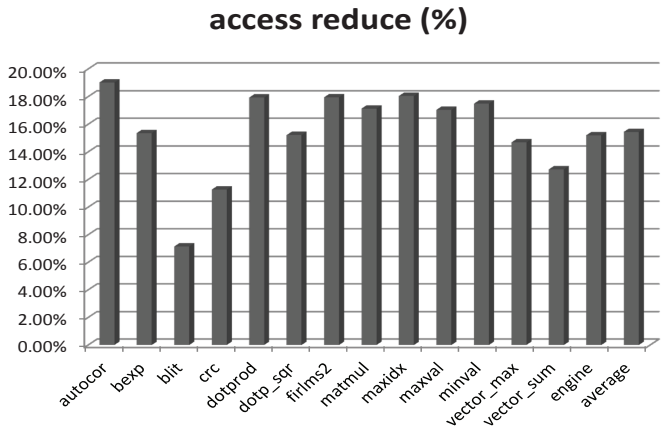

Fig. 4. Cache access reduction.

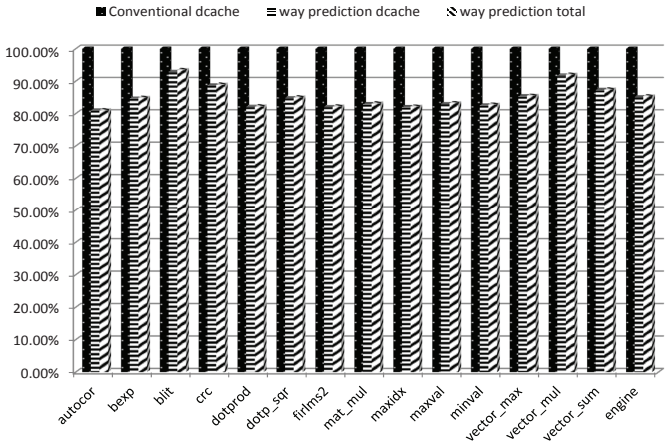

Fig. 5. Cache power reduction.

Fig. 4 reports the cache unnecessary access reduction. The more unnecessary access is reduced, the more power savings can get. And the unnecessary access reduction was up to 19.02% and 15.45% on average.

Fig. 5 reports the cache power reduction compared to the conventional cache. The way prediction total power also includes the WPM, which was actually negligible with respect to the cache power consumption. And the power reduction was up to 18.97% and 14.70% on average.

A prediction miss causes a cache miss. And these operations clearly impact miss rates, which in turn affect the overall performance of the memory hierarchy. The impact of the miss rates on the total execution time (in cycles) for the various benchmarks is shown in the Fig. 6. We notice that the performance penalty was always below 0.5%, and with an average of 0.09%. In other words, this way prediction cache provided comparable performance with the conventional cache.

Although our proposal requires additional hardware overhead, such as the TRB and the WRB, it is almost trivial. The additional hardware TRB is just a three-entry 10 bits width

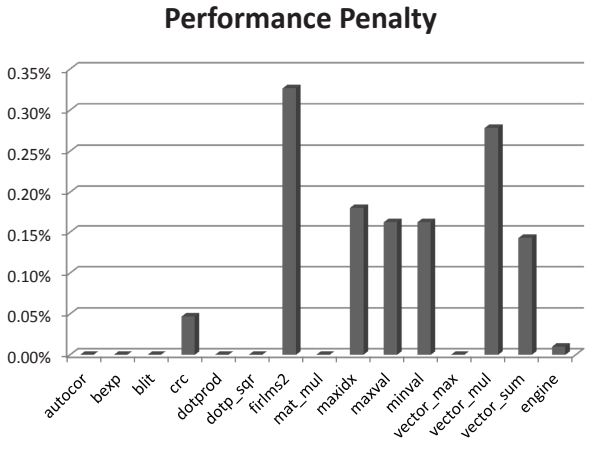

Fig. 6. Execution time increase.

small buffer rather than a memory. The size of the WRB is even less than TRB.

The area and delay overheads are shown in the Table III. In comparison with the conventional data cache, the area overhead was only 0.52%.

TABLE III
AREA AND DELAY OVERHEADS

Overheads	Area		Delay
	Value(μm^2)	Percentage(%)	Value(ns)
Conventional	269445	100	1.10
WP	270839	100.52	1.25

The additional delay in cache access for our proposal was also small, and the delay overhead was 0.15ns. Considering the fact that the reduction in power consumption obtained with this method, we think this extra overheads in area and delay are acceptable.

V. CONCLUSION

For embedded DSP, set-associative caches consume a significant amount of the total power dissipation, and the key to power reduction is to pinpoint the matching way without probing all of the ways. This paper has derived a way prediction method to reduce level 1 set-associative data cache dynamic power while maintaining high performance. It is demonstrated that the proposed method provides a reduction in power of up to 18.97% and 14.70% on average, respectively, with negligible area overheads.

ACKNOWLEDGMENT

This work was supported by National Natural Science Foundation of China under Grant 612774025.

REFERENCES

[1] C. J. Zhang, V. Frank, J. Yang and W. Najjar, "A Way-Halting Cache for Low-Energy High-Performance Systems," *in Proc. ISLPED04, California, USA*, 2004, pp. 126-131.

[2] M. Zhang, X. T. Chang and G. Zhang, "Reducing Cache Energy Consumption by Tag Encoding in Embedded Processors," *in Proc. ISLPED07, Portand, Oregon, USA*, 2007, pp. 367-370.

[3] R. K. Megalingam, K. B. Deepu, I. P. Joseph and V. Vandana, "Phased Set Associative Cache Design For Reduced Power Consumption," *in Proc. ICCSIT09, Los Alamitos, USA*, 2009, pp. 551-556.

[4] R. Min, W. B. Jone and Y. M. Hu, "Phased Tag Cache: An Efficient Low Power Cache System," *in Proc. ISCAS04, Los Alamitos, USA*, 2004, pp. 805-808.

[5] J. Yang and R. Gupta, "Frequent Value Locality and its Applications," *ACM Trans. Embedded Computing Systems*, vol. 1, no. 1, pp. 79-105, Nov. 2002.

[6] J. Ye and T. Watanabe, "A Variable Bitline Data Cache for Low Power Design," *in Proc. APCPRME10, Los Alamitos, USA*, 2010, pp. 174-177.

[7] Y. J. Chang and F. P. Lai, "Dynamic Zero-Sensitivity Scheme for Low-Power Cache Memories," *IEEE Micro*, vol. 25, no. 4, pp. 20-32, Jul. 2005.

[8] E. Witchel, S. Larsen, A. C. Scott and K. Asanovic, "Direct Addressed Caches for Reduced Power Consumption," *in Proc. ISM01, Los Alamitos, USA*, 2001, pp. 124-133.

[9] M. Loghi, P. Azzoni and M. Poncino, "Tag Overflow Buffering: Reducing Total Memory Energy by Reduced-Tag Matching," *IEEE Trans. VLSIS*, vol. 17, no. 5, pp. 728-732, Apr. 2009.

[10] J. W. Kwak and Y. T. Jeon, "Compressed tag architecture for low-power embedded cache systems," *Journal of Systems Architecture*, vol. 56, pp. 419-428, 2010.

[11] J. Gu, H. Guo and P. Li, "An on-chip instruction cache design with one-bit tag for low-power embedded systems," *Microprocessor and Microsystems*, vol. 35, pp. 382-391, 2009.

[12] J. Villarreal, R. Lysecky, S. Cotterell and F. Vahid, "A study on the loop behavior of embedded programs," *Department of Computer Science and Engineering, University of California, Riverside, Tech. Rep. UCR-CSE-01-03*, Dec. 2001.

[13] TMS320C62x DSP Library Programmers Reference, Texas Instruments Inc, 2003. Available: http://www.ti.com/lit/ug/spru402b/spru402b.pdf.

[14] A. Malik, B. Moyer and D. Cermak, "A Low Power Unified Cache Architecture Providing Power and Performance Flexibility," *in Proc. ISLPED00, Rapallo, Italy*, 2000, pp. 241-243.