ISSN(Online): 2320 - 9801
ISSN (Print) : 2320 - 9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 6, June 2016

Low Complexity Architecture for Similar Tag Bits in Cache Memories Using BWA

Elsa John, Sharon Mathew

M.Tech Scholar (VLSI & Embedded System), Dept. of ECE., IIET, M.G University, Kottayam, Kerala, India

Assistant Professor, Dept. of ECE, IIET, Nellikuzhy, Kerala, India

ABSTRACT: As the transient error rate is increasing, it is essential to prevent transient errors and provide a correction mechanism for the circuits, particularly SRAM cache memories. The largest structures in current microprocessors are caches and are thus more vulnerable to the transient errors. The tag bits in cache memories are also vulnerable to transient errors but less efforts have been made to reduce their vulnerability. In this paper, we propose to exploit the same tag bits to improve error protection capability of the tag bits in caches. During the fetching of data from the main memory, the adjacent cache lines are checked to see whether they have the same tag bits as those of the data fetched. This is done using the Butterfly-formed Weight Accumulator (BWA). For further use, this same tag bit information is stored in the caches as extra bits. The same tag bit information is used to recover from the error in the tag bits when an error is detected. The proposed scheme has less area and delay when compared to the recent implementations.

KEYWORDS*:* Cache memory, tag bits, transient errors, data comparison, tag matching**.**

I. Introduction

With the scaling of technology to nanometer, the vulnerability of computer systems to transient errors are becoming larger [5]. Particularly, the cache memories are more vulnerable because their operating voltage is very low and their sizes are increasing due to popular use of multilevel cache hierarchy and multi-core architecture even in embedded/mobile systems [6]. To deal with the transient errors, the cache memories employ several error protection mechanisms such as parity codes and single bit error correction and double bit error detection (SEC-DED) codes but these schemes are not efficient in terms of area and delay. Thus many techniques have been proposed to reduce such inefficiency. In order to reduce the vulnerability of data bits considerable efforts have been made but the errors in tag bits are also critical for data integrity. For example, transient errors in tag bits can incur false misses in dirty cache lines, and consequently, stale data can be consumed. However, a complex protection mechanism for tag bits can degrade performance due to their long latency. Clearly, providing high reliability with a low-cost mechanism to tag bits is necessary to guarantee system integrity and maintain system performance. Due to the spatial locality of programs, there are many tag bits whose values are the same with those of other tag bits in adjacent cache sets. In other words, when a cache line is accessed or replaced, another cache line whose tag bits are the same with those of the accessed cache line is very likely to be found in an adjacent cache set. This phenomenon is called tag bits similarity in this paper.

We exploit the tag bits similarity to improve reliability of tag bits against transient errors. When an error is detected by the conventional parity check bits, the error can be corrected if the same tag bits are present in an adjacent cache set. Faulty tag bits are simply replaced with intact tag bits from the adjacent cache set. To exploit same tag bits for transient error protection, we augment the conventional cache architecture with four simple hardware components. First, a shifter located after the cache decoder or an up/down counter in the execution stage is required to access cache lines in an upper and/or lower cache set than currently accessed cache set. Second, an encoder to generate the similarity information between tag bits is needed. Third, a small circuit is necessary to handle similarity bits on cache replacements. Finally, an error correction unit corrects transient errors in tag bits using the same tag bits from adjacent cache set. Data comparison is widely used in computing systems to perform many operations such as tag matching in a cache memory and the virtual-to-physical address translational look-aside buffer. The tag comparison is done with the help of Butterfly-formed Weight Accumulator (BWA) [2], which contains a number of half adders (HAs) to count the number of one's. This will help to reduce the area and delay of the proposed architecture.

This paper is organized as follows. Section II briefly provides the related works. Section III details tag bits errors classified into four subgroups. Section IV describes our proposed cache architecture exploiting prevalent same tag bits and BWA. Section V presents our simulation results for our scheme and previous work. Finally, Section VI concludes this paper.

II. RELATED WORK

ICR has been proposed in [4], and Fig. 1 simplifies its mechanism. The basic idea of ICR is to replicate frequently accessed cache lines into other cache lines, which are predicted not to be accessed further, called dead blocks. Spaces of the dead blocks in the L1 data cache are recycled to maintain duplicates of frequently used data bits against transient errors. To avoid significant performance degradation by dead block miss predictions, ICR uses a prediction strategy. This scheme can be applied to protect tag bits because replicated locations are calculated by fixed formulation and replicated cache lines can be used to correct tag bits errors in the active lines. However, the dead block prediction technique is based on a prediction, and thus ICR can increase cache miss rates and the number of write-backs resulting in large performance loss and energy consumption increase.

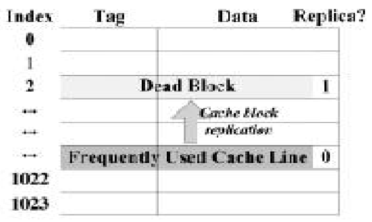


Fig.1. Replication of ICR scheme

To enhance reliability of tag bits, Wang *et al*. [9] proposed tag replication buffer (TRB) scheme. They exploit the address locality of memory access and duplicate most recently accessed tag bits in their TRB structure, and thereby correcting errors with replica in TRB. Sector caches exploit tag bits similarity to conciliate low tag implementation cost [8].

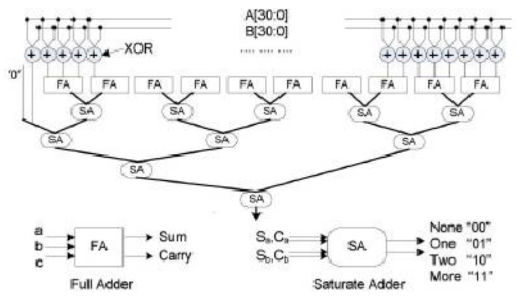


Fig. 2. Direct compare (DC) circuit constructed of full adders and SAs.

The comparison method used in [1] is the direct compare method [3] as shown in Fig. 2. Saturating adder (SA) is the basic building block of direct compare design. This adder is similar to a regular arithmetic compressor. The difference is that the output value of the counter has an upper bound, i.e., any sum value exceeding this upper bound will be ignored. With this upper bound limit it is possible to further optimize the circuit. The inputs and outputs of the adder are both 2-bit binary words represented with C and S for each bit, where C is the most significant bit (MSB) and S is the least significant bit(LSB).

ISSN(Online): 2320 - 9801
ISSN (Print) : 2320 - 9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 6, June 2016

III. TAG BIT ERRORS

Recently, a standard taxonomy for error detection classifies errors as undetected, detected but un-correctable (DUE), or detected and correctable errors. Single event upsets in logic path can potentially cause silent data corruptions (SDCs) in unmitigated system [10]. SDCs are undetected errors that lead to incorrect machine results. DUE occurs when detected errors exceed error protection capability. Tag arrays are typically protected by error protection codes, such as parity or SEC–DED codes.

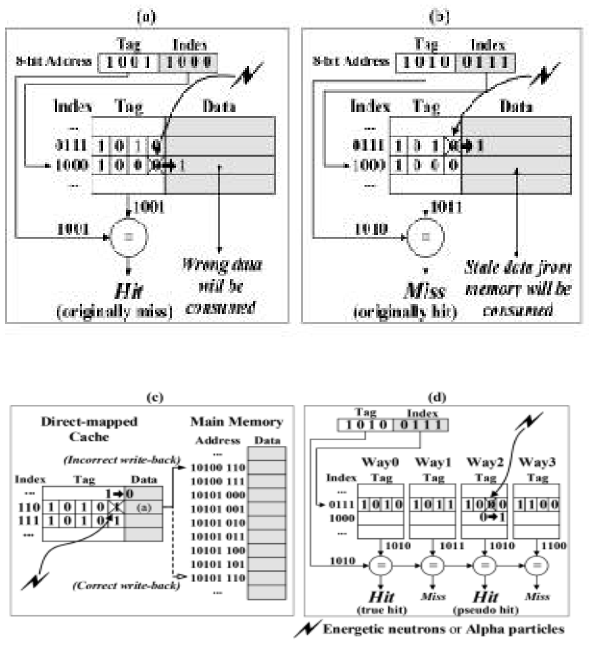


Fig .3. Effects of tag bits errors. (a) Pseudo-hit. (b) Pseudo-miss. (c) Replacement error. (d) multi-hit.

Transient errors in tag bits raise the possibilities of pseudo-hits, Pseudo-misses [7], replacement errors [11], and multi-hits. Fig. 3(a) shows an example of a pseudo-hit. A pseudo-hit refers to a hit that is actually a miss in the absence of a transient error. The pseudo-hit may lead the data path to use the incorrectly matched data. However, pseudo-hit can be detected by storing check bits (parity or SEC–DED codes) for each row of tag bits. When an access for a given input results in a hit, the corresponding check bits of the tag bits are read out and then compared with the check bits computed from input tag bits. Any mismatch denotes that the tag bits have been corrupted and the hit is surely a pseudo-hit. A pseudo-miss refers to a miss that is actually a hit when there is no transient error. Fig. 3(b) shows an example of a pseudo-miss. Pseudo-misses may cause a data integrity problem in a write-back data cache, which holds dirty data. A pseudo-miss on dirty data triggers a fetch and use of stale data from a lower level. In case of a write-through cache, however, a pseudo-miss does not cause a problem. It just incurs a cache miss. If tag bits are corrupted after their corresponding cache line is modified, the modified data can be written back to a wrong location when write-back caches are used. Obviously, stored data in backing storage will be lost and data modification cannot be reflected in backing storage. This kind of error type is classified as a replacement error, as shown in Fig. 3(c) [11]. In addition, transient errors may result in multi-hit errors when caches are associative. Fig. 3(d) shows an example of a multi-hit in a four-way set associative cache. At most one tag bit must be matched in one cache set, but it is possible that multiple tag bits are matched in a single set due to a pseudo-hit. Multi-hit errors occur infrequently, but their rate can be increased in highly associative caches.

IV. PROPOSED APPROACH

As we mentioned, our scheme refers to tag bits and STI bits in upper and lower sets of a new cache line, finds and invalidates STI bits pointing to an evicted line, and generates proper STI bits for the new tag bits. Fig.4 represents the

ISSN(Online): 2320 - 9801
ISSN (Print) : 2320 - 9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 6, June 2016

detailed operation in a four way set associative cache.). Four-bit STI is attached into each entry of tag array in a four-way set associative cache. When data (a) are fetched from the main memory, upper and lower sets are referred to find the same tag bits. In this example, the upper set has the same tag bits as newly fetched tag bits. After checking the lower and upper cache sets, STI bits of data (a) are filled with a proper STI code. At this time, STI bit of existing lines, whose tag value is the same with the new tag bits, is also filled with a proper STI code. STI code of the upper right side set were originally 0000, no same tag, and thus corresponding tag bits were vulnerable to errors. After data (a) are loaded, however, the STI bits are encoded to 1001 due to the same tag bits in the lower left side set, and the corresponding tag bits are protected by exploiting the same tag bits. Therefore, both tag bits whose values are the same can be recovered from errors by fetching other intact tag bits. Note that all these procedures are performed only when a new cache line is loaded on a cache miss. Since a cache miss makes quite large stall cycles even in the first-level cache of multilevel cache hierarchy, the process can be done while pipelines are stalled. The BWA is used for the comparison of tag bits and this reduces the complexity of the architecture.

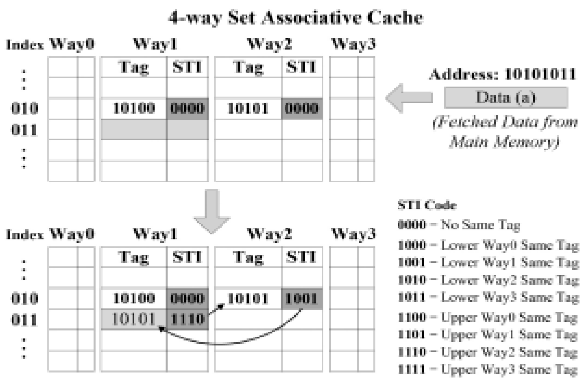

4-way Set Associative Cache
Index
Way0
Way1
Way2
Way3
Tag
STI
010
011
10100
0000
10101
Address: 10101011
Data (a)
(Fetched Data from Main Memory)
1110
1001
STI Code
0000 – No Same Tag
1000 – Lower Way0 Same Tag
1001 – Lower Way1 Same Tag
1010 – Lower Way2 Same Tag
1011 – Lower Way3 Same Tag
1100 – Upper Way0 Same Tag
1101 – Upper Way1 Same Tag
1110 – Upper Way2 Same Tag
1111 – Upper Way3 Same Tag

Fig.4. Detailed operation in a four way set associative cache.

Detailed Architecture

In this section, we present our proposed architecture and explain additional components required. To exploit tag bits similarity for transient error protection of tag bits, the conventional cache structure is augmented with four simple hardware components. Fig. 5 shows the schematic micro-architecture design of our scheme called *SimTag* (similar tag bits). To implement SimTag, first, a shifter is added to access a lower or upper cache set. Second, STI Encoder is needed to generate STI bits. Third, STI replacement handler is required to modify STI bits in an upper or lower cache set on replacements. Finally, an error correction unit is built for error recovery.

Detailed operation of each component is described below.

1) Shifter: By changing the output signals of the cache decoder, cache lines in a lower or upper set can be accessed. Thus, our proposed architecture needs a shifter. On cache misses, the shifter shifts the output signals of the decoder to the left and to the right sequentially so as to generate STI bits. For error recovery, it also shifts the decoder output signals but shift direction is determined by the STI set location bit. Note that overheads of error recovery can be ignored, since transient error rate is quite low in SRAM cells. This approach is simple and intuitive.

ISSN(Online): 2320 - 9801
ISSN (Print) : 2320 - 9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 6, June 2016

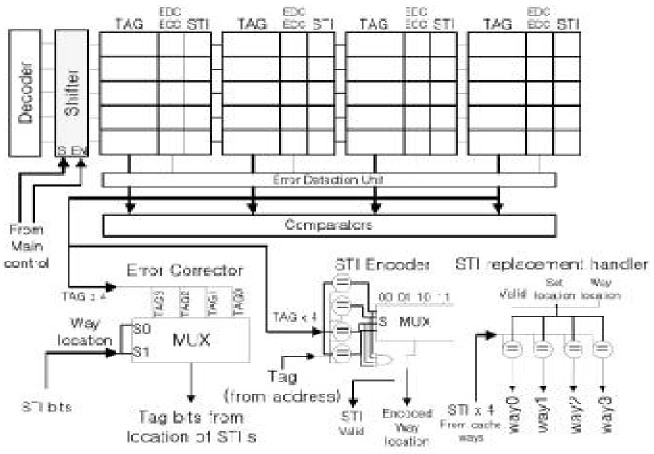


Fig.5. Detailed architecture of BWA-SIMTAG.

2) Comparator: A new butterfly-formed weight accumulator is used for the matching of the data protected with an error correcting code (ECC). The basic function of BWA is to count the number of 1's among its input bits. It consists of multiple stages of HAs as shown in Fig.6, where each output bit of a HA is associated with a weight. The HAs in a stage are connected in a butterfly form so as to accumulate the carry bits and sum bits of the upper stage separately. in other words, both inputs of HA in a stage, except the first stage, are either carry bits or sum bits computed in the upper stage. Fig.7. shows the first and second level circuits for a (8,4) code. Taking the outputs of preceding circuits, the decision unit finally determines if the incoming tag matches the retrieved code word by considering the four ranges of

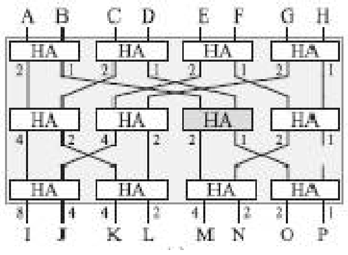


Fig.6. General structure of BWA.

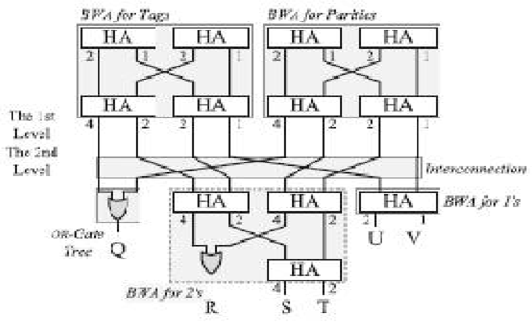


Fig.7. First and second level circuits for a (8,4) code

the hamming distance. The truth table for the decision unit is described in Table I. Since U and V cannot be set simultaneously, such cases are implicitly included in do not care terms in Table I.

TABLE I
TRUTH TABLE OF THE DECISION UNIT FOR A (8, 4) CODE

Q OR R OR S	T	U	V	Decision
0	0	0	x	Match
	0	1	x	Fault
	1	0	0	Fault
	1	0	1	Mismatch
	1	1	x	Mismatch
1	x	x	x	Mismatch

3) STI Encoder: To generate STI bits, the STI encoder compares the tag bits of cache missed data with other tag bits located in lower and upper sets while pipelines are stalled due to cache misses. If there are matching tag bits in an adjacent set, an STI valid bit is set to one and STI way location bits are generated using a multiplexer. STI set location bit can be obtained from the main controller.
4) STI Replacement Handler: As we discussed before, STI bits in a lower or upper set must be updated on cache replacements. The STI replacement handler checks all STI bits in the adjacent sets. If certain STI bits point to the tag bits of the replaced cache line, then we simply invalidate their STI valid bits and generate new STI bits by finding other same tag bits.
5) Error Corrector: To recover tag bits from errors, uncorrupted tag bits should be obtained from an adjacent cache set using STI bits (if the same tag bits exist). With STI way location bits, the exact location having the same tag bits can be obtained. After fetching right tag bits, the corrupted tag
bits are replaced with them.
6) Main Controller: To support extra components, the main controller is slightly modified to generate necessary control signals. On cache misses or tag bits errors, the pipelines are stalled and, at the same time, the main controller signals the additional shifter to access adjacent sets.

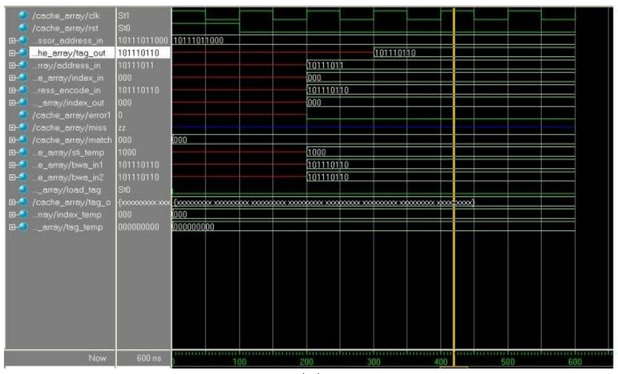
V. SIMULATION RESULTS

In order to compare the BWA-SIMTAG with that of previous paper [1], we have employed the use of BWA architecture for the purpose of tag matching. The simulation result of BWA-SIMTAG with normal operation is shown in Fig.8. (a) and the error correction from main memory is shown in Fig.8.(b).

Table II. Comparison between SIMTAG and BWA-SIMTAG

Design	Delay (ns)	Area(no of slices)
SIMTAG	11.045	474
BWA-SIMTAG	10.9	437

The comparison table shown in Table II describes that the technique using BWA architecture has less delay and area than the previous architecture using direct compare. . In Fig.8.(a), clk, rst and processor address in are the inputs. bwa in1 and bwa in2 are the inputs to the BWA. Index indicates the index number of each cache line. Sti temp indicates the sti bit of each cache lines. Tag out indicates the corrected tag from the tag array after detecting the error. Since there is no error the match indicates 000 and the corresponding tag is obtained at the output as tag out. In Fig.8.(b), there is an error in the tag and thus the match value became 010. The correct tag is then obtained at the output as tag out by considering the sti bits in each cache line.


(a)

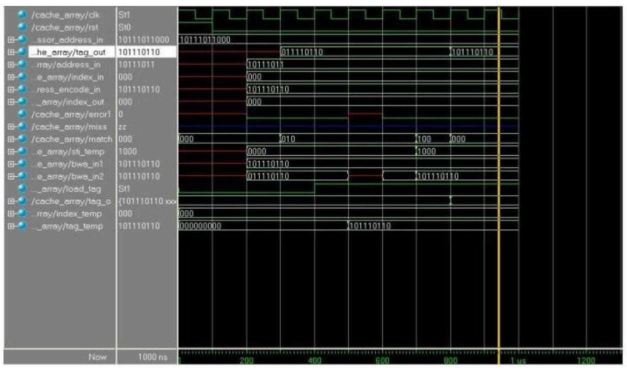

(b)

Fig.9. Simulation result of BWA-SIMTAG (a) normal operation. (b) error correction from main memory

VI. CONCLUSION

Due to the increasing trend of transient error rates, it is becoming important to provide error detection and correction c capability for hardware circuits, especially for cache memories. However, most of the previous techniques focus only on data bits without considering tag bits corruption. Most tag bits in the data cache have their replica in adjacent cache sets and we exploit this tag bit similarity against transient errors. The method of tag matching is done using the BWA architecture. Using this architecture, the circuit had less area and delay compared to the previous technique that used direct compare method.

REFERENCES

1. Jeongkyu Hong, Jesung Kim, and Soontae Kim, Member, IEEE “Exploiting Same Tag Bits to Improve the Reliability of the Cache Memories”, IEEE Trans. vol. 23, no. 2, February 2015.
2. Byeong Yong Kong, Jihyuck Jo, Hyewon Jeong, Mina Hwang, Soyoung Cha, Bongjin Kim, and In-Cheol Park, “Low-Complexity Low-Latency Architecture for Matching of Data Encoded With Hard Systematic Error-Correcting Codes”, IEEE Trans. Vol. 22, no. 7, July 2014.
3. Wei Wu, Dinesh Somasekhatr, and Shin-Lien Lu, “Direct compare of information coded with error correcting codes”, IEEE Trans. Vol. 20,no. 11, November 2012.
4. W. Zhang, S. Gurumurthi, M. Kandemir, and A. Sivasubramaniam, “ICR: In-cache replication for enhancing data cache reliability,” in Proc. Int. Conf. Dependable. Syst. Netw., 2003, pp. 291–300.
5. P. Shivakumar, “Modeling the effect of technology trends on the soft error rate of combinational logic,” in Proc. Int. Conf. Dependable Syst. Netw., 2002, pp. 389–398.
6. R. Kessler, “The alpha 21264 microprocessor,” IEEE Micro, vol. 19, no. 2, pp. 24–36, Apr. 1999.
7. L. Hung, M. Goshima, and S. Sakai, “Mitigating soft errors in highly associative cache with CAM-based tag,” in Proc. IEEE Int. Conf. Comput. Des., Oct. 2005, pp. 342–347.
    a. Seznec, “Decoupled sectored caches: Conciliating low tag implementation cost and low miss ratio,” in Proc. 21st Ann. Int. Symp. Comput. Archit., 1994, pp. 384–393.

8. S. Wang, J. Hu, and S. Ziavras, "TRB: Tag replication buffer for enhancing the reliability of the cache tag array," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 20, no. 4, pp. 643–654, Apr. 2012.
9. P. Meaney, S. B. Swaney, P. N. Sanda, and L. Spainhower, "IBM z990 soft error detection and recovery," IEEE Trans. Device Math. Rel., vol. 5, no. 3, pp. 419–427, Sep. 2005.
10. H. Asadi, V. Sridharan, M. B. Tahoori, and D. Kaeli, "Vulnerability analysis of L2 cache elements to single event upsets," in Proc. Des., Autom., Test Eur., 2006, pp. 1–6.

BIOGRAPHY

Elsa John is an M-Tech scholar in VLSI and Embedded System in the Electronics and Communication Department, IIET, M. G. University. She received B-Tech degree in 2014 from IIET, M. G. University, Kottayam, Kerala. Her research interests are VLSI, low power designs and HDL languages etc.

Sharon Mathew is an Assistant Professor in the Electronics and Communication Department, IIET, M. G. University. She received B-Tech degree in 2010 from VJCET, Vazhakulam, M. G. University, Kottayam, Kerala and M.Tech in 2013 from Sri Ramakrishna Engineering college, Coimbatore, Anna University. Her research interests are VLSI design, low power etc.