International Journal of VLSI System Design and Communication Systems

ISSN 2322-0929
Vol.03, Issue.06,
August-2015,
Pages:0892-0896

www.ijvdcs.org

Error Detection and Correction using STI in Cache Memory

D. VISHWAKALA[1], CH. SURESH[2], K. HYMAVATHI[3]

[1]PG Scholar, Dept of VLSI System Design, Swami Ramananda Tirtha Institute of Science and Technology, Telangana, India, Email: vishwakala431@gmail.com.

[2]Assistant Professor, Dept of ECE, Swami Ramananda Tirtha Institute of Science and Technology, Telangana, India.

[3]Assoc Prof & HOD, Dept of ECE, Swami Ramananda Tirtha Institute of Science and Technology, Telangana, India.

Abstract: With continued technology scaling to the nanometer regime, computer systems are becoming vulnerable to transient errors. Especially, cache memories are vulnerable because they operate at low voltage levels and their sizes increase due to popular use of multilevel cache hierarchy and multi-core architecture even in embedded/mobile systems. To combat against transient errors, cache memories typically employ error protection mechanisms, such as parity codes and single-bit error correction and double-bit error detection (SEC–DED) codes. However, these schemes are not efficient in terms of area overhead and error protection coverage. Thus, many techniques are proposed to reduce such inefficiency and enhance protection coverage. With the trend of increasing transient error rate, it is becoming important to prevent transient errors and provide a correction mechanism for hardware circuits, especially for SRAM cache memories. Caches are the largest structures in current microprocessors and, hence, are most vulnerable to the transient errors. Tag bits in cache memories are also exposed to transient errors but a few efforts have been made to reduce their vulnerability. In this paper, we propose to exploit prevalent same tag bits to improve error protection capability of the tag bits in the caches. When data are fetched from the main memory, it is checked if adjacent cache lines have the same tag bits as those of the data fetched. This same tag bit information is stored in the caches as extra bits to be used later. When an error is detected in the tag bits, the same tag bit information is used to recover from the error in the tag bits. The proposed scheme has small area, energy, and performance overheads with error protection coverage of 97.9% on average. Even with large working sets and various cache sizes, our scheme shows protection coverage of higher than 95% on average.

Keywords: Cache memory, reliability, tag bits, transient Errors.

I. INTRODUCTION

With continued technology scaling, caches are becoming more vulnerable to transient errors. There have been many efforts made to address transient errors in the data arrays of the caches. However, errors in the tag bits of the caches are critical for data integrity, too. For example, transient errors in the tag bits can lead to false misses in the dirty cache lines and, consequently, stale data can be consumed. Therefore, addressing transient errors in the tag bits are critical for correction execution. By our experiments with embedded benchmarks on an Intel X-scale-based simulator, most tag bits in the data caches have their replica in other cache sets. In other words, when a cache line is accessed or replaced, we can find an adjacent cache line with the same tag bits as those of the cache line accessed in a upper or lower cache set than the current set. This is called tag bits similarity in this paper. Tag bits similarity can be exploited for improving tag bits vulnerability against transient errors. For instance, when an error is detected using the conventional parity check bits, the error could be corrected if the same tag bits were present in one of adjacent cache lines. Faulty tag bits are simply replaced with correct tag bits from the adjacent cache line for error correction. To exploit similar tag bits for transient error protection, we augment the conventional cache architecture with four simple hardware components. To access cache lines in a upper and/or lower cache set than currently accessed cache set, a shifter right after the decoder of a cache or a up/down counter is required. Second, an encoder for generating similarity information between tag bits is needed. Third, a small circuit is necessary for handling similarity bits on cache replacements. Finally, an error correction unit corrects transient errors in the tag bits using the same tag bits from adjacent cache lines. These extra components are simple structures and incur little energy, area, and latency overheads. We evaluated our proposed scheme with in-cache replication (ICR), which was originally proposed to reduce data array vulnerability but can also be applied to reduce tag bits vulnerability. From our experimental results, our scheme shows high error protection coverage of 97% with no virtual performance hit while ICR degrades overall system performance by around 10% and increases DRAM energy consumption by around 20%, on average.

II. RELATED WORK

Different techniques are proposed to protect against transient errors in microprocessors. Protection is generally achieved by employing redundancy; this redundancy may be

in time, in area, or in information. Error Detection Code (EDC) and Error Correction Code (ECC) are used widely for protecting caches against transient errors. However, the conventional ECC protection imposes significant area and latency penalties, making it practical only for large memories and second-level (L2) caches where the increased latency has little impact on performance. To prevent latency increasing, first level (L1) caches tend to employ parity check codes that allow bit error detection, but no correction. Bhattacharya et al. investigate in detail multi-bit soft error rates in large L2 caches and propose a framework based on the amount of redundancy present in the memory hierarchy.

Table I. Error Protection Techniques for Tag Bits

Technique	No Detection		Parity		SECDED		Parity+SimTag		SECDED+SimTag		Parity+ICR		SECDED+ICR	
Cache Line	*Clean*	*Dirty*	*Clean*	*Dirty*	*Clean*	*Dirty*	*Clean*	*Dirty*	*Clean*	*Dirty*	*Clean*	*Dirty*	*Clean*	*Dirty*
False Hit	Error		Hit → Miss		Hit → Miss		Hit → Miss		Hit → Miss		Hit → Miss		Hit → Miss	
False Miss	Ignore	Error	Ignore	Error	Ignore	Correct only 1-Bit Error	Ignore	Correct Odd bits error if same tag exists	Ignore	Correct 2-bit error if same tag exists	Ignore	Correct Odd bits error if Replica exists	Ignore	Correct 2-bit error if replica exists
Replacement Error	Ignore	Error	Ignore	Error	Ignore	Correct only 1-Bit Error	Ignore	Correct Odd bits error if same tag exists	Ignore	Correct 2-bit error if same tag exists	Ignore	Correct Odd bits error if Replica exists	Ignore	Correct 2-bit error if replica exists

Despite the fact that most of the previous work has studied effectiveness in terms of performance, energy, and area overheads, it targets data bits reliability with the assumption that tag bits are intact. However, tag bits also are vulnerable in caches and they have different inherent properties compared to data bits. Kim et al. classify tag bits faults into pseudo-hit, pseudo miss (also called false-hit or false-miss), and multi-hit. Asadi et al. present L1 and L2 cache vulnerability computation algorithms and also deal with algorithms forag vulnerability computation. They analyze in detail the sources of tag bits vulnerability. In-Cache Replication (ICR) has been proposed to replicate frequently accessed cache blocks to dead blocks. Replicated blocks can be used to correct tag bits errors in the active blocks. However, the dead block prediction technique is not always accurate. Thus, ICR increases cache miss and write-back rates resulting in large performance loss and increased energy consumption.

III. OUR PROPOSED APPROACH

A. Effects of tag bits corruptions

Transient errors in tag bits manifest themselves as false hits, false-misses, and replacement errors. A false-miss makes cache hit as a cache miss because of transient error in tag bits. Consequently, the data path gets wrong data on a read and updates a wrong location on a write. A false-hit refers to a cache hit that is actually a miss in the absence of a transient error. If tag bits are corrupted after the line is modified, it may write back to a wrong location in the next level of memory, which is classified as a replacement error. Table I shows tag bits error protection techniques including our proposed scheme and ICR. Except for no detection, clean cache lines do not need error recovery. If erroneous data are in a clean cache line, they can be recovered by invalidating the cache line and by fetching correct data from the next level of memory. If an error occurs in a dirty cache line, hardware exception will be generated and an error handling mechanism will take over for error recovery. Parity check code can cover transient errors on clean caches but it cannot protect dirty cache lines. Single Error Correction Double Error Detection (SECDED) can detect 2-bit errors and correct 1-bit errors.

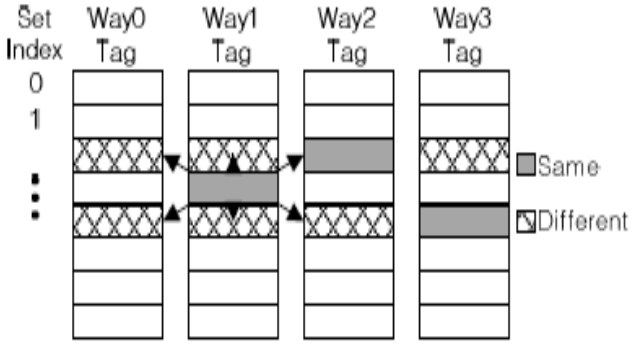

Fig.1. Same tag bits in the adjacent sets

These error codes can be augmented with our proposed technique. If single or multi-bit errors are detected by parity or SEC-DED, our technique will correct the errors using the same tag bits from adjacent cache lines. Our scheme and ICR use location information for error protection. However our scheme exploits prevalent same tag bits while ICR replicates tag bits into other locations by force.

B. Exploiting spatial locality

It is highly probable that same tag bits exist in adjacent cache sets (see Fig.1). This is a consequence of spatial locality of programs. The basic idea of our scheme is to exploit the same tag bits in an adjacent set for correcting erroneous tag bits. Additional bits are required to encode location information which points to exact location of the same tag bits in a upper or lower set. These extra bits are called "Same Tag Information" (STI). STI bits consist of three logical parts; a valid bit, a set location bit, and way location bits. The valid bit indicates that tag bits have the same bits in an adjacent set. The set location bit denotes a lower or upper set and way location bits represent a specific cache way which has the same tag bits. To find same tag bits and set STI bits properly, extra components are required. The tag bits of the missed data are compared with the tag bits of adjacent sets during fetching data from the next level of memory on a cache miss. If there is a match, the STI bits for

the missed data are generated and stored in the data cache. It is possible that replaced tag bits are indicated by the STI bits of adjacent sets (A). If this situation is not handled properly, the STI bits will point non-existing tag bits. To solve this problem, another extra component reads STI bits from lower and upper sets A. If there are matching STI bits, new STI bits are generated. All of these procedures are performed while pipelines are stalled due to cache misses. Therefore, there is no performance degradation virtually.

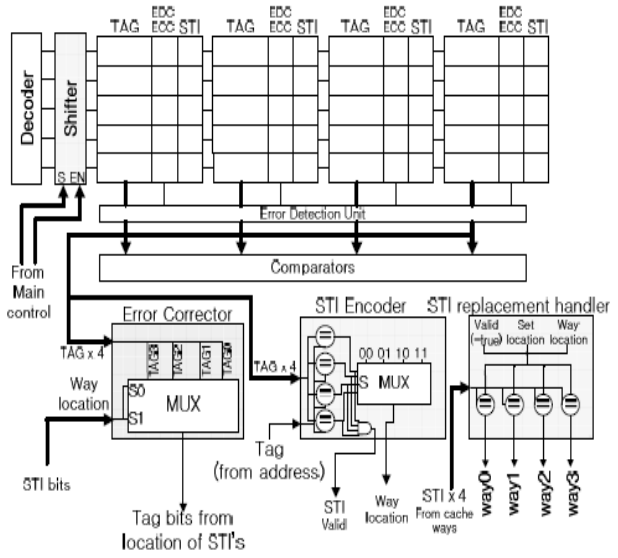

Fig.2. Detailed architecture of Sim Tag.

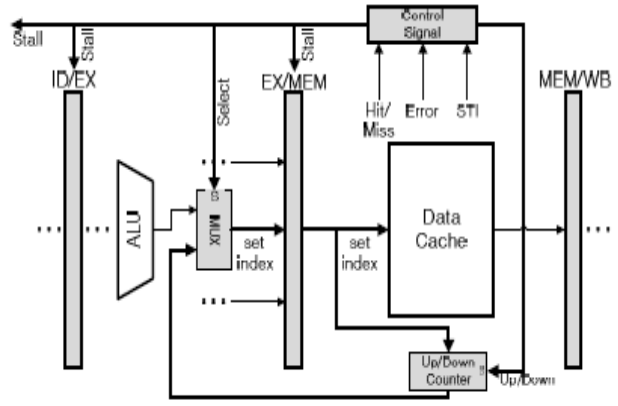

Fig.3. Counter-based technique to access adjacent sets

C. Proposed architecture

In this subsection, we present our proposed architecture and explain additional components which are required to implement it. Fig.2 shows the micro-architecture-level schematic of our architecture called **Sim Tag**. Detailed operation of each component is described below. **Shifter** Our approach uses a shifter for accessing lower or upper cache lines. This is simple and intuitive but it may increase the critical path of data cache access. An alternative approach is to use a counter for hiding the decoding latency. Fig.3 shows the counter-based technique for set index control. Basic operation is same as in the shifter approach but it operates in parallel with cache access.

STI Encoder: To generate STI bits, STI encoder compares the tag bits of cache missed data with the tag bits from lower and upper sets during pipelines are stalled due to cache misses.

STI Replacement Handler: STI replacement handler checks the STI bits in the upper and lower sets on cache replacement. If the STI bits point to replaced tag bits in question, then simply invalidate the STI valid bits and generate new STI bits by finding other same tag bits.

Error Corrector: When errors are detected, this component fetches uncorrupted tag bits from an adjacent cache set by using STI bits (if same tag bits exist) for error correction.

Main Controller: On cache misses or tag bits errors, the pipelines are stalled and, at the same time, the main controller signals the additional shifter (or counter) to access adjacent sets.

There is little area overhead due to the additional components. A set index bits-wide shifter or counter is put into the cache. A 4-to-1 tag bits-wide multiplexer is used for tag matching in the Error Corrector. STI Encoder uses a 2-bits multiplexer, a 1-bit AND gate and tag bits-wide comparators.

Table II. Baseline System Configuration

Configuration Parameter	Value
Processor	
Functional unit	1 Integer/Floating-point ALU 1 Multiply-Accumulate (MAC) Unit
Frequency	600MHz
Cache and Memory Hierarchy	
L1 Instruction Cache	32KB, 32-way, 32byte blocks
L1 Data Cache	8KB, 4-way, 32byte blocks Write-back
Memory	32M SDRAM

Also, STI bits-wide comparators are required for STI Replacement Handler. A few bits are required for the way location. These bits depend on the number of sets inside the data cache. If there are M sets, the size of way location bits is log2 (M)

IV. RESULTS

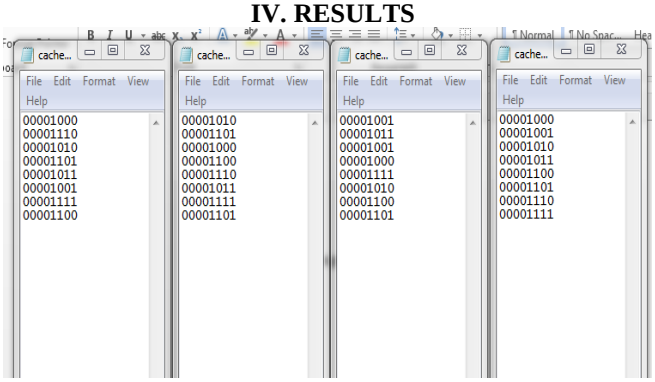

Fig.4. Cache memory.

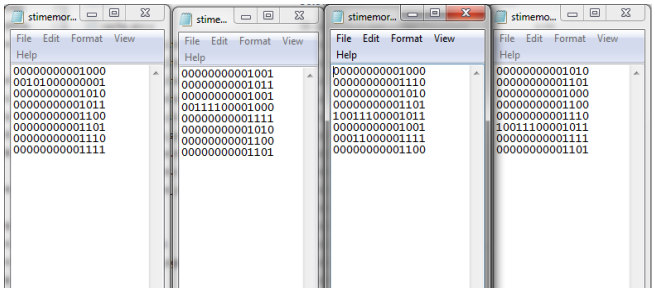

Fig.5. STI memory

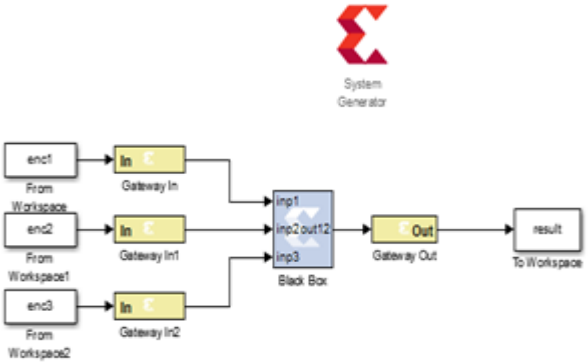

Fig.6. Encoder Design.

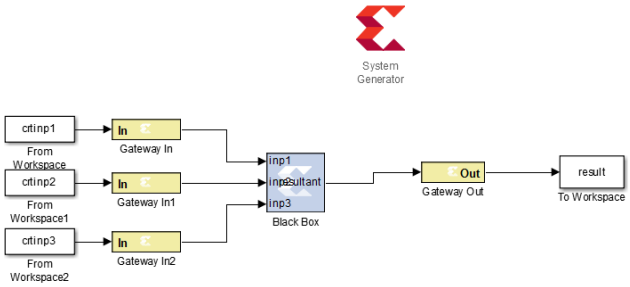

Fig.7 Corrector.

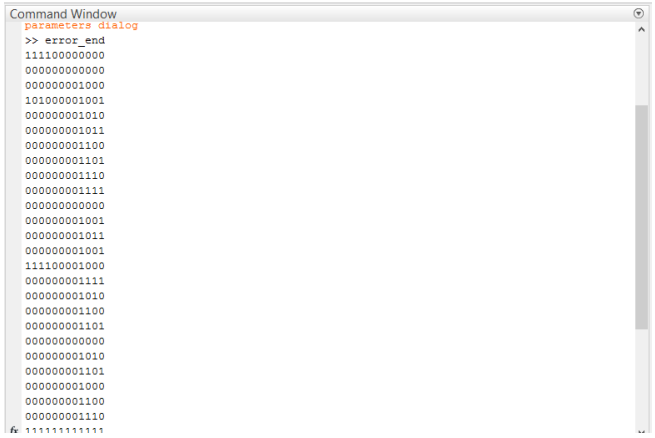

Fig.8. Error correction result

V. CONCLUSION AND FUTURE SCOPE

A. Conclusion

It is becoming important to provide error detection and correction capability for hardware circuits, especially for cache memories. Traditionally, parity or SEC–DED code has been widely used for caches against transient errors. Many techniques are proposed to reduce performance, energy, and area overheads of the conventional error protection mechanisms. However, most of the techniques consider only data bits without considering tag bits corruption. To improve tag bits reliability, this paper exploits tag bits similarity. Most of the tag bits in the data caches have their replica in adjacent cache sets due to spatial locality of programs. Hence, when an error is detected using the conventional parity check bits, the error can be corrected if the same tag bits were present in adjacent cache sets. Faulty tag bits are simply replaced with correct tag bits from the adjacent cache sets for error correction. We evaluate and compare our proposed architecture with the ICR scheme. With the trend of increasing soft error rate, it is becoming important to provide error detection and correction 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 caches have their replica in adjacent cache sets from our experiments. We exploit this tag bits similarity against transient errors. Faulty tag bits are simply replaced with correct tag bits from the adjacent cache lines for error correction.

B. Future scope

Processor caches already play a critical role in the performance of today's computer systems. At the same time, the data integrity of words coming out of the caches can have serious consequences on the ability of a program to execute correctly, or even to proceed. The integrity checks need to be performed in a time-sensitive manner to not slow down the execution when there are no errors as in the common case, and should not excessively increase the power budget of the caches which is already high. A novel solution to this problem by allowing in-cache replication, wherein reliability can be enhanced without excessively slowing down cache accesses or requiring significant area cost increases. The mechanism is fairly power efficient in comparison to other alternatives as well. In particular, the solution replicates data that is in active use within the cache itself while evicting those that may not be needed in the near future. Our experiments show that a large fraction of the data read from the cache has replicas available with this optimization.

VI. REFERENCES

[1] Jeongkyu Hong, Jesung Kim, and Soontae Kim, Member, IEEE, "Exploiting Same Tag Bits to Improve the Reliability of the Cache Memories", IEEE Transactions on Very Large Scale Integration (VLSI) Systems.

[2] Z. Herczeg, A. Kiss, D. Schmidt, N. Wehn, and T. Gyimóthy, "XEEMU: An improved X-Scale power simulator," in PATMOS (Lecture Notes in Computer Science), N. Azémard and L. J. Svensson, Eds. Springer-Verlag, 2007, pp. 300–309.

[3] S. Kim and A. Somani, "Area efficient architectures for information integrity in cache memories," in Proc. Int. Symp. Comput, Archit., 1999, pp. 246–255.

[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] B. Gold, M. Ferdman, B. Falsafi, and K. Mai, "Mitigating multi-bit soft errors in L1 caches using last-store prediction," in Proc. Workshop Archit. Support Giga scale Integr, 2007, pp. 1–8.
[6] S. Mukherjee, J. Emer, T. Fossum, and S. Reinhardt, "Cache scrubbing in microprocessors: Myth or necessity," in Proc. Int. Symp. Dependable Comput, 2004, pp. 37–42.
[7] C. Slayman, "Cache and memory error detection, correction, and reduction techniques for terrestrial servers and workstations," IEEE Trans. Device Math. Rel., vol. 5, no. 3, pp. 397–404, Sep. 2005.
[8] N. Wang and S. Patel, "ReStore: Symptom-based soft error detection in microprocessors," IEEE Trans. Dependable Sec. Comput., vol. 3, no. 3, pp. 188–201, Jul. 2006.
[9] 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.
[10] T. Austin, "DIVA: A reliable substrate for deep submicron micro-architecture design," in Proc. 32nd Annu. Int. Symp. Micro-architecture, 1999, pp. 196–2007.
[11] O. Ergin, O. Unsal, X. Vera, and A. Gonzaez, "Exploiting narrow values for soft error tolerance," IEEE Comput. Archit. Lett., vol. 5, no. 2, pp. 1–12, Dec. 2006.
[12] C. Weaver, J. Emer, S. S. Mukherjee, and S. K. Reinhardt, "Techniques to reduce the soft errors rate in a high-performance microprocessor," in Proc. 31st Annu. Int. Symp. Comput, Archit., Jun. 2004, pp. 264–275.

Author's Profile:

D.Vishwakala, PG Scholar, Dept of VLSI System Design, Swami Ramananda Tirtha Institute of Science and Technology, Telangana, India,
Email: vishwakala431@gmail.com.

Ch. Suresh, Received B.Tech degree in Electronics & Communication Engineering from sri venkateshwara engineering college, surypet, Nalgonda and M.Tech degree in VLSISD Systems from netaji institute of engineering college, nalgonda ,and currently working as an Assistant Professor, Department of Electronics and Communication Engineering in Swami Ramananda Tirtha Institute of Science and Technology, Nalgonda. Area of interest includes communication Systems.

K. Hymavathi, Received B.E degree in Electronics & Communication Engineering from Osmania University, Hyderabad and M.Tech degree in Digital Systems and Computer Electronics from Jawaharlal Nehru Technological University, Hyderabad and currently working as an Associate Professor and Head of the Department of Electronics and Communication Engineering in Swami Ramananda Tirtha Institute of Science and Technology, Nalgonda. Area of interest includes Analog and Digital Communications.