Research and Implement of Cache Fault Tolerance Technique

Jianlei Wei

Research and Development Dept. ORANGE Technology(Tianjin) CO., LTD, Tianjin, China

* Corresponding author: eddiewei2018@126.com

Abstract. For the space application, cache rams are easy to be affected by single event upset (SEU). To protect against SEU error, it is necessary to design fault tolerance cache. Multiple bit-flips faults are increasing as integrated circuits continue to scale into the deep sub-micron regime, the error detection ability need to be improved, so an interleaving grouping parity error detection code scheme is proposed in this paper, each 32-bit data and each 19-bit tag adding 4 valid bits adopt 4 parity bits respectively, the adjacent 4 bits data are assigned to different groups, so all less than 8 bits burst faults can be detected. The hit structure and updating structure of the cache memory are redesigned. No timing penalty occurs since parity checking is performed in parallel with tag comparison. The experimental results show that, by using the proposed technique, there is a small performance loss, but the advantage is the higher reliability valuable for safety system. The failure rate of the cache memory is lower by 6 orders of magnitude.

1. Introduction
The most microprocessors demand large cache memories for higher performance, but cache rams are easy to be affected by SEU. To protect cache memories against SEU error, it is necessary to design fault tolerance cache. The fault techniques including Hamming SEC-DED (single error correction and double error detection) code [1], BCH (Bose-Chaudhuri-Hocquenghem) DEC (double error correction) code [2], TMR (Triple Modular Redundancy)[3] and parity error-detection code[4] are usually adopted to protect cache memories. For the large cache memories, if the ECC (Error correction Code) or TMR techniques are used, the memory storing check bits are too large to be ignored. In addition, the ECC encoding and decoding have a high timing overhead, When the ECC technique are adopted, the typical critical (timing) path of a processor, accessing cache memory, will be more severity. In fact, cache memory is the external memory’s duplication, When a data in cache has a fault, the correct data can be fetched from the external memory again, and the less parity bits alleviate the storage overhead. So to minimize complexity and area overhead, the parity error detection code is adopted. When a fault occurs during a cache access, a cache miss is forced. In dense ram blocks, it is possible that one SEU hit can cause multiple errors [5], typically in adjacent cells. Therefore, an interleaving grouping parity error detection code scheme is proposed, each 32-bit data and each 19-bit tag adding 4 valid bits adopt 4 parity bits respectively, the adjacent 4 bits data are assigned to different groups, so all less than 8 bits burst faults can be detected. The hit structure and updating structure of the cache memory are redesigned. No timing penalty occurs since parity checking is performed in parallel with tag comparison. Experimental results show although there is a small performance loss, the advantage is the higher reliability valuable for safety system. By using the proposed technique, the failure rate of the cache memory is lower by 6 orders of magnitude.
2. Interleaving grouping parity error detection code

To loading instruction in parallel with accessing data, the instruction cache (Icache) is separated from the data cache (Dcache). They are all 16KB, 2-way set-association (4-word lines), as shown in figure 1. Icache/Dcache is divided into tag array and data array. Each tag line stores the high 19 bits of 32-bit physical address, the middle 9 bits physical address indicate in which line the data is located. Each line of data array includes 4 words, the 4 valid bits show whether the corresponding instruction/data is valid. If the high 19 bits physical address is same with the tag value, and the valid bit is valid, a cache hitting is generated.

![Figure 1. I/D cache 0/1 way structure.](image)

A parity bit can only detect odd faults, to detect multiple bits error, the grouping coding is proposed. Because the most multiple bits error are the adjacent bits error, the interleaving grouping technique is adopted. Each 32-bit data and each 23-bit (19-bit tag adding 4 valid bits) tagv uses 4 parity bits respectively. Figure 2 shows the interleaving grouping mode per 32-bit data, the tagv grouping mode is similar. In the 4 groups, as long as one group has a odd fault, this fault can be detected. Therefore, all less than 8-bit burst errors can be detected.

![Figure 2. The interleaving grouping mode per 32-bit data.](image)

For each n-bit parity group (parity bits are included), the SEU failure rate is:

\[
P_p = 1 - (1 - p)^n + \frac{C_n^1 p(1-p)^{n-1} + C_n^2 p^2(1-p)^{n-2} + ... + C_n^n p^n}{n}
\]

The parameter n is odd, for each 32-bit data divided into 4 groups, n=9, namely, each group has 9 bits. The variable p is the SEU failure rate of one bit data. The order of magnitude of SEU failure rate in LEO is 10^-7 error/(bit·day), it can be increased in some abnormal regions, here p=10^-6 error/(bit·day) is selected, so P9=3.60e-11. The SEU failure rate of Icache/Dcache data array is:

\[
P_{data} = 1 - (1 - P_p)^{36} = 5.90e-7 error/(bit·day).
\]

For the tagv, 3 parity groups all have 7 bits, P7=2.10e-11, one parity group has 6 bits, P6=1.50e-11, the SEU failure rate of Icache/Dcache tagv array is:

\[
P_{tag} = 1 - (1 - P_p)^{28} (1 - P_p)^{29} = 7.99e-8 error/(bit·day).
\]

Therefore, the SEU failure rate of Icache/Dcache memory is Pcache=1-(1-Pdata)*(1-Ptag)=6.70e-7 error/(bit·day). But the SEU failure rate of Icache/Dcache memory without fault tolerance design is P=1-(1-p)^4096*32+1024*23=1.43e-1 error/(bit·day). The SEU failure rate is lower by 6 orders of magnitude.
3. Fault tolerance hit structure design

Icache only has read operation, instructions are not rewritten, so read hit structure needs to be designed. But processor can read from and write to Dcache, so read hit structure and write hit one all need to be designed.

3.1. Read hit structure design

Icache read hit structure is same with that of Dcache. Before adding fault tolerance design, as long as instruction’s PC value (or data’s physical address) is same with tag in one way, and the word in the hit way is valid, a cache hit occurs.

To add fault tolerance design, the hit conditions need to be redefined: (1) PC value (or physical address) is matching with the corresponding line tag value in one way; (2) the instruction/data is valid; (3) tagv parity checking is correct; (4) data parity checking is correct. (1) shows a cache hit must be only one way tag matching. The 0 way tag and the 1 way tag may be uniform due to SEU faults, namely, the double hit case occurs, so double hit must have a fault. (3) and (4) indicate if a parity error is detected, a cache miss is forced and the instruction/data is fetched from external memory. Figure 4 shows the fault tolerance read hit structure. In figure 4, the “set” is the line offset address, after the corresponding line is decoded, the two ways’ tag, valid bit, Tag parity bits (Tpar), data and data parity bits (Dpar) are read synchronously. Therefore, no timing penalty occurs since the tag and data parity checking are performed in parallel with tag matching.

3.2. Writing hit structure design

The store instruction can be divided into two kinds: one is STB and STH, they show the write byte and write half word operation respectively; another is ST and STD, they show the write word and write double words operation. The Dcache uses write-through policy. When a STB/STH instruction hit
occurs, Dcache will be updated by Read-Modify-Write (RMW), and so the data must be valid. But for ST/STD instruction, if physical address is matched, dcache will be hit and updated, the new data will be set to valid.

To add fault tolerance design, the write hit conditions need to be redefined. The new write hit conditions for the STB/STH are: (1) physical address of the write data is matching with the tag value in one way; (2) the data is valid; (3) tagv parity checking is correct; (4) data parity checking is correct. The new write hit conditions for the ST/STD are: (1) physical address of the write data is matching with the tag value in one way; (2) tagv parity checking is correct. Figure 5 shows the write hit structure. For STB/STH, to ensure correctness of the data used to piece together, the data need to be checked. For ST/STD, even if the data in dcache is erroneous, the fault will also be removed since a new 32-bit word or double words would be updated to dcache, the data checking is unnecessary.

![Figure 5. Cache writing hit structure.](image)

4. Update policy redefinition

4.1. Read update policy

For the read operation, if a cache miss occurs due to no tag matching or data invalid, the miss data is updated to cache by LRU policy. If SEU faults lead to a cache miss, not only the miss data is updated, but also the erroneous data are corrected. Tag and data of the two ways are all possible to have bit-flips; there are 24–1=15 error cases. If the 15 error cases are taken into account one by one, a larger timing penalty will be occurred in the critical timing path of a processor. To minimize the update circuit’s complexity and correct the faults in time, the read update policy is shown as table 1.

| Parity cases | Hit cases | Read update                      |
|--------------|-----------|---------------------------------|
| Error(tag and/or data) | -- | external memory return ok          |
| Right (tag) | Unvalid hit | Update hit way |

![Table 1. Cache read update policy.](image)
Parity cases | Read update
---|---
--- | ---
Hit cases | external memory return ok | external memory return error
and data) | miss | LRU | No update
Double hit | Update 0 way, Flush 1 way | Flush two ways
hit | No update | No update

If the tag and/or data parity checking are erroneous, a cache miss need to be forced, and the data needs to be fetched from external memory. If the accurate data is returned from external memory, one way is updated and another way is flushed. To remove all faults, only the update data is valid in the same line of the update way, and all 4 valid bits are invalid in the flush way. In this design, if the SEU faults occurs, we assume the 0 way is updated and the 1 way is flushed. If external memory cannot return an accurate data due to a data access exception, the two ways can but be flushed. If cache is double hit, then at least one way has a fault, so cache is updated according to parity checking error case.

| Parity cases | writing update |
|---|---|
| | Hit case | Dcache update cases |
| Stb/sth error(tag and/or data) | -- | Flush two ways |
| right | miss | No update |
| hit | Update hit way |
| Double hit | Flush two ways |
| St/stdlib Error (tag) | -- | Flush two ways |
| Right | miss | No update |
| hit | Update hit way |
| Double hit | Update 0way, Flush 1 way |

4.2. Writing update policy
The writing update policy can be divided into two cases. For the STB/STH instruction, When tag and/or data parity checking are erroneous, a cache miss is forced, Dcache doesn’t update. To correct the faults in time, the corresponding lines of the two ways are flushed. Table 2 shows Dcache writing update policy. When double hit occurs, at least one way has a fault, so the relevant lines of the ways need to be flushed. For the ST/STD instruction, if tag parity checking is erroneous, a cache miss is also forced; the corresponding lines are flushed too. When double hit occurs, we can update 0 way and flush 1 way, for the update way, only the update data in the corresponding line is valid. Flush operations decrease the system performance, but the probability of SEU faults generation is low, to improve microprocessor reliability, the small performance loss is tolerable.
5. Experimental results
To evaluate the reliability and performance of our approach, a program about multiplying two 3x3 matrices is written. 4 test cases are constructed using this program. In the 4 test cases, some faults are injected into Icache tag memory, Icache data memory, Dcache tag memory and Dcache data memory respectively. The 4 test cases are executed in the processor with fault-tolerance cache and with cache no using fault-tolerance mechanism respectively, as shown in figure 6. The case 1 is injection double hit faults into the Icache tag, if cache has an error detection mechanism, the execution time is 1127ns, and the result is true; otherwise, the erroneous instructions are executed, and the result is inaccurate. The case 2 is injection the detectable faults into Icache data, if the error detection mechanism is adopted, the cache miss is forced, the case execution time is long, but the result is true; otherwise, a undefined instruction trap is generated, the test case can’t be ended normally. The case 3 is injection some detectable faults into Dcache tag, if cache has an error detection design, a cache miss is forced; but even if cache does not contain an error detection design, due to no tag matching, these data are also from the external memory, so the result is also accurate. The case 4 is injection some detectable faults into Dcache data, if there is not an error detection design, the case will use the erroneous data in the Dcache, then the result is not true. By adding the error detection mechanism, there is a small performance loss, but the higher reliability is obtained desired for high-safety system.

6. Conclusions
This paper presented an interleaving grouping parity error detection code scheme, each 32-bit data and each 19-bit tag adding 4 valid bits adopt 4 parity bits respectively, the adjacent 4 bits data are assigned to different groups, so all less than 8 bits burst faults can be detected. The hit structure and updating structure of the cache memory are redesigned. No timing penalty occurs since parity checking is performed in parallel with tag comparison. The experimental results show that, by using the proposed technique, there is a small performance loss, but the advantage is the higher reliability valuable for safety system. The failure rate of the cache memory is lower by 6 orders of magnitude.

References
[1] M.Y. Hsiao, A class of optimal minimum odd- weight- column SEC-DED codes. IBM J. Res. Develop., 14:395-401 (1970)
[2] X.Wang, G. Xiao, Error Correcting Code Principles and Methods. Xi’an: Xidian University Press. (2003)
[3] D. Roberts, N. S. Kim, and T. Mudge, On-chip cache device scaling limits and effective fault repair techniques in future nanoscale technology. Microprocessors and Microsystems 32:244-253 (2008)
[4] J.Gaisler, A portable and fault-tolerant microprocessor based on the SPARC v8 architecture. In Dependable Systems and Networks. DSN 2002. Proceedings. International Conference on, 409-415 (2002)
[5] A. D. Tipton, J.A. Pellish, R.A. Reed, R.D. Schrimpf, R.A. Weller, M.H. Mendenhall et al, Multiple-bit upset in 130nm CMOS technology. IEEE Trans. On Nucl. Sci., 53:3259-3264 (2006)