Evaluation of Parity Encoding Schemes via a Instruction TLB Simulator

***Abstract*—This document is a model and instructions for LaTeX. This and the IEEEtran.cls file define the components of your paper [title, text, heads, etc.]. *CRITICAL: Do Not Use Symbols, Special Characters, Footnotes, or Math in Paper Title or Abstract.**

***Index Terms*—soft errors, TLB, parity encoding, false positives, false negatives**

I. Introduction

Critical embedded systems are computers designed to have optimal performance despite some hard constraints, due to their failure being extremely dangerous to human lives or the environment [1]. Some of these might be located in harsh environments, such as space, radioactive mineral extraction sites or places where there is a high incidence of ionizing radiation, cosmic rays or intense electromagnetic activity. These surroundings are responsible for frequent occurrences of soft errors in the computers' memories. They do not cause damage to the hardware itself, but change the data that is stored in them, hence the "soft" description [2].

The TLB (Translation Lookaside Buffer) is an important memory component that is present in current computer architectures. It is a small cache memory usually located inside a MMU (Memory Management Unit) and helps to accelerate the processing by storing some frequently used virtual addresses, known as VPNs (Virtual Page Numbers) [3]. It can be implemented as a CAM (Content Addressable Memory), mapping VPNs to their corresponding physical page numbers [4]. Usually, there are data TLBs, to facilitate the access to data page tables, and instruction TLBs to facilitate access to instruction page tables [5].

When it's corrupted by soft errors, an instruction TLB suffers a change in its stored values, or entries. It may affect the value of just one bit in an entry, which is called a Single Bit Upset (SBU), or simultaneously multiple and usually adjacent bits, in a Multiple Bit Upset (MBU). [6].

Parity encoding or Error Correction Codes (ECCs) are applied to avoid disastrous consequences due to the soft errors. They encode and store the encoded incoming values in the TLB.

II. Soft Errors and their Effects

As said previously, soft errors happen due to the influence of ionizing radiation or cosmic rays over a memory component. As this paper relates mainly to instruction TLBs, it is the memory component which the soft error effects were viewed upon. The following examples are also simplified with a byte word for better understanding.

As a regular way in a TLB, when the incoming value is found in it, it produces a hit and the associated instruction page is retrieved and processed by the CPU. As shown in (1), the incoming VPN 00011001 is stored in the TLB and thus recognized, producing a hit.

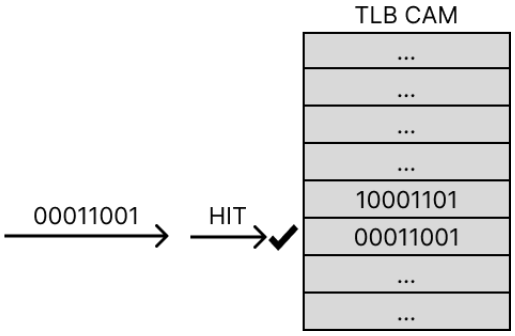

Fig. 1. Example of a TLB hit.

When the requested value cannot be found in the TLB, a miss occurs, causing a page fault which prompts the MMU to retrieve the associated page from the disk memory, which is a slow process in itself. The missed requested value is then stored in the TLB as it was just referenced. In (2), the VPN 00011010 cannot be found in any entries of the TLB, thus a miss is produced and the page needs to be retrieved from the disk memory.

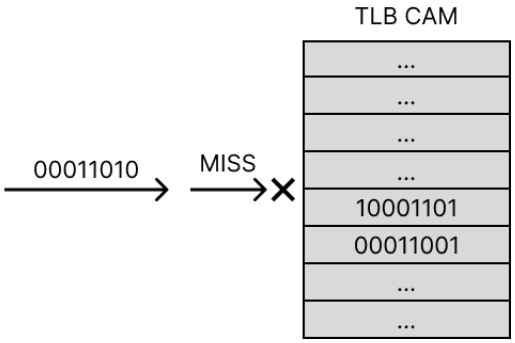

Fig. 2. Example of a TLB miss.

The false negative, or a false miss, happens when an error-stricken VPN in the TLB is requested by the CPU. The VPN will not be recognized in the TLB and thus, it will cause a page fault, causing brief slowness in the processing. As shown in (3), the requested VPN is stored in the TLB, however, it was afflicted by a double adjacent error in its two LSBs. Because

of that, there is no match and a TLB miss is prompted, causing a page walk and slowness in processing.

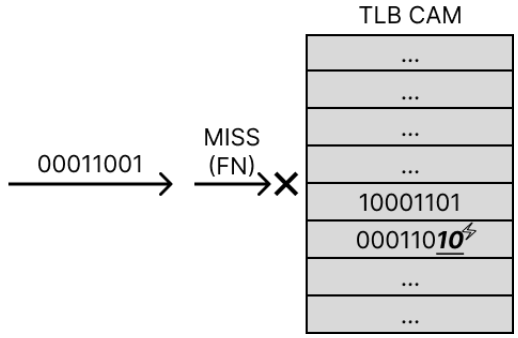

Fig. 3. Example of a TLB false negative.

The false positive, or a false hit, happens when an error-stricken VPN in the TLB is not requested by the CPU but is recognized in the TLB. Because of the TLB hit, the associated page with instructions is then retrieved and executed. However, the retrieved page is associated with the VPN before the error, thus executing instructions unrelated to the requested VPN and possibly causing silent data corruption or system freezing, far more dangerous than the previous effect [11]. In (4), a VPN of value 00011010 is requested but absent in the TLB. However, because of a double adjacent error, a stored VPN of value 00011001 had its value altered to 00011010, turning it into the requested VPN. Due to that, a TLB hit happens and the page frame is returned, but the page frame is related to the VPN 00011001. The instructions are of an unrelated context and their execution by the CPU may cause the effects cited just above.

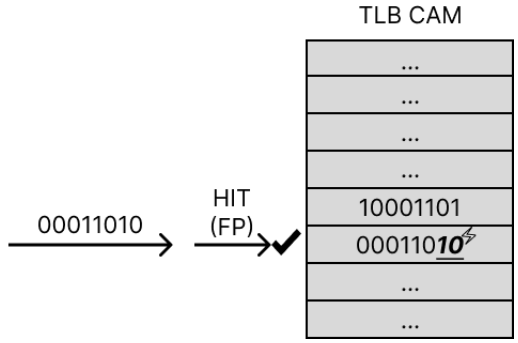

Fig. 4. Example of a TLB false positive.

III. Related work

[7] developed two schemes capable of protecting instruction TLBs from different quantities of soft errors. The first one aims at protecting TLBs from SBUs and is recommended to be applied in a TLB without an embedded parity bit. It uses all of the bits from the incoming VPN to compute its parity and encodes it by replacing the word's MSB with the calculated parity. As it protects the TLB from false positives due to single bit errors (or, by extension, errors in odd quantities), it is not efficient in protecting from false positives of double errors. In (5) the scheme is shown as a successive calculation of XOR gates to obtain the parity of the 32-bit word and store it in the word's MSB.

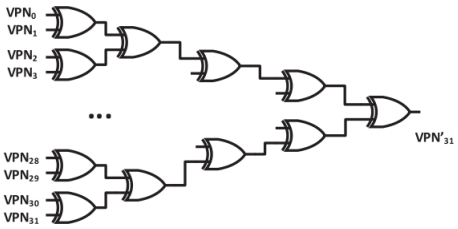

Fig. 5. First parity encoding scheme elaborated in [7].

The second scheme, however, protects TLBs from false positives caused by single, double adjacent and triple adjacent errors and is directed to be used in TLBs with an embedded parity bit mechanism. Instead of using all the bits as the first one does, it uses alternate bits from the incoming VPN, depending on the length of the word, and encodes it similarly, storing the computed value in the MSB of the word. The encoded word is then stored and the parity bit from the TLB mechanism is calculated. In (6) the scheme is shown as another calculation of XOR gates, with fewer, however. The odd bits from a 32-bit word are used to compute the parity, which is then set to the MSB to encode the word. When the encoded word is stored in the TLB, the memory itself calculates the word's parity and sets a parity bit in its entry, thus shielding the entry with a double parity protection.

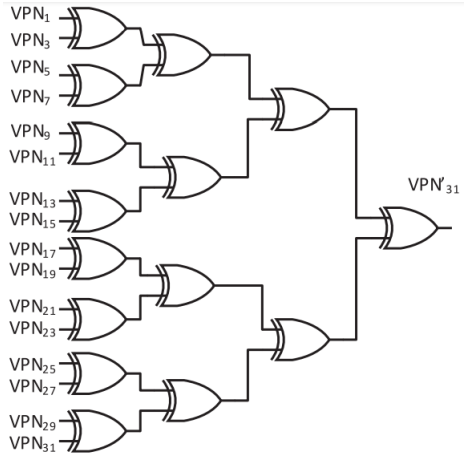

Fig. 6. Second parity encoding scheme for TLBs with parity bit calculation, elaborated in [7].

Later, [8] further developed the previous schemes and into a new one. It computes the parity of the odd and even bits separately and stores them in the first two MSBs from the incoming word. It provides full protection from false positives in single, double adjacent, double and triple error occurrences.

[9] explored the spatial locality principle and proposed reduced versions of the previous scheme, using 4, 8 12 and

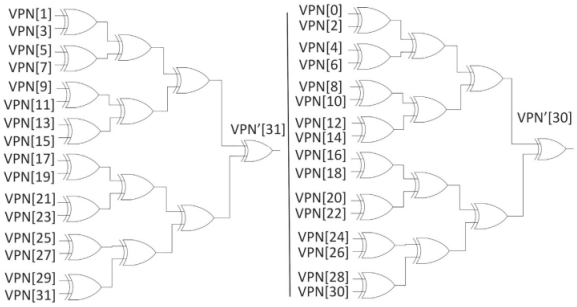

Fig. 7. Parity encoding scheme for TLBs improved in [7].

16 LSBs from the VPN to encode the word the same way the previous scheme works. The results have shown that, with the exception of the 4-LSB scheme, the proposed schemes were effective in protecting TLBs from false positives.

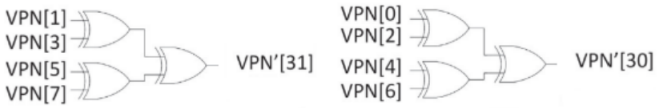

Fig. 8. Parity encoding scheme for TLBs presented in [9].

It must be noted that the mentioned papers have only included the results related to false positives. This paper proposes to evaluate all of the previously presented schemes with different benchmarks and show the false positives and false negatives detected during their simulation.

IV. Experiments

The experiments were conducted similarly to the ones presented in [7], [8] and [9]. A simulator was built to behave as a TLB, with the inclusion of the error injection campaign and ingestion of instruction trace files for its reading and writing procedures. The TLB also has a variable maximum size of entries, but for the simulation, the chosen size is 8 entries.

The trace files were generated using the Pin dynamic instrumentation tool [10]. The traces in the files are converted to binary words of 32 bits for the encoding process.

For each iteration of a simulation run, a soft error is injected into a randomly chosen bit from a random VPN entry in the TLB in a randomly chosen line from the selected trace file. The injected errors vary from a single bit error to a double adjacent error and a triple adjacent error.

An iteration stops whenever a false positive or a false negative is detected, when an error-stricken VPN is removed from the TLB, when it is not possible to inject an error due to the random nature of the error injection or when the trace file is completely consumed by the simulator.

V. Results

References

[1] D. Feitosa et al., "Design Approaches for Critical Embedded Systems: A Systematic Mapping Study," Communications in Computer and Information Science, pp. 243–274, 2018, doi: https://doi.org/10.1007/978-3-319-94135-6_12.

[2] R. C. Baumann, "Soft errors in advanced semiconductor devices-part I: the three radiation sources," IEEE Transactions on Device and Materials Reliability, vol. 1, no. 1, pp. 17–22, Mar. 2001, doi: https://doi.org/10.1109/7298.946456.

[3] A. S. Tanenbaum and H. J. Bos, Modern Operating Systems, 4th Edition. Pearson Higher Education, 2015.

[4] B. Jacob, S. W. Ng, and D. T. Wang, "Cache Case Studies," Memory Systems, pp. 301–312, 2008, doi: https://doi.org/10.1016/b978-012379751-3.50008-4.

[5] J. B. Chen, A. Borg, and N. P. Jouppi, "A simulation based study of TLB performance," ACM SIGARCH Computer Architecture News, vol. 20, no. 2, pp. 114–123, Apr. 1992, doi: https://doi.org/10.1145/146628.139708.

[6] S. Satoh, Y. Tosaka, and S. A. Wender, "Geometric effect of multiple-bit soft errors induced by cosmic ray neutrons on DRAM's," IEEE Electron Device Letters, vol. 21, no. 6, pp. 310–312, Jun. 2000, doi: https://doi.org/10.1109/55.843160.

[7] A. Sánchez-Macián, L. A. Aranda, P. Reviriego, V. Kiani and J. A. Maestro, "Enhancing Instruction TLB Resilience to Soft Errors," in IEEE Transactions on Computers, vol. 68, no. 2, pp. 214-224, 1 Feb. 2019, doi: 10.1109/TC.2018.2874467.

[8] A. Sánchez-Macián, L. A. Aranda, P. Reviriego, and J. A. Maestro, "Reducing false positives due to double adjacent errors in instruction TLBs," Microelectronics Reliability, vol. 102, p. 113494, Nov. 2019, doi: https://doi.org/10.1016/j.microrel.2019.113494.

[9] W. F. d. Almeida, *Tratando erros em TLBs de instruções utilizando um método de paridade simples*. Instituto Federal de Educação, Ciência e Tecnologia do Ceará, 2022. [Online]. Available: www.biblioteca.ifce.edu.br/index.asp?codigo_sophia=108524

[10] C.-K. Luk et al., "Pin: building customized program analysis tools with dynamic instrumentation," ACM SIGPLAN Notices, vol. 40, no. 6, pp. 190–200, Jun. 2005, doi: https://doi.org/10.1145/1064978.1065034.

[11] V. Kiani and P. Reviriego, "Improving Instruction TLB Reliability with Efficient Multi-bit Soft Error Protection," Microelectronics Reliability, vol. 93, pp. 29–38, Feb. 2019, doi: https://doi.org/10.1016/j.microrel.2018.12.011.