UM SIMULADOR DE TLB PARA AVALIAÇÃO DE CÓDIGOS CORRETORES DE ERROS

Hélio Lopes Alves[(1)]; Otávio Alcântara de Lima Júnior[(2)]
Bolsista[(1)]; Instituto Federal de Educação, Ciência e Tecnologia do Ceará, *campus* Maracanaú; hélio.lopes.alves07@aluno.ifce.edu.br.
Orientador[(2)]; Instituto Federal de Educação, Ciência e Tecnologia do Ceará, *campus* Maracanaú; otavio@ifce.edu.br.

1. RESUMO. A TLB (*Translation Lookaside Buffer*) é uma estrutura de memória cache que armazena instruções e que, em ambientes de alto risco, pode ser afetada por erros que invertem seus bits armazenados. O uso de códigos corretores de erros (CCEs) é importante e necessário para mitigar os efeitos maléficos desses erros, como uma falta de página, corrupção de dados ou congelamento de sistema, por exemplo. O objetivo deste trabalho é desenvolver um programa que simule o funcionamento de uma TLB, além de injetar erros pseudoaleatórios, para avaliar o desempenho dos CCEs implementados pela sua taxa de correção de erros. Uma estrutura para simular uma TLB foi construída usando a linguagem Java, acompanhada dos CCEs a serem testados. Neste trabalho, os códigos testados foram a família de CCEs PHICC (*Parity Hamming Interleaving Correction Codes*). Os testes foram feitos em um processo simulado de leitura e escrita na TLB, iterado 500.000 vezes, com inserção de 1 a 8 erros pseudoaleatórios nos bits, para cada CCE. É feita a contagem de casos de falsos positivos, falsos negativos e erros substituídos na memória durante as iterações. Os resultados mostraram que todos os códigos conseguem corrigir totalmente até dois erros adjacentes, além de manter uma taxa de correção semelhante ao trabalho original.
PALAVRAS-CHAVE: TLB. CCE. PHICC. Simulação.

2. INTRODUÇÃO

Existem sistemas embarcados em regiões de alto risco que estão mais suscetíveis a erros de processamento devido ao efeito da radiação ionizante nos componentes de memória. Um dos erros possíveis é o MBU (*Multiple Bit Upset*), um evento que corrompe a memória invertendo alguns bits armazenados em suas células (SEIFERT et al., 2008). Este evento pode causar desde falhas mínimas, como uma simples falta de página, até falhas mais graves, como congelamento do sistema ou corrupção de dados. Para evitar tais consequências, faz-se necessário o uso de códigos corretores de erros (CCEs) para mitigar os efeitos causados pelos MBUs (SÁNCHEZ-MACIÁN et al., 2019a, p. 1).

Os MBUs podem afetar a TLB (*Translation Lookaside Buffer*), uma memória cache importante para o processamento por manter páginas virtuais frequentemente utilizadas próximas e acelerar o processamento dos dados. Quando a TLB possui o auxílio de um CCE, as informações armazenadas são os endereços codificados, podendo ser bytes ou matrizes de bits dependendo do código implementado (SÁNCHEZ-MACIÁN et al., 2019b, p. 3).

Uma forma de avaliar a eficiência de um CCE é verificando sua taxa de correção de erros durante o funcionamento de uma memória auxiliada por ele. Há o caso de um erro de falso negativo, em que a entrada solicitada está na cache porém corrompida e, portanto, diferente da entrada solicitada. Outro caso, o de falso positivo, ocorre quando uma entrada corrompida é igual à entrada solicitada, esse caso sendo mais perigoso pois é possível que ocorra a execução de instruções não

solicitadas e consequentemente corrupção silenciosa dos dados (SÁNCHEZ-MACIÁN et al., 2019b, p. 2).

O objetivo deste trabalho é desenvolver um programa que simule o funcionamento de uma memória cache como a TLB, além de injetar erros pseudoaleatórios, para avaliar o desempenho dos CCEs implementados pela sua taxa de correção de erros.

3 METODOLOGIA/RESULTADOS

O código foi implementado na linguagem Java. A estrutura foi construída para emular uma memória cache de tamanho regulável, com política de substituição LRU (*Least Recently Used*). Para os testes realizados, o tamanho escolhido para a memória foi de 8 posições.

Durante uma iteração da simulação, o programa acessa um arquivo contendo uma lista de 807 endereços e com eles realiza os processos de leitura e escrita na memória cache. Em um momento pseudoaleatório da iteração, são injetados erros em uma linha pseudoaleatória da memória.

Os erros são injetados pseudoaleatoriamente, variam de 1 a 8 erros em bits adjacentes e são feitas 500.000 iterações para cada quantidade de erros. Uma iteração é interrompida na simulação em alguns casos: quando há a detecção de um falso positivo ou de um falso negativo, ou quando uma entrada com erro é substituída e removida da memória. Para cada simulação, ocorre a contagem para cada caso de interrupção.

Os CCEs são avaliados pela quantidade de falsos positivos e falsos negativos detectados durante a simulação. Para este projeto, foi testada a família de CCEs PHICC (MAGALHÃES, 2021).

5. RESULTADOS E DISCUSSÃO

Abaixo, estão as tabelas com os resultados obtidos a partir da simulação de teste para cada CCE da família PHICC. Foi possível observar que os testes obtiveram correção semelhante às do trabalho original, visto que todos apresentam correção total em até dois erros.

Tabela 1 – Resultados da simulação do CCE PHICC(40, 16).

Erros injetados	Falsos positivos	Falsos negativos	Erros substituídos
1	0	0	497.247
2	0	0	497.221
3	1.106	4.249	491.828
4	3.142	14.706	479.470
5	3.541	15.404	478.364
6	4.051	24.849	468.481
7	3.084	22.647	471.633
8	2.338	25.459	469.578

Fonte: autor, 2022.

Tabela 2 – Resultados da simulação do CCE PHICC(44, 16).

Erros injetados	Falsos positivos	Falsos negativos	Erros substituídos
1	0	0	497.162
2	0	0	497.178
3	216	1.399	495.677
4	270	3.638	493.445
5	299	6.072	490.821

6	132	4.881	492.141
7	258	9.910	487.044
8	272	13.371	483.675

Fonte: autor, 2022.

Tabela 3 – Resultados da simulação do CCE PHICC(36, 16).

Erros injetados	Falsos positivos	Falsos negativos	Erros substituídos
1	0	0	497.143
2	0	0	497.213
3	282	3.420	493.552
4	819	19.220	477.331
5	795	15.252	481.157
6	852	22.713	473.827
7	1.187	24.828	471.325
8	1.829	27.174	468.339

Fonte: autor, 2022.

Tabela 4 – Resultados da simulação do CCE PHICC(32, 16).

Erros injetados	Falsos positivos	Falsos negativos	Erros substituídos
1	0	0	497.162
2	0	0	497.266
3	839	5.679	490.823
4	2.716	20.083	474.577
5	3.594	25.016	468.822
6	1.118	35.030	461.366
7	1.657	34.965	460.836
8	1.447	34.306	461.884

Fonte: autor, 2022.

4. CONSIDERAÇÕES FINAIS

Para trabalhos futuros, os objetivos são: fazer a refatoração do código-fonte para implementar outras configurações de memórias cache, outras políticas de substituição, por exemplo; inserir novos códigos corretores de erros para também avaliá-los e comparar seus resultados com os obtidos neste trabalho; e expandir a quantidade de iterações.

5. REFERÊNCIAS

MAGALHÃES, Philippe de Souza. **PHICC: Uma família de Códigos Corretores de Erros para dispositivos de memória**. 2021. Dissertação (Mestrado em Ciência da Computação) – Instituto Federal de Educação, Ciência e Tecnologia do Ceará, Fortaleza, CE, 2021.

SÁNCHEZ-MACIÁN, A. et al. Enhancing Instruction TLB Resilience to Soft Errors. **IEEE Transactions on Computers**, v. 68, n. 2, p. 214-224, fev. 2019a. DOI: https://doi.org/10.1016/j.microrel.2019.113494. Disponível em: https://ieeexplore.ieee.org/document/8488602. Acesso em: jan. 2022.

SÁNCHEZ-MACIÁN, A. et al. Reducing false positives due to double adjacent errors in instruction TLBs. **Microelectronics Reliability**, v. 102, n. 113494, nov. 2019b. DOI: https://doi.org/10.1016/j.microrel.2019.113494. Disponível em: https://www.sciencedirect.com/science/article/abs/pii/S0026271419302975. Acesso em: jan. 2022.

SEIFERT, N. et al. Multi-Cell Upset Probabilities of 45nm High-k + Metal Gate SRAM Devices in Terrestrial and Space Environments. **2008 IEEE International Reliability Physics Symposium**, p. 181-186, abr. 2008. DOI: https://doi.org/10.1109/RELPHY.2008.4558882. Disponível em: https://ieeexplore.ieee.org/document/4558882. Acesso em: ago. 2022.