Algoritmo de Detecção e de Correção de Falhas em Memórias S-RAM para Sistemas Embarcados CLC (Column-Line-Code).

Alexandre Augusto da Penha Coelho, Helano de Souza Castro, Felipe Gaspar Alan e Silva, Victor Dall Mass Fernandes, Maria Luciene Lira Monteiro, Filipe Maciel Lins e Guilherme Bastos Cavalcante.

Laboratório de Engenharia de Sistemas de Computação

Departamento de Engenharia de Teleinformática

Universidade Federal do Ceará

Resumo-Esse artigo apresentará um novo tipo de algoritmo para correção de erros em memorias S-RAM para sistemas embarcados, o CLC (Column-Line-Code). CLC é um código corretor que utiliza conhecimentos sobre código de Hamming(H), código de Hamming estendido (HE), paridade(P) e código Matrix (Ma). O código utiliza as estratégias citadas de modo que combinadas permitirão maior eficácia na correção dos dados. Essa eficácia foi testada e comprovada a partir de simulações de padrões de erros para os casos mais prováveis de erros em memorias S-RAM devido aos efeitos da incidência de radiação nas mesmas.

I. INTRODUÇÃO

Circuitos integrados de alto desempenho que são caracterizados por altas frequências de operação, níveis de baixa voltagem e pequenos ruídos tornam-se cada vez mais sensíveis a falhas devido aos avanços da microeletrônica. A maioria dos componentes eletrônicos comerciais é inapropriada para operar em ambientes expostos a radiação, pois a exposição a partículas alpha e nêutrons [3] podem afetar não só a memória, como a lógica num todo. Além disso, o impacto dessas partículas na camada de silício podem também causar variações na voltagem do circuito.

Grande parte das memórias é sensível a falhas transitórias em hardware. Esses eventos são uma grande preocupação para aplicações espaciais, podendo ocasionar falhas na transmissão de informação e perda de desempenho do sistema. Dois tipos de erro gerados por radiação que afetam memórias são SEU(Single Error Upset) [2] e MEU(Multiple Error Upset) [6]. Com os avanços e a utilização da microeletrônica e dos sistemas embarcados, houve um aumento na taxa de erros em memórias tornando a analise de MEU bastante relevante. Para auxiliar na manutenção do funcionamento das memórias, de modo que elas estejam protegidas dessas eventuais falhas, são utilizados códigos [3], [5].

Hamming [13] é uma estratégia de correção que pode ser utilizada com a finalidade de prevenir SEUs devido a sua eficiência e simplicidade.

O código Matrix [9],[1],[14] é um código que poderá atender a necessidade de se corrigir dois erros numa memória .

Uma vez que há uma probabilidade considerável de que ocorram três falhas numa memoria S-RAM devido aos efeitos de radiação, será necessário um algoritmo mais eficiente que Hamming e Matrix.

Nesse artigo, será apresentada uma nova estratégia de detecção e correção para múltiplas falhas. Sua utilização é baseada em Hamming estendido e paridade formando uma matriz. Foram realizados testes de verificação para detecção e correção onde se atingiu 100% de eficiência em correção para três erros.

Também, foram analisados os casos em que ocorrem quatro e cinco erros, e tivemos como resultado taxas de correção superior ao código Matrix. A detecção mostrou resultados satisfatórios, pois a cobertura de detecção foi 100% em todos os casos analisados.

O restante do artigo apresentará a metodologia da aplicação do código, seu algoritmo, explicando o codificador, o decodificador e as estratégias utilizadas para detecção e correção das palavras. Além disso, serão apresentados os padrões de erro que foram utilizados para verificar a eficácia do CLC, além de gráficos comparativos com os códigos Matrix e Hamming em correção e detecção.

II. TRABALHOS RELACIONADOS

O código de Hamming(7,4) é o código base das séries de corretores e detectores de erro.

Hamming estendido é um código derivado do Hamming e tem como foco a aplicação em sistemas mais sujeitos a SEC-DED (Single Error Correction – Double Error Detection). As codificações de Hamming e do Hamming estendido são similares, diferindo no acréscimo de um bit nas palavras codificadas em Hamming estendido (8,4).

A codificação é realizada a partir do rearranjo dos bits da palavra original, através de operações XOR a fim de gerar os bits da palavra codificada. Na decodificação de Hamming, verifica-se a palavra codificada, e, usando operações XOR, recompomos a palavra original.

A utilização do Hamming estendido também é abordada na criação de algoritmos SEC-DED-TAED[11], em que as palavras são codificadas e ordenadas em uma matriz que permite detectar até três erros adjacentes na palavra código, pois a partir do bit de paridade gerado no Hamming estendido, é feita uma síndrome que detecta um erro.

A incidência de radiação [3],[8] em memorias S-RAM pode ocasionar em erros adjacentes. Isso torna necessária a utilização de algoritmos de correção com cobertura maior que um, portanto o código de Hamming (8,4) não é recomendado, já que apesar de ser possível detectarmos dois erros não poderemos corrigir a informação completamente.

O código Matrix é utilizado em memórias S-RAM de 32 bits, 64 bits e 128 bits. Esse código consiste na utilização de conhecimentos envolvendo código de Hamming e paridade, onde a palavra é gerada a partir do rearranjo dos bits da palavra de Hamming codificada. O código Matrix tem os bits da palavra codificada organizadas no formato similar ao de uma matriz. Essa estrutura é composta por quatro linhas com palavras de Hamming codificadas e uma linha adicional com os bits de paridade das colunas que auxiliam na detecção dos erros nos bits adjacentes. Esse rearranjo permite uma maior eficiência na detecção e na correção de erros adjacentes, sem comprometer o rendimento.

Esse código tem foco na aplicação em sistemas sujeitos a DEC-DED (Double Error Correction – Double Error Detection)[6]. Sua decodificação consiste na verificação das síndromes dos check bits e dos bits de paridade, de modo que, se não houver qualquer anomalia nas palavras verificadas, ocorrerá reorganização dos bits que compunham a palavra original.

III. TÉCNICA PROPOSTA

CLC tem sua palavra codificada composta por 40 bits como mostra a figura 1. Sendo 16 da palavra que se deseja codificar (C), oito de paridade (P), quatro de paridade de palavra (Pa) e 12 check bits (CB).

C_1	C_2	C_3	C_4	CB_1	CB_2	CB_3	Pa_1
C_5	C_6	C_7	C_8	CB_4	CB_5	CB_6	Pa_2
C_9	C_{10}	C_{11}	C_{12}	CB_7	CB_8	CB_9	Pa_3
C_{13}	C_{14}	C_{15}	C_{16}	CB_{10}	CB_{11}	CB_{12}	Pa_4
P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8

Figura 1. Modelo da palavra codificada CLC.

Para o código CLC, utilizamos uma palavra de 16 bits, dividida em quatro palavras menores de quatro bits. Essas palavras de quatro bits serão codificadas em Hamming estendido e os bits serão realocados de acordo com a tabela acima. Em seguida, calculamos os seus CB's.

$CB_1 = C_2 \oplus C_3 \oplus C_4$

$CB_2 = C_1 \oplus C_3 \oplus C_4$

$CB_3 = C_1 \oplus C_2 \oplus C_4$

Como utilizamos Hamming estendido, as Pa's são calculadas fazendo operação XOR com todos os elementos da linha.

$Pa_1 = C_1 \oplus C_2 \oplus C_3 \oplus C_4 \oplus CB_1 \oplus CB_2 \oplus CB_3$

$Pa_2 = C_5 \oplus C_6 \oplus C_7 \oplus C_8 \oplus CB_4 \oplus CB_5 \oplus CB_6$

$Pa_3 = C_9 \oplus C_{10} \oplus C_{11} \oplus C_{12} \oplus CB_7 \oplus CB_8 \oplus CB_9$

$Pa_4 = C_{13} \oplus C_{14} \oplus C_{15} \oplus C_{16} \oplus CB_{10} \oplus CB_{11} \oplus CB_{12}$

Fazendo operações XOR com os elementos das colunas encontramos os valores da paridade da coluna.

$P_1 = C_1 \oplus C_5 \oplus C_9 \oplus C_{13}$

$P_2 = C_2 \oplus C_6 \oplus C_{10} \oplus C_{14}$

$P_3 = C_3 \oplus C_7 \oplus C_{11} \oplus C_{15}$

$P_4 = C_4 \oplus C_8 \oplus C_{12} \oplus C_{16}$

$P_5 = CB_1 \oplus CB_4 \oplus CB_7 \oplus CB_{10}$

$P_6 = CB_2 \oplus CB_5 \oplus CB_8 \oplus CB_{11}$

$P_7 = CB_3 \oplus CB_6 \oplus CB_9 \oplus CB_{12}$

O sistema de verificação do CLC consiste em verificar cada linha e cada coluna a partir da geração de síndromes de paridade (SP) e síndrome dos check bits (SCB). Para isso, devemos primeiro calcular os CB's da palavra codificada ('CB) e em seguida comparar com os CB's gerados anteriormente, dessa forma a relação da SCB é:

$CB_1 \oplus' CB_1$.

De maneira semelhante, a SP é igual a:

$P_1 \oplus' P_1$.

A detecção de erros na palavra codificada dá-se quando as síndromes geradas em linhas e colunas tem valores diferentes de zero. Além dessas, uma síndrome é gerada para casos onde temos três erros na palavra codificada, essa é a síndrome da paridade da palavra (SPa), onde o calculo dela segue-se com o mesmo principio das ultimas mostradas.

Supondo o caso em que a palavra "1000100001111111" é transmitida, ao codifica-la temos que seus check bits CB_1-CB_{12} serão iguais a "011011100111", os bits de paridade P_1-P_8 serão "10000111" e os bits de paridade da palavra Pa_1-Pa_4 serão "1101". Supondo que durante a leitura da palavra codificada os bits C_1, C_2 e C_3 apresentam falhas, agora C_1 é igual a 0, C_2 a 1 e C_3 a 1. Calculando 'CB_1-'CB_{12} tem-se "011011100111", calculando 'P_1-'P_8 tem-se "01100110". Usando o algoritmo de verificação do CLC podemos detectar as três falhas no processo e corrigi-las.

É realizado o cálculo das síndromes de check bits e de paridade, de modo que o algoritmo verifique a linha e a coluna que possuemos erros.

Fazendo a SCB temos:

(011011100111) $\oplus$ (011011100111) = (000000000000).

Fazendo a SP temos:

(10000111) $\oplus$ (01100111) = (11100000)

O algoritmo conclui que há três erros em alguma linha. No entanto, a SCB nos mostra um resultado inconclusivo, pois aparentemente todas as linhas estão corretas. Dessa forma, o algoritmo recorre a SPa, para saber se realmente houve modificação nas linhas. Para isso, calcula-se 'Pa_1-'Pa_4 e temos "0101". Fazendo a SPa temos:

(1101) $\oplus$ (0101) = (1000)

A partir disso e em conjunto com os dados das síndromes geradas, o algoritmo do CLC conclui que na primeira linha, houve três erros nos três primeiros bits da primeira linha da palavra codificada. Em seguida algoritmo inverte os bits filtrados pelas linhas e pelas colunas, e a palavra codificada é decodificada.

A decodificação consiste em pegar os quatro primeiros bits das quatro primeiras linhas que foram codificadas e reordena-las para o formato original da palavra que foi transmitida.

A figura 2 mostra o processo de funcionamento do CLC.

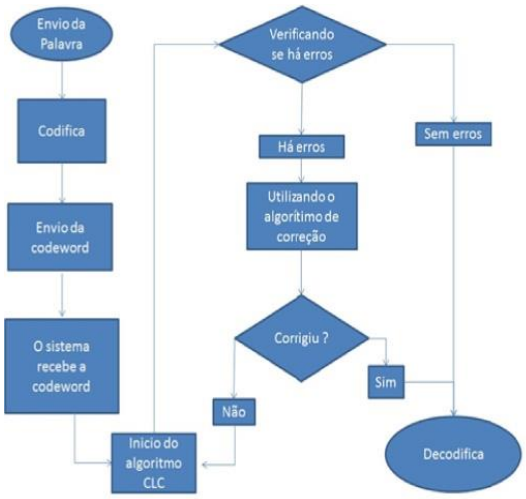

Figura 2. Fluxograma mostrando a sequência de processos para a verificação da palavra.

IV. EXPERIMENTOS COM INJEÇÃO DE FALHAS

Para a realização dos testes e validação da técnica proposta neste artigo, escolhemos padrões de erros com fator de distância menor ou igual a um. Tanto falhas singulares como falhas múltiplas foram injetadas.

A verificação do código CLC foi escrita em SystemVerilog, com algoritmos de inserção de falhas complexos, permitindo que a área de cobertura da verificação fosse de aproximadamente 100%.

A seguir, temos os padrões de erro que foram utilizados nos testes envolvendo o CLC. Os padrões de dois e três erros tiveram eficácia máxima. Já os padrões de quatro e cinco erros, não atingiram a eficiência máxima de correção. Os padrões utilizados foram.

Tabela I Padrão de 2 Erros.

0	1	2	3	4	5	6	7
8	9	10	11	12	13	14	15
16	17	18	19	20	21	22	23
24	25	26	27	28	29	30	31
32	33	34	35	36	37	38	39

Tabela II Padrão de 2 Erros.

0	1	2	3	4	5	6	7
8	9	10	11	12	13	14	15
16	17	18	19	20	21	22	23
24	25	26	27	28	29	30	31
32	33	34	35	36	37	38	39

Tabela III Padrão de 2 Erros.

0	1	2	3	4	5	6	7
8	9	10	11	12	13	14	15
16	17	18	19	20	21	22	23
24	25	26	27	28	29	30	31
32	33	34	35	36	37	38	39

Tabela IV Padrão de 3 Erros.

0	1	2	3	4	5	6	7
8	9	10	11	12	13	14	15
16	17	18	19	20	21	22	23
24	25	26	27	28	29	30	31
32	33	34	35	36	37	38	39

Tabela V Padrão de 3 Erros.

0	1	2	3	4	5	6	7
8	9	10	11	12	13	14	15
16	17	18	19	20	21	22	23
24	25	26	27	28	29	30	31
32	33	34	35	36	37	38	39

Tabela VI Padrão de 3 Erros.

0	1	2	3	4	5	6	7
8	9	10	11	12	13	14	15
16	17	18	19	20	21	22	23
24	25	26	27	28	29	30	31
32	33	34	35	36	37	38	39

Tabela VII Padrão de 4 Erros.

0	1	2	3	4	5	6	7
8	9	10	11	12	13	14	15
16	17	18	19	20	21	22	23
24	25	26	27	28	29	30	31
32	33	34	35	36	37	38	39

Tabela VIII Padrão de 4 Erros.

0	1	2	3	4	5	6	7
8	9	10	11	12	13	14	15
16	17	18	19	20	21	22	23
24	25	26	27	28	29	30	31
32	33	34	35	36	37	38	39

Tabela IX Padrão de 5 Erros.

0	1	2	3	4	5	6	7
8	9	10	11	12	13	14	15
16	17	18	19	20	21	22	23
24	25	26	27	28	29	30	31
32	33	34	35	36	37	38	39

Tabela X Padrão de 5 Erros.

0	1	2	3	4	5	6	7
8	9	10	11	12	13	14	15
16	17	18	19	20	21	22	23
24	25	26	27	28	29	30	31
32	33	34	35	36	37	38	39

No tópico a seguir mostraremos, em números, a eficiência de correção e de detecção do CLC.

V. RESULTADOS E ANÁLISE

Após a realização dos testes com os padrões escolhidos, coletamos os resultados obtidos e formulamos gráficos de correção e de detecção que serão mostrados a seguir.

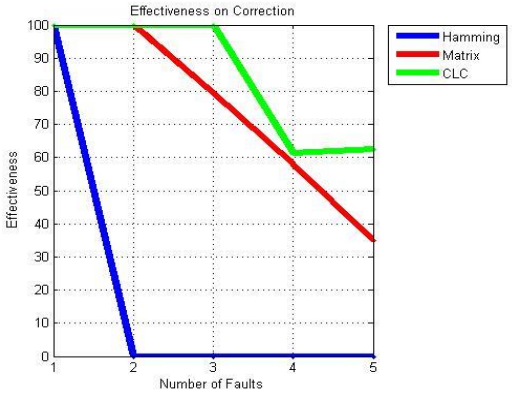

Figura 3. Gráfico de correção.

Tabela XI Tabela de correção.

N° de erros	Eficácia Hamming	Eficácia Matrix	Eficácia CLC
1	100%	100%	100%
2	0%	100%	100%
3	0%	79,31%	100%
4	0%	57,86%	61,32%
5	0%	35,02%	62,52%

Observamos na figura 3 e na tabela XI que o código corrige 100% das falhas se ocorrerem até três erros, como dito anteriormente. Além disso, temos que a porcentagem de correção para cinco erros é maior que para quatro.

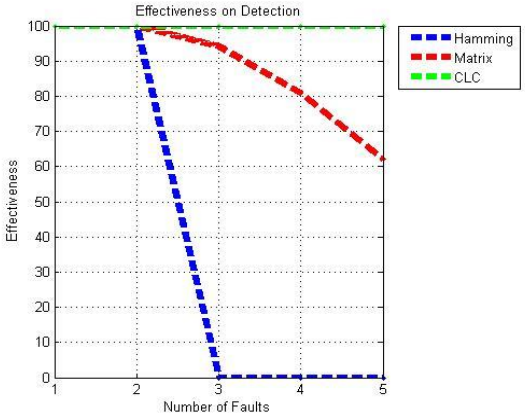

Figura 4. Gráfico de detecção.

Tabela XII Tabela de detecção.

N° de Erros	Eficácia Hamming	Eficácia Matrix	Eficácia CLC
1	100%	100%	100%
2	100%*	100%	100%
3	0%	94,01%	100%
4	0%	80,71%	100%
5	0%	62,20%	100%

*Hamming estendido detecta até dois bits

Observamos na figura 4 e na tabela XII a eficiência do algoritmo de detecção do CLC, onde se espera que a detecção se mantenha com 100% até oito bits.

Os resultados obtidos do experimento mostraram-se bastante satisfatórios, pois CLC mostrou-se mais eficaz que os códigos de Hamming e Matrix tanto em correção quanto em detecção como pode ser observado na figura 5.

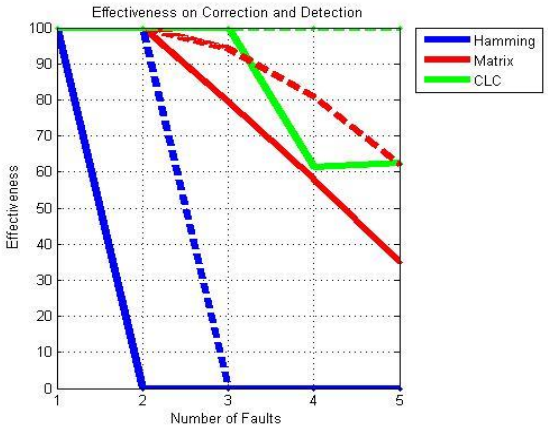

Figura 5. Gráfico relacionando a correção e a detecção do algoritmo.

Isso pode ser justificado devido ao código CLC possuir mais ferramentas de detecção que os outros códigos comparados. Com isso, o

algoritmo de correção terá mais informações para analisar e para realizar as correções necessárias.

VI. CONCLUSÃO

Esse artigo apresentou um novo código corretor de erros, o CLC. Ele utiliza conhecimentos sobre o código Matrix, que por sua vez utiliza os códigos de Hamming e de paridade. O CLC também utiliza conhecimentos sobre Hamming estendido, sendo esse fundamental para permitir a correção de 100% dos casos para erros em três bits na palavra codificada.

Além disso, a cobertura de detecção de erro do CLC mostrou-se bastante abrangente, haja vista que a detecção e a correção do código tem eficiência superior ao do Matrix e do Hamming.

O CLC mostra que é um código com lógica relativamente simples e que apresenta resultados satisfatórios.

VII. REFERÊNCIAS

[1]Costas A. Argyrides, Pedro Reviriego, Dhiraj K. Pradhan, Juan Antonio Maestro, "Matrix-Based Codes for Adjacent Error Correction," IEEE Trans. Nucl. Sci., vol. 57, no. 4, pp. 01, Aug. 2010.

[2]P. Ferreyra, C. Marques, R. Ferreyra, and J. Gaspar, "Failure map functions and accelerated mean time to failure tests: New approaches for improving the reliability estimation in systems exposed to single event upsets," IEEE Trans. Nucl. Sci., vol. 52, no. 1, pp. 494–500, Feb. 2005.

[3]P. Hazucha and C. Svensson, "Impact of CMOS technology scaling on the atmospheric neutron soft error rate," IEEE Trans. Nucl. Sci., vol. 47, no. 6, pp. 2586–2594, Dec. 2000.

[4]S. Satoh, Y. Tosaka, and S. Wender, "Geometric effect of multiple-bit soft errors induced by cosmic ray neutrons on DRAM's," IEEE Electron Device Lett., vol. 21, no. 6, pp. 310–312, Jun. 2000.

[5]D. Radaelli, H. Puchner, S. Wong, and S. Daniel, "Investigation of multibit upsets in a 150 nm technology SRAM device," IEEE Trans. Nucl. Sci., vol. 52, no. 6, pp. 2433–2437, Dec. 2005.

[6]P. Reviriego, J. A. Maestro, and C. Cervantes, "Reliability analysis of memories suffering multiple bit upsets," IEEE Trans. Device Mater. Rel., vol. 7, no. 4, pp. 592–601, Dec. 2007.

[7]R. Hentschke, F. Marques, F. Lima, L. Carro, A. Susin, and R. Reis, "Analyzing area and performance penalty of protecting different digital modules with hamming code and triple modular redundancy," in Proc. 15th Symp. on Integrated Circuits and Systems Design, 2002, pp. 95–100.

[8]J. Karlsson, P. Liden, P. Dahlgren, R. Johansson, and U. Gunneflo, "Using heavy-ion radiation to validate fault-handling mechanisms," IEEE Micro., vol. 14, no. 1, pp. 8–11, 1994, 13–23.

[9]Costas Argyrides, Hamid R. Zarandi, Dhiraj K. Pradhan,"Matrix Codes: Multiple Bit Upsets Tolerant Method for SRAM Memories" 22nd IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems

[10]Zhen Wang, Mark Karpovsky, Fellow, IEEE, and Ajay Joshi, Member, IEEE Nonlinear Multi-Error Correction Codes for Reliable MLC nand Flash Memories" IEEE Transactions on Very Large Scale Integratrion (VLSI) systems, vol. 20, no. 7, july 2012.

[11]Alfonso Sánchez-Macián, Member, IEEE, Pedro Reviriego, Member, IEEE, and Juan Antonio Maestro, Member, IEEE "Hamming SEC-DAED and Extended Hamming SEC-DED-TAED codes through selective shortening and bit placement".

[12] Mojtaba Valinataj and Saeed Safari "Fault Tolerant Arithmetic Operations with Multiple Error Detection and Correction" 22nd IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems, 2007

[13]Jatinder Singh and Jaget Singh "A Comparative Study of Error Detection and Correction Coding Techniques" 2012 Second International Conference on Advanced Computing & Communication Technologies

[14] Costas Argyrides, Member, IEEE, Dhiraj K. Pradhan, Fellow, IEEE, and Taskin Kocak "Matrix Codes for Reliable and Cost Efficient Memory Chips", IEEE Transaction Very Large Scale Integration (VLSI) Systems, vol. 19, no. 3, March 2011.