

UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Tiago M. Rohde

**CÓDIGO CORRETOR DE ERROS MULTI BIT UPSET COM
DUPLA VERIFICAÇÃO**

Santa Maria, RS
2020

Tiago M. Rohde

**CÓDIGO CORRETOR DE ERROS MULTI BIT UPSET COM DUPLA
VERIFICAÇÃO**

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação (PPGCC) da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção do título de **Mestre em Ciência da Computação**.

Orientador: Prof. Dr. João Baptista dos Santos Martins

Santa Maria, RS

2020

Rohde, Tiago M.

Código Corretor de Erros multi bit upset com dupla verificação /
por Tiago M. Rohde. – 2020.

85 f.: il.; 30 cm.

Orientador: João Baptista dos Santos Martins

Dissertação (Mestrado) - Universidade Federal de Santa Maria,
Centro de Tecnologia, Pós-Graduação em Ciência da Computação, RS,
2020.

1. MBU. 2. SEE. 3. SRAM. 4. Código Corretor de Erros. I. Bap-
tista dos Santos Martins, João. II. Código Corretor de Erros multi bit
upset com dupla verificação.

© 2020

Todos os direitos autorais reservados a Tiago M. Rohde. A reprodução de partes ou do todo
deste trabalho só poderá ser feita mediante a citação da fonte.

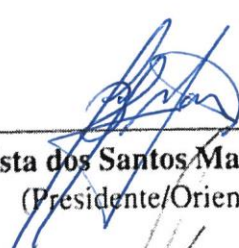
E-mail: tiago.mri@hotmail.com

Tiago Mallmann Rohde

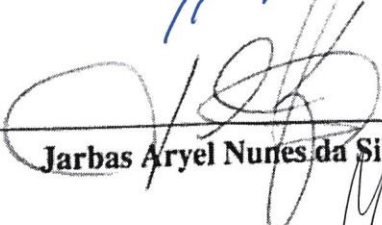
**CÓDIGO CORRETOR DE ERROS MULTI BIT UPSET COM DUPLA
VERIFICAÇÃO**

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação (PPGCC) da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção do título de **Mestre em Ciência da Computação**.

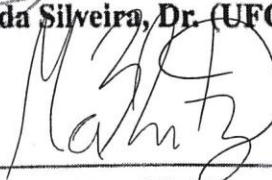
Aprovado em 09 de Setembro de 2020



João Baptista dos Santos Martins, Dr. (UFSM)
(Presidente/Orientador)



Jarbas Aryel Nunes da Silveira, Dr. (UFC)



Mateus Beck Rutzig, Dr. (UFSM)

Santa Maria, RS

2020

RESUMO

CÓDIGO CORRETOR DE ERROS MULTI BIT UPSET COM DUPLA VERIFICAÇÃO

AUTOR: TIAGO M. ROHDE

ORIENTADOR: JOÃO BAPTISTA DOS SANTOS MARTINS

Grandes avanços tecnológicos vêm sendo desenvolvidos, especialmente no cunho espacial. O padrão CubeSat permitiu que diversas entidades também participassem do desenvolvimento dessas tecnologias. Com isso, os circuitos que integram os mais diversos sistemas vêm evoluindo aceleradamente e a quantidade de células lógicas dentro do mesmo espaço físico aumenta, ou seja, os circuitos estão cada vez mais densos. No espaço, a radiação cósmica é mais intensa, fazendo com que os erros provindos do impacto de partículas altamente carregadas tornassem cada vez mais preocupantes, tendo em vista que Códigos Corretores de Erros precisam ser cada vez mais eficazes para garantir a integridade da informação. Esta pesquisa expõe o desenvolvimento de um algoritmo de dupla verificação, onde os bits que checam os dados são previamente verificados por outros bits de paridade, com o intuito de atenuar os falsos positivos durante o processo de correção dos bits afetados pelos *Single Event Effect* e consequentemente ampliar a cobertura de erros de *upsets* dentro da *Static Random Access Memmory*. As simulações, foram desenvolvidas no MATLAB e contiveram-se na programação de alto nível, demonstraram que o algoritmo possui diferentes índices de correção comparado a técnicas já consolidadas no meio científico. A cobertura de erros ficou distribuída em 97% em um raio 3X3 para todas as possibilidades possíveis, 55% em um raio 4X4 para todas as possibilidades de até 9 erros e 68,08% de correção considerando as 90.485.041 simulações que compreendem impactos únicos e múltiplos impactos de partículas de energia altamente carregadas.

Palavras-chave: MBU. SEE. SRAM. Código Corretor de Erros.

ABSTRACT

CÓDIGO CORRETOR DE ERROS MULTI BIT UPSET COM DUPLA VERIFICAÇÃO

AUTHOR: TIAGO M. ROHDE

ADVISOR: JOÃO BAPTISTA DOS SANTOS MARTINS

Major technological advances have been developed, especially in the space field. The CubeSat standard allowed several entities to also participate in the development of these technologies. Thus, the circuits that integrate the most diverse systems have been evolving rapidly and the amount of logical cells within the same physical space increases, that is, the circuits are increasingly dense. In space, cosmic radiation is more intense, causing the errors stemmed from the impact of highly charged particles become increasingly worrying, given that the Error Correcting Codes need to be increasingly effective to ensure the information integrity. This research exposes the development of a double check algorithm, where the bits that check the data are previously checked by other parity bits, in order to mitigate false positives during the process of correcting the bits affected by the Single Event Effect and consequently expand the coverage of upsets errors within the Static Random Access Memmory. The simulations, were developed in MATLAB and were contained in the high level programming, showed that the algorithm has different adjustment rates compared to already established techniques in the scientific environment. The coverage of errors was distributed in 97% in a 3X3 radius for all the possibilities, 55% in a 4X4 radius for all possibilities of up to 9 errors and 68,08% of correction contemplating the 90.485.041 simulations that comprise single and multiple impacts of highly charged energy particles.

Keywords: MBU. SEE. SRAM. Error Correction Code.

LISTA DE FIGURAS

Figura 1 –	Evolução da quantidade de células lógicas por espaço físico	13
Figura 2 –	Relação entre o tamanho do circuito e a probabilidade de erro	13
Figura 3 –	Possíveis configurações para CubeSats	17
Figura 4 –	Sistemas presentes nos satélites	18
Figura 5 –	Colisão de partículas em diferentes tamanhos de memórias	20
Figura 6 –	Padrão dos MBU verificados durante as simulações	22
Figura 7 –	Efeito de SEE do tipo SE no circuito	23
Figura 8 –	Relação do número de células afetadas devido a tecnologia do chip	24
Figura 9 –	Relação do número de células afetadas devido a construção do chip	24
Figura 10 –	MBU na mesma palavra X MBU em palavras distintas	26
Figura 11 –	Formato dos erros na tecnologia 65 nm	26
Figura 12 –	Padrão de erro para FPGA de 45 nm	27
Figura 13 –	Processo de armazenamento	29
Figura 14 –	Funcionamento de um CCE	30
Figura 15 –	Matriz Identidade (I)	31
Figura 16 –	Matriz Geradora (G)	31
Figura 17 –	Transposta de Q	32
Figura 18 –	Matriz de paridade H	32
Figura 19 –	Vetor de correção designado para correção de erro	32
Figura 20 –	Representação TMR na memória	34
Figura 21 –	Representação dos Dados antes e depois de aplicar <i>Interleaving</i>	34
Figura 22 –	Estrutura do código Matrix	36
Figura 23 –	Estrutura do CLC(16,40)	37
Figura 24 –	Estrutura do CLC(16,39)	37
Figura 25 –	Matriz Bidimensional	38
Figura 26 –	Distribuição dos bits dentro da matriz usando 4 bits de paridade	39
Figura 27 –	Distribuição dos bits dentro da matriz usando 5 bits de paridade	39
Figura 28 –	Matriz binária	41
Figura 29 –	Conjunto C após ser calculado utilizando os bits dos dados D quaisquer.	45
Figura 30 –	Dependências dos cálculos	46
Figura 31 –	Exemplo alocação dos dados	47
Figura 32 –	Exemplo Cálculo do C, utilizando os Bits da Mensagem M	47
Figura 33 –	Exemplo Calculo do PC e PD	47
Figura 34 –	Exemplo Calculo do HPC e BC	48
Figura 35 –	Exemplo Calculo do CC E BP	48
Figura 36 –	Matriz com <i>Interleaving</i>	48
Figura 37 –	Possibilidade de auto anulação.	50
Figura 38 –	Fluxograma de Codificação dos bits	50
Figura 39 –	Integridade do C	52
Figura 40 –	Integridade da PC	53
Figura 41 –	Tamanho da Janela de Erros de ordem 3	57
Figura 42 –	Bateria de testes	57
Figura 43 –	Possibilidades para 2 Erros de <i>upsets</i> dentro do raio de alcance do SEE 3X3	58
Figura 44 –	Possibilidades para 9 Erros de <i>upsets</i> dentro do raio de alcance do SEE 3X3	58
Figura 45 –	Comparação da eficiência com e sem o <i>Interleaving</i>	59

Figura 46 –	Possibilidades de <i>upsets</i> em uma Janela de Erros 4X4	60
Figura 47 –	Comparação da eficiência pelo tamanho do raio do impacto	61
Figura 48 –	Cobertura de erros outras técnicas	61
Figura 49 –	Cobertura de Erros por Técnica	62
Figura 50 –	Bits a mais utilizados como paridade	63
Figura 51 –	Teste de 2 MBU de nível 2.....	64
Figura 52 –	Exemplo de colisão	65
Figura 53 –	Comparação da cobertura em múltiplos eventos.....	66
Figura 54 –	Correção de 100% das configurações de erros	67
Figura 55 –	comparação tamanho distintos de Janela de Erro	68
Figura 56 –	Exemplo 1: MBU na memória utilizando a técnica proposta	69
Figura 57 –	Exemplo 2: MBU na memória utilizando a técnica proposta	70
Figura 58 –	Exemplo 3: MBU na memória utilizando a técnica proposta	70
Figura 59 –	Exemplo 4: MBU na memória utilizando a técnica proposta	71
Figura 60 –	Exemplo 5: MBU na memória utilizando a técnica proposta	72
Figura 61 –	Exemplo 6: MBU na memória utilizando a técnica proposta	73
Figura 62 –	Integridade dos Dados.	84
Figura 63 –	Fluxo geral para o tratamento de MBU	85

LISTA DE TABELAS

Tabela 1 –	Correções dos modelos de erros Janela 3X3	59
Tabela 2 –	Correções dos modelos de erros Janela 4X4	60
Tabela 3 –	Correções de erros para dois eventos: MBU + SBU	66

LISTA DE ABREVIATURAS E SIGLAS

ASIC	<i>Application Specific Integrated Circuit</i> - Circuito Integrado de Aplicação Específica
BC	Base dos Checadores
BP	Base Primos
C	Checadores
CC	Checador do Checador
CCE	Código Corretor de Erros
CI	Circuito Integrado
CLC	Código Linha Coluna
CPU	<i>Central Processor Unit</i>
D	Dados
DC	Dados Codificados
FIT	<i>Failures In Time</i>
FPGA	<i>Field Programmable Gate Array</i> - Arranjo de Portas Programáveis em Campo
HE	<i>Hard Error</i>
HPC	<i>Hamming</i> da Paridade da Coluna
LEO	<i>Low Earth Orbit</i>
MBU	<i>Multi Bit Upset</i>
MC	Mensagem Codificada
OBC	<i>On-Board Computer</i>
PC	Paridade da Coluna
PD	Paridade dos Dados
SBU	<i>Single Bit Upset</i>
SE	<i>Soft Error</i>
SEE	<i>Single Event Effect</i>
SEU	<i>Single Event Upsets</i>
SER	<i>Soft Error Rate</i>
SRAM	<i>Static Random Access Memmory</i>
TID	<i>Total Ionizing Dose</i>
TMR	<i>Triple Modular</i> - Redundância Modular Tripla
UFSM	Universidade Federal de Santa Maria

SUMÁRIO

1	INTRODUÇÃO	12
1.1	OBJETIVO GERAL	15
1.2	OBJETIVOS ESPECÍFICOS	15
2	CONTEXTUALIZAÇÃO	16
3	REVISÃO TÉCNICA	23
3.1	CONSEQUÊNCIAS DA INCIDÊNCIA DA RADIAÇÃO SOBRE A SRAM ...	23
3.2	CÓDIGOS CORRETORES DE ERRO	28
3.2.1	TÉCNICA DE HAMMING	29
3.2.2	TÉCNICA DE HAMMING ESTENDIDO	32
3.2.3	TÉCNICA DE PARIDADE	33
3.2.4	TÉCNICA DE REDUNDÂNCIA	33
3.2.5	TÉCNICA <i>INTERLEAVING</i>	34
4	TRABALHOS RELACIONADOS	35
4.1	UTILIZAÇÃO DO CCE	35
4.2	CÓDIGO <i>MATRIX</i>	36
4.3	CÓDIGO LINHA COLUNA	36
4.4	CLC UTILIZANDO HAM-E (8,13)	37
4.5	CÓDIGO DE APAGAMENTO	38
4.6	MATRIZ BIDIMENSIONAL	38
4.7	EXTENSÃO DO CÓDIGO HAMMING	39
5	CÓDIGO CORRETOR DE ERROS COM DUPLA VERIFICAÇÃO	40
5.1	CODIFICAÇÃO	41
5.1.1	Checadores (C)	42
5.1.2	Paridade da Coluna (PC)	43
5.1.3	Paridade dos Dados (PD)	43
5.1.4	Hamming da Paridade da Coluna (HPC)	44
5.1.5	Base do C (BC)	44
5.1.6	Checador do Checador (CC) e Base primos (BP)	45
5.1.7	Exemplo de Codificação	46
5.1.8	<i>Interleaving</i>	48
5.2	DECODIFICAÇÃO	50
5.2.1	Tratamento do C	51
5.2.2	Tratamento da PC	52
5.2.3	Verificação dos Dados	54
6	SIMULAÇÕES E ANÁLISE DOS DADOS	56
6.1	OUTRAS SIMULAÇÕES	62
6.2	CONSIDERAÇÕES SOBRE AS SIMULAÇÕES	66
6.3	SITUAÇÃO EXEMPLO	69
6.3.1	Exemplo 1	69
6.3.2	Exemplo 2	69
6.3.3	Exemplo 3	70
6.3.4	Exemplo 4	71
6.3.5	Exemplo 5	71
6.3.6	Exemplo 6	72

7	CONCLUSÃO	74
7.1	TRABALHOS FUTUROS	75
	REFERÊNCIAS	76
	APÊNDICES.....	83

1 INTRODUÇÃO

As mudanças que a tecnologia trouxe e traz ao nosso cotidiano são observadas facilmente, desde objetos de uso diário às mais incríveis invenções. Elas estão intrínsecas em qualquer proposta de desenvolvimento de novos aparatos tecnológicos. Presenciamos, nas últimas décadas, uma rápida evolução da eletrônica, dos computadores pessoais e equipamentos de modo geral. Vivenciamos hoje uma transformação que caminha em direção a um infinito de possibilidades.

A miniaturização dos mais diversos tipos de circuitos permitiram um rápido avanço na velocidade de processamento da informação, além disso, novos paradigmas na *Era da Informação* vem sendo concebidos. Se antigamente não era possível o desenvolvimento de satélites por parte de pequenas empresas e universidades, a atual realidade mostra-se totalmente diferenciada.

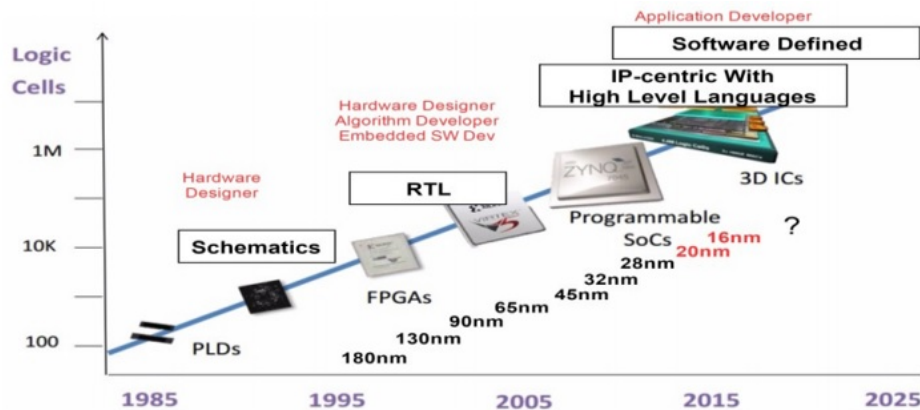
Os nanosatélites vêm sendo utilizados como ferramentas de ensino pelos mais diversos tipos de entidades, além disso, aplicações comerciais e científicas também são uma realidade. Entretanto vários empecilhos são enfrentados, não somente no desenvolvimento de sistemas espaciais mas também por qualquer tipo de aplicação que utiliza circuitos eletrônicos.

O desenvolvimento de tais tecnologias trazem consigo alguns obstáculos os quais necessitam de técnicas e métodos diferentemente dos utilizados outrora. Por exemplo, os sistemas de comunicações espaciais possuem maior probabilidade de ocorrer erro nas informações contidas nos circuitos eletrônicos (AGUIRRE, 2017), ou seja, o que décadas atrás era pertinente ao uso em sistemas espaciais atualmente demonstram-se obsoletos.

A miniaturização dos circuitos permite mais células no mesmo espaço físico, permitindo assim maiores funções lógicas no mesmo microchip. A evolução dos dispositivos torna mais denso seus componentes, na Figura 1 é exposto a relação entre a quantia de células lógicas e o tamanho do chip, é salientado que esta relação é específica para dispositivos de lógica programável.

A radiação cósmica, que atinge constantemente o planeta terra, traz consigo partículas de alta energia, as quais atingem os circuitos. Quando ocorre essa colisão existe a possibilidade de perturbação no fluxo correto de elétrons dentro do dispositivo ocasionando falha na lógica que está sendo processada. Dessa forma a confiabilidade do resultado fica comprometido, ainda mais em sistemas espaciais onde a radiação é mais intensa.

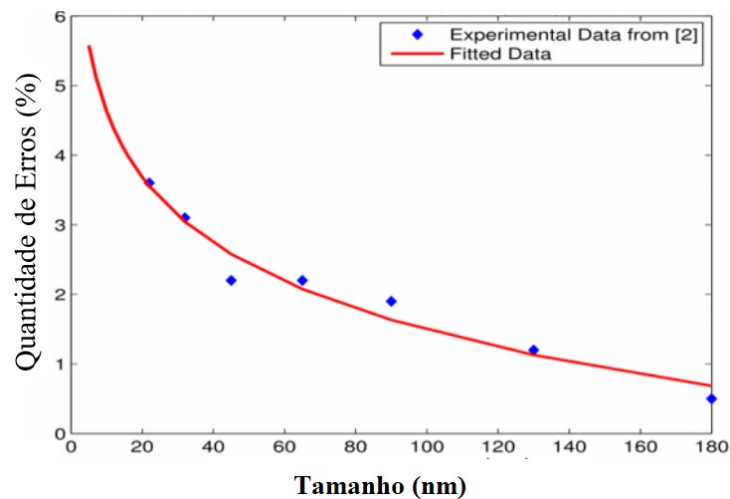
Figura 1 – Evolução da quantidade de células lógicas por espaço físico



Fonte: Adaptado de (SANTOS, 2017)

A pesquisa de IBE et al. (2010) verificou a relação de erros com a miniaturização dos circuitos e constatou que quanto mais denso for, maiores são as probabilidades de ocorrência de erro na memória. Na Figura 2, extraída do artigo NEALE; SACHDEV (2014) é possível verificar qual a quantidade de erros em relação ao tamanho da litografia.

Figura 2 – Relação entre o tamanho do circuito e a probabilidade de erro



Fonte: Adaptado de (NEALE; SACHDEV, 2014)

Para ter noção dos prejuízos que este tipo de evento pode causar à pesquisa de SANTOS (2017) traz algumas situações: Em 2008, um erro na leitura de um sensor fez com que a altitude de uma aeronave comercial fosse alterada bruscamente ferindo diversos passageiros.

Diante disso, os sistemas espaciais necessitam estar preparados para os erros que competem o ambiente. A tolerância à falhas tem por objetivo manter o sistema funcionando corretamente e perfazendo as tarefas atribuídas, lidando com os erros (SIDDIQUI; BAIG, 2011),

para aumentar a tolerância às falhas devido a inversão do bit causada pela radiação já existem técnicas, dentre elas temos os Códigos Corretores de Erros (CCE).

As perturbações que ocorrem podem ser detectadas e até corrigidas pelos CCE, dependendo como ocorrem. Basicamente, a inversão de um ou mais bits podem ocorrer na transmissão ou no armazenamento dentro da memória. Se na transmissão o erro for detectado, mas não houver como corrigi-lo, basta fazer a retransmissão. Entretanto se ao ler a informação da memória for detectado inconsistência existe a necessidade de correção, pois não há como extrair as informações novamente de sua origem como, por exemplo, de sensores cuja a atual leitura pode não ser suficiente para tomada de decisões.

Nessa perspectiva, o procedimento exposto neste estudo demonstra que uma segunda verificação é necessária para ampliar a confiabilidade do algoritmo. Uma dupla verificação dos bits faz-se necessária para atenuar os falsos positivos. Ou seja, os bits que validam os dados precisam ser validados previamente para suprimir parte dos falsos positivos. A dupla verificação é dada através de: Checagem dos bits de paridade principais e Checagem dos dados utilizando os bits de paridade que foram previamente filtrados.

Assim sendo, com o intuito de mitigar os possíveis erros oriundos dos agentes externos aos circuitos como, por exemplo, as radiações cósmicas presentes na órbita terrestre, a pesquisa desenvolvida é de um algoritmo com dupla verificação de Código Corretor de Erros, o qual detecta e realiza a correção dos erros de *Multi Bit Upset* (MBU) na *Static Random Access Memory* (SRAM). Salienta-se que o desenvolvimento do algoritmo bem como as simulações efetuadas foram desenvolvidas no MATLAB e conteve-se nas simulações de alto nível.

Na sequência serão abordados os assuntos pertinentes, os quais conduzirão o leitor desde o embasamento teórico até a compreensão das simulações efetuadas. Para tal, o texto foi dividido em capítulos. No segundo capítulo, **Contextualização** será abordados os principais aspectos que permeiam os satélites, bem como a relação entre os erros devido a radiação.

No terceiro capítulo, **Revisão Técnica**, são contempladas as pesquisas relacionadas aos erros na memória devido aos impactos de partículas altamente carregadas, tal como a origem e a importância dos CCE e as principais técnicas provindas desse método.

Em **Trabalhos Relacionados**, quarto capítulo, é demonstrada a aplicabilidade dos CCE, além disso, são expostas algumas das principais técnicas para correção dos erros de *upsets* na memória.

Na sequência, o quinto capítulo denominado **Proposta**, é demonstrado como ocorre a

codificação e decodificação da mesma maneira que os cálculos envolvidos e o fluxo necessário para manter a eficiência da técnica proposta.

Simulações e Análise dos Dados, disposto no capítulo 6, apresenta como foram efetuadas as simulações bem como a extração dos resultados e a comparação com outras técnicas verificadas ao longo do estudo.

Por fim, o último capítulo contempla a **Conclusão** expondo as considerações finais e as verificações para os trabalhos futuros que não foi possível desenvolver neste âmbito devido a escassez de tempo.

1.1 OBJETIVO GERAL

Desenvolvimento de um novo Código Corretor de Erros de *Multi Bit Upset* em Memória Estática de Acesso Aleatório com dupla verificação de erros diminuindo os falsos positivos e consequentemente aumentando o índice de confiabilidade .

1.2 OBJETIVOS ESPECÍFICOS

Com o propósito de realizar esta pesquisa bem como cumprir o objetivo geral da abordagem proposta os objetivos específicos para atingir os resultados finais são:

- Desenvolver um algoritmo que detecte e corrija erros de MBU em SRAM.
- Ampliar o índice de confiabilidade no algoritmo de correção de erros de MBU.
- Contribuir para a elaboração de novos métodos de bits de checagem de dados.
- Propor novos parâmetros para metodologias de codificação/decodificação.

2 CONTEXTUALIZAÇÃO

Os satélites são utilizados para os mais diversos fins desempenhando funções às quais impactam no nosso dia a dia. Dentre os mais diversos objetivos podemos destacar o uso de GPS, previsão do tempo e catástrofes naturais, monitoramento do clima, telecomunicações comerciais e militares dentre outras milhares de aplicações (PIOVESAN, 2017).

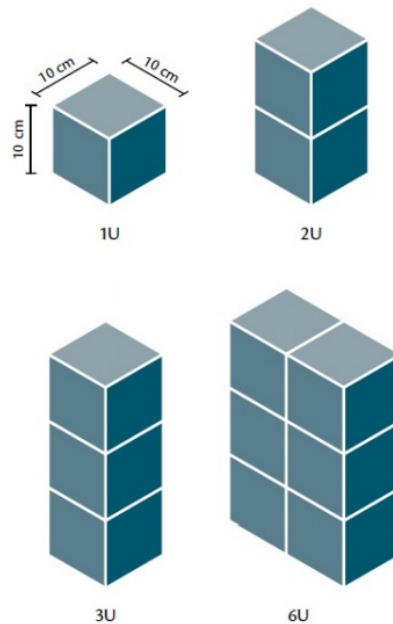
O início da era espacial com a utilização de satélites geoestacionários para a comunicação ocorreu na década de 60 (FORTES, 2016) e desde então o desenvolvimento para diversos fins vem sendo estudado. Com o avanço da tecnologia, eles foram divididos em categorias sendo considerada sua massa para divisão de categoria (PROCHNOW; DURÃO; SCHUCH, 2006), abaixo como são subdivididos:

- Satélites Grandes: possuem mais de uma tonelada
- Satélites Médios: entre 500Kg e 1000Kg
- Satélites Pequenos: menos de 500Kg, os quais possuem 5 subdivisões:
 - Minisatélites: entre 100Kg e 200Kg
 - Microsatélites: entre 10Kg e 100Kg
 - Nanosatélites: entre 1Kg e 10Kg
 - Picosatélites: entre 100g e 1Kg
 - Femtosatélites: menores do que 100g

Quanto menor o satélite menores são os custos, tendo em vista que se economiza no material para o desenvolvimento bem como com o lançamento até a órbita. Este custo reduzido vem possibilitando e estimulando pesquisas de pequenos satélites por parte de indústrias, organizações governamentais e instituições acadêmicas (ALVES, 2019), o que até então era inviável para grande parte dos atuais envolvidos, considerando o lançamento de satélites de médio porte.

Ainda em ALVES (2019) é possível verificar que o padrão Cubesat está em ascensão. Cubesat é categorizado como nanosatélite com massa próxima a 1,33 Kg e tamanho 10 cm de aresta. Esse tamanho é especificado como sendo 1U, existindo a possibilidade, ainda, de juntar mais de uma unidade formando configurações distintas como, por exemplo, 2U, 3U e 6U. Vejamos na Figura 3 as configurações expostas.

Figura 3 – Possíveis configurações para CubeSats



Fonte: Adaptado de (ALVES, 2019)

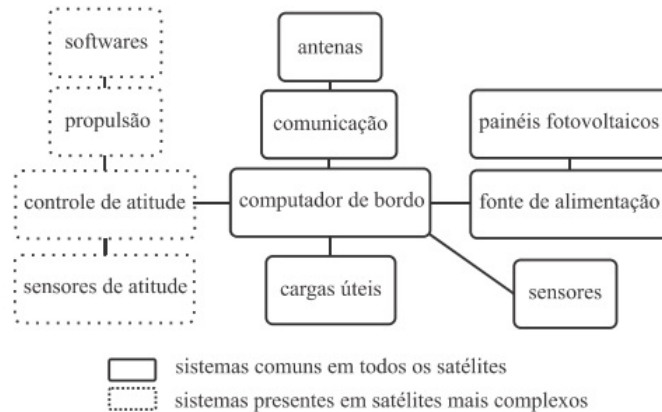
Esse padrão de satélite foi concebido em 1999 para auxiliar e incentivar as universidades, estudantes e pesquisadores no desenvolvimento na área da ciência espacial devido aos custos reduzidos. Os primeiros CubeSats foram enviados ao espaço no lançamento chamado Rocket, em 2003, e desde então esse padrão vem sendo popularizado (MOTA, 2017).

Os satélites são compostos por diversos dispositivos e com variados objetivos, de acordo com sua missão, ele pode ter mais ou menos sistemas, ou seja, a carga útil que compõe o satélite depende da sua missão. Na Figura 4 podemos observar os sistemas que compõem um satélite, a esquerda os sistemas que são utilizados em satélites mais complexos e a direita o que comumente está intrínseco nos sistemas espaciais.

Além disso, na Figura 4 é possível observar que o computador de bordo, ou OBC do inglês *On-Board Computer*, é a peça central que interliga todos os outros sistemas. Em JUNIOR (2019) destaca-se que é de sua responsabilidade: guiamento, navegação e controle do foguete, o qual tem como objetivo levar o satélite até o seu destino. Além disso, o computador de bordo também deve tomar as decisões que competem a missão espacial.

Ele, o OBC, é formado por vários componentes. Dentre os principais tem-se: processador, memória de boot, memória de trabalho, memória de armazenamento para os dados obtidos e barramento de dados (MOTA, 2017). Também é de sua responsabilidade supervisionar as

Figura 4 – Sistemas presentes nos satélites



Fonte: Adaptado de (PIOVESAN, 2017)

tarefas que os diferentes subsistemas executam e garantir o correto funcionamento do satélite (Osman; Mohamed, 2017).

Os microcontroladores, que estão presentes nos satélites, podem ser definidos como um pequeno computador presente em um único Circuito Integrado (CI), o qual contém um núcleo de processamento, memória e periféricos de entrada e saída (ARAÚJO, 2015).

Os CIs digitais podem ser basicamente divididos em duas categorias, sendo elas: Dispositivos de lógica fixa e lógica programável. O primeiro deles refere-se aos circuitos desenvolvidos para um único fim onde sua lógica é pré-definida no processo de fabricação, já os dispositivos de lógica programável possuem interconexões configuráveis podendo ser utilizados nas mais diversas aplicações. (MAXFIELD, 2004)

O *Application Specific Integrated Circuits* (ASIC), ou em português Circuito Integrado de Aplicação Específica, é um circuito desenvolvido para uma aplicação, de lógica fixa. O custo para o desenvolvimento e a fabricação de um chip desta categoria é alto, devido a necessidade de criar máscaras específicas para cada projeto (TAVARES, 2018; ARAGÃO, 1998). Além disso, qualquer alteração necessária no circuito após sua fabricação requer uma nova versão sendo necessária a fabricação completa do novo circuito.

Já, o *Field Programmable Gate Array* (FPGA) é programável após a sua fabricação, ou seja, consiste em uma *Central Process Unit* (CPU) com instruções pré-estabelecidas e blocos lógicos programáveis onde é possível obter qualquer comportamento lógico digital desde que se preserve a integridade do sinal (ARAGÃO, 1998) (MAXFIELD, 2004). Nesse sentido, vale ressaltar que os CIs integram sistemas embarcados que são utilizados em projetos de sistemas eletrônicos, tendo como principal função a execução de uma aplicação e/ou conjunto de

aplicações, relacionadas a um único sistema (DODD; MASSENGILL, 2003).

As aplicações espaciais expõem o hardware a condições de radioatividade, íons pesados se chocam fisicamente ao equipamento podendo causar perturbações no dispositivo em questão e alterar o fluxo correto do sistema. Além disso, *píons* e *múons* também podem causar esses efeitos. A interação entre o chip e as partículas radioativas de maior energia podem causar dano permanente nos chips, fazendo com que ocorra um incorreto funcionamento (AGUIRRE, 2017).

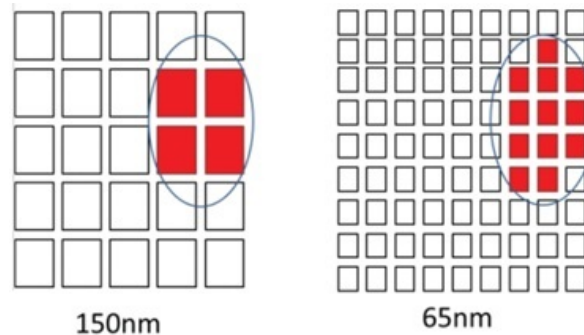
Os nanossatélites ficam na órbita terrestre, mais especificamente na região *Low Earth Orbit* (LEO) que compreende a altitude de 200 km a 1000 km (AGUIRRE, 2017). Projetos como o Delfi-C3 objetivaram o uso desse tipo de satélite a uma distância aproximada de 600 km de altitude e foram coletadas informações relevantes para estudos científicos como, por exemplo, o uso de filme fino de célula solar. Esse satélite começou suas operações em abril de 2008 (BOUWMEESTER; AALBERS; UBBELS, 2008).

Quando ocorre uma falha em um sistema espacial é crucial que o sistema se ajuste evitando a parada do sistema. CIs dedicados como os ASICs não podem alterar a forma como estão estruturados, entretanto os FPGAs podem se reprogramar ou serem reprogramados por ASICs, para manter o seu funcionamento com funções reduzidas e mantendo sua operacionalidade (BRAGGIO, 2014; DUMITRIU; KIRISCHIAN; KIRISCHIAN, 2014).

Os circuitos apresentados são passíveis de erros oriundos de agentes externos e/ou internos, esta pesquisa concentra-se nos agentes externos, mais especificamente, nos impactos de partículas energizadas. Ao longo dos anos foi verificado que a possibilidade de perturbação no circuito, causando a inversão dos bits em mais células adjacentes se ampliaram sobremaneira. A miniaturização dos circuitos aproxima as células, demonstrando que quanto menor a área de célula de memória maiores as chances de ocorrerem erros adjacentes. Conforme (AMAGASAKI et al., 2016) nos demonstra na Figura 5. A colisão de elétrons continua a mesma, entretanto, o impacto nas células de memória é alterado, comparação de uma mesma descarga elétrica em duas tecnologias de fabricação de memória: 150 nm e 65 nm.

Desde a década de 1960, quando começaram a ser observados esses efeitos em nível atmosférico, Wallmark e Marcus, já demonstravam interesse na verificação deste tipo de alteração. Com o avanço da miniaturização, esses eventos passaram a ser observados também em nível terrestre. Desde aquela época existia a preocupação com o aumento da complexidade que é inversamente proporcional ao tamanho do circuito (WALLMARK; MARCUS, 1962).

Figura 5 – Colisão de partículas em diferentes tamanhos de memórias



Fonte: Adaptado de (AMAGASAKI et al., 2016)

Isso ocorre porque elétrons, prótons e nêutrons chocam-se aos CIs em função das radiações cósmicas que provém de regiões entre galáxias, emitidas por erupções solares e/ou de outros planetas (STASSINOPOULOS; RAYMOND, 1988). Esses eventos ocorrem quando há choques de partículas de alta energia em qualquer tipo de circuito e são denominados *Single Event Effects* (SEE) ou *Single Event Upsets* (SEU). O efeito dessas descargas pode passar despercebido e/ou causar a interrupção completa do funcionamento do sistema. Isso ocorre em função do choque dessas partículas carregadas com regiões sensíveis no circuito (DODD; MASSENGILL, 2003; ALEXANDRESCU, 2011; CHORASIA; JASANI; SHAH, 2017).

Soft Error (SE) e *Hard Errors* (HE) podem ser consideradas subcategorias do SEE. O primeiro refere-se as falhas lógicas, devido alguma anomalia de descarga elétrica. Esse tipo de erro não causa danos físicos ao circuito, entretanto, pode alterar os bits que estão sendo processados e/ou armazenados no circuito e, conseqüentemente, alterar o comportamento esperado do mesmo. Podemos considerar este tipo de evento temporário (TORRES, 2013).

Já o segundo, o HE danifica fisicamente o circuito interrompendo seu funcionamento e/ou apresentando dados inválidos como verdadeiros, tornando o circuito inutilizável (TORRES, 2013). Em ambos os casos, a causa dos erros é a mesma. Os SE são passíveis, em alguns casos, de serem detectados e corrigidos antes do dado ser efetivamente alterado no processamento da aplicação. Na categoria HE, a consequência é a pior possível, pois implica o dano físico, o que, via de regra, desencadeará na interpretação equivocada dos dados e/ou a interrupção do serviço.

A área de efeito depende da tecnologia utilizada, memórias menores e conseqüentemente que armazenam mais bits no mesmo espaço físico tendem a possuir uma área relativamente maior influenciada pelo SEE do que memórias que ocupam menos espaço físico, ou seja,

memórias mais densas têm maiores probabilidades de múltiplos erros. Conforme demonstrado anteriormente na Figura 5, a pesquisa de REVIRIEGO; MAESTRO (2009) deixa claro que a relação entre múltiplos erros devido a algum tipo de SEE afetam bits adjacentes e que a distância entre os bits afetados depende exclusivamente da parte física fazendo com que quanto menor o espaço utilizado por um bit na memória torna-se mais agressivo o impacto da radiação.

Efeitos de SE, que ocorrem seguidamente em um dispositivo, podem causar um HE. Para além, é possível ainda que choques de partículas de alta energia que não apresentam SEE instantaneamente, acumulem cargas em regiões específicas do dispositivo. Esse acúmulo pode desencadear um *Total Ionizing Dose* (TID), conseqüentemente, a migração das vacâncias e o deslocamento atômico, resultam em um dano físico (AGUIRRE, 2017; MOTA, 2017).

A interpretação equivocada dos bits pode ocorrer devido aos SEE. As alterações em valores de aplicações resultam em dados não confiáveis para um fim específico. Um SEE não detectado pode acarretar o desvio da rota de um satélite causando prejuízos incalculáveis ou simplesmente não causarem nenhuma mudança notória.

Nesse sentido, o SEE pode afetar uma única célula de memória, ou seja, o bit armazenado naquele local físico específico será invertido causando o que é denominado de *Single-Bit Upset* (SBU). Entretanto, dependendo de como ocorre essa colisão das partículas carregadas, células adjacentes também podem ser afetadas, causando um *Multiple-Bit Upset* (MBU) (LIU et al., 2015). Assim, quando um único bit da memória é alterado denominamos SBU e quando dois ou mais bits são afetados chamamos de MBU.

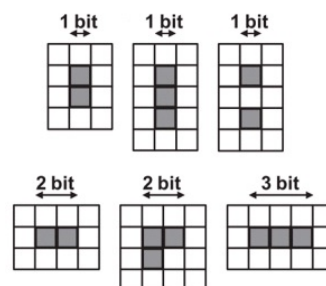
MBU não são previsíveis, mas seguem algumas regras. O impacto de partículas de alta energia pode alterar o estado de uma célula da memória. O MBU dar-se-á exclusivamente nas células adjacentes ao local principal afetado, sendo que uma única colisão de partículas não pode afetar células com uma grande distância entre si (MAVIS et al., 2008; IBE et al., 2010; RAO et al., 2014; XIAO et al., 2015).

É salientado que dois impactos podem afetar células distantes, sendo ele SBU ou MBU. Nessa perspectiva existe a necessidade de verificar os dados dentro da memória de tempos em tempos, dependendo da aplicação que está sendo averiguada, essa técnica chama-se *Memory scrubbing* (MARIANI; BOSCHI, 2005; STODDARD et al., 2017). Outra característica que devemos considerar é que a quantidade de bits afetado pelo MBU, para esta pesquisa, foi denominada como **nível de MBU**, ou seja, para 2 erros é dito MBU nível 2, 3 erros MBU nível 3 e assim sucessivamente, para 1 único erro este é chamado de SBU.

Todavia, sabendo que não é possível definir quando e como ocorrerão os SEE, ainda assim, é possível fazer uma previsão aproximada da quantidade de SEE em um determinado circuito. SILVA (2018) e MAHESHWARI; KOREN; BURLESON (2004) nomeiam como *Soft Error Rate* (SER) a unidade que mede os referidos erros e é dada em *Failures In Time* (FIT) no qual 1 FIT representa um SEE por bilhão de horas. Pode-se determinar que 1 FIT pode ocorrer a cada 114 anos, aproximadamente. Entretanto, quanto menor a área de armazenamento do bit em um circuito maiores as taxas de FIT. A título de exemplificação a faixa provável para os dispositivos eletrônicos é de 100 a 100.000 FITs. Para melhorar a precisão é possível efetuar cálculos, e até testes, aproximando o SER do CI em questão.

Além disso, o layout do circuito bem como a orientação espacial do microchip impactam na forma como ocorrem os erros. Em YOSHIMOTO et al. (2012) uma estimativa da taxa de FIT é simulada, é verificado que células NPN e PNP divergem na influência de MBU, a Figura 6 demonstra os padrões de erros averiguados.

Figura 6 – Padrão dos MBU verificados durante as simulações



Fonte: Adaptado de (YOSHIMOTO et al., 2012)

Para MONTEALEGRE et al. (2015), é possível verificar que aplicações com pouca ou nenhuma possibilidade de reparo, quando ocorrer um HE ou um SE, podem se utilizar de um FPGA para adaptá-lo em situações extremas como, por exemplo, em missões espaciais nas quais tudo estará à mercê de intensas mudanças de temperatura e radiação, além de forças cinéticas e outras situações imprevisíveis.

Diante do exposto é necessário verificar algumas questões de operacionalidade dos satélites, em específico os CubeSats que devido ao baixo custo de produção, muitas vezes, não possuem características de *Radiation Hardened* (MOTA, 2017). Os circuitos que possuem uma proteção à radiação nativa, devido ao processo de fabricação, são mais caros e consequentemente inviáveis muitas vezes.

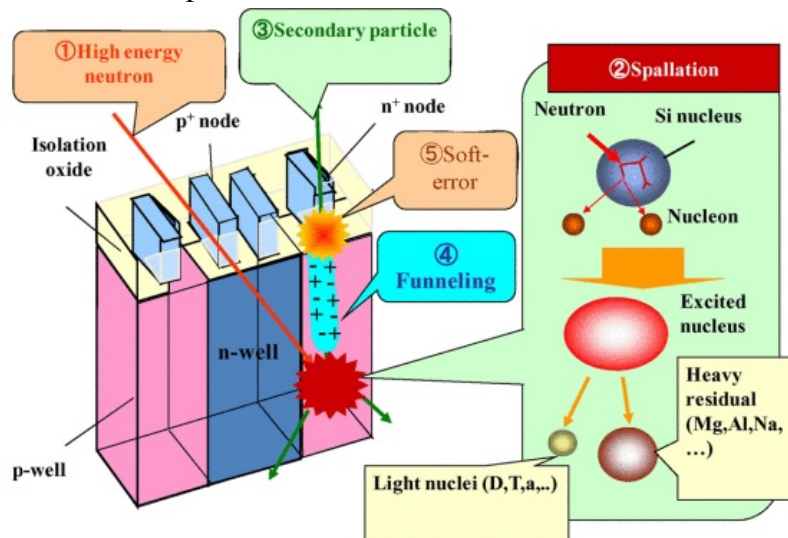
3 REVISÃO TÉCNICA

Cada circuito possui suas especificidades, conforme já demonstrado na Figura 5; o tamanho da célula impacta diretamente na alteração dos bits uma vez que a área de influência da radiação se mantém a mesma. Além disso, cada projeto tem suas especificações, porém todos necessitam de uma forma de detecção e correção dos erros. A verificação de erros oriundos dos SEE deve ser efetuada. Esse capítulo abordará, primeiramente, pesquisas em diferentes tecnologias e radiações e na sequência algumas técnicas para solucionar esses erros.

3.1 CONSEQUÊNCIAS DA INCIDÊNCIA DA RADIAÇÃO SOBRE A SRAM

Conforme já dito, partículas de energia altamente carregadas podem se chocar ao circuito, a Figura 7 nos demonstra a ilustração gráfica de como ocorre esse impacto, expondo um evento do tipo SE.

Figura 7 – Efeito de SEE do tipo SE no circuito

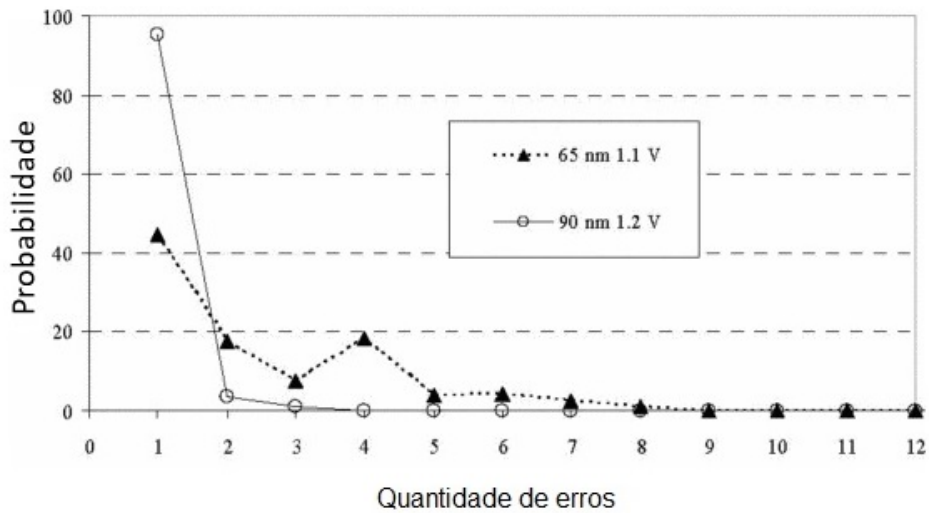


Fonte: Adaptado de IBE et al. (2010)

Em várias pesquisas utilizando tecnologias e padrões de testes diferenciados é possível observar que o aumento do número de erros provindos de um único SEE é significativamente menor a medida que mais células são afetadas. A Figura 8 expõe a comparação da quantidade de erros que podem ocorrer em duas tecnologias: 90 nm e 65 nm. A maior incidência de erro ocorre em um único bit alterado, ou seja, por um SBU. Para além, conforme o ângulo de incidência da radiação com o CI há uma variação do nível de MBU. Na sequência são expostos

algumas pesquisas que validam essa verificação.

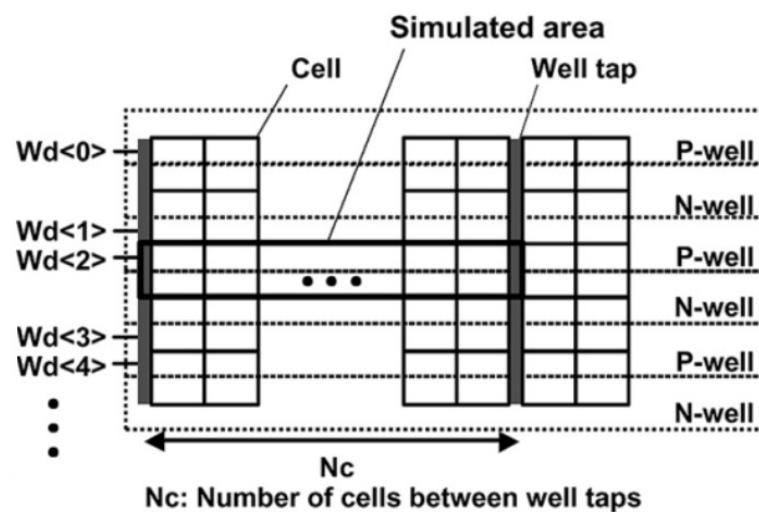
Figura 8 – Relação do número de células afetadas devido a tecnologia do chip



Fonte: Adaptado de Georgakos et al. (2007)

Em Osada et al. (2004) é testado a influência de raios cósmicos em SRAM. A tecnologia de 130 nm é averiguada. A cada colisão de raio cósmico a verificação de SBU/MBU é feita. Resultados demonstram que quase metade das inversões verificadas são SBU, além disso, a construção da SRAM pode diminuir a quantidade de bits afetado. Na figura 9 é demonstrado a relação do número de células entre as derivações dos poços N e P. Nesse experimento, a maior quantidade de erros é de 4 *upsets*.

Figura 9 – Relação do número de células afetadas devido a construção do chip



Fonte: Adaptado de Osada et al. (2004)

A pesquisa de TIPTON et al. (2008) fez experimentos em ângulos distintos de incidência da radiação em dispositivos de SRAM usando tecnologia de 65 nm, sendo elas: 2 no sentido horizontal, 2 no sentido vertical, e por fim 1 perpendicularmente. Perturbação múltipla induzida por íons pesados são averiguados utilizando a técnica de Monte Carlo para ambiente espacial. Cada uma das orientações espaciais demonstraram resultados diferentes durante os experimentos. Verificou-se que quando incididos perpendicularmente 90 % dos eventos são SBU, além disso, considerando 1 e 2 erros representam 98 % dos casos analisados.

Em KATO et al. (2019) um estudo com tecnologia de 20 nm analisou 4 formas distintas de afetar os bits em relação aos ângulos da radiação, concluiu-se que a quantidade de bits afetado depende diretamente do ângulo do circuito em relação a radiação aplicada, expondo majoritariamente os erros como SBU.

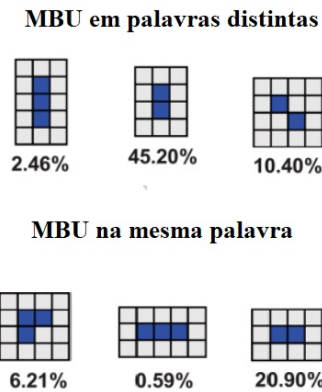
Em TONFAT et al. (2017) foram analisados 9 formas diferentes de incidência de radiação em um FPGA Artix-7 de 28 nm, no pior caso o valor foi de 83,66%, e no melhor caso foi 97,57% que são eventos de SBU. Considerando que quanto menos bits afetados melhor a situação, o experimento demonstrou que o pior caso somando as possibilidades de 1 e 2 bits com *upsets* se tem 95,83% das possibilidades, ou seja, todos os modelos de erros que contenham 1 e 2 bits representam mais de 95% das possibilidades, restando 4,17% possibilidades para 3 ou mais bits afetados pela incidência de radiação.

Na pesquisa de TIPTON et al. (2008) testes de aceleração de nêutrons em SRAM com tecnologia de 90 nm para calcular as taxas de FIT são efetuados em três graus diferenciados. Para todos os ângulos majoritariamente 1 erro é apontado, além disso, conforme altera o ângulo de incidência os tamanhos de erro são alterados.

Um FPGA de 28 nm do modelo Artix-7 foi verificado em TONFAT et al. (2016), os testes contiveram-se em três ângulos distintos, sendo eles: 0°, 45° e 60°, onde o impacto de partículas de íons pesados com baixa transferência de energia linear foi testado. Os resultados demonstraram que predominaram os erros do tipo SBU destacando o pior caso para quase 95% das validações.

A eficácia na detecção/tratamento de MBU aumenta se dividirmos a palavra. Em YOSHIMOTO et al. (2011) foi simulado induções de erros utilizando as ferramentas *iRoC TFIT* e *Synopsys 3-D TCAD*. A estrutura das palavras foram divididas com a intenção de melhorar as taxas de FIT. A Figura 10 expõe os padrões de erros averiguados na mesma palavra e em palavras distintas.

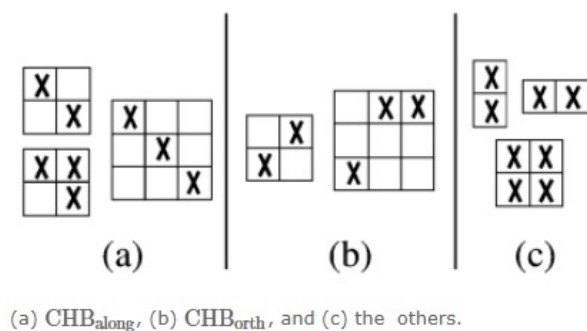
Figura 10 – MBU na mesma palavra X MBU em palavras distintas



Fonte: Adaptado de (YOSHIMOTO et al., 2011)

HARADA et al. (2012) efetuou testes em dois ângulos distintos na tecnologia de 65 nm para a incidência de radiação: 0° e 60° . Verificou-se que a maior incidência é de menores erros e conforme aumenta a quantidade de células afetadas a probabilidade diminui. Para além, o formato dos erros foi avaliado, o qual demonstrou que os *Upsets* ficam em uma distância máxima de uma matriz 3X3, a Figura 11 nos demonstra o formato dos erros verificados. **CHBalong** é descrito pelos autores como Padrão quadriculado ao longo do feixe de nêutrons e está representado em (a), **CHBorth** é descrito como Padrão quadriculado ortogonal ao feixe de nêutrons representado por (b) e os outros padrões averiguados estão representados em (c).

Figura 11 – Formato dos erros na tecnologia 65 nm



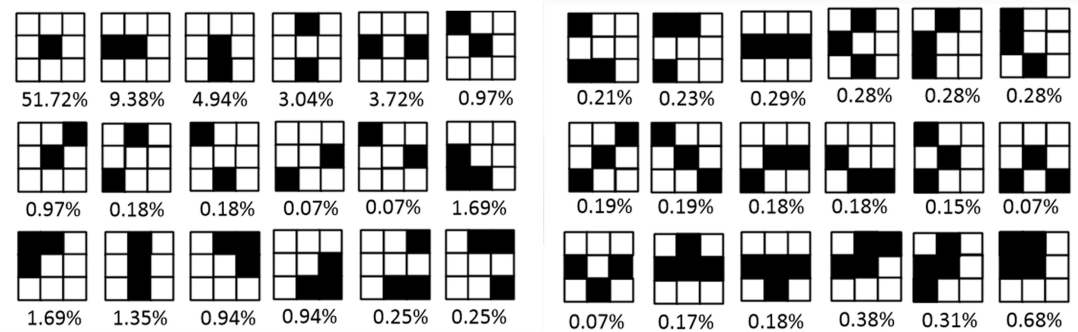
Fonte: Adaptado de (HARADA et al., 2012)

RAO et al. (2014) usou a tecnologia de 45nm. Um FPGA Xilinx Virtex-6 foi utilizado durante os experimentos, demonstrando que aproximadamente 50% dos erros apontam uma única inversão. Para 2 alterações são registrados quase 25% e para 3 registra-se próximo a 11%.

Vale ressaltar que nos testes verificados em RAO et al. (2014), nos quais a distribuição de energia utilizada para os experimentos se dá através do padrão *JEDEC89a*. A partir dos

resultados extraídos é possível verificar que erros de nível 4 MBU, para 45 nm, restringem-se a uma área de 3X3. A Figura 12 nos mostra os padrões de erros verificados e a probabilidade de ocorrência de cada configuração de célula de memória afetada.

Figura 12 – Padrão de erro para FPGA de 45 nm



Fonte: Adaptado de (RAO et al., 2014)

Ao observar a figura verifica-se em proporção maior que 50%, os erros de SE ocorrem em uma única célula de memória. Conforme aumenta o número de células afetadas diminui a probabilidade de ocorrer esse tipo de erro. O que chama a atenção, nesse caso, é que um SBU é facilmente corrigido (contemplando grandes partes dos casos) e os MBU, que possuem vários formatos, são mais difíceis de serem corrigidos.

Basicamente, todos os cálculos de verificação utilizam portas XOR, ou seja, quando ocorre um evento existe a possibilidade de auto anulação da verificação. Imagine que ao sofrer um SEE a informação altera o estado de um bit, consequentemente, o resultado do XOR também será alterado. Entretanto, se no mesmo conjunto outro bit sofrer outra alteração qualquer o resultado XOR será novamente invertido à posição inicial, anulando a detecção.

Para resolução dos erros existem técnicas que vêm sendo desenvolvidas há muito tempo as quais tentam garantir a integridade dos dados. Na tentativa de melhorar a integridade dos dados, por vezes, prejudica-se o armazenamento e o desempenho de aplicações.

Na sequência, uma breve exploração de alguns métodos. Vale ressaltar que conforme existe a necessidade de aumentar a confiabilidade da informação, a complexidade e o custo computacional também aumentam, tanto para processamento quanto para armazenamento da memória (MACHADO, 2016).

3.2 CÓDIGOS CORRETORES DE ERRO

Na década de 40, computadores eram máquinas de alto custo utilizadas para atividades de grande relevância, por exemplo, calcular a órbita de um planeta em volta do sol. O Laboratório Bell de Tecnologia possuía esse tipo de equipamento na época, e Richard W. Hamming tinha acesso a esses nos finais de semana. Especificamente, em 1947, Richard perdeu dois finais de semana por erros nos cartões perfurados e, nesse momento, ele proferiu as seguintes palavras: “Maldição, se as máquinas podem detectar um erro, porque não podemos localizar a posição do erro e corrigi-lo” (MILIES, 2009).

A partir desse ponto na história, Hamming começou a desenvolver um método de detecção e correção de erros, chamado Codificação Hamming, o qual consegue detectar e corrigir um erro. Atualmente, Códigos Corretores de Erro (CCE) são utilizados nas mais diversas áreas do conhecimento como a Matemática, Computação, Engenharias, entre outras.

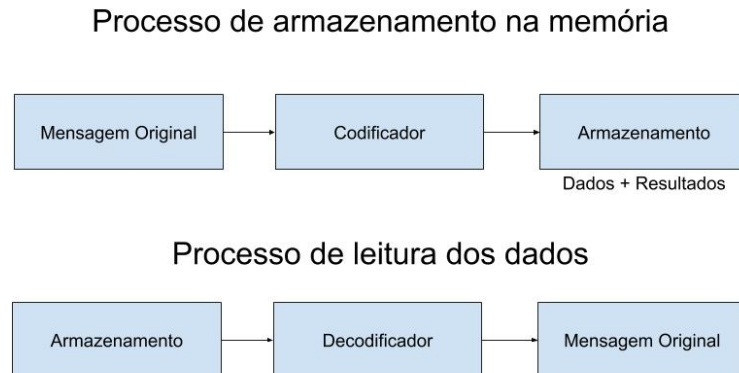
Nessa perspectiva, os CCE's foram desenvolvidos com o intuito de preservar a integridade dos dados (MAHESHWARI; KOREN; BURLESON, 2004) e evitar que os mesmos sejam corrompidos, desde que o erro não seja físico, ou seja, um SE pode ser tratado pelo CCE enquanto o HE não.

Um exemplo: para armazenar 1 e 2 com a utilização de técnicas podemos partir de uma simples equação em que $X = 1 + 2$. É evidente que X equivale a 3, dessa forma não são armazenados apenas os dados, mas sim resultados de contas matemáticas, possibilitando a comparação do resultado e dos dados propriamente ditos. Se um SE ocorrer e inverter um elemento da equação, transformando o 1 em 4, isso implicará a ocorrência de um erro. De tal forma é possível analisar e recalculer a equação $3 = 4 + 2$ e verificar que a mesma está incorreta e assim detectar o erro.

A base do esquema de correção em memórias pode ser verificado na Figura 13 onde o processamento da mensagem para armazenar na memória é intitulado de "Codificador" e a leitura/detecção do erro pode ser denominado "Decodificador".

Para um CCE ser considerado eficiente, e conseqüentemente garantir a integridade da informação, ele necessita atingir 100% de correção, no seu raio de atuação, das inversões causadas nos bits devido ao MBU, ou seja, para considerarmos que o algoritmo atinge a correta correção é imprescindível que ele consiga corrigir todos os formatos e número de erros dentro do raio que ele atua. Na literatura existem algoritmos com correções parciais após um número

Figura 13 – Processo de armazenamento



Fonte: AUTOR

acentuado de erros, esses algoritmos serão tratados mais adiante.

Vale ressaltar que as técnicas de verificação servem para armazenamento ou transporte da informação. O armazenamento é um dos fatores mais complexos devido ao fato de que quando transportamos a mensagem de A para B, e os erros ocorridos pelo meio de transmissão não permitirem a recuperação dela, existe a possibilidade de refazer a transmissão.

Diferentemente da transmissão, o armazenamento é mais delicado, pois não é possível recuperar a informação de outro lugar, tem-se apenas os dados e seus respectivos validadores. Se o erro for muito acentuado existe a possibilidade de detecção de *upset*, entretanto determinar quais bits foram alterados pode ser impossível de definir, dependendo do grau de MBU que a memória foi exposta.

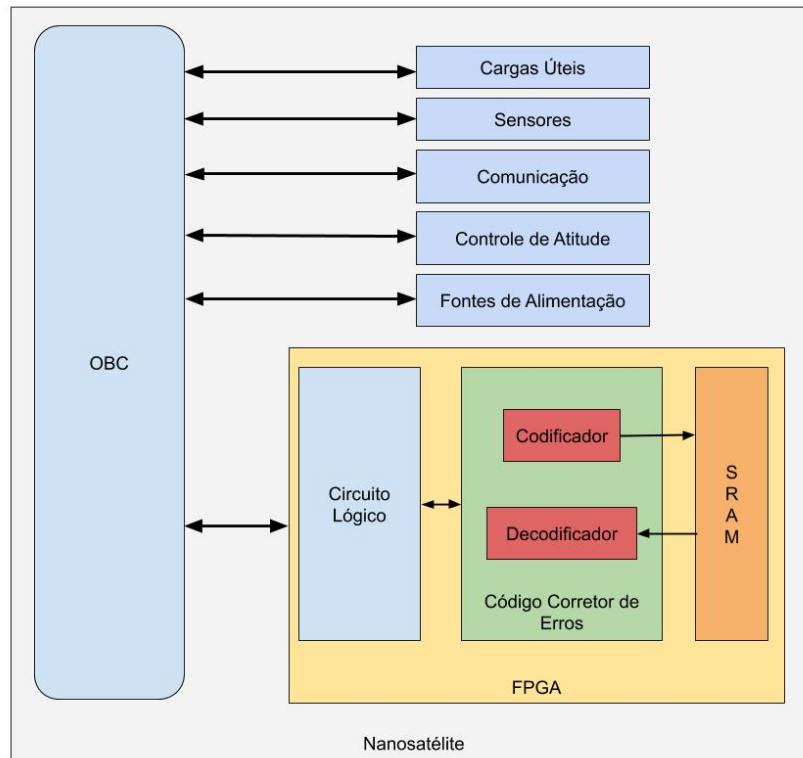
Na Figura 14 é representado o esquema de como funciona o CCE no ambiente espacial. O CCE fica entre o OBC e a memória fazendo a codificação e a decodificação a cada gravação e leitura das informações. O CI que faz esse intermédio entre o OBC e a memória pode ser um FPGA, por exemplo.

Existem vários métodos de recuperação da informação devido a SEE, a seguir serão expostas as principais técnicas.

3.2.1 TÉCNICA DE HAMMING

É possível utilizar 2 métodos matemáticos para encontrar os bits de verificação de Hamming. Sendo capaz de detectar e corrigir um erro dentro de uma palavra de dados. É caracterizado por $\text{Ham}(M,N)$, onde M representa a quantia de bits de dados úteis e N a quantia total de

Figura 14 – Funcionamento de um CCE



Fonte: AUTOR

bits. N pode ainda ser definido como $M + K$, neste caso K representa a quantidade de bits Hamming (SILVA, 2018).

Como primeira forma os bits de Hamming são introduzidos de forma intercalada na mensagem. A primeira posição é um dos bits de Hamming, o próximo bit de checagem será alocado no dobro do valor dessa posição e assim sucessivamente, ou seja, as posições dos bits de hamming são alocados nos exponenciais da base 2. Em uma representação matemática é possível dispor a mensagem codificada, optou-se por exemplificar o Ham (4,7) o que possibilita uma compreensão mais fácil, da seguinte forma: H1 H2 D1 H3 D2 D3 D4, sendo H um bit derivado da codificação de Hamming e D o dado pertencente a uma mensagem qualquer (S. CASTRO et al., 2016).

As seguintes posições são alocadas para os bits de checagem: 1^o , 2^o e 4^o e os dados são colocados de forma sequencial nos locais disponíveis. O H1 valida ele próprio e alterna 1 bit, valida o próximo e alterna 1 bit e assim sucessivamente. O H2 valida de dois em dois bits e o H3 válida de 4 em 4 bits, ou seja, a posição que o Hamming se encontra determinará quantos bits ele valida e intercala. Em outro caso de checagem, com mais bits, a lógica continua a mesma,

porém, para a validação do H4, que estaria na oitava posição, serão calculados os próximos 8 bits, alternando de 8 em 8.

A validação ocorre através de portas XOR, ou seja, para o H1 teremos: $H1 \oplus D1 \oplus D2 \oplus D4$. Como o H1 ainda não possui valor específico, é atribuído a ele o valor 0 para esse cálculo (S. CASTRO et al., 2016). Quando houver alteração em alguns dos dados verificados por H1, o mesmo também será alterado durante o recálculo. Todos os bits de dados possuem bits de checagem em comum podendo, assim, detectar o bit que foi invertido.

Uma outra forma de realizar o cálculo de Hamming acontece através da matriz geradora G e a matriz de paridade H (MOREIRA; FARRELL, 2006; NEUBAUS; FREUDENBERGES; KUHN, 2007). A matriz G é dada pela seguinte equação $G = [(I2^K - k - 1Q)]$, onde I é uma matriz Identidade de ordem $(2^K - k - 1)$ e Q é uma matriz formada por $(2^K - k - 1)$ linhas com os vetores de peso. Para Ham (4,7) temos $G = [I4Q]$, a Figura 15 expõe a estrutura inicial.

Figura 15 – Matriz Identidade (I)

1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1

Fonte: Adaptado de (SILVA, 2018)

Os vetores de peso, utilizados na matriz Q, possíveis para a distância 3 são 2^3 , consequentemente pode-se utilizar: 000, 001, 010, 011, 100, 101, 110 e 111. Os quatro vetores escolhidos para exemplificação de Q, foram: 011, 101, 110 e 111. Assim, a matriz geradora assumirá a forma da Figura 16 (SILVA, 2018):

Figura 16 – Matriz Geradora (G)

1	0	0	0	0	1	1
0	1	0	0	1	0	1
0	0	1	0	1	1	0
0	0	0	1	1	1	1

Fonte: Adaptado de (SILVA, 2018)

A mensagem codificada (MC) é obtida através do produto vetorial entre a mensagem (de quatro bits) e a matriz G. Para exemplificar a mensagem $M = 1000$, logo, $MC = 1000011$.

A verificação de erros por esse método é dada através da matriz de paridade H gerada através da transposta de Q, juntamente com uma matriz identidade de ordem K, Utilizando os dados, citados anteriormente, temos a Q^T da seguinte maneira conforme Figura 17 :

Figura 17 – Transposta de Q

0	1	1	1
1	0	1	1
1	1	0	1

Fonte: Adaptado de (SILVA, 2018)

Na sequência, ao lado da Figura 17 há uma matriz identidade de ordem 3, representada na Figura 18.

Figura 18 – Matriz de paridade H

0	1	1	1	1	0	0
1	0	1	1	0	1	0
1	1	0	1	0	0	1

Fonte: Adaptado de (SILVA, 2018)

A Validação de erros pode ser averiguada através do produto da matriz H com a transposta da MC, $V = [H * MC^T]$. Se o resultado for um vetor V nulo não foi detectado erro na mensagem. A mensagem original, nesse caso, são os 4 primeiros bits da MC. Entretanto, se V retornar algum valor não nulo deve ser aplicado um XOR entre a mensagem incorreta MC e o vetor de correção designado conforme será demonstrado na Figura 19.

Figura 19 – Vetor de correção designado para correção de erro

V	Vetor de Correção
[0 0 0]	[0 0 0 0 0 0]
[0 0 1]	[0 0 0 0 0 1]
[0 1 0]	[0 0 0 0 1 0]
[0 1 1]	[1 0 0 0 0 0]
[1 0 0]	[0 0 0 1 0 0]
[1 0 1]	[0 1 0 0 0 0]
[1 1 0]	[0 0 1 0 0 0]
[1 1 1]	[0 0 1 0 0 0]

Fonte: Adaptado de (SILVA, 2018)

3.2.2 TÉCNICA DE HAMMING ESTENDIDO

A utilização de Hamming estendido proporciona a possibilidade de detecção de dois erros dentro da mesma mensagem, entretanto, os mesmos não poderão ser corrigidos, sendo

que um único erro será detectado e corrigido. A técnica é definida da seguinte forma: $Ham = E(M, N + 1)$. Os cálculos do método são igualmente efetuados, conforme a técnica anterior, com a adição de 1 bit de paridade ao fim da mensagem (ESMAEELI et al., 2011). A técnica de paridade é utilizada para diversos fins e será explorada na sequência.

3.2.3 TÉCNICA DE PARIDADE

De acordo com RAVI; PARGUNARAJAN (2017), a técnica de paridade consiste na adição de mais um bit ao final de uma mensagem. Nessa mensagem é calculado o XOR entre todos os elementos. A título de exemplo:

Uma mensagem $X = 101001$, realizados os cálculos de XOR entre todos os elementos verifica-se que o resultado é 1. Então, a mensagem codificada resulta em 1010011. Na decodificação ignora-se o último elemento e é feito o cálculo, identificando se este é equivalente com o último dígito da mensagem codificada ou não.

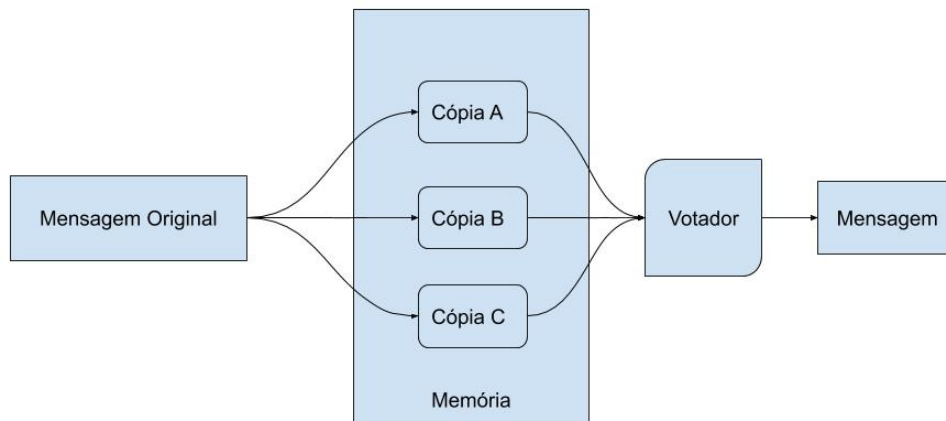
Assim sendo, é possível detectar erros ímpares, mas não é possível determinar qual a posição deste erro dentro de qualquer parte de uma estrutura de dado. Esse cálculo sozinho não traz grandes benefícios, entretanto, se combinado com outras técnicas, potencializa a eficácia na detecção de erros. As técnicas de Hamming, Hamming Estendido e Paridade são provenientes de cálculos e podem ser utilizadas de forma interligada para aumentar a eficácia dos CCE.

3.2.4 TÉCNICA DE REDUNDÂNCIA

A técnica de *Triple Modular Redundancy* (TMR), ou em português Redundância Modular Tripla, consiste em armazenar o mesmo conjunto de bits 3 vezes dentro da memória, possibilitando um sistema de votação (LYONS; VANDERKULK, 1962; SIDDIQUI; BAIG, 2011). Uma mensagem M é armazenada em três locais distintos dentro da memória A, B e C, ao ler M os erros são verificados por um votador que seleciona como verdadeiro o valor majoritário. Já que são três votantes, nunca haverá empate. A Figura 20 demonstra um esquema simbólico do sistema.

Na situação hipotética um bit da mensagem salvo no local A é alterado devido alguma anomalia no sistema, este por sua vez será votado juntamente com B e C, considerando que B e C estão corretos a mensagem estará preservada. Esse método aumenta a necessidade de espaço de armazenamento em 200%, pois duas cópias a mais são salvas na memória o que por vezes não é possível devido a arquitetura do sistema.

Figura 20 – Representação TMR na memória



Fonte: AUTOR

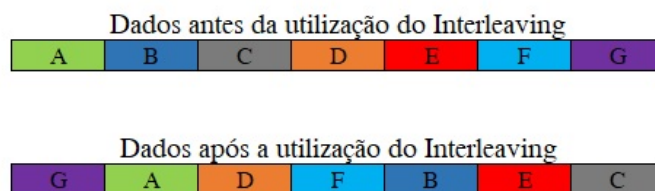
Quando possível é necessário garantir a integridade da informação com essa técnica, ainda, é possível utilizar esse método com outras técnicas, sendo para cada uma das mensagens salvas (A, B e C) uma técnica distinta.

3.2.5 TÉCNICA *INTERLEAVING*

A técnica de *Interleaving* consiste numa reorganização dos dados dentro da memória. Essa garante que quando ocorrer um MBU, a área atingida será um ponto específico e os dados nela contidos serão de fragmentos espalhados ou até mesmo fragmentado em várias mensagens. Esses serão tratados como vários SBU's. Erros de SBU's são mais fáceis de serem corrigidos porque podem ser tratados com Ham-E (RADAELLI et al., 2005).

Ao utilizar *Interleaving*, o algoritmo torna-se mais complexo devido a necessidade de alternar os locais dos bits na fase de armazenamento e conseqüentemente reorganizá-los quando solicitada leitura. A Figura 21 representa graficamente como os dados poderão ser dispostos utilizando essa técnica.

Figura 21 – Representação dos Dados antes e depois de aplicar *Interleaving*



Fonte: AUTOR

4 TRABALHOS RELACIONADOS

Na sequencia, são descritas algumas técnicas que utilizam procedimentos já referenciados no capítulo anterior, demonstrando melhor eficácia no tratamento de MBU's em memórias. Primeiramente será abordado a utilização dos CCE e logo após será contemplado algumas técnicas já existentes.

4.1 UTILIZAÇÃO DO CCE

A utilização de CI no espaço requer alguns cuidados em relação a incidência da radiação, dentre eles possuímos a possibilidade de blindagem eletromagnética. Existem algumas categorias de blindagens (TURER; AYDIN, 2015): Campo magnético DC, campo magnético de baixa frequência, campo elétrico de baixa frequência e ondas planas. Essas blindagens requerem o uso de alguns materiais que alteram o peso e o tamanho do satélite, o que por vezes pode ser inviável. Imagine revestir uma CubeSat com chumbo, conseqüentemente o peso sera drasticamente alterado não sendo possível mantê-lo em 1,33 kg. Além disso, um tipo de material para uma determinada blindagem pode ser ineficiente para a outra categoria, isso trás a necessidade de mais de uma camada de blindagem, inviabilizando o seu lançamento.

Os CCE são utilizados para os mais diversos fins e para tal a sua aplicabilidade é ampla. Com foco para o cunho espacial, podemos verificar a sua importância em pesquisas como OLIVEIRA MIRANDA (2016) cuja tolerância a falhas para a determinação de atitude de um nanossatélite é muito relevante. Essa atitude é o que define sua orientação espacial onde colhe-se dados de sensores e a partir dos resultados o algoritmo toma as decisões cabíveis. Diante disso, um SEE pode alterar sua percepção espacial e conseqüentemente fazendo-o tomar as decisões erradas.

Conforme já abordado, o uso do FPGA pode ser um ótimo aliado a problemas provindos de SEE no espaço, entretanto pode trazer alguns problemas, pois conforme SANTOS (2017) os FPGAs podem possuir matrizes densas de células o que o torna mais sensível a erros conforme já exposto na Figura 5.

Na pesquisa MONTEALEGRE et al. (2015) é possível verificar que uma das qualidades do FPGA é a reconfiguração de partes, mantendo uma fração das funcionalidades operacionais, enquanto outra é ajustada devido à, por exemplo, um HE. Além disso, é possível que a recon-

figuração do FPGA seja feita por ele mesmo, um ASIC ou ainda pela estação terrestre. De fato, técnicas precisam ser estabelecidas para que se mantenha a integridade das informações contidas na memória dos satélites. Na sequência algumas técnicas são abordadas.

4.2 CÓDIGO *MATRIX*

O código *Matrix* (ARGYRIDES; ZARANDI; PRADHAN, 2007) combina a técnica de Hamming e de Paridade criando uma matriz disposta com os dados, cada linha é verificada pela técnica de Hamming, permitindo a localização e correção de 1 bit errôneo. Já, a coluna produz uma paridade simples informando quando há números ímpares de erros, tendo em vista que a paridade simples usa XOR como base e que erros pares se auto anulam.

A Figura 22 nos demonstra como os dados são dispostos, onde D representa os dados, H representa os bits, utilizando a técnica de Hamming e P representa a paridade simples. Nessa pesquisa foi exposto que para uma palavra de 32 bits de dados a correção de 100% dos erros ocorre para até 2 inversões, sendo que a partir da terceira inversão existe uma parcialidade dos casos na correção efetiva.

Figura 22 – Estrutura do código Matrix

D1	D2	D3	D4	H1	H2	H3
D5	D6	D7	D8	H4	H5	H6
D9	D10	D11	D12	H7	H8	H9
D13	D14	D15	D16	H10	H11	H12
P1	P2	P3	P4			

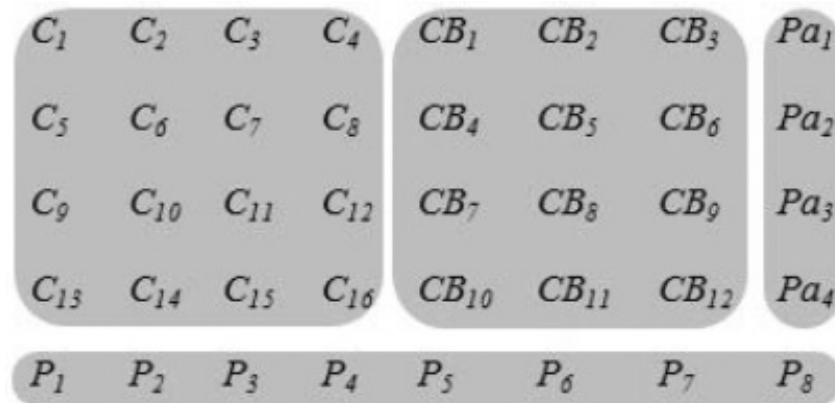
Fonte: Adaptado de (ARGYRIDES; ZARANDI; PRADHAN, 2007)

4.3 CÓDIGO LINHA COLUNA

O código linha coluna (CLC), desenvolvido por S. CASTRO et al. (2016), apresenta uma nova técnica de detecção de erros utilizando matrizes binárias na confecção da palavra codificada. Esse método permite a utilização de 16 bits de dados resultando na codificação de 40 bits totais, considerando dados e validadores. Conforme Figura 23, os dados são dispostos em 4 partes diferente: C representa os bits da mensagem, CB são os bits utilizando Hamming, Pa é a paridade da linha e P representa a paridade da coluna.

O autor utiliza a técnica de Ham-E(4,7+1) e dispõe os dados, de forma separada, dentro da matriz nas primeiras 4 linhas, colocando os dados a esquerda, os bits de Hamming no meio

Figura 23 – Estrutura do CLC(16,40)



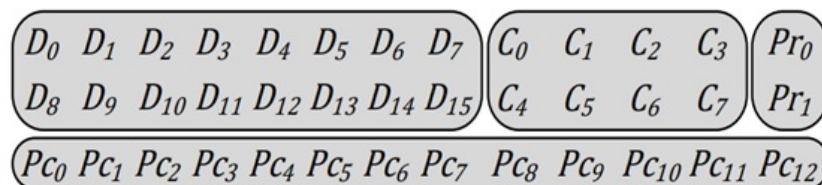
Fonte: Adaptado de (S. CASTRO et al., 2016)

e o bit de paridade ao final, utilizando a última linha como paridade de cada coluna. Testes com a ferramenta *Synopsis Business VCS* demonstram que essa técnica detecta assertivamente a totalidade dos dados que obtiveram 4 bits invertidos por erros de MBU. Entretanto, a correção dessa totalidade de garantia da integridade da informação fica limitada quando ocorrem 2 erros.

4.4 CLC UTILIZANDO HAM-E (8,13)

Testes utilizando uma disposição de dados diferente dentro de uma matriz binária foram verificados por SILVA (2018). Nesse contexto, o autor validou a utilização de menor espaço em memória, dispondo de 16 bits de dados mais 23 bits de checagem, resultando em um total de 39 bits. A Figura 24 mostra esta reorganização.

Figura 24 – Estrutura do CLC(16,39)



Fonte: Adaptado de (SILVA, 2018)

Igual ao procedimento anterior é observado na Figura 24, sendo D os dados, C os bits de Hamming, Pr a paridade da linha e Pc a paridade da coluna. O C e o Pr são calculados com o Ham-E, apenas são separados dentro da matriz os bits de dados dos de checagem. A validação deu-se através do MATLAB, mantendo uma detecção assertiva da totalidade dos erros de MBU

para 4 erros, aumentando a eficácia da correção de 2 para 3 erros de MBU.

4.5 CÓDIGO DE APAGAMENTO

Para RAO et al. (2014) a correção dos erros não é uma preocupação. A atrelação dos bits dar-se-á por toda a estrutura do FPGA, a detecção ocorre através de bits de validação e então o bloco inteiro é descartado. Para recuperação da informação são utilizados os códigos de apagamento que recriam o bloco original usando os validadores presentes nos outros blocos de dados dentro do FPGA. Entretanto, efeitos de MBU em mais de um bloco de memória podem estabelecer mensagens corrompidas, impedindo sua reconstrução e consequentemente a perda de todas as informações.

4.6 MATRIZ BIDIMENSIONAL

Em Zhang et al. (2019) são verificados 32 bits de dados totalizando 64 bits a serem armazenados. Na figura 25 é demonstrado como os bits são armazenados dentro da matriz para serem calculados. A e B são destinados aos dados, Da e Db são as verificações na horizontal e por fim o Ca e Cb são as verificações na horizontal.

Este método atingiu, durante as simulações, a correção de 100 % dos erros para dois *upsets*. Para maior quantidade de inversões existe uma parcialidade na correção dos erros, obtendo aproximadamente 90 % de correção para 3 erros.

Figura 25 – Matriz Bidimensional

data bits								check bits			
a0	a1	a2	a3	a4	a5	a6	a7	Da0	Da1	Da2	Da3
a8	a9	a10	a11	a12	a13	a14	a15	Da4	Da5	Da6	Da7
b0	b1	b2	b3	b4	b5	b6	b7	Db0	Db1	Db2	Db3
b8	b9	b10	b11	b12	b13	b14	b15	Db4	Db5	Db6	Db7

Ca0	Ca1	Ca2	Ca3	Ca4	Ca5	Ca6	Ca7
Cb0	Cb1	Cb2	Cb3	Cb4	Cb5	Cb6	Cb7

} parity bits

Fonte: Adaptado de (Zhang et al., 2019)

4.7 EXTENSÃO DO CÓDIGO HAMMING

Em Sanchez-Macian; Reviriego; Maestro (2012) é verificado a possibilidade de maximizar a detecção e correção dos erros em conjuntos de bits utilizando a técnica de Hamming com a correta distribuição dos bits dentro da memória. Na Figura 26 é demonstrada a distribuição dos bits para uma palavra de 8 bits utilizando 4 bits adicionais possibilitando a detecção de dois bits errados. Na Figura 27 possui uma palavra de 8 bits utilizando 5 bits adicionais permite a detecção de três erros.

Figura 26 – Distribuição dos bits dentro da matriz usando 4 bits de paridade

Bit placement												Detection	
1	2	3	4	5	6	7	8	9	10	11	12	1/11	9%
1	12	2	3	6	8	7	9	4	10	5	11	9/11	82%

Fonte: Adaptado de (Sanchez-Macian; Reviriego; Maestro, 2012)

Figura 27 – Distribuição dos bits dentro da matriz usando 5 bits de paridade

Bit placement													Detection	
1	2	3	4	5	6	7	8	9	10	11	12	p	1/11	9%
6	8	1	7	11	3	5	9	2	4	p	10	12	9/11	82%

Fonte: Adaptado de (Sanchez-Macian; Reviriego; Maestro, 2012)

5 CÓDIGO CORRETOR DE ERROS COM DUPLA VERIFICAÇÃO

Este capítulo trata da apresentação de um novo algoritmo de Código Corretor de Erros com dupla verificação para SRAM, primeiramente será abordado como é desenvolvido a codificação dos bits, após as etapas da decodificação e por fim exemplos de tratamento de erros.

Uma característica que foi notada durante as pesquisas foi a necessidade, quase que exclusivamente, de validar os dados. Como choques de partículas de alta energia podem causar perturbações em quaisquer bits salvos na memória e que os bits de checagem estão salvos juntamente com os dados, existe a possibilidade de inversões nos bits checadore, ocasionando falsos positivos. Esse novo método prevê a criação de bits de checagem para os dados e um segundo nível de verificação que valida os bits de checagem. O tratamento de um falso positivo como um erro pode desencadear em um dado verdadeiro considerado errado e o recalculo corromper o dado.

Dessa forma, é possível que os checadore sejam previamente validados antes de verificar a integridade dos dados garantindo que falsos positivos tenham sido previamente tratados. Além disso, é possível observar a técnica de Hamming, onde quanto menor a quantidade de bits de dados maior a carga dos bits de paridade, relativamente. Dessa forma as verificações foram para 64 bits de dados, onde pode ser possível a utilização de 4 conjuntos de 16 bits ou 2 conjuntos de 32 bits, por exemplo.

Diante disso, a codificação foi desenvolvida para 64 bits de dados, sendo que ao gerar a informação codificada resultará em 144 bits ao total, tendo um acréscimo de 80 bits, ou seja, para a codificação dos 64 bits de dados é necessário um acréscimo de 125% de bits de paridade.

Anteriormente já dito, cálculos matemáticos devem ser realizados para armazenarmos os dados dentro da memória, juntamente com seus respectivos bits de checagem. A disposição dos dados dentro de uma matriz permite checagens estratégicas, fazendo com que o mesmo bit da *string* de dados seja verificado por diversos validadores, além disso, a verificação computacional é mais simples, o que economiza recursos para o processamento da decodificação. Na Figura 28 é observada a matriz de dados já preenchida com os bits de paridade, cada conjunto expresso na imagem será explanado a seguir.

Conforme já dito, o MBU não pode ser previsto, muito menos antecipar seu efeito dentro de um CI, pois ele pode ocorrer em qualquer parte do circuito. Entretanto, é possível adaptar o codificador e decodificador de um CCE baseado em algumas regras que o choque de partículas

Figura 28 – Matriz binária

D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	C1	C2	C3	C4	BC1	BP3	PD1
D12	D13	D14	D15	D16	D17	D18	D19	D20	D21	D22	C5	C6	C7	C8	BC2	BP4	PD2
D23	D24	D25	D26	D27	D28	D29	D30	D31	D32	D33	C9	C10	C11	C12	BC3	BP5	PD3
D34	D35	D36	D37	D38	D39	D40	D41	D42	D43	D44	C13	C14	C15	C16	BC4	BP6	PD4
D45	D46	D47	D48	D49	D50	D51	D52	D53	D54	D55	C17	C18	C19	C20	BC4	BP7	PD5
D56	D57	D58	D59	D60	D61	D62	D63	D64	BP1	BP2	C21	C22	C23	C24	BC6	BP8	PD6
PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	PC10	PC11	PC12	PC13	PC14	PC15	BP10	BP9	PD7
HPC1	HPC2	HPC3	HPC4	HPC1	HPC2	HPC3	HPC4	CC1	CC2	CC3	CC4	CC5	CC6	CC7	CC8	CC9	PD8
D1...D64	Dados																
C1...C24	Checadores																
PC1...PC11	Paridade da Coluna dos Dados																
PC12...PC15	Paridade da Coluna dos Checadores																
PD1...PD8	Paridade dos Dados																
HPC1...HPC4	Hamming da Paridade da Coluna dos Dados																
HPC1...HPC4	Hamming da Paridade da Coluna dos Dados																
BC1...BC6	Base do C																
CC1...CC9	Checador do Checador																
BP1...BP10	Base Primos																

Fonte: AUTOR

altamente carregados segue. A principal regra que norteia a técnica para uma melhor eficácia é que quando ocorre um MBU este dar-se-á exclusivamente em células próximas.

Quando é alterada a tecnologia de um circuito é possível que o tamanho físico onde os dados serão salvos seja impactado e conseqüentemente as células de memória estarem mais próximas, ou seja, a densidade do circuito afeta diretamente a aplicação da técnica conforme já exposto na Figura 5. Dessa forma é salientado que a inversão binária ocorre dentro da memória, onde estão armazenados os dados e também os bits de paridade. De fato, o erro pode ocorrer em qualquer parte da memória podendo ser algum dado, a paridade propriamente dita ou os dois ao mesmo tempo.

O método foi validado utilizando simulação sendo o mesmo algoritmo para verificação das simulações e correção dos erros. O próximo capítulo irá abordar as simulações efetuadas bem como os resultados extraídos.

Diante disso, a seguir é descrito como é efetuado os cálculos para preenchimento da matriz já exposta na Figura 28, lembrando que os dados D, são a única informação acessível pelo circuito, todos os outros conjuntos são calculados e validados pelo codificador e decodificador.

5.1 CODIFICAÇÃO

Existem bits de paridade dependentes de cálculos que envolvem outros conjuntos de paridade, por isso a seqüência do cálculo para a codificação deve ser a descrita a seguir. Ob-

servemos a Figura 28, onde: D representa os dados, que são organizados sequencialmente em cada uma de suas respectivas posições, a partir desses bits são calculados os outros 7 conjuntos que são os seguintes:

5.1.1 Checadores (C)

Os Checadores (C) representam os bits de verificação usando a técnica Hamming, cada linha é verificada individualmente, cada bit representado por C valida os bits de dados estratégicos e o cálculo expresso na Equação 5.1 é feito para a primeira linha que se estende para as outras linhas:

$$\begin{aligned}
 C1 &= D1 \oplus D2 \oplus D4 \oplus D5 \oplus D7 \oplus D9 \oplus D11 \\
 C2 &= D1 \oplus D3 \oplus D4 \oplus D6 \oplus D7 \oplus D10 \oplus D11 \\
 C3 &= D2 \oplus D3 \oplus D4 \oplus D8 \oplus D9 \oplus D10 \oplus D11 \\
 C4 &= D5 \oplus D6 \oplus D7 \oplus D8 \oplus D9 \oplus D10 \oplus D11
 \end{aligned}
 \tag{5.1}$$

A partir desse cálculo é possível observar que a alteração de um bit do conjunto D , devido a um erro de SBU ou MBU, implicará na alteração de alguns bits de checagem específicos para cada dado por linha, ou seja, é possível mapear qual foi o bit alterado desde que seja 1 erro por linha. Estes, por sua vez, serão recalculados na decodificação. Os bits de checagem são alterados da seguinte maneira:

- $D1$ implica na alteração de $C1$ e $C2$.
- $D2$ implica na alteração de $C1$ e $C3$.
- $D3$ implica na alteração de $C2$ e $C3$.
- $D4$ implica na alteração de $C1$, $C2$ e $C3$.
- $D5$ implica na alteração de $C1$ e $C4$.
- $D6$ implica na alteração de $C2$ e $C4$.
- $D7$ implica na alteração de $C1$, $C2$ e $C4$.
- $D8$ implica na alteração de $C3$ e $C4$.
- $D9$ implica na alteração de $C1$, $C3$ e $C4$.

- D10 implica na alteração de C2, C3 e C4.
- D11 implica na alteração de C1, C2, C3 e C4.

Por exemplo, um SBU no D27 (que está na quinta posição da linha) afetará diretamente o C9 e C12 (que são a primeira e a quarta posição do conjunto C), podendo assim ajustar o bit afetado pela radiação. De mesmo modo o D60 também está na quinta posição da linha, sendo detectado de mesma forma pela primeira e a quarta posição do conjunto C, ou seja, pelo C21 e o C24 respectivamente. É salientado que o cálculo da última linha que compreende apenas 9 bits é efetuado ignorando os últimos dois bits alocado como dados nas linhas anteriores, dessa forma o recálculo para a última linha não possui as configurações de correção para os últimos dois bits.

5.1.2 Paridade da Coluna (PC)

A Paridade da Coluna (PC) faz a paridade simples da coluna dos dados, analisemos o cálculo da Equação 5.2 que é replicado para as outras colunas:

$$PC1 = D1 \oplus D12 \oplus D23 \oplus D34 \oplus D45 \oplus D56 \quad (5.2)$$

PC1 até PC11 faz a detecção para o conjunto D e os demais bits ajudam a mapear erros nos checkadores como, por exemplo, para o PC12 que é calculado conforme a Equação 5.3:

$$PC12 = C1 \oplus C5 \oplus C9 \oplus C13 \oplus C17 \oplus C21 \quad (5.3)$$

Esta paridade simples da coluna consegue identificar que a coluna está com erros ímpares, erros pares não são detectados e também não é possível identificar qual bit foi alterado. Mesmo com a detecção limitada é de extrema importância para auxiliar no mapeamento de erros, sua necessidade será tratada na decodificação.

5.1.3 Paridade dos Dados (PD)

Assim como possuímos a paridade da coluna também necessitamos da paridade da linha, expresso pela Paridade dos Dados(PD). Uma paridade simples dos dados é efetuada para cada linha de dados, a Equação 5.4 é efetuada:

$$PD1 = D1 \oplus D2 \oplus D3 \oplus D4 \oplus D5 \oplus D6 \oplus D7 \oplus D8 \oplus D9 \oplus D10 \oplus D11 \quad (5.4)$$

O cálculo exposto é estendido para as demais linhas. Como é calculado uma paridade simples é possível averiguar erros ímpares e conseqüentemente, da mesma forma que a PC, não é possível determinar onde ocorreu o erro. Esse conjunto auxilia na prevenção de falsas correções e será explanado mais detalhadamente na decodificação. Salienta-se que o PD7 e o PD8, que estão nas últimas duas linhas às quais não possuem bits de dados, não estão sendo utilizados.

5.1.4 Hamming da Paridade da Coluna (HPC)

Neste ponto é finalizada a verificação direta dos dados, os próximos conjuntos de bits são para verificar a integridade disposta nas paridades já explicitadas. O Hamming da Paridade da Coluna (HPC), disposto duas vezes (Redundância) dentro da matriz, utiliza a técnica de Hamming para validar os bits PC, considerando do PC1 ao PC11, limitando-o a paridade da coluna dos dados, ignorando a paridade dos Checadores. O cálculo do HPC é exposto na Equação 5.5:

$$\begin{aligned}
 HPC1 &= PC1 \oplus PC2 \oplus PC4 \oplus PC5 \oplus PC7 \oplus PC9 \oplus PC11 \\
 HPC2 &= PC1 \oplus PC3 \oplus PC4 \oplus PC6 \oplus PC7 \oplus PC10 \oplus PC11 \\
 HPC3 &= PC2 \oplus PC3 \oplus PC4 \oplus PC8 \oplus PC9 \oplus PC10 \oplus PC11 \\
 HPC4 &= PC5 \oplus PC6 \oplus PC7 \oplus PC8 \oplus PC9 \oplus PC10 \oplus PC11
 \end{aligned}
 \tag{5.5}$$

O HPC permite a detecção e a correção de até um erro dentro do PC1 até PC11, este por sua vez é duplicado dentro da matriz para evitar falsos positivos, na decodificação será tratado esta necessidade.

5.1.5 Base do C (BC)

Conforme já explicitado, a PC12 até a PC15 efetua o cálculo de paridade simples da coluna do C, a Base do C (BC) faz a checagem na horizontal, ou seja, o BC faz a paridade simples para a linha do C. A título de exemplificação o BC1 é calculado conforme a Equação 5.6, as demais linhas são calculadas da mesma forma.

$$BC1 = C1 \oplus C2 \oplus C3 \oplus C4 \tag{5.6}$$

Se a alteração binária ocorrer em algum dos bits de paridade representados por C será

possível efetuar um mapeamento detectando se ocorreu alguma alteração, ou seja, ao inverter algum bit do conjunto C, as paridades BC e a PC (12 a 15) reconhecerão o erro e acionaram o tratamento, utilizando o Checador do Checador (CC) e Base primos (BP), os quais serão explanados na sequência.

5.1.6 Checador do Checador (CC) e Base primos (BP)

Todavia, utilizando apenas os conjuntos PC e BC para detecção de problemas no conjunto C, não há recursos suficientes para garantir a integridade dos checadores, esta deficiência é sanada pelo CC e BP.

O CC realiza um cálculo, utilizando um vetor de pesos, o qual é validado com o pré cálculo do C. Este vetor é composto da seguinte maneira: {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23 e 24}. Cada posição dos Checadores valida a sua respectiva posição dentro do vetor de peso. Se o bit respectivo do checador for 1, o peso do vetor é acionado e se for 0 é ignorado, a soma final do vetor é salva no formato binário dentro da matriz representada por CC1 ao CC9.

O vetor BP utiliza o mesmo princípio, entretanto, o peso do vetor é distribuído pelos primeiros 24 números primos alternados, vejamos como ele é disposto: {2, 89, 3, 83, 5, 79, 7, 73, 11, 71, 13, 67, 17, 61, 19, 59, 23, 53, 29, 47, 31, 43, 37 e 41}. Imaginemos que após calcular a paridade dos dados representada pelos checadores, a segunda linha composta pelo C5 ao C8 resulte na seguinte sequência: 0101. Para o CC será adicionado o 6 e o 8 e para o BP será adicionado o 79 e o 73.

Para além, um exemplo completo de codificação para o conjunto CC e BP é realizado. O conjunto C é previamente calculado utilizando as regras já citadas resultando nos bits expressos na Figura 29.

Figura 29 – Conjunto C após ser calculado utilizando os bits dos dados D quaisquer.

1	1	0	1
0	1	1	0
1	1	1	1
0	0	0	1
0	1	0	1
0	0	1	1

Fonte: AUTOR

A soma prévia para o CC está disposto na Equação 5.7 e para o BP na Equação 5.8. A

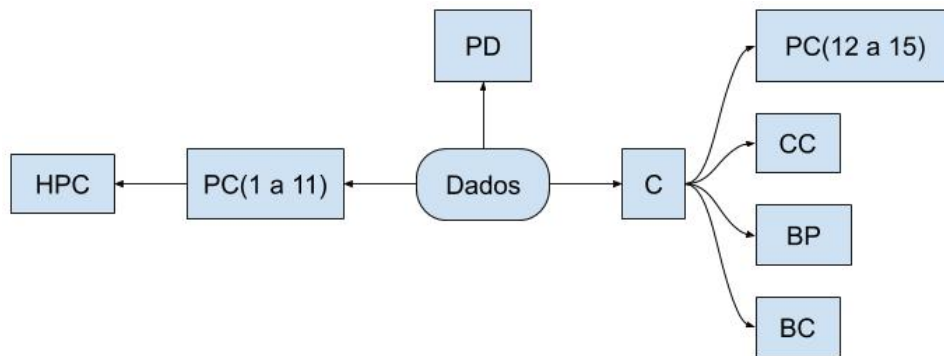
soma dos valores para o CC resulta em 163 e para o BP 659. Transformando o valor decimal 163 para binário, temos como resultado 10100011, sabendo que existem 9 bits reservados para o CC será alocado o seguinte valor: 010100011. Já, o 659 para binário resulta em 1010010011, o qual é disposto nos locais indicados na Figura 28.

$$CC = 1 + 2 + 0 + 4 + 0 + 6 + 7 + 0 + 9 + 10 + 11 + 12 + 0 + 0 + 0 + 16 + 0 + 18 + 0 + 20 + 0 + 0 + 23 + 24 = 163 \quad \{010100011_2\} \quad (5.7)$$

$$BP = 2 + 89 + 0 + 83 + 0 + 79 + 7 + 0 + 11 + 71 + 13 + 67 + 0 + 0 + 0 + 59 + 0 + 53 + 0 + 47 + 0 + 0 + 37 + 41 = 659 \quad \{1010010011_2\} \quad (5.8)$$

Neste ponto, os cálculos de paridades estão completos, resumidamente os dados são dispostos sequencialmente dentro da matriz. O C, PC(do 1 ao 11) e PD checam os dados, já HPC checa o PC(do 1 ao 11). PC(12 a 15), BC, BP e CC validam o C. Vejamos na representação gráfica expressa pela Figura 30 as dependências de checagem.

Figura 30 – Dependências dos cálculos.



Fonte: AUTOR

5.1.7 Exemplo de Codificação

Para exemplificação de como resulta a codificação explanada anteriormente será efetuada uma codificação, passo a passo para cada um dos conjuntos citados. Uma mensagem qualquer M, que precise ser armazenada na memória, possuindo o seguinte conteúdo: {1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0} é verificada. A Figura 31 mostra como os dados são alocados sequencialmente dentro da matriz.

Figura 31 – Exemplo alocação dos dados

1	0	1	1	1	0	1	1	1	1	0									
1	0	1	0	1	0	1	0	1	1	0									
1	0	1	1	1	0	1	1	1	0	0									
1	0	0	1	1	0	1	1	0	0	0									
1	0	0	1	1	0	1	1	0	0	1									
0	0	0	0	0	1	1	1	0											

Fonte: AUTOR

Em seguida, o conjunto C é calculado, utilizando a técnica de Hamming, é preenchido os bits de validação nos seus devidos lugares. A Figura 32 expõe o resultado dos cálculos descritos anteriormente.

Figura 32 – Exemplo Cálculo do C, utilizando os Bits da Mensagem M

1	0	1	1	1	0	1	1	1	1	0	1	1	1	1					
1	0	1	0	1	0	1	0	1	1	0	0	0	1	0					
1	0	1	1	1	0	1	1	1	0	0	1	0	0	0					
1	0	0	1	1	0	1	1	0	0	0	0	1	0	1					
1	0	0	1	1	0	1	1	0	0	1	1	0	1	0					
0	0	0	0	0	1	1	1	0			1	0	1	1					

Fonte: AUTOR

A Paridade da Coluna é calculada. A Paridade dos Dados também, os resultados são demonstrados na Figura 33.

Figura 33 – Exemplo Calculo do PC e PD

1	0	1	1	1	0	1	1	1	1	0	1	1	1	1					
1	0	1	0	1	0	1	0	1	1	0	0	0	1	0					0
1	0	1	1	1	0	1	1	1	0	0	1	0	0	0					1
1	0	0	1	1	0	1	1	0	0	0	0	1	0	1					1
1	0	0	1	1	0	1	1	0	0	1	1	0	1	0					0
0	0	0	0	0	1	1	1	0			1	0	1	1					1
1	0	1	0	1	1	0	1	1	0	1	0	0	0	1					0
																			0

Fonte: AUTOR

O HPC utiliza os valores calculados na etapa anterior resultando em $\{0, 0, 0, 1\}$. Este por sua vez é colocado duas vezes, o principal e o secundário, para garantir a redundância necessária para a decodificação. A paridade BC também é calculada, a Figura 34 é exposta.

Por fim, o CC e o BP são calculados. A soma resultante do vetor de pesos é de 160 para o CC e 476 para o BP. Convertidos para o formato binário tem-se para o CC e o BP respectivamente 010100000 e 0111011100. A Figura 35 demonstra como é finalizado o cálculo de bits para a mensagem M.

Figura 34 – Exemplo Calculo do HPC e BC

1	0	1	1	1	0	1	1	1	1	0	1	1	1	1	0	0	0
1	0	1	0	1	0	1	0	1	1	0	0	0	1	0	1	0	0
1	0	1	1	1	0	1	1	1	0	0	1	0	0	0	1	0	1
1	0	0	1	1	0	1	1	0	0	0	0	1	0	1	0	0	1
1	0	0	1	1	0	1	1	0	0	1	1	0	1	0	0	0	0
0	0	0	0	0	1	1	1	0	0	0	1	0	1	1	1	0	1
1	0	1	0	1	1	0	1	1	0	1	0	0	0	1	0	0	0
0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	1	0	0

Fonte: AUTOR

Figura 35 – Exemplo Calculo do CC E BP

1	0	1	1	1	0	1	1	1	1	0	1	1	1	1	0	1	0
1	0	1	0	1	0	1	0	1	1	0	0	0	1	0	1	1	0
1	0	1	1	1	0	1	1	1	0	0	1	0	0	0	1	0	1
1	0	0	1	1	0	1	1	0	0	0	0	1	0	1	0	1	1
1	0	0	1	1	0	1	1	0	0	1	1	0	1	0	0	1	0
0	0	0	0	0	1	1	1	0	0	1	1	0	1	1	1	1	1
1	0	1	0	1	1	0	1	1	0	1	0	0	0	1	0	0	0
0	0	0	1	0	0	0	1	0	1	0	1	0	0	0	0	0	0

Fonte: AUTOR

5.1.8 Interleaving

Para potencializar os ganhos de correção é necessário garantir que conjuntos sensíveis não estejam com erros ao mesmo tempo, ou seja, se ocorrer um *upset* no conjunto PC e ao mesmo tempo no dado correspondente ele poderá anular a detecção.

Para garantir a correta detecção é utilizada a técnica de *Interleaving* que consiste em reorganizar os dados, considerando alguns fatores. Isso garante que quando houver um impacto de partículas, o erro ocorra em conjuntos distintos de bits, podendo ser tratado como um ou mais SBU no lugar de um MBU, o que facilita a correção (RADAELLI et al., 2005). As regras que foram utilizadas para a criação da matriz final, a qual é armazenada dentro da memória, demonstrada na Figura 36, serão abordadas na sequência, para tal observamos a imagem.

Figura 36 – Matriz com *Interleaving*

D28	PC1	C24	C5	PC4	D27	D35	HPC3	HPC4	BC1	BC2	BC3	HPC1	HPC2	HPC3	CC1	BP3	PD1
C6	D2	C19	C10	D58	D15	D34	HPC1	HPC2	BC4	BC5	BC6	D3	D16	HPC4	CC2	BP4	PD2
C11	D48	C14	C20	D17	D51	D52	D31	D43	D33	D53	D21	D11	D59	CC9	CC3	BP5	PD3
C16	PC2	C9	C15	PC5	D32	D55	D36	D26	D49	D1	PC9	D24	D39	D29	CC4	BP6	PD4
C21	D5	C18	C23	D47	D4	D38	D61	D18	D8	C1	D7	D41	D9	PC7	CC5	BP7	PD5
C2	D30	C7	C12	D6	D62	D19	D42	D54	D22	D25	D37	D60	D14	D56	CC6	BP8	PD6
C4	D40	C13	C3	D64	D10	D44	PC8	D23	D57	PC11	D45	D13	D50	BP2	CC7	BP9	PD7
C22	PC3	C17	C8	PC6	D12	D46	PC12	PC13	PC14	PC15	D63	D20	BP1	PC10	CC8	BP10	PD8

Fonte: AUTOR

A distância utilizada para o tratamento depende da área afetada pelo SEE, consequen-

temente o tamanho do circuito impacta diretamente nesta característica. Dos experimentos observados nesta pesquisa como, por exemplo, as verificações de RAO et al. (2014) expostas na Figura 12, verificou-se que a distância máxima afetada pelo impacto da radiação é de duas casas em relação ao erro principal, ou seja, os bits afetados encontram-se em uma submatriz de ordem 3.

É salientado que a tecnologia envolvida nesse experimento é para um FPGA de 45 nm onde a distribuição de energia utilizada para os testes foi *JEDEC89a*. Assim sendo, alguns bits devem estar afastados para garantir que um SEE não cause perturbações em dois validadores ao mesmo tempo, o indicativo é a distância do raio que não poderá atingir o conjunto de bits analisados. Vejamos:

- A coluna de dados não pode estar próxima. Exemplo: D1, D12, D23, D34, D45 e D56 precisam estar afastados entre si, isso garante que a paridade da coluna detectará o erro, evitando a auto anulação.
- Os checadores representados por C são validados pelo PC(12 ao 15), CC, BP e BC e conseqüentemente o C precisa estar fora do raio de alcance dos seus validadores.
- CC e BP devem estar longes do BC e PC(12 ao 15) para evitar falsos positivos.
- BC deve estar distante do PC(12 ao 15), evitando assim a auto anulação em caso de SEE.
- PD verifica a paridade simples dos dados da linha, então PD precisa estar fora do raio de alcance dos bits de dados da linha correspondente.
- Existe uma redundância do HPC e conseqüentemente elas devem estar longe uma da outra.
- O HPC valida o PC(1 ao 11), devendo estes estar longe um do outro.
- As paridades da coluna dos dados dispostas do PC1 ao PC11 não podem estar próximas entre si, pois a técnica de Hamming corrige apenas 1 erro por conjunto de bits.

Além da regra de conjuntos descrita anteriormente é necessário garantir que a auto anulação do C não seja válida. Vejamos a Figura 37 A, a qual é uma possibilidade de erro provindo de um MBU. Nesta situação é possível a detecção de erro pelos auxiliares BC4 e BC6. Entretanto na Figura 37 (B), eles são anulados em linha e coluna, impossibilitando a detecção do Erro.

Figura 37 – Possibilidade de auto anulação.

C1	C2	C3	C4	BC1
C5	C6	C7	C8	BC2
C9	C10	C11	C12	BC3
C13	C14	C15	C16	BC4
C17	C18	C19	C20	BC5
C21	C22	C23	C24	BC6
PC12	PC13	PC4	PC5	

(A)

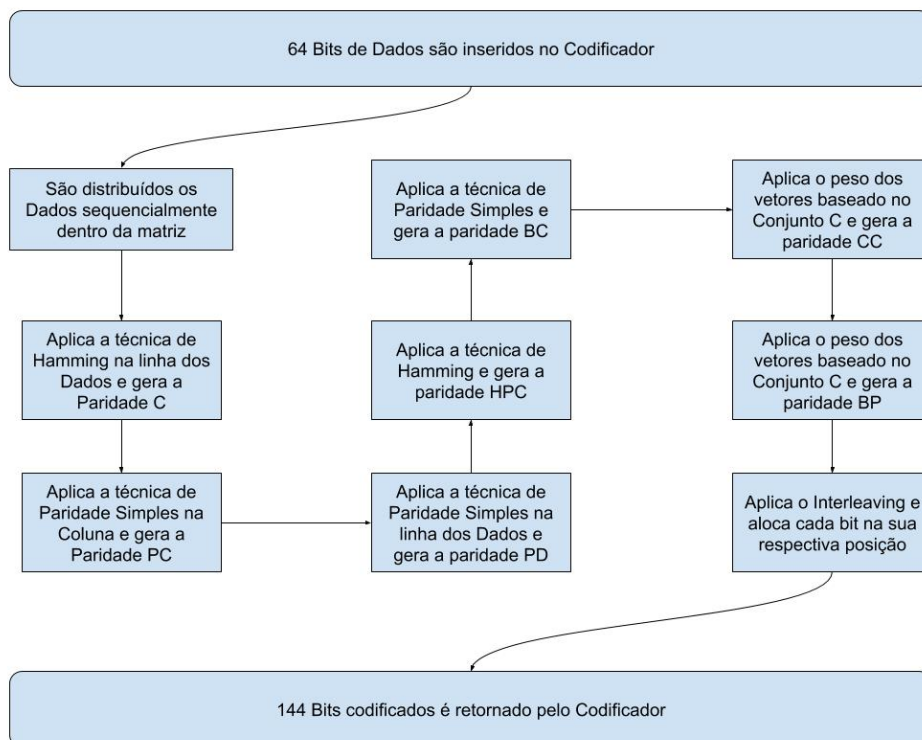
C1	C2	C3	C4	BC1
C5	C6	C7	C8	BC2
C9	C10	C11	C12	BC3
C13	C14	C15	C16	BC4
C17	C18	C19	C20	BC5
C21	C22	C23	C24	BC6
PC12	PC13	PC4	PC5	

(B)

Fonte: AUTOR

Por fim é apresentado o fluxograma geral para a codificação dos bits de dados utilizando as técnicas descritas, o qual está ilustrado na Figura 38.

Figura 38 – Fluxograma de Codificação dos bits



Fonte: AUTOR

5.2 DECODIFICAÇÃO

Para efetuarmos a decodificação é necessário seguir um caminho alternativo ao caminho da codificação. Para compreensão da técnica vamos nomenclar como **Dados Codificados (DC)** os dados colhidos da memória, após o desembaralhamento que foi aplicado pelo *Inter-*

leaving, lembrando que estes podem ou não conter erros. Do DC é extraído apenas os bits do conjunto D e gerado uma nova paridade para os dados extraídos, utilizando as técnicas citadas anteriormente.

Ao compararmos os validadores dos DC com os dados recalculados, não havendo alteração em nenhum bit, a informação é declarada como intacta, garantindo a integridade da informação, ou seja, não teve nenhuma alteração nos bits que foram guardados na memória. Entretanto, se o binário de um não corresponder ao binário do outro é necessário o tratamento da informação. O tratamento segue um fluxo que torna mais compreensível a decodificação. Dessa forma, abaixo estão divididos em subseções.

5.2.1 Tratamento do C

Primeiramente recalculamos o BC e o PC(12 ao 15) a partir do conjunto DC. Se os valores contidos no recálculo forem condizentes com os valores do DC considera-se os bits do conjunto C intactos. Caso contrário, existem três possibilidades, lembrando que apenas uma delas pode ocorrer por SEE, sendo elas:

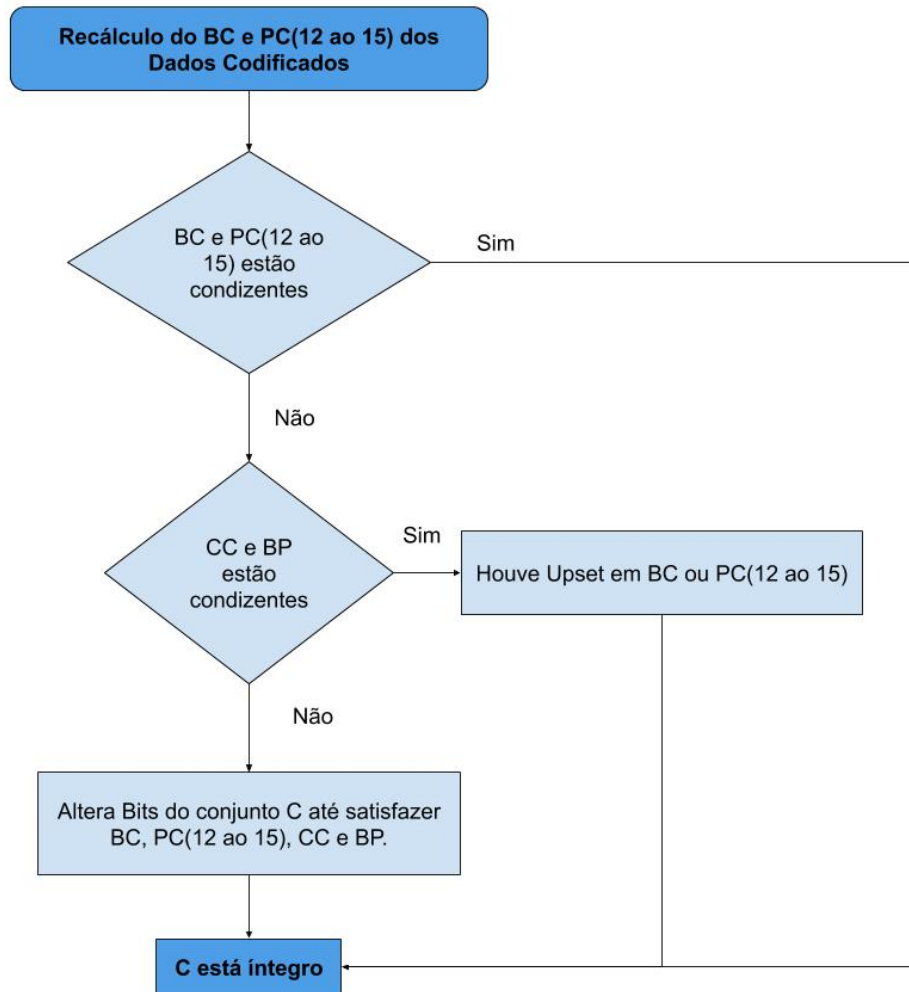
- Houve alterações no conjunto BC
- Houve alterações no conjunto PC, especificamente do 12 ao 15.
- Houve alterações no Conjunto C

É salientado que qualquer erro que ocorra nos bits de conjunto C, seguindo as regras já citadas, garantem a detecção pelo mapeamento da paridade simples como, por exemplo, a representação da Figura 37 (A). Vejamos o fluxograma expresso na Figura 39.

Se algum erro for detectado é necessário o recálculo do CC e BP a partir do conjunto DC, pois pode ter ocorrido a possibilidade de erro no conjunto BC ou PC(12 ao 15). É enfatizado que o erro nunca ocorrerá nos dois validadores (BC e PC12 ao 15) ao mesmo tempo, pois pelas regras citadas anteriormente o respingo de erros não alcança os dois conjuntos, além disso, erros no CC e BP garantem a integridade do C que é inicialmente validada pelo BC e PC(12 ao 15).

Ainda assim, se os valores de CC e BP não condizem com o recálculo existe a necessidade do ajuste no conjunto C. O ajuste estará contemplado quando as validações de C expressos na Figura 30 estiverem satisfeitas. Isto é, deve ser alterado os bits do Conjunto C até que paridades BC, PC(12 ao 15), CC e BP ao serem recalculados estejam de acordo com os bits do DC.

Figura 39 – Integridade do C.



Fonte: AUTOR

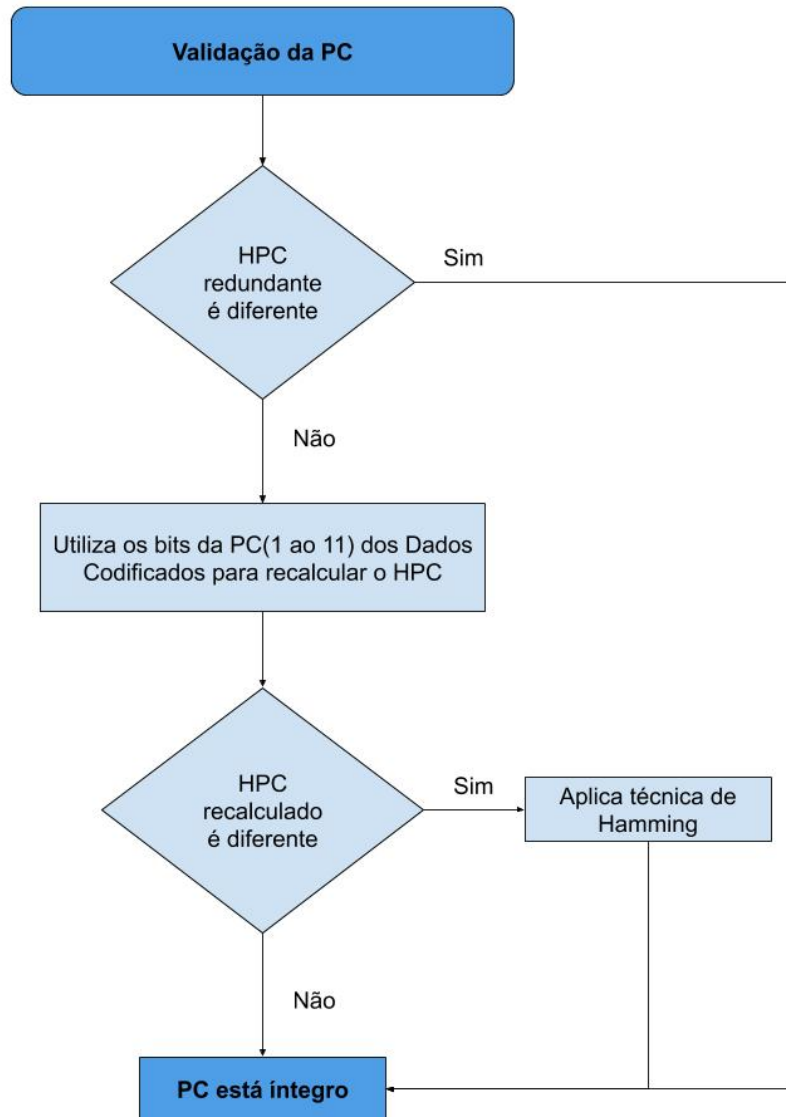
5.2.2 Tratamento da PC

Após a garantia de integridade do C é efetuado os cálculos que validam a paridade PC. Tendo o HPC redundante na memória, é possível compará-las e se não forem iguais houve erro em algum dos HPC e a PC está íntegro, entretanto se estão iguais é necessário refazer o cálculo do HPC, utilizando os bits da PC presentes no DC, vejamos o fluxograma da Figura 40.

A seguir as configurações de correção para as alterações do HPC, essas correções são dados pelo tratamento usando a técnica de Hamming. A título de exemplificação, se a PC5 for alterada será validado seu erro através do HPC1 e HPC4 que estarão divergentes entre o recálculo e DC

- PC1 implica na alteração de HPC1 e HPC2.

Figura 40 – Integridade da PC.



Fonte: AUTOR

- PC2 implica na alteração de HPC1 e HPC3.
- PC3 implica na alteração de HPC2 e HPC3.
- PC4 implica na alteração de HPC1, HPC2 e HPC3.
- PC5 implica na alteração de HPC1 e HPC4.
- PC6 implica na alteração de HPC2 e HPC4.
- PC7 implica na alteração de HPC1, HPC2 e HPC4.
- PC8 implica na alteração de HPC3 e HPC4.

- PC9 implica na alteração de HPC1, HPC3 e HPC4.
- PC10 implica na alteração de HPC2, HPC3 e HPC4.
- PC11 implica na alteração de HPC1, HPC2, HPC3 e HPC4.

5.2.3 Verificação dos Dados

Por fim, a parte mais importante, que faz a verificação da integridade dos dados. Até esse momento, os cálculos foram necessários para validar os validadores dos Dados. Essa primeira verificação garante que os dados não sofreram falsos positivos, ou seja, a informação correta não vai ser acusada de estar incorreta.

Os checadores dos dados devem ser os que foram validados nos cálculos anteriores, ou seja, recalculamos o C utilizando os dados colhidos na memória, este é comparado com o C previamente tratado. A paridade PC é recalculada da mesma forma que o C e efetuada as comparações. A comparação ocorre linha a linha e coluna a coluna e se detectado alterações é aplicada a técnica de Hamming, conforme já explicitado na codificação. A exemplo, a alteração da posição 7 de um dado em qualquer linha implicará na divergência da primeira, segunda e quarta posição do C. Analisemos o fluxograma da Figura 62 disponível no Apêndice A.

A correção ocorre da primeira linha em diante. Ao final da verificação utilizando o C é verificada a integridade dos dados pela PC e da PD. Se uma dessas duas verificações estiverem incoerentes, a validação da primeira linha em diante é desfeita e ocorre na forma inversa da última linha até a primeira.

Uma nova averiguação da PC e da PD ocorrem, pois ainda assim existe a possibilidade de ocorrer falsos positivos como, por exemplo, o D4 e o D10 que estão próximos e na mesma linha, se houver um *upset* o que ocorre é o seguinte:

- O D4 implica na alteração do C1, C2 e C3
- O D10 implica na alteração do C2, C3 e C4
- A alteração dos dois bits implica em uma anulação do C2 e C3 apontando divergência apenas no C1 e C4.
- A configuração de erro do D5 é a alteração do C1 e do C4.

Dessa forma, o D5 é apontado erroneamente como errado e a PC irá detectar que houveram falsos positivos. As alterações novamente são desfeitas e o algoritmo deixa de se basear pelo conjunto C utilizando como norteador o conjunto PC.¹

¹ Disponível no Apêndice B a Figura 63 que compreende o Fluxograma Geral para a Decodificação.

6 SIMULAÇÕES E ANÁLISE DOS DADOS

Com o intuito de elucidar desenvolvimento dos testes, bem como a comparação com outras técnicas, este capítulo busca descrever como foram realizados os experimentos e os resultados obtidos, comparar as técnicas consolidadas no meio científico e por fim alguns dos exemplos verificados.

Primeiramente precisamos definir como ocorrem os erros e como eles serão aplicados dentro da matriz para efetuarmos as simulações. Conforme já fundamentado nesta pesquisa, todos os erros ocorrem de forma próxima, mais especificamente adjacentes ao erro principal. Além disso, a maioria das pesquisas encontradas na literatura demonstram que a distância máxima do erro principal é de duas linhas e duas colunas. Diante disso, foi determinado dois parâmetros para validar os testes, sendo eles:

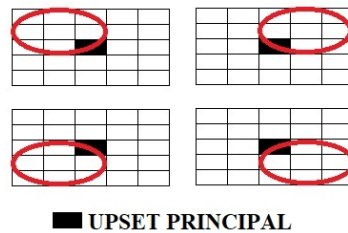
- **Janela de Erros:** Tamanho do raio em que o SEE pode ocorrer.
- **Erros:** Quantidade de erros dentro da Janela de Erros.

Já dito, o número de células de memória afetados pelo MBU depende exclusivamente do tamanho físico da memória, explicitado na Figura 5. Na pesquisa de SILVA (2018) e de NEALE; SACHDEV (2014) foram verificados algoritmos de tratamento de MBU. Nas duas pesquisas são referenciados as simulações apenas como erros adjacentes e não é exposto a distância máxima do erro principal averiguada.

Além disso, a Figura 6 disponível no capítulo 2 e as Figuras 10, 11 e 12 disponíveis no capítulo 3 trazem modelos de erros em outras pesquisas analisadas, demonstrando que grande parte dos erros oriundos da radiação mantêm-se restritos a uma distância máxima de uma submatriz de ordem 3. Diante disso, foi adotado como padrão para as simulações de comparação a janela de erros de tamanho 3 e 4. A Figura 41 ilustra como a tratativa de erros ocorre levando em consideração uma janela de erros de tamanho 3.

Toda modelagem do algoritmo foi desenvolvida no MATLAB[®] 2018, bem como a bateria de testes foi gerada nessa plataforma. A título de exemplificação um subconjunto foi utilizado da matriz principal, onde a nova matriz é quadrada de ordem três, considerando que esse é o tamanho afetado pelo SEE. Nesta nova submatriz, realizou-se a aplicação de todas as possibilidades de erros possíveis, testando a correção a cada nova iteração. Após a validação desta sub-matriz, deslocou-se a mesma em uma coluna e realizou-se novamente os testes até

Figura 41 – Tamanho da Janela de Erros de ordem 3

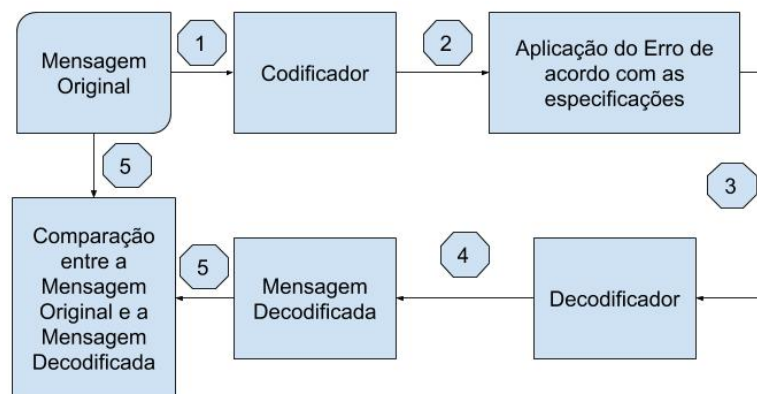


Fonte: AUTOR

alcançar o final do conjunto, onde incrementa-se uma linha e retorna-se à coluna inicial, realizando os testes novamente, até o final das colunas. Ao chegar ao final do conjunto maior, a verificação dos erros está concluída.

A Figura 42 expressa o passo a passo generalizado para as simulações de correção que o algoritmo foi submetido. Primeiramente uma mensagem qualquer de 64 bits é codificada, utilizando a metodologia já abordada anteriormente, em seguida o resultado da codificação, ou seja, os 144 bits resultantes da operação são submetidos aos erros.

Figura 42 – Bateria de testes



Fonte: AUTOR

Os bits apontados como *upsets* seguem as configurações de erros da Janela de Erros em concordância com a quantidade de Erros que por sua vez são invertidos na mensagem simulando o erro. Após a inserção dos erros é acionado o decodificador que refaz os cálculos traduzindo os 144 bits da mensagem codificada nos 64 bits da mensagem decodificada.

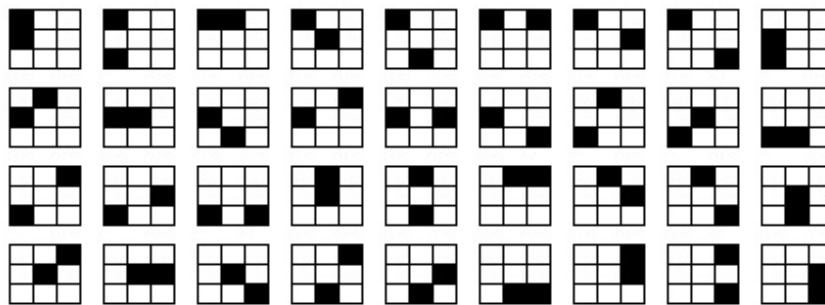
A comparação da Mensagem Original com a Mensagem Decodificada ocorre e se os bits forem iguais tem-se que o decodificador conseguiu solucionar os problemas de *upsets*. Conseqüentemente, se as mensagens forem divergentes é somado 1 a quantia de configurações

de erros não solucionáveis.

É salientado que o decodificador retorna somente os bits que considera corretos depois do tratamento. Não foi implementado a possibilidade de ajuste nos bits checadores e nem a detecção se houve ou não sucesso na decodificação.

A título de exemplificação: para 1 erro temos 144 possibilidades que são todos os bits possíveis dentro da matriz. Para 2 *upsets* com janela de erro de ordem 3, ou seja, para submatriz afetada pelo SEE de tamanho 3X3, existem 36 possibilidades dentro da submatriz contemplando um total de 1.356 formatos diferentes na matriz principal (8 linhas e 18 colunas), as formas possíveis de erros são exibidas na Figura 43.

Figura 43 – Possibilidades para 2 Erros de *upsets* dentro do raio de alcance do SEE 3X3



Fonte: AUTOR

De igual forma para 9 erros tem-se apenas 1 possibilidade na sub matriz, representada na Figura 44, totalizando 96 possibilidades na matriz principal.

Figura 44 – Possibilidades para 9 Erros de *upsets* dentro do raio de alcance do SEE 3X3



Fonte: AUTOR

O primeiro conjunto de testes gerados no MATLAB foi todos os erros possíveis, de 1 a 9 erros, na janela 3X3. A Tabela 1 descreve a quantidade de modelos de erros possíveis para cada quantidade de erros averiguada, além disso, a quantidade não solucionados e a cobertura de correção também é exposta. A título de comparação foi verificado a correção com *Interleaving* e sem *Interleaving*.

Perante o exposto na Tabela 1 é possível validar que o algoritmo desenvolvido permite a correção de 100% das possibilidades até 4 erros considerando que o SEE afete uma janela

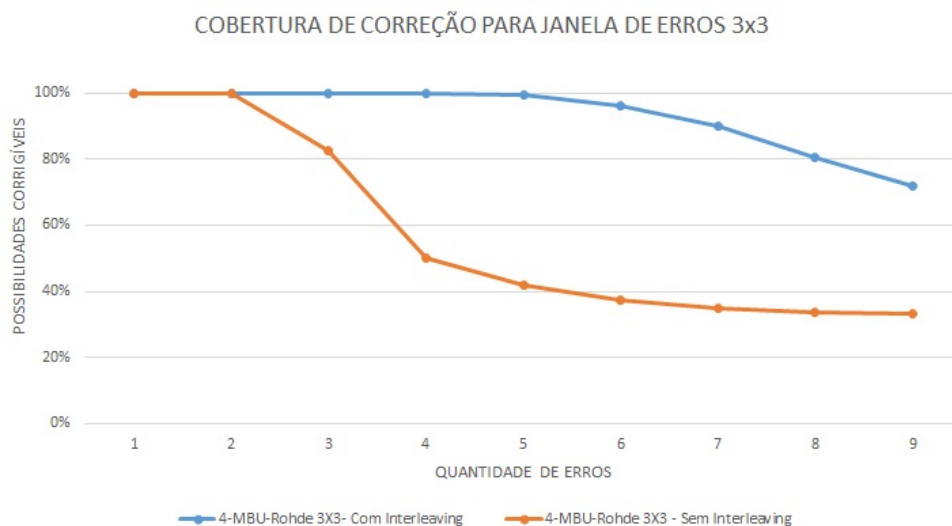
Tabela 1 – Correções dos modelos de erros Janela 3X3

Quantidade de Erros	Possibilidades de Erros Janela 3X3	Não corrigíveis Com <i>Interleaving</i>	Cobertura Com <i>Interleaving</i>	Não corrigíveis Sem <i>Interleaving</i>	Cobertura Sem <i>Interleaving</i>
# 1	144	0	100%	0	100%
# 2	1.356	0	100%	0	100%
# 3	4.964	0	100%	868	82,51%
# 4	9.621	0	100%	4.816	49,94%
# 5	11.076	78	99,30%	6.426	41,98%
# 6	7.894	290	96,33%	4.944	37,37%
# 7	3.456	350	89,87%	2.253	34,81%
# 8	864	168	80,56%	573	33,68%
# 9	96	27	71,88%	64	33,33%

Fonte: AUTOR

máxima de ordem 3. Aliás, somando todas as possibilidades de erros têm-se 39.471 possibilidades, das quais utilizando o *Interleaving* 913 configurações não são corrigíveis expondo um percentual de correção maior que 97% para todas as possibilidades de *upsets* em uma janela de ordem 3.

Para demonstrar a eficiência do *Interleaving* utilizando as regras já citadas a Figura 45 mostra a comparação da corrigibilidade do algoritmo utilizando como parâmetro a Janela de Erros 3x3, nela podemos verificar que é expressiva a diferença do índice de cobertura de erros utilizando a correta distribuição dos conjuntos de bits dentro da matriz.

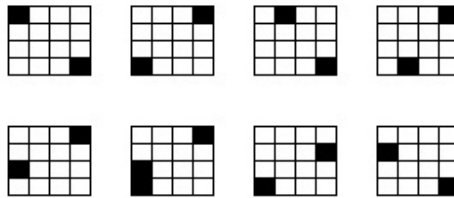
Figura 45 – Comparação da eficiência com e sem o *Interleaving*

Fonte: AUTOR

Devemos levar em consideração que de fato erros provindos da radiação são apenas adjacentes e que uma janela de erros maior que 3X3 possibilita configurações de erros que

não sejam próximos como, por exemplo, a ilustração da Figura 46 considerando uma Janela de Erros 4X4.

Figura 46 – Possibilidades de *upsets* em uma Janela de Erros 4X4



Fonte: AUTOR

Assim sendo, a título de comparação e verificação foram efetuadas as simulações de erros em uma Janela 4X4 da mesma forma que para a Janela de erros 3X3, obviamente a quantidade de modelos de erros resultantes é maior bem como as possibilidades de cobertura de erros são alteradas, a Tabela 2 expõe os resultados obtidos.

Tabela 2 – Correções dos modelos de erros Janela 4X4

Quantidade de Erros	Possibilidades de Erros Janela 4X4	Configurações Não corrigíveis	Cobertura
# 1	144	0	100%
# 2	2.436	0	100%
# 3	18.104	516	97,14%
# 4	79.206	6.278	92,07%
# 5	231.696	38.126	83,54%
# 6	485.184	135.812	72,00%
# 7	757.056	307.351	59,40%
# 8	901.404	473.084	47,51%
# 9	829.456	519.704	37,34%

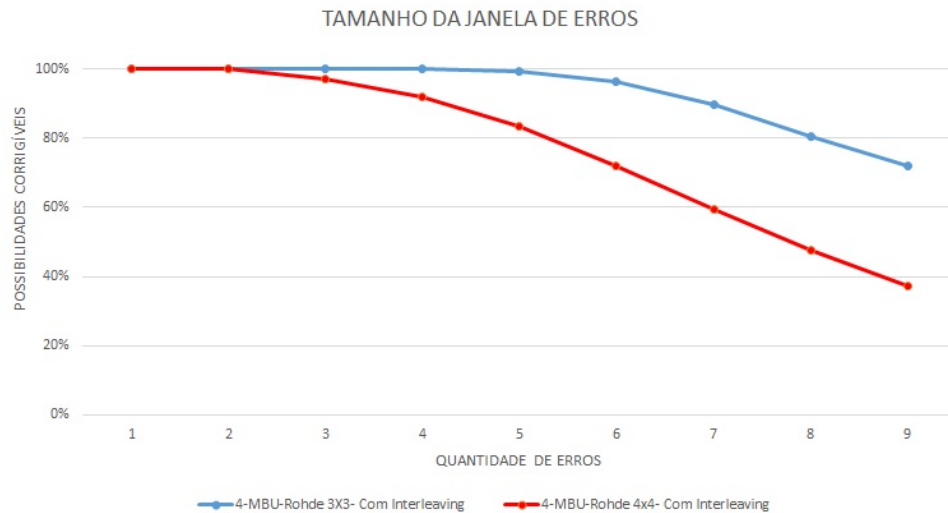
Fonte: AUTOR

A verificação de erros para uma Janela de Erros de ordem 4 não validou a correção de erros para o algoritmo sem o *Interleaving*, devido a alta quantidade de cálculos para todas as simulações efetuadas nesse estudo o tempo de constatação da execução do MATLAB foi muito elevado, além disso, a técnica verificada sem o embaralhamento dos dados para uma Janela de Erros 3x3 demonstrou ser ineficaz . 3.304.686 possibilidades são possíveis para todas as combinações de erros para Janela de Erros de ordem 4, destas 1.480.871 configurações não são corrigíveis obtendo um percentual de 55,18% de capacidade de correção do algoritmo para esses parâmetros. Lembrando que todas as formas de erros possíveis para a Janela de Erros de ordem 3 estão inclusas nas configurações da Janela de Erros de ordem 4.

Sabemos que o tamanho físico do circuito pode impactar na distância que os bits serão

afetados pela colisão de partículas altamente carregadas, de tal forma a Figura 47 nos mostra a expressiva diferença de correção dos erros na Janela de Erros 3x3 e 4x4, ambas utilizando o *Interleaving*.

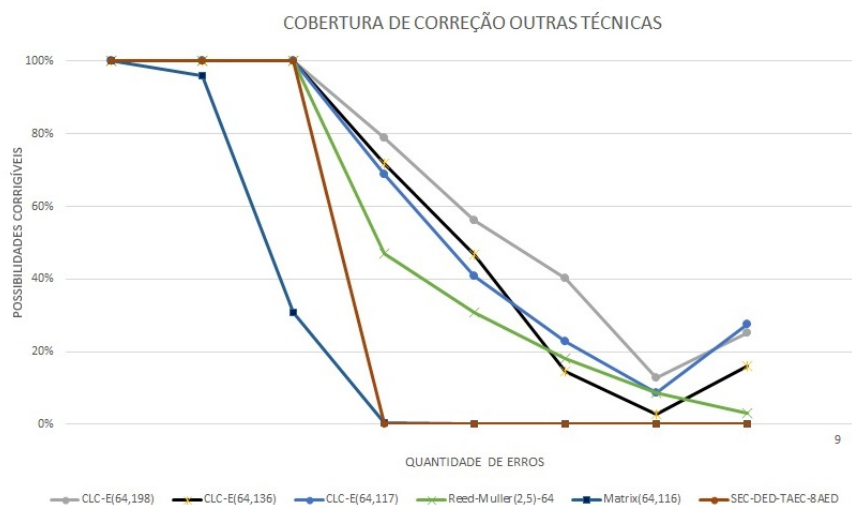
Figura 47 – Comparação da eficiência pelo tamanho do raio do impacto



Fonte: AUTOR

Na trabalho de SILVA (2018) foram verificados 5 algoritmos distintos, enquanto em NEALE; SACHDEV (2014) foi verificado mais uma técnica. É possível verificar na Figura 48 os índices de correções estipulados pelas simulações. Salienta-se que os dados manifestados nas pesquisas mencionadas não apresentaram as regras de embaralhamento.

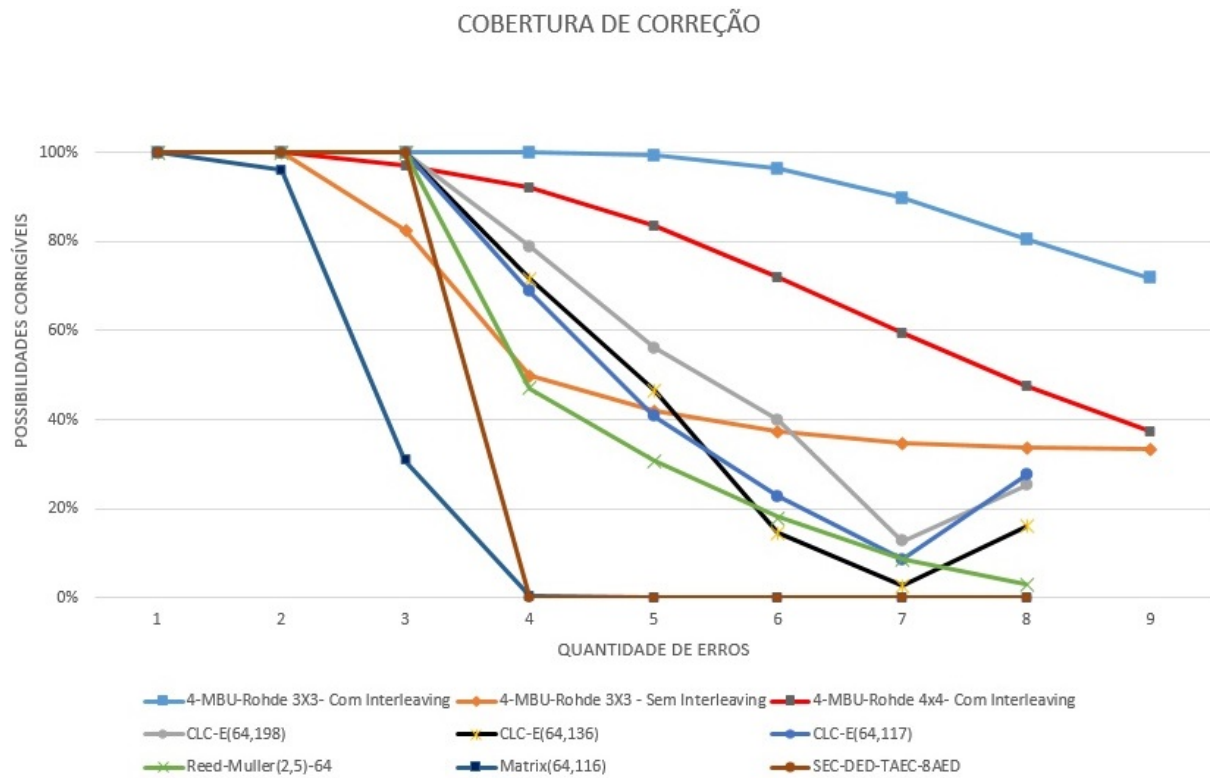
Figura 48 – Cobertura de erros outras técnicas



Fonte: AUTOR

Além disso, uma comparação com todas as técnicas é efetuada, a Figura 49 demonstra a comparação com os índices de correção.

Figura 49 – Cobertura de Erros por Técnica



Fonte: AUTOR

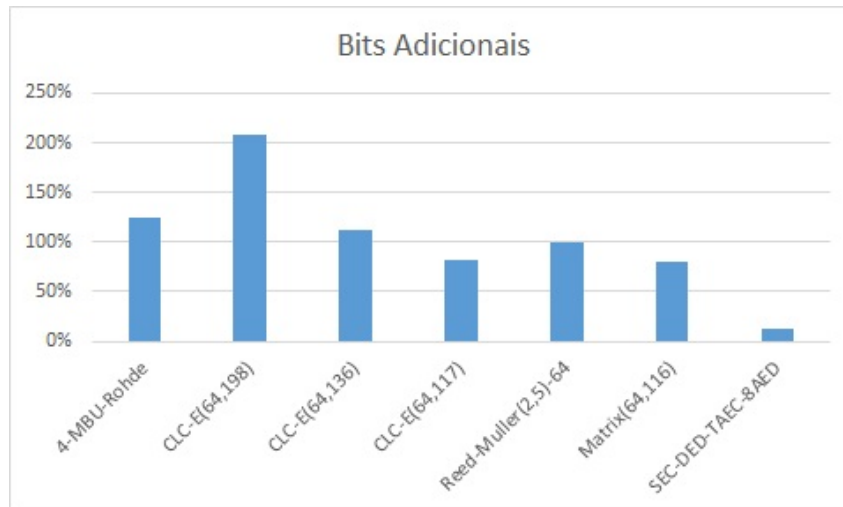
Os bits adicionais que são utilizados para geração da paridade e são armazenados juntamente com a informação podem trazer vários empecilhos, dentre eles a utilização de memória para carga redundante diminuindo o espaço útil para a informação. Essa carga adicional é necessária tendo em vista que aumenta as possibilidades de correção. Vejamos uma comparação da carga adicional entre os algoritmos avaliados na Figura 50.

6.1 OUTRAS SIMULAÇÕES

A fim de obter maiores informações referentes a técnica desenvolvida foram efetuadas outras simulações que abrangem parâmetros diferentes que podem ser necessários dependendo da tecnologia envolvida no microchip, ou seja, dependendo do *design* das células de memória e a litografia do circuito novas configurações de erros podem ser passíveis de verificação.

Uma característica importante que deve ser levada em conta é que dois SEE podem ocorrer, em tempos diferentes, impactando ainda mais o tratamento dos erros. Diante disso, é

Figura 50 – Bits a mais utilizados como paridade



Fonte: AUTOR

necessária uma varredura de tempos em tempos para averiguar possíveis problemas e determinar as soluções cabíveis, conforme já dito é denominada técnica de *scrubbing*. Em LIU et al. (2017) foi verificada que a eficiência do CCE pode ser ampliado; se varreduras na memória ajustando os bits que foram afetados ocorrerem.

O período necessário para a varredura da memória depende da tecnologia envolvida no circuito, conforme já dito, cálculos matemáticos podem ser feitos para determinar o SER do CI o que determinará como e quando essa varredura deverá ser feita.

As simulações descritas a seguir levam em consideração os eventos em tempos distintos considerando que o CCE não efetuou a varredura entre eles, ou seja, o impacto de partículas pode alterar um bit qualquer e quando ocorre outro impacto é estabelecido que este pode ocorrer em qualquer local da memória, desconsiderando a proximidade com o primeiro evento.

Para exemplificação de dois impactos de MBU nível 2 temos a seguinte rotina de testes: É aplicado a primeira configuração de erros da Janela de Erros, temos como exemplo a Figura 51A, em seguida uma nova aplicação de erro ocorre a qual é demonstrada na Figura 51B, então ocorre a verificação. A Figura 51C e 51D são as próximas verificações, assim segue sucessivamente até o fim de todas as possibilidades para a Figura 51A.

Finalizada todas as possibilidades para a Figura 51A outra configuração de erro possível é verificada, o ciclo acaba quando todas as possibilidades de combinações possíveis são averiguadas.

Diante ao exposto na literatura tem-se que a maior probabilidade de ocorrência de SEE

Figura 51 – Teste de 2 MBU de nível 2

		PC1	C24	C5	PC4	D27	D35	HC3	HC4	BC1	BC2	BC3	HPC1	HPC2	HPC3	CC1	BP3	PD1
		D2	C19	C10	D58	D15	D34	HC1	HC2	BC4	BC5	BC6	D3	D16	HPC4	CC2	BP4	PD2
A	C11	D48	C14	C20	D17	D51	D52	D31	D43	D33	D53	D21	D11	D59	CC9	CC3	BP5	PD3
	C16	PC2	C9	C15	PC5	D32	D55	D36	D26	D49	D1	PC9	D24	D39	D29	CC4	BP6	PD4
	C21	D5	C18	C23	D47	D4	D38	D61	D18	D8	C1	D7	D41	D9	PC7	CC5	BP7	PD5
	C2	D30	C7	C12	D6	D62	D19	D42	D54	D22	D25	D37	D60	D14	D56	CC6	BP8	PD6
	C4	D40	C13	C3	D64	D10	D44	PC8	D23	D57	PC11	D45	D13	D50	BP2	CC7	BP9	PD7
	C22	PC3	C17	C8	PC6	D12	D46	PC12	PC13	PC14	PC15	D63	D20	BP1	PC10	CC8	BP10	PD8
		PC1	C24	C5	PC4	D27	D35	HC3	HC4	BC1	BC2	BC3	HPC1	HPC2	HPC3	CC1	BP3	PD1
		D2	C19	C10	D58	D15	D34	HC1	HC2	BC4	BC5	BC6	D3	D16	HPC4	CC2	BP4	PD2
B	C11	D48	C14	C20	D17	D51	D52	D31	D43	D33	D53	D21	D11	D59	CC9	CC3	BP5	PD3
	C16	PC2	C9	C15	PC5	D32	D55	D36	D26	D49	D1	PC9	D24	D39	D29	CC4	BP6	PD4
	C21	D5	C18	C23	D47	D4	D38	D61	D18	D8	C1	D7	D41	D9	PC7	CC5	BP7	PD5
	C2	D30	C7	C12	D6	D62	D19	D42	D54	D22	D25	D37	D60	D14	D56	CC6	BP8	PD6
	C4	D40	C13	C3	D64	D10	D44	PC8	D23	D57	PC11	D45	D13	D50	BP2	CC7	BP9	PD7
	C22	PC3	C17	C8	PC6	D12	D46	PC12	PC13	PC14	PC15	D63	D20	BP1	PC10	CC8	BP10	PD8
		PC1	C24	C5	PC4	D27	D35	HC3	HC4	BC1	BC2	BC3	HPC1	HPC2	HPC3	CC1	BP3	PD1
		D2	C19	C10	D58	D15	D34	HC1	HC2	BC4	BC5	BC6	D3	D16	HPC4	CC2	BP4	PD2
C	C11	D48	C14	C20	D17	D51	D52	D31	D43	D33	D53	D21	D11	D59	CC9	CC3	BP5	PD3
	C16	PC2	C9	C15	PC5	D32	D55	D36	D26	D49	D1	PC9	D24	D39	D29	CC4	BP6	PD4
	C21	D5	C18	C23	D47	D4	D38	D61	D18	D8	C1	D7	D41	D9	PC7	CC5	BP7	PD5
	C2	D30	C7	C12	D6	D62	D19	D42	D54	D22	D25	D37	D60	D14	D56	CC6	BP8	PD6
	C4	D40	C13	C3	D64	D10	D44	PC8	D23	D57	PC11	D45	D13	D50	BP2	CC7	BP9	PD7
	C22	PC3	C17	C8	PC6	D12	D46	PC12	PC13	PC14	PC15	D63	D20	BP1	PC10	CC8	BP10	PD8
		PC1	C24	C5	PC4	D27	D35	HC3	HC4	BC1	BC2	BC3	HPC1	HPC2	HPC3	CC1	BP3	PD1
		D2	C19	C10	D58	D15	D34	HC1	HC2	BC4	BC5	BC6	D3	D16	HPC4	CC2	BP4	PD2
D	C11	D48	C14	C20	D17	D51	D52	D31	D43	D33	D53	D21	D11	D59	CC9	CC3	BP5	PD3
	C16	PC2	C9	C15	PC5	D32	D55	D36	D26	D49	D1	PC9	D24	D39	D29	CC4	BP6	PD4
	C21	D5	C18	C23	D47	D4	D38	D61	D18	D8	C1	D7	D41	D9	PC7	CC5	BP7	PD5
	C2	D30	C7	C12	D6	D62	D19	D42	D54	D22	D25	D37	D60	D14	D56	CC6	BP8	PD6
	C4	D40	C13	C3	D64	D10	D44	PC8	D23	D57	PC11	D45	D13	D50	BP2	CC7	BP9	PD7
	C22	PC3	C17	C8	PC6	D12	D46	PC12	PC13	PC14	PC15	D63	D20	BP1	PC10	CC8	BP10	PD8

Upset

Fonte: AUTOR

é para SBU, em segundo lugar tem-se MBU de nível 2 e assim sucessivamente, perante isso foi realizado as verificações, considerando esses parâmetros. Na sequência o resultado das simulações:

- Dois SBU's possuem 10.296 possibilidades, o algoritmo efetuou a correção de todas elas obtendo 100% de cobertura para 2 *upsets* em tempos distintos.
- Três SBU's possuem 487.344 possibilidades das quais 17.190 não são corrigíveis, ou seja, para três impactos do tipo SBU a técnica possui mais de 96% de cobertura.
- Dois impactos de MBU nível 2 com janela 3x3 possuem 1.838.736 possibilidades, das quais 162.174 configurações não possuem correção, ou seja, para dois impactos que contenham 2 *upsets* com janela de ordem 3 tem aproximadamente 91% de cobertura.
- Dois impactos de MBU, sendo um nível 2 e outro nível 3 com janela 3x3 possuem 6.731.184 possibilidades, das quais 1.095.189 configurações não possuem correção, ou

seja, para dois impactos que contenham 2 e 3 *upsets* respectivamente com janela de ordem 3 tem aproximadamente 83,72% de cobertura.

- Dois impactos de MBU nível 3 com janela 3x3 possuem 24.641.296 possibilidades, das quais 6.710.938 configurações não possuem correção, ou seja, para dois impactos que contenham 3 *upsets* com janela de ordem 3 tem aproximadamente 72,76% de cobertura.
- Dois impactos de MBU, sendo um nível 4 e outro nível 3 com janela 3x3 possuem 47.758.644 possibilidades, das quais 18.371.021 configurações não possuem correção, ou seja, para dois impactos que contenham 4 e 3 *upsets* respectivamente com janela de ordem 3 tem aproximadamente 61,53% de cobertura.

Além disso, foi verificado o impacto que contempla a janela de ordem 3 com um segundo evento, sendo esse segundo evento exclusivamente SBU. Ou seja, a primeira inversão dos bits dentro da matriz contempla as formas já descritas anteriormente na Tabela 1 e na segunda etapa mais 1 bit é alterado.

Essa simulação de multi colisões foi efetuada de forma exaustiva. Assim sendo, foi aplicado uma das configurações de *upsets* da Janela de Erros e após todas as outras possibilidades de SBU, assim sucessivamente. Como exemplo de verificação a Figura 52 expõe uma configuração de 3 *upsets* (destacados em total preto), todas as outras posições vagas, as 141, foram alteradas e verificadas.

Figura 52 – Exemplo de colisão

D25	PC1	C24	C5	PC4	D27	D35	HC3	HC4	BC1	BC2	BC3	HPC1	HPC2	HPC3	CC1	BP3	PD1
C6	D2	C19	C10	D53	D15	D34	HC1	HC2	BC4	BC5	BC6	D3	D16	HPC4	CC2	BP4	PD2
C11	D48	C14	C20		D81	D82	D31	D43	D33	D53	D21	D11	D59	CC9	CC3	BP5	PD3
C16	PC2	C9		PC5	D32	D55	D36	D36	D49	D4	PC9	D24	D39	D29	CC4	BP6	PD4
C21	D5	C18	C23	D47		D38	D61	D18	D8	C1	D7	D41	D9	PC7	CC5	BP7	PD5
C2	D30	C7	C12	D6	D62	D19	D42	D84	D32	D28	D37	D60	D14	D56	CC6	BP8	PD6
C4	D40	C13	C3	D64	D10	D44	PC8	D23	D57	PC11	D45	D13	D50	BP2	CC7	BP9	PD7
C22	PC3	C17	C8	PC6	D12	D46	PC12	PC13	PC14	PC15	D63	D20	BP1	PC10	CC8	BP10	PD8

Fonte: AUTOR

Obtendo como total 5.673.384 verificações, sendo 1.037.021 configurações não solucionáveis, obteve-se 81,72% de cobertura para dois impactos, levando em consideração o primeiro evento como janela de ordem 3 e o segundo sendo um SBU. A Tabela 3 demonstra detalhadamente a quantidade de validações para cada caso e a sua respectiva porcentagem de correção.

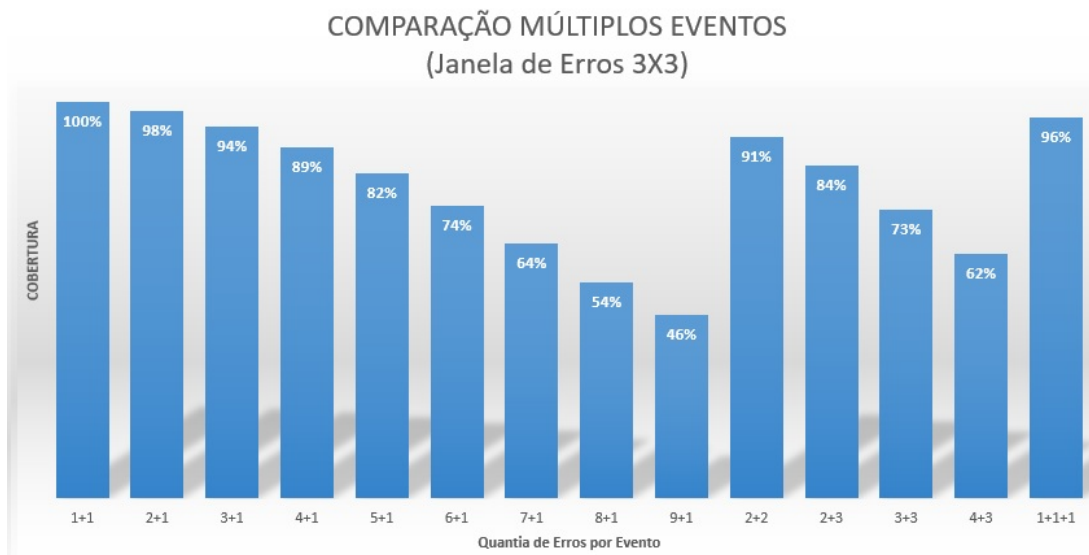
Se a varredura da memória não for efetuada adequadamente múltiplos eventos podem causar perturbações que alteram os bits sem considerar o tamanho do raio que impacta as células, aqui denominado de Janela de Erros. A Figura 53 faz a comparação da correção entre múltiplos eventos, considerando o tamanho da Janela de Erros 3X3 para cada impacto.

Tabela 3 – Correções de erros para dois eventos: MBU + SBU

Nível MBU	Possibilidades de Erros MBU + SBU	Possibilidades Não Corrigíveis	Cobertura
# 1	10.296	0	100%
# 2	195.264	4.618	97,63%
# 3	714.816	44.803	93,73%
# 4	1.385.424	157.793	88,61%
# 5	1.594.944	288.313	81,92%
# 6	1.136.736	299.291	73,67%
# 7	497.664	177.968	64,23%
# 8	124.416	56.792	54,35%
# 9	13.824	7.443	46,15%

Fonte: AUTOR

Figura 53 – Comparação da cobertura em múltiplos eventos



Fonte: AUTOR

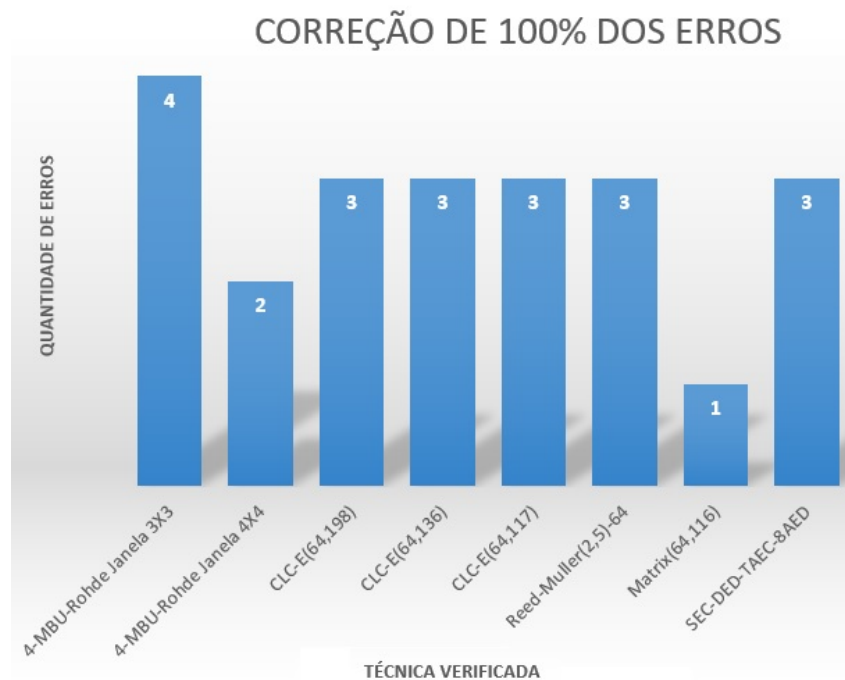
6.2 CONSIDERAÇÕES SOBRE AS SIMULAÇÕES

Na Figura 49 foi possível verificar que a técnica desenvolvida nesta pesquisa, especificamente para Janela de Erros de ordem 4, constatando a quantidade de 3 erros não obteve 100% de correção, demonstrando que outras técnicas possuem maior cobertura de erros para essa especificidade. Entretanto, para todas as outras quantidades de erros, ela demonstrou-se muito mais eficaz.

Sabe-se ainda que a integridade da informação só é garantida quando se obtém 100% de correção para determinada situação, tendo em vista que dependendo do nível de MBU que o circuito é submetido algumas configurações de erros não são solucionadas causando a ilegitimidade da informação. A Figura 54 efetua uma comparação entre as simulações realizadas e as

pesquisas efetuadas, considerando apenas o número de erros que obtém-se 100% de solução.

Figura 54 – Correção de 100% das configurações de erros



Fonte: AUTOR

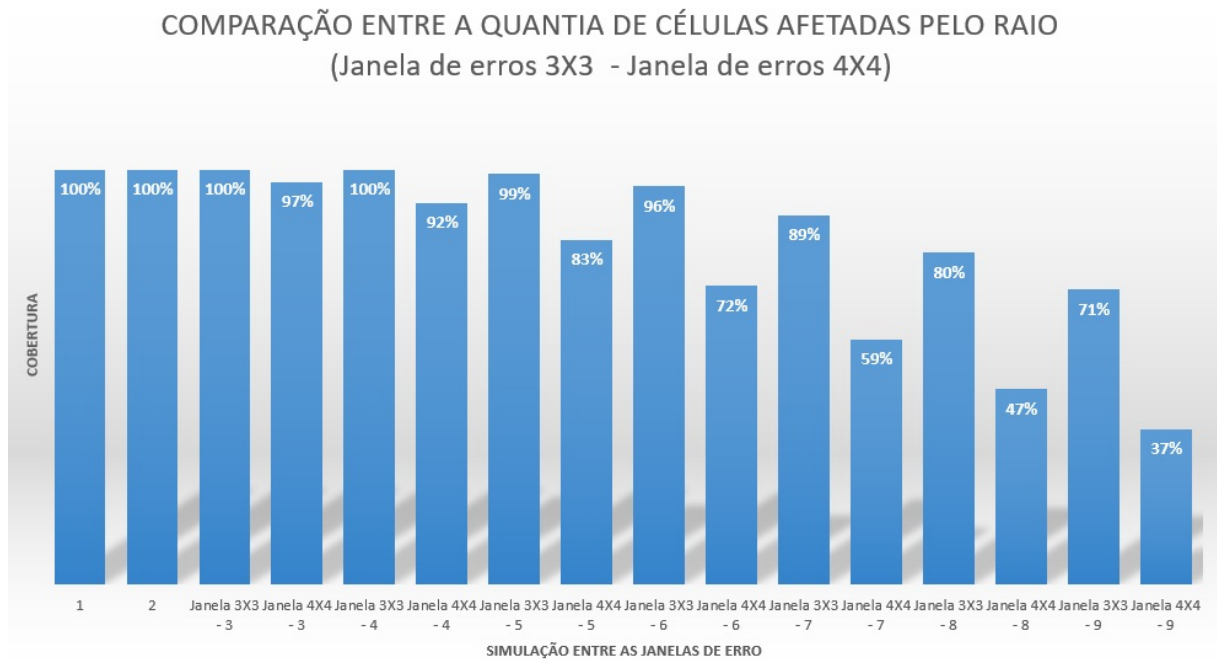
Além disso, a Figura 55 faz uma comparação entre a capacidade de correção do algoritmo, considerando as Janelas de Erros, demonstrando que o raio que afeta os bits impacta diretamente na eficiência do decodificador. Levando em consideração que o *Interleaving* foi alocado de forma manual, verificando o que se julga a melhor alocação para cada bit, é possível que uma verificação de todas as possibilidades de reorganização possam interferir diretamente no índice de correção para Janelas de Erros maiores.

Ao total foi efetuado 3.344.157 de simulações que contemplam apenas um impacto de partículas altamente carregadas das quais 1.481.784 são configurações que não foi possível a correção, considerando apenas as simulações que utilizaram a técnica de *Interleaving*. Diante disso, 55,69% de todas as possibilidades únicas de impacto que foram verificadas possuem cobertura de correção.

Além disso, considerando múltiplos impactos, totalizou 87.140.884 verificações, sendo 27.393.533 configurações não corrigíveis, expondo um percentual de cobertura de erros de aproximadamente 68,56%.

Totalizando 90.485.041 simulações, o decodificador não solucionou 28.875.317 de configurações testadas, expondo um percentual aproximado de 68,08% de cobertura de erros, ou

Figura 55 – comparação tamanho distintos de Janela de Erro



Fonte: AUTOR

seja, em mais de 68% dos casos simulados obteve-se a a devida correção dos erros de *upsets* na memória provindos da radiação cósmica.

Aliás, é nítida a eficiência do algoritmo, mesmo sem a utilização da técnica de *Interleaving* verificado na Figura 49, para múltiplos erros em um único impacto como, por exemplo, para 6 erros apenas uma das técnicas verificadas possui melhor porcentagem de cobertura e mesmo assim a diferença é pouca. Se tratando de mais erros a metodologia proposta se mostra mais eficiente mesmo sem o *Interleaving*. Ou seja, para uma grande quantidade de erros a técnica proposta é mais eficiente mesmo sem reorganizar os bits.

O uso da reorganização dos bits proporcionou um ganho muito significativo tendo em vista que o SEE ocorre em células adjacentes e que quanto mais espalhados os erros pelos conjuntos de bits, sejam de paridade ou dos dados propriamente ditos, melhores são as condições de tratamento do erro pelo CCE. É salientado que dois ou mais erros no mesmo conjunto pode causar a ineficiência da técnica, consequentemente quanto mais distante os bits que pertencem a mesma checagem maiores são os índices de corretibilidade.

Na sequência é apresentada algumas das formas de simulações que foram efetuadas utilizando os métodos já descritos.

6.3 SITUAÇÃO EXEMPLO

Ao ocorrer um SEE dentro da memória este poderá ser em qualquer local. Diante disso existem muitas configurações possíveis para os erros, na sequência são descritos alguns exemplos de possibilidades de *upsets* e quais os passos utilizado pelo decodificador para efetuar o ajuste.

6.3.1 Exemplo 1

Um MBU que afete os dados de acordo com a Figura 56 é verificado, este por sua vez afeta os seguintes bits: D28, PC1, C6 e D2. De acordo com a decodificação, o tratamento do erro ocorrerá da seguinte forma:

Figura 56 – Exemplo 1: MBU na memória utilizando a técnica proposta

		C24	C5	PC4	D27	D35	HPC3	HPC4	BC1	BC2	BC3	HPC1	HPC2	HPC3	CC1	BP3	PD1
		C19	C10	D58	D15	D34	HPC1	HPC2	BC4	BC5	BC6	D3	D16	HPC4	CC2	BP4	PD2
C11	D48	C14	C20	D17	D51	D52	D31	D43	D33	D53	D21	D11	D59	CC9	CC3	BP5	PD3
C16	PC2	C9	C15	PC5	D32	D55	D36	D26	D49	D1	PC9	D24	D39	D29	CC4	BP6	PD4
C21	D5	C18	C23	D47	D4	D38	D61	D18	D8	C1	D7	D41	D9	PC7	CC5	BP7	PD5
C2	D30	C7	C12	D6	D62	D19	D42	D54	D22	D25	D37	D60	D14	D56	CC6	BP8	PD6
C4	D40	C13	C3	D64	D10	D44	PC8	D23	D57	PC11	D45	D13	D50	BP2	CC7	BP9	PD7
C22	PC3	C17	C8	PC6	D12	D46	PC12	PC13	PC14	PC15	D63	D20	BP1	PC10	CC8	BP10	PD8

Fonte: AUTOR

1. Erro ao recalcular o conjunto C, apontando inconsistência pelo BC2 e o PC13.
2. Solicitação de ajuste com o auxílio do CC e BP.
3. Após ajustar o conjunto C é verificado que existe divergência ao recalcular o HPC, apontando erro no HPC1 e HPC2. PC1 é ajustado.
4. Utilizando as regras do cálculo de Hamming, o conjunto C ajusta o D2 da primeira linha e o D28 da terceira linha.
5. Última checagem verifica o conjunto PC e PD para evitar falso positivo.
6. A matriz está com os erros de MBU sanados.

6.3.2 Exemplo 2

Um MBU que afete os dados de acordo com a Figura 57 é verificado, este por sua vez afeta os seguintes bits: BC6, D3, D59 e D24. De acordo com a decodificação, o tratamento do

erro ocorrerá da seguinte forma:

Figura 57 – Exemplo 2: MBU na memória utilizando a técnica proposta

D28	PC1	C24	C5	PC4	D27	D35	HPC3	HPC4	BC1	BC2	BC3	HPC1	HPC2	HPC3	CC1	BP3	PD1	
C6	D2	C19	C10	D58	D15	D34	HPC1	HPC2	BC4	BC5				D16	HPC4	CC2	BP4	PD2
C11	D48	C14	C20	D17	D51	D52	D31	D43	D33	D53	D21	D11			CC9	CC3	BP5	PD3
C16	PC2	C9	C15	PC5	D32	D55	D36	D26	D49	D1	PC9			D39	D29	CC4	BP6	PD4
C21	D5	C18	C23	D47	D4	D38	D61	D18	D8	C1	D7	D41	D9	PC7	CC5	BP7	PD5	
C2	D30	C7	C12	D6	D62	D19	D42	D54	D22	D25	D37	D60	D14	D56	CC6	BP8	PD6	
C4	D40	C13	C3	D64	D10	D44	PC8	D23	D57	PC11	D45	D13	D50	BP2	CC7	BP9	PD7	
C22	PC3	C17	C8	PC6	D12	D46	PC12	PC13	PC14	PC15	D63	D20	BP1	PC10	CC8	BP10	PD8	

Fonte: AUTOR

1. Erro ao recalcular o conjunto C, apontando inconsistência pelo BC6.
2. Solicitação de ajuste com o auxílio do CC e BP.
3. CC e BP estão íntegros, falso positivo detectado. Não altera o conjunto C.
4. O recálculo do HPC está condizente, não há necessidade de alterações.
5. Utilizando as regras do cálculo de Hamming, o conjunto C ajusta o D3 da primeira linha, o D24 da terceira linha e por fim o D59 da sexta linha.
6. Última checagem verifica o conjunto PC e PD para evitar falso positivo.
7. A matriz está com os erros de MBU sanados.

6.3.3 Exemplo 3

Um MBU que afete os dados de acordo com a Figura 58 é verificado, este por sua vez afeta os seguintes bits: D56, CC6 e BP8. De acordo com a decodificação, o tratamento do erro ocorrerá da seguinte forma:

Figura 58 – Exemplo 3: MBU na memória utilizando a técnica proposta

D28	PC1	C24	C5	PC4	D27	D35	HPC3	HPC4	BC1	BC2	BC3	HPC1	HPC2	HPC3	CC1	BP3	PD1
C6	D2	C19	C10	D58	D15	D34	HPC1	HPC2	BC4	BC5	BC6	D3	D16	HPC4	CC2	BP4	PD2
C11	D48	C14	C20	D17	D51	D52	D31	D43	D33	D53	D21	D11	D59	CC9	CC3	BP5	PD3
C16	PC2	C9	C15	PC5	D32	D55	D36	D26	D49	D1	PC9	D24	D39	D29	CC4	BP6	PD4
C21	D5	C18	C23	D47	D4	D38	D61	D18	D8	C1	D7	D41	D9	PC7	CC5	BP7	PD5
C2	D30	C7	C12	D6	D62	D19	D42	D54	D22	D25	D37	D60	D14				PD6
C4	D40	C13	C3	D64	D10	D44	PC8	D23	D57	PC11	D45	D13	D50	BP2	CC7	BP9	PD7
C22	PC3	C17	C8	PC6	D12	D46	PC12	PC13	PC14	PC15	D63	D20	BP1	PC10	CC8	BP10	PD8

Fonte: AUTOR

1. Conjunto C está correto ao ser calculado pelo BC e PC12 a 15.
2. O recálculo do HPC está condizente, não há necessidade de alterações.
3. Utilizando as regras do cálculo de Hamming o conjunto C ajusta o D56 da sexta linha.
4. Última checagem verifica o conjunto PC e PD para evitar falso positivo.
5. A matriz está com os erros de MBU sanados.

6.3.4 Exemplo 4

Um MBU que afete os dados de acordo com a Figura 59 é verificado, este por sua vez afeta os seguintes bits: D34, HC1 e D31. De acordo com a decodificação, o tratamento do erro ocorrerá da seguinte forma:

Figura 59 – Exemplo 4: MBU na memória utilizando a técnica proposta

D28	PC1	C24	C5	PC4	D27	D35	HPC3	HPC4	BC1	BC2	BC3	HPC1	HPC2	HPC3	CC1	BP3	PD1
C6	D2	C19	C10	D58	D15			HPC2	BC4	BC5	BC6	D3	D16	HPC4	CC2	BP4	PD2
C11	D48	C14	C20	D17	D51	D52		D43	D33	D53	D21	D11	D59	CC9	CC3	BP5	PD3
C16	PC2	C9	C15	PC5	D32	D55	D36	D26	D49	D1	PC9	D24	D39	D29	CC4	BP6	PD4
C21	D5	C18	C23	D47	D4	D38	D61	D18	D8	C1	D7	D41	D9	PC7	CC5	BP7	PD5
C2	D30	C7	C12	D6	D62	D19	D42	D54	D22	D25	D37	D60	D14	D56	CC6	BP8	PD6
C4	D40	C13	C3	D64	D10	D44	PC8	D23	D57	PC11	D45	D13	D50	BP2	CC7	BP9	PD7
C22	PC3	C17	C8	PC6	D12	D46	PC12	PC13	PC14	PC15	D63	D20	BP1	PC10	CC8	BP10	PD8

Fonte: AUTOR

1. Conjunto C está correto ao ser calculado pelo BC e PC12 a PC15.
2. É verificado que existe divergência entre os dois conjuntos do HPC. PC1 a PC11 está íntegro.
3. Utilizando as regras do cálculo de Hamming o conjunto C ajusta o D31 da terceira linha e o D34 da quarta linha.
4. Última checagem verifica o conjunto PC e PD para evitar falso positivo.
5. A matriz está com os erros de MBU sanados.

6.3.5 Exemplo 5

Um MBU que afete os dados de acordo com a Figura 60 é verificado, este por sua vez afeta os seguintes bits: D1, PC9 e D11. De acordo com a decodificação, o tratamento do erro ocorrerá da seguinte forma:

Figura 60 – Exemplo 5: MBU na memória utilizando a técnica proposta

D28	PC1	C24	C5	PC4	D27	D35	HPC3	HPC4	BC1	BC2	BC3	HPC1	HPC2	HPC3	CC1	BP3	PD1
C6	D2	C19	C10	D58	D15	D34	HPC1	HPC2	BC4	BC5	BC6	D3	D16	HPC4	CC2	BP4	PD2
C11	D48	C14	C20	D17	D51	D52	D31	D43	D33	D53	D21		D59	CC9	CC3	BP5	PD3
C16	PC2	C9	C15	PC5	D32	D55	D36	D26	D49			D24	D39	D29	CC4	BP6	PD4
C21	D5	C18	C23	D47	D4	D38	D61	D18	D8	C1	D7	D41	D9	PC7	CC5	BP7	PD5
C2	D30	C7	C12	D6	D62	D19	D42	D54	D22	D25	D37	D60	D14	D56	CC6	BP8	PD6
C4	D40	C13	C3	D64	D10	D44	PC8	D23	D57	PC11	D45	D13	D50	BP2	CC7	BP9	PD7
C22	PC3	C17	C8	PC6	D12	D46	PC12	PC13	PC14	PC15	D63	D20	BP1	PC10	CC8	BP10	PD8

Fonte: AUTOR

1. Conjunto C está correto ao ser calculado pelo BC e PC12 a PC15.
2. O recálculo do HPC aponta divergência entre o recalculado e o que estava na memória. HPC1, HPC2 e HPC4 estão diferente, é ajustado o PC9.
3. Utilizando as regras do cálculo de Hamming o conjunto C ajusta o D8 da primeira linha.
4. Última checagem verifica o conjunto PC e PD para evitar falso positivo.
5. Falso positivo detectado desfaz alteração nos dados (D8) e corrige da última linha para a primeira.
6. Última checagem verifica o conjunto PC e PD para evitar falso positivo.
7. Falso positivo detectado novamente, desfaz alteração nos dados (D8). Começa a orientação através do conjunto PC.
8. Faz as alterações na coluna detectada pelo PC em comum acordo com o conjunto C.
9. Última checagem verifica o conjunto PC e PD para evitar falso positivo.
10. A matriz está com os erros de MBU sanados.

6.3.6 Exemplo 6

Um MBU que afete os dados de acordo com a Figura 61 é verificado, este por sua vez afeta os seguintes bits: BP5, BP7 e PD5. De acordo com a decodificação, o tratamento do erro ocorrerá da seguinte forma:

1. Conjunto C está correto ao ser calculado pelo BC e PC12 a PC15.
2. O recálculo do HPC está condizente, não há necessidade de alterações.

Figura 61 – Exemplo 6: MBU na memória utilizando a técnica proposta

D28	PC1	C24	C5	PC4	D27	D35	HPC3	HPC4	BC1	BC2	BC3	HPC1	HPC2	HPC3	CC1	BP3	PD1
C6	D2	C19	C10	D58	D15	D34	HPC1	HPC2	BC4	BC5	BC6	D3	D16	HPC4	CC2	BP4	PD2
C11	D48	C14	C20	D17	D51	D52	D31	D43	D33	D53	D21	D11	D59	CC9	CC3		PD3
C16	PC2	C9	C15	PC5	D32	D55	D36	D26	D49	D1	PC9	D24	D39	D29	CC4	BP6	PD4
C21	D5	C18	C23	D47	D4	D38	D61	D18	D8	C1	D7	D41	D9	PC7	CC5		
C2	D30	C7	C12	D6	D62	D19	D42	D54	D22	D25	D37	D60	D14	D56	CC6	BP8	PD6
C4	D40	C13	C3	D64	D10	D44	PC8	D23	D57	PC11	D45	D13	D50	BP2	CC7	BP9	PD7
C22	PC3	C17	C8	PC6	D12	D46	PC12	PC13	PC14	PC15	D63	D20	BP1	PC10	CC8	BP10	PD8

Fonte: AUTOR

3. Utilizando as regras do cálculo de Hamming, o conjunto C verifica que está tudo condizente.
4. Última checagem verifica apenas o conjunto PC, pois C está correto.
5. A matriz está com os erros de MBU sanados.

7 CONCLUSÃO

A necessidade da utilização de satélites está cada vez mais intensa, desde pesquisas até aplicações militares. O uso de nanosatélites do tipo CubeSat permitiu que diversas entidades pudessem se inserir nas pesquisas no ramo espacial. Com o avanço dessa tecnologia, a miniaturização dos circuitos permite maiores ganhos de performance e economia de energia, todavia esse ganho propicia alguns problemas, principalmente no ambiente espacial devido ao ambiente hostil e radioativo.

O impacto de partículas energizadas provindo da radiação cósmica nos circuitos pode gerar distúrbios elétricos, criando assim o *upset* nos bits armazenados na memória, consequentemente a informação é alterada, necessitando uma forma de corrigir esses erros. Ao longo desta pesquisa compreendemos um pouco sobre os Códigos Corretores de Erros e suas especificidades os quais tentam garantir a integridade da informações devido aos erros oriundos de agentes externos a aplicação.

Uma forma diferenciada para averiguação dos erros foi testada, com a utilização de vetores de peso permitiram um segundo nível de checagem, gerando maiores ganhos na utilização dos CCE. A dupla verificação permite uma melhor performance devido ao fato de que muitos falsos positivos são descartados previamente, antes do tratamento dos bits que compõe o conjunto dos dados.

Os resultados obtidos através das simulações desenvolvidas no MATLAB demonstraram-se promissores, a comparação com as diferentes técnicas já existentes permitiu demonstrar os ganhos possíveis com uma pré verificação dos bits de checagem. Além disso, diversos parâmetros de testes foram verificados, permitindo uma melhor compreensão como um todo.

Tendo em vista que a maior probabilidade é de uma única célula afetada, as simulações demonstraram que para múltiplos impactos o algoritmo consegue manter um bom nível de cobertura de erros, principalmente para dois impactos do tipo SBU onde consegue correção total dos erros e para três impactos de SBU a correção mantém-se acima de 96%. Isto aumenta a eficácia para casos onde a técnica de Memory scrubbing não consegue fazer a varredura em curtos períodos de tempos.

Além disso, a Janela de Erro 3x3 contemplou total correção até 4 erros de *upset*, mantendo um nível de correção maior que 97% para todas as possibilidades dentro desse raio de atuação. Em um total de 90.485.041 de simulações foi obtido aproximadamente 68% de cor-

reções dentre as quais foram verificados impactos únicos e múltiplos impactos. É salientado ainda que a comparação dos resultados utilizando a técnica de *Interleaving* e sem ela são expressivos, demonstrando que a sua utilização é de extrema importância para a correção desse tipo de problema.

7.1 TRABALHOS FUTUROS

A reorganização dos bits dado pela técnica de *Interleaving* foi desenvolvida de forma manual, testando as posições estratégicas, levando em consideração as regras de distanciamento, até que foi encontrado o modelo disposto na Figura 36 que obteve um bom resultado. Diante disso, uma verificação da alocação diferente dos bits pode melhorar o desempenho do algoritmo, para tal deve ser levado em consideração que existem $144!$ possibilidades, o que demandará um alto custo computacional se testadas todas as combinações. Técnicas de Inteligência Artificial podem ser utilizadas para mitigar as possíveis combinações. Além do mais, em alguns casos como, por exemplo, a troca do bit HPC1 pelo HPC2 não será obtido nenhuma mudança no decodificador, sabendo que esses dois bits são tratados da mesma forma o que demandará tempo para verificar combinações de alocação que obtém os mesmos resultados.

A implementação física em um FPGA em comparação com a produção de um ASIC também é um objetivo considerado para um estudo futuro tendo em vista que o algoritmo pode se comportar diferentemente nesses dois tipos de circuitos no quesito desempenho e eficiência energética. A análise da implementação física de desempenho e mitigação de erros do tipo SE e HE poderão demonstrar demandas ainda não verificadas no CCE desenvolvido nessa pesquisa.

Outro ponto muito importante é que quanto mais distribuído os erros entre os conjuntos de bits, melhor a performance de correção dos *upsets*. A utilização de dois conjuntos de dados pode melhorar a possibilidade de correção e até manter o mesmo índice de correção para Janelas de Erros maiores. Por exemplo, dois conjuntos de 64 bits codificados com a técnica proposta resultam em dois conjuntos de 144 bits. Se alocar os dados utilizando o *Interleaving* a matriz de 288 bits pode proporcionar distâncias grandes entre bits sensíveis, permitindo maior exatidão na correção.

REFERÊNCIAS

- AGUIRRE, F. R. **Estudo sobre distribuição de cargas em semicondutores sujeitos a radiação ionizante**. 2017. Tese (Doutorado em Ciência da Computação) — Universidade de São Paulo.
- ALEXANDRESCU, D. A comprehensive soft error analysis methodology for SoCs/ASICs memory instances. In: IEEE 17TH INTERNATIONAL ON-LINE TESTING SYMPOSIUM, 2011. **Anais...** [S.l.: s.n.], 2011. p.175–176.
- ALVES, A. C. R. **Controle de bordo para nanossatélites: desenvolvimento e aplicação ao projeto conasat**. 2019. Mestrado em Engenharia Mecatrônica — Universidade Federal do Rio Grande do Norte - Brasil.
- AMAGASAKI, M. et al. An area compact soft error resident circuit for FPGA. **2016 International Conference on IC Design and Technology (ICICDT)**, [S.l.], p.1–4, June 2016.
- ARAGÃO, A. C. d. O. S. **Uma arquitetura sistólica para solução de sistemas lineares implementada com circuitos FPGAs**. 1998. Tese (Doutorado em Ciência da Computação) — Universidade de São Paulo.
- ARAÚJO, J. D. B. d. **Protótipo de rastreador solar de um eixo baseado em microcontrolador**. 2015. Dissertação (Mestrado em Ciência da Computação) — Universidade Federal do Rio Grande do Norte.
- ARGYRIDES, C.; ZARANDI, H. R.; PRADHAN, D. K. Matrix Codes: multiple bit upsets tolerant method for sram memories. In: IEEE INTERNATIONAL SYMPOSIUM ON DEFECT AND FAULT-TOLERANCE IN VLSI SYSTEMS (DFT 2007), 22. **Anais...** [S.l.: s.n.], 2007. p.340–348.
- BOUWMEESTER, J.; AALBERS, G.; UBBELS, W. Preliminary mission results and project evaluation of the delfi-c3 nano-satellite. In: S SYMPOSIUM, 4. **Proceedings...** [S.l.: s.n.], 2008. n.1.
- BRAGGIO, A. A. **FPGA: processo de configuração com desenvolvimento de protótipo em placa de circuito impresso**. 2014. B.S. thesis — Universidade Tecnológica Federal do Paraná.

CHORASIA, J.; JASANI, K.; SHAH, A. Realization of various error mitigation techniques for SRAM based FPGA. In: INTERNATIONAL CONFERENCE FOR CONVERGENCE IN TECHNOLOGY (I2CT), 2017. **Anais...** [S.l.: s.n.], 2017. p.55–59.

DODD, P. E.; MASSENGILL, L. W. Basic mechanisms and modeling of single-event upset in digital microelectronics. **IEEE Transactions on Nuclear Science**, [S.l.], v.50, n.3, p.583–602, June 2003.

DUMITRIU, V.; KIRISCHIAN, L.; KIRISCHIAN, V. Decentralized run-time recovery mechanism for transient and permanent hardware faults for space-borne FPGA-based computing systems. **2014 NASA/ESA Conference on Adaptive Hardware and Systems (AHS)**, [S.l.], p.47–54, July 2014.

ESMAEELI, S. et al. A multi-bit error tolerant register file for a high reliable embedded processor. In: IEEE INTERNATIONAL CONFERENCE ON ELECTRONICS, CIRCUITS, AND SYSTEMS, 2011. **Anais...** [S.l.: s.n.], 2011. p.532–537.

FORTES, J. M. P. **Efeito da modelagem probabilística de parâmetros na análise da interferência produzida por múltiplos satélites geostacionários em receptores do Serviço Fixo Terrestre**. 2016. Tese (Doutorado em Ciência da Computação) — PUC-Rio.

Georgakos, G. et al. Investigation of Increased Multi-Bit Failure Rate Due to Neutron Induced SEU in Advanced Embedded SRAMs. In: IEEE SYMPOSIUM ON VLSI CIRCUITS, 2007. **Anais...** [S.l.: s.n.], 2007. p.80–81.

HARADA, R. et al. Angular Dependency of Neutron-Induced Multiple Cell Upsets in 65-nm 10T Subthreshold SRAM. **IEEE Transactions on Nuclear Science**, [S.l.], v.59, n.6, p.2791–2795, 2012.

IBE, E. et al. Impact of Scaling on Neutron-Induced Soft Error in SRAMs From a 250 nm to a 22 nm Design Rule. **IEEE Transactions on Electron Devices**, [S.l.], v.57, n.7, p.1527–1538, July 2010.

JUNIOR, A. J. A. T. **Desenvolvimento e Simulação de um Computador de Bordo para o Veículo Lançador de Microsatélites (VLM)**. 2019. Mestrado em de Computação e Sistemas — UNIVERSIDADE ESTADUAL DO MARANHÃO - Brasil.

- KATO, T. et al. Neutron-Induced Multiple-Cell Upsets in 20-nm Bulk SRAM: angular sensitivity and impact of multiwell potential perturbation. **IEEE Transactions on Nuclear Science**, [S.l.], v.66, n.7, p.1381–1389, 2019.
- LIU, R. et al. Investigation of Single-Bit and Multiple-Bit Upsets in Oxide RRAM-Based 1T1R and Crossbar Memory Arrays. **IEEE Transactions on Nuclear Science**, [S.l.], v.62, n.5, p.2294–2301, Oct 2015.
- LIU, S. et al. Reliability analysis of memories suffering MBUs for the effect of negative bias temperature instability. In: ASIA AND SOUTH PACIFIC DESIGN AUTOMATION CONFERENCE (ASP-DAC), 2017. **Anais...** [S.l.: s.n.], 2017. p.87–92.
- LYONS, R. E.; VANDERKULK, W. The Use of Triple-Modular Redundancy to Improve Computer Reliability. **IBM Journal of Research and Development**, [S.l.], v.6, n.2, p.200–209, 1962.
- MACHADO, D. A. **Uma abordagem de dígitos verificadores e códigos corretores no ensino fundamental**. 2016. Tese (Doutorado em Ciência da Computação) — Universidade de São Paulo.
- MAHESHWARI, A.; KOREN, I.; BURLESON, W. Accurate estimation of soft error rate (SER) in VLSI circuits. In: IEEE INTERNATIONAL SYMPOSIUM ON DEFECT AND FAULT TOLERANCE IN VLSI SYSTEMS, 2004. DFT 2004. PROCEEDINGS., 19. **Anais...** [S.l.: s.n.], 2004. p.377–385.
- MARIANI, R.; BOSCHI, G. Scrubbing and partitioning for protection of memory systems. In: IEEE INTERNATIONAL ON-LINE TESTING SYMPOSIUM, 11. **Anais...** [S.l.: s.n.], 2005. p.195–196.
- MAVIS, D. G. et al. Multiple Bit Upsets and Error Mitigation in Ultra-Deep Submicron SRAMS. **IEEE Transactions on Nuclear Science**, [S.l.], v.55, n.6, p.3288–3294, Dec 2008.
- MAXFIELD, C. **The design warrior's guide to FPGAs: devices, tools and flows**. [S.l.]: Elsevier, 2004.
- MILIES, C. P. Breve introduçaoa Teoria dos Códigos Corretores de Erros. **Colóquio de Matemática da Região Centro-Oeste**, [S.l.], 2009.

MONTEALEGRE, N. et al. In-flight reconfigurable FPGA-based space systems. In: NASA/ESA CONFERENCE ON ADAPTIVE HARDWARE AND SYSTEMS (AHS), 2015. **Anais...** [S.l.: s.n.], 2015. p.1–8.

MOREIRA, J. C.; FARRELL, P. G. **Essentials of error-control coding**. [S.l.]: John Wiley & Sons, 2006.

MOTA, D. F. M. **OPENOBC**: uma arquitetura de um computador de bordo open source e de baixo custo para o padrao cubesat. 2017. Mestrado em Engenharia de Teleinformática — Universidade Federal do Ceará.

NEALE, A.; SACHDEV, M. A 0.4 V 75 kbit SRAM macro in 28 nm CMOS featuring a 3-adjacent MBU correcting ECC. In: IEEE 2014 CUSTOM INTEGRATED CIRCUITS CONFERENCE. **Proceedings...** [S.l.: s.n.], 2014. p.1–4.

NEUBAUS, A.; FREUDENBERGES, J.; KUHN, V. **Coding theory, algorithms, architectures and application**. [S.l.]: Wiley & sons inc, 2007.

OLIVEIRA MIRANDA, B. C. de. Modelagem e Validação de um Sistema de Determinação de Atitude com Tolerância a Falhas para o NanosatC-Br2. , [S.l.], 2016.

Osada, K. et al. SRAM immunity to cosmic-ray-induced multierrors based on analysis of an induced parasitic bipolar effect. **IEEE Journal of Solid-State Circuits**, [S.l.], v.39, n.5, p.827–833, 2004.

Osman, D. A. M.; Mohamed, S. W. A. Hardware and software design of Onboard Computer of ISRASAT1 CubeSat. In: INTERNATIONAL CONFERENCE ON COMMUNICATION, CONTROL, COMPUTING AND ELECTRONICS ENGINEERING (ICCCCEE), 2017. **Anais...** [S.l.: s.n.], 2017. p.1–4.

PIOVESAN, T. **Otimização de fontes de alimentação de alto rendimento para nano satélites cubesat 1U**. 2017. Mestrado em Engenharia Elétrica — UNIVERSIDADE FEDERAL DE SANTA MARIA - Brasil.

PROCHNOW, S. L.; DURÃO, O. S. C.; SCHUCH, N. J. Miniaturização De Satélites. **Centro Regional Sul de Pesquisas Espaciais-CRSPE/INPE-MCT**, [S.l.], p.p10, 2006.

RADAELLI, D. et al. Investigation of multi-bit upsets in a 150 nm technology SRAM device. **IEEE Transactions on Nuclear Science**, [S.l.], v.52, n.6, p.2433–2437, Dec 2005.

RAO, P. M. B. et al. Protecting SRAM-based FPGAs against multiple bit upsets using erasure codes. In: ACM/EDAC/IEEE DESIGN AUTOMATION CONFERENCE (DAC), 2014. **Anais...** [S.l.: s.n.], 2014. p.1–6.

RAVI, P. S.; PARGUNARAJAN, K. Adaptive threshold multi bit flipping for low density parity check codes. In: INTERNATIONAL CONFERENCE ON ADVANCES IN COMPUTING, COMMUNICATIONS AND INFORMATICS (ICACCI), 2017. **Anais...** [S.l.: s.n.], 2017. p.294–298.

REVIRIEGO, P.; MAESTRO, J. A. Study of the Effects of Multibit Error Correction Codes on the Reliability of Memories in the Presence of MBUs. **IEEE Transactions on Device and Materials Reliability**, [S.l.], v.9, n.1, p.31–39, March 2009.

S. CASTRO, H. d. et al. A correction code for multiple cells upsets in memory devices for space applications. In: IEEE INTERNATIONAL NEW CIRCUITS AND SYSTEMS CONFERENCE (NEWCAS), 2016. **Anais...** [S.l.: s.n.], 2016. p.1–4.

Sanchez-Macian, A.; Reviriego, P.; Maestro, J. A. Enhanced Detection of Double and Triple Adjacent Errors in Hamming Codes Through Selective Bit Placement. **IEEE Transactions on Device and Materials Reliability**, [S.l.], v.12, n.2, p.357–362, 2012.

SANTOS, A. F. d. Análise do uso de redundância em circuitos gerados por síntese de alto nível para FPGA programado por SRAM sob falhas transientes. , [S.l.], 2017.

SIDDIQUI, K. S.; BAIG, M. A. FRAM based TMR (triple modular redundancy) for fault tolerance implementation. In: INTERNATIONAL CONFERENCE ON INFORMATION AND COMMUNICATION TECHNOLOGIES, 2011. **Anais...** [S.l.: s.n.], 2011. p.1–5.

SILVA, F. G. A. E. Um código extensível para correção de Multiple Bit Upsets em memórias. **DETE - Dissertações defendidas na UFC**, [S.l.], 2018.

STASSINOPOULOS, E. G.; RAYMOND, J. P. The space radiation environment for electronics. **Proceedings of the IEEE**, [S.l.], v.76, n.11, p.1423–1442, Nov 1988.

- STODDARD, A. et al. A Hybrid Approach to FPGA Configuration Scrubbing. **IEEE Transactions on Nuclear Science**, [S.l.], v.64, n.1, p.497–503, 2017.
- TAVARES, Y. A. **Projeto e Implementação de um Field-Programmable Gate Array (FPGA)**. 2018. B.S. thesis — Universidade Federal do Rio Grande do Norte.
- TIPTON, A. D. et al. Device-Orientation Effects on Multiple-Bit Upset in 65 nm SRAMs. **IEEE Transactions on Nuclear Science**, [S.l.], v.55, n.6, p.2880–2885, 2008.
- TIPTON, A. D. et al. Increased Rate of Multiple-Bit Upset From Neutrons at Large Angles of Incidence. **IEEE Transactions on Device and Materials Reliability**, [S.l.], v.8, n.3, p.565–570, 2008.
- TONFAT, J. et al. Analyzing the influence of the angles of incidence on SEU and MBU events induced by low LET heavy ions in a 28-nm SRAM-based FPGA. In: EUROPEAN CONFERENCE ON RADIATION AND ITS EFFECTS ON COMPONENTS AND SYSTEMS (RADECS), 2016. **Anais...** [S.l.: s.n.], 2016. p.1–6.
- TONFAT, J. et al. Analyzing the Influence of the Angles of Incidence and Rotation on MBU Events Induced by Low LET Heavy Ions in a 28-nm SRAM-Based FPGA. **IEEE Transactions on Nuclear Science**, [S.l.], v.64, n.8, p.2161–2168, 2017.
- TORRES, F. E. Desenvolvimento de um sistema de emulação de single event upsets em dispositivos cots baseado na metodologia code emulating upsets. **Biblioteca Digital UFMG**, [S.l.], 2013.
- TURER, I.; AYDIN, K. Electromagnetic shielding properties of satellites. In: INTERNATIONAL CONFERENCE ON RECENT ADVANCES IN SPACE TECHNOLOGIES (RAST), 2015. **Anais...** [S.l.: s.n.], 2015. p.401–404.
- WALLMARK, J. T.; MARCUS, S. M. Minimum Size and Maximum Packing Density of Nonredundant Semiconductor Devices. **Proceedings of the IRE**, [S.l.], v.50, n.3, p.286–298, March 1962.
- XIAO, L. et al. Hardened design based on advanced orthogonal Latin code against two adjacent multiple bit upsets (MBUs) in memories. In: SIXTEENTH INTERNATIONAL SYMPOSIUM ON QUALITY ELECTRONIC DESIGN. **Anais...** [S.l.: s.n.], 2015. p.485–489.

YOSHIMOTO, S. et al. Multiple-bit-upset and single-bit-upset resilient 8T SRAM bitcell layout with divided wordline structure. In: IEEE 17TH INTERNATIONAL ON-LINE TESTING SYMPOSIUM, 2011. **Anais...** [S.l.: s.n.], 2011. p.151–156.

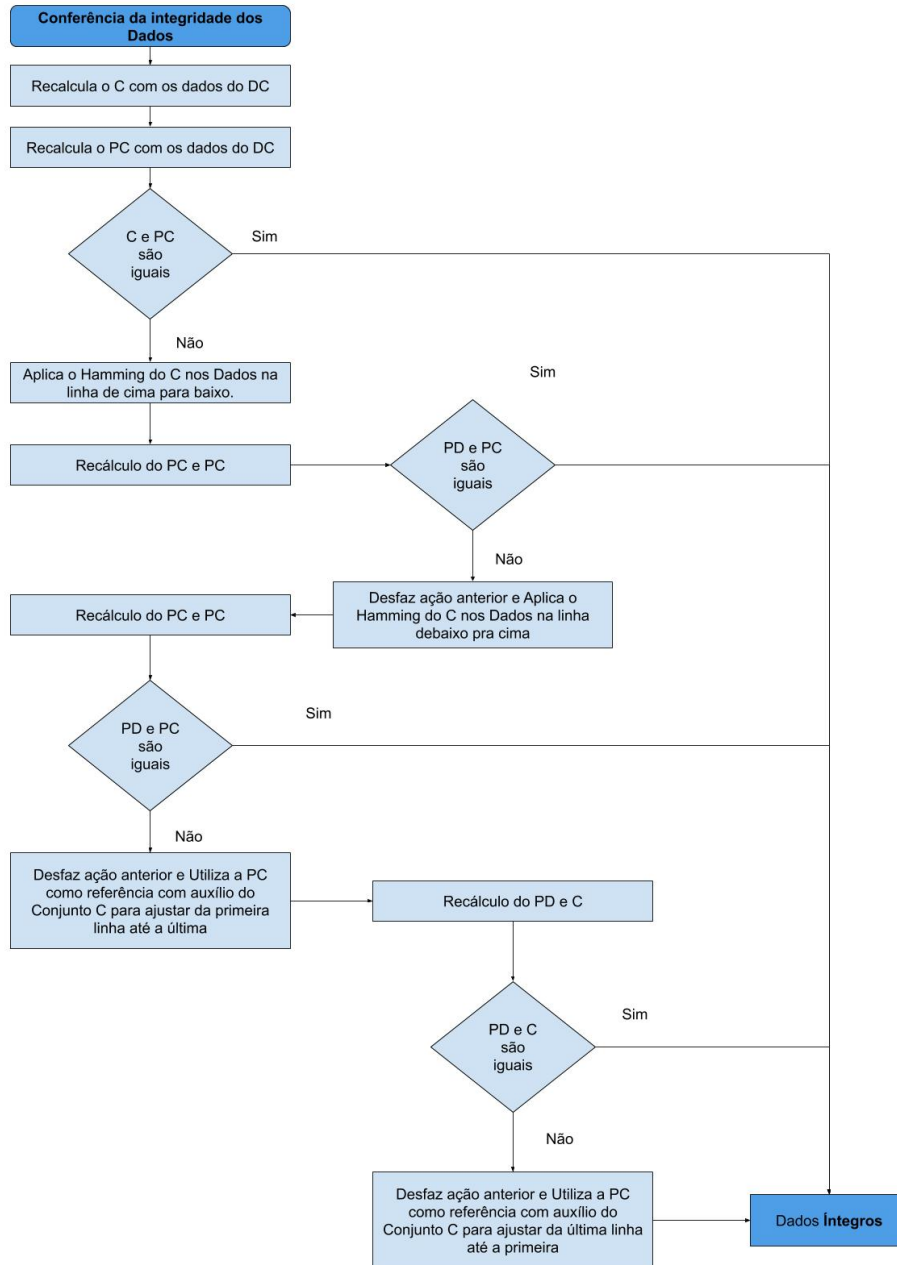
YOSHIMOTO, S. et al. Neutron-induced soft error rate estimation for SRAM using PHITS. In: IEEE 18TH INTERNATIONAL ON-LINE TESTING SYMPOSIUM (IOLTS), 2012. **Anais...** [S.l.: s.n.], 2012. p.138–141.

Zhang, F. et al. Multi-bit Upset Mitigation with Double Matrix Codes in Memories for Space Applications. In: IEEE INTERNATIONAL CONFERENCE ON UNMANNED SYSTEMS AND ARTIFICIAL INTELLIGENCE (ICUSAI), 2019. **Anais...** [S.l.: s.n.], 2019. p.146–149.

APÊNDICES

APÊNDICE A – Fluxograma para o tratamento de erros nos Dados

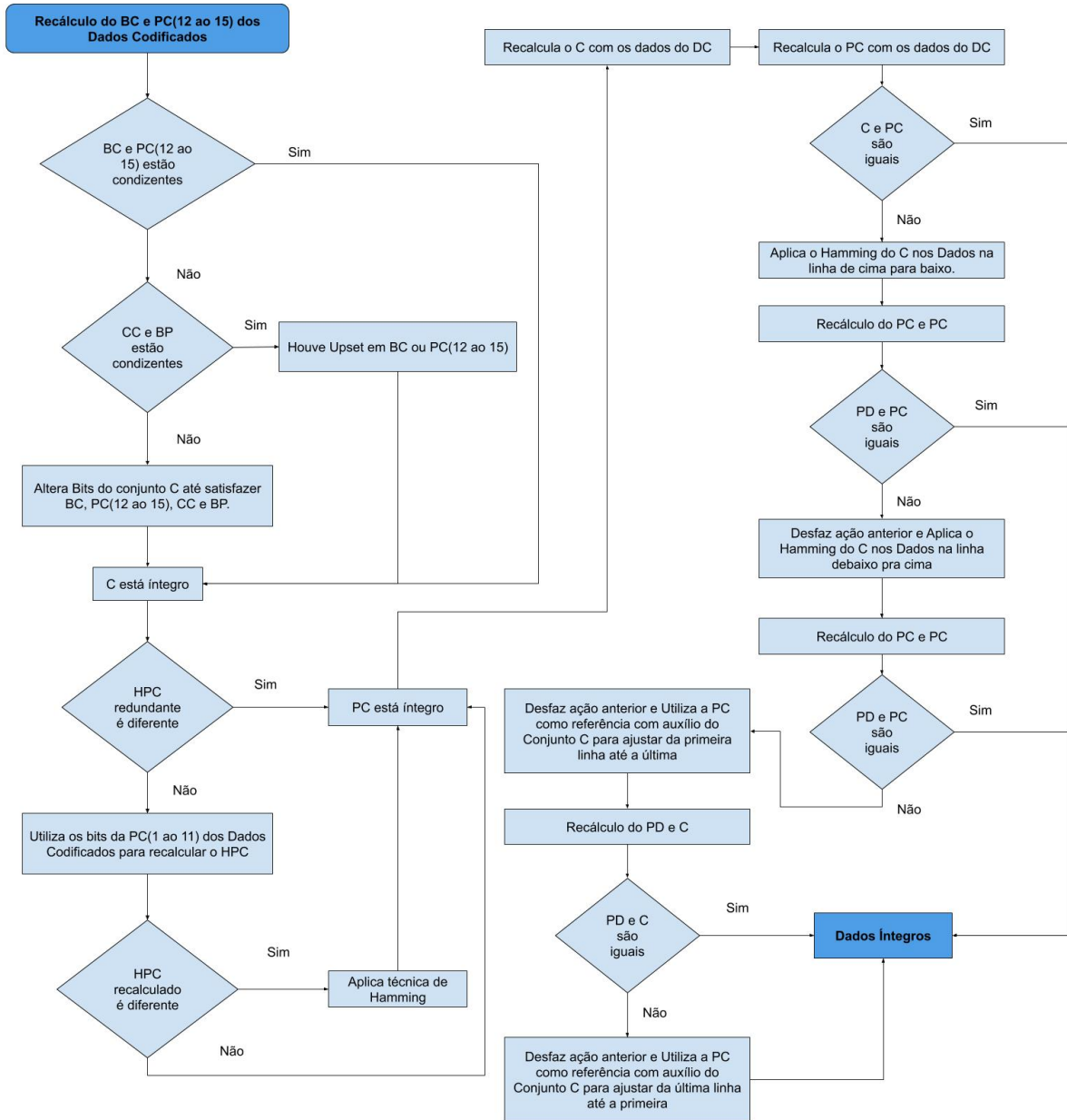
Figura 62 – Integridade dos Dados.



Fonte: AUTOR

APÊNDICE B – Fluxograma Geral para o tratamento de erros

Figura 63 – Fluxo geral para o tratamento de MBU



Fonte: AUTOR