UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
CENTRO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO

Redundância Modular Dupla Baseada em Paridade para Transmissão de Dados em Sistemas de Processamento de Bordo em Nanossatélites

Alex Carlos Rodrigues Alves

Orientador: Prof. Dr. Luiz Felipe de Queiroz Silveira

Tese de Doutorado apresentada ao Programa de Pós-Graduação em Engenharia Elétrica e de Computação da UFRN (área de concentração: Engenharia de Computação) como parte dos requisitos para obtenção do título de Doutor em Ciências.

Número de ordem PPgEEC: D374
Natal, RN, dezembro de 2024

Universidade Federal do Rio Grande do Norte - UFRN
Sistema de Bibliotecas - SISBI
Catalogação de Publicação na Fonte. UFRN - Biblioteca Central Zila Mamede

Alves, Alex Carlos Rodrigues.
Redundância modular dupla baseada em paridade para transmissão de dados em sistemas de processamento de bordo em Nanossatélites / Alex Carlos Rodrigues Alves. - 2024.
132f.: il.

Tese (doutorado) - Universidade Federal do Rio Grande do Norte, Centro de Tecnologia, Programa de Pós-Graduação em Engenharia Elétrica e de Computação, Natal, 2024.
Orientação: Prof. Dr. Luiz Felipe de Queiroz Silveira.

1. DMR - Tese. 2. Paridade - Tese. 3. TMR - Tese. 4. Nanossatélites - Tese. 5. Transmissão de Dados - Tese. 6. Tolerância a Falhas - Tese. I. Silveira, Luiz Felipe de Queiroz. II. Título.

RN/UF/BCZM CDU 004

Elaborado por Jackeline dos Santos Pinheiro da Silva Maia Cavalcanti - CRB-15/317

Agradecimentos

Aos meus pais, Paulo Alves e Maria Miriam (*in memorian*), e ao meu irmão, Paulo Roberto, por todo amor e incentivo, por serem minha base.

À minha esposa, Angelica, por todo carinho e atenção, por me incentivar durante toda essa caminhada e por me apoiar nos momentos de incerteza.

A toda minha família e amigos pelo apoio durante esta trajetória.

Ao professor Márcio pela disponibilidade e por proporcionar a plataforma para os testes realizados durante o desenvolvimento da tese.

Ao meu orientador, professor Luiz Felipe, por toda orientação durante minha jornada no doutorado, pela paciência e pela atenção.

Ao professor Samaherni, coorientador desta tese, por ter me recebido na UFRN ainda no mestrado e por toda colaboração em minha formação acadêmica.

À CAPES, pelo apoio financeiro.

"Life comes from within your heart and desire..."

Pearl Jam

Resumo

A crescente demanda por capacidade de processamento em sistemas embarcados para nanossatélites tornou comum a utilização de SoCs (*Systems-on-Chip*) comerciais de prateleira (COTS, *Commercial-Off-The-Shelf*), os quais são compostos por unidade de processamento *hard-core* (CPU, *Central Processing Unit*) e lógica reconfigurável (FPGA, *Field-Programmable Gate Arrays*) integradas em um mesmo chip. Um ponto de atenção nesses SoCs diz respeito às interfaces de comunicação entre CPU e FPGA, as quais são implementadas na área de lógica reconfigurável e podem sofrer com erros ocasionados pela radiação em ambiente espacial. Diferentes tipos de redundância podem ser empregados para mitigar os efeitos da radiação em barramentos de comunicação *on-chip*, destacando-se a redundância da informação e a redundância de *hardware*. Contudo, apesar de apresentarem considerável capacidade de correção, códigos de redundância de informação modernos podem apresentar uma alta complexidade lógica. Ademais, técnicas de redundância de *hardware* como a TMR (*Triple Modular Redundancy*) resultam em um aumento da área ocupada pelo sistema e um *overhead* de energia. Essas características podem impactar o desenvolvimento de sistemas de nanossatélites, os quais possuem restrições de massa, potência, peso e custo. Nesse contexto, neste trabalho é apresentada uma abordagem de Redundância Modular Dupla (DMR, *Dual Modular Redundancy*) baseada em paridade para aplicação em interfaces de comunicação de SoCs COTS, objetivando o aumento da confiabilidade da transmissão de dados. Para isso, buscou-se propor uma solução com baixa complexidade lógica e com um menor consumo de área e de energia quando comparada com a TMR. Diferentes versões da técnica foram desenvolvidas tomando como base bits de paridade e a DMR. Para cada uma dessas versões, foram desenvolvidas expressões matemáticas para as probabilidades de correção e de erro, bem como à probabilidade de detecção para duas das versões. A análise matemática foi validada por meio da comparação com resultados de simulações em *scripts* Python, levando-se em consideração diferentes taxas de erros. Além disso, a técnica proposta foi comparada com a TMR em termos de probabilidades de correção. Os resultados mostram que, para taxas específicas, a abordagem proposta possui valores aproximados aos da TMR e que, mesmo com a retransmissão de dados com possíveis erros, o número total de bits transmitidos é menor. Como forma de verificar a área ocupada e o consumo de energia, foram realizadas implementações em *hardware* no SoC Xilinx Zynq-7000. Tais implementações evidenciam uma menor utilização de recursos de *hardware* e um menor consumo energético em comparação com a TMR.

Palavras-chave: DMR, Paridade, TMR, Nanossatélites, Transmissão de Dados, Tolerância a Falhas.

Abstract

The growing demand for processing capacity in embedded systems for nanosatellites has made it common to use Commercial-Off-The-Shelf (COTS) SoCs (Systems-on-Chip), which are composed of a hard-core processing unit (CPU, Central Processing Unit) and reconfigurable logic (FPGA, Field-Programmable Gate Arrays) integrated on the same chip. One point of attention in these SoCs concerns the communication interfaces between CPU and FPGA, which are implemented in the reconfigurable logic area and may suffer from errors caused by radiation in a space environment. Different types of redundancy can be employed to mitigate the effects of radiation in on-chip communication buses, highlighting information redundancy and hardware redundancy. However, despite presenting considerable correction capacity, modern information redundancy codes can present high logical complexity. Furthermore, hardware redundancy techniques such as TMR (Triple Modular Redundancy) increase the system's area and the energy overhead. These characteristics may impact the development of nanosatellite systems, which have mass, power, weight, and cost constraints. In this context, this work presents a parity-based Dual Modular Redundancy (DMR) approach for application in COTS SoC communication interfaces, aiming at increasing data transmission reliability. To this end, it was sought to propose a solution with low logical complexity and lower area and energy consumption compared to TMR. Different versions of the technique were developed based on parity bits and DMR. For each version, mathematical expressions for the correction and error probabilities were developed, as well as the detection probability for two of the presented versions. The mathematical analysis was validated by comparisons with the results of simulations in Python scripts, considering different error rates. Furthermore, the proposed technique was compared with TMR in terms of correction probabilities. The results show that, for specific rates, the proposed approach has values close to those of TMR and that, even with the retransmission of data with possible errors, the total number of transmitted bits is lower. In order to verify the occupied area and the energy consumption, hardware implementations were performed on the Xilinx Zynq-7000 SoC. Such implementations demonstrate a lower utilization of hardware resources and energy consumption than TMR.

Keywords: DMR, Parity, TMR, Nanosatellites, Data Transmission, Fault Tolerance.

Sumário

Sumário i

Lista de Figuras iii

Lista de Tabelas v

Lista de Símbolos e Abreviaturas vii

1 Introdução 1
- 1.1 Motivação . . . 4
- 1.2 Objetivos . . . 5
- 1.3 Contribuições . . . 5
- 1.4 Organização do Texto . . . 6

2 Nanossatélites e a Radiação em Sistemas Espaciais 7
- 2.1 CubeSats . . . 7
- 2.2 Efeitos da Radiação em Componentes Eletrônicos . . . 9
- 2.3 Tolerância a Falhas . . . 11
- 2.4 Processamento Híbrido em Sistemas Espaciais . . . 14
- 2.5 AMBA AXI . . . 18
- 2.6 Resumo . . . 19

3 Estado da Arte 20
- 3.1 Resumo . . . 23

4 PDMR: Uma Abordagem DMR Baseada em Paridade 25
- 4.1 Regras de Formação da Palavra-Código . . . 27
  - 4.1.1 Regra *RF1* . . . 27
  - 4.1.2 Regra *RF2* . . . 29
- 4.2 A Paridade como Critério de Seleção . . . 29
- 4.3 O Bit de Indicação de Erro . . . 33
- 4.4 Otimizando a Solução . . . 37
- 4.5 Expandido a Capacidade de Detecção . . . 41
- 4.6 Aplicação com Elementos Redundantes . . . 42
- 4.7 Resumo . . . 44

5 Análise Matemática da Proposta PDMR **46**
5.1 Cálculos de Probabilidade para PDMR-B 47
5.1.1 Correção 47
5.1.2 Erro 59
5.2 Cálculos de Probabilidade para PDMR-G 59
5.2.1 Correção 59
5.2.2 Detecção 67
5.2.3 Erro 76
5.3 Cálculos de Probabilidade para PDMR-C 77
5.3.1 Correção 77
5.3.2 Detecção 77
5.3.3 Erro 83
5.4 Resumo 83

6 Experimentos e Resultados **85**
6.1 Cálculos Matemáticos 86
6.2 Simulações em *Software* 86
6.2.1 Comparação Entre Valores Simulados e Calculados 88
6.2.2 Capacidade de Correção da PDMR e da TMR 93
6.2.3 Análise de Retransmissão 94
6.3 Implementações em *Hardware* 96
6.3.1 Plataforma de *Hardware* e Sistemas Projetados 96
6.3.2 Resultados de Consumo de Área e Potência 100
6.4 Estudo de Caso com Elementos Redundantes 103
6.5 Comparações Adicionais 104
6.6 Resumo 106

7 Conclusão **108**
7.1 Trabalhos Futuros 109
7.2 Publicação Relacionada 109

Referências bibliográficas **110**

Lista de Figuras

2.1 Possíveis configurações de CubeSats 8
2.2 CubeSat 1U com subsistema de comunicação (TT&C), ADCS, OBC, carga útil (*Payload*) e subsistema de energia (EPS) 8
2.3 Simplificação dos conceitos de falha, erro e defeito 11
2.4 Exemplo de Redundância Modular Tripla (TMR) 12
2.5 Exemplo simplificado de uma implementação FPSoC 14
2.6 Arquitetura dos canais de escrita para o AMBA AXI 18
2.7 Arquitetura dos canais de leitura para o AMBA AXI 19
2.8 Interfaces e interconexão AXI para um sistema típico 19

4.1 Diagrama de transmissão de dados com TMR 25
4.2 Circuito lógico votador TMR para um bit de entrada 26
4.3 Diagrama de transmissão de dados com DMR 27
4.4 Formato de uma palavra-código de N_d bits com bits de paridade 28
4.5 Fluxo de bits para o dado binário $0b100101$ transmitido com a PDMR . . 30
4.6 Exemplo de implementação da PDMR com seletor de canal simples para um dado de N bits dividido em grupos com três bits de informação e um bit de paridade 32
4.7 MUX para seleção de canal do grupo 0 apresentado no exemplo da Figura 4.6 32
4.8 Relação entre o número de portas lógicas e bits de informação (N) para diferentes valores de N_p 34
4.9 Relação entre o número de entradas de porta e bits de informação (N) para valores distintos de N_p 34
4.10 Circuito lógico para o bit de indicação de erro no grupo i (h_i) 36
4.11 Formato do dado de saída 38
4.12 Diagrama de transmissão de dados com PDMR em que $N_\mathrm{p} = N_\mathrm{e}$ 39
4.13 Exemplo da PDMR com indicação de erro para um dado de entrada com três bits de infomação 39
4.14 Exemplo de comparador para o receptor considerando três bits de informação 42
4.15 Diagrama de transmissão de dados de elementos redundantes com TMR . 43
4.16 Diagrama de transmissão de dados de elementos redundantes com PDMR 43

5.1 Exemplo para transmissão do dado $0b00$ com paridade par para a PDMR-B 51
5.2 Posições em que os *bitflips* podem acontecer em um dado de N_g bits em um dos canais 52

6.1 Diagrama da simulação em *software* baseada em *scripts* Python 88
6.2 NRMSE para a versão PDMR-B 91
6.3 NRMSE para as versões PDMR-G e PDMR-C 92
6.4 IPs gerados com auxílio do Vivado (a) codificador e (b) decodificador. . . 97
6.5 Sistema para emulação de transmissão de dados com implementações PDMR (blocos em vermelho) 98
6.6 *Design* de blocos para emulação de injeção de falhas usando os blocos PDMR desenvolvidos (marcados em vermelho) 98
6.7 Exemplo de fluxo de dados para o teste de injeção de falhas 99
6.8 Ambiente de teste de emulação incluindo redundância na interconexão com blocos PDMR (em vermelho) 99
6.9 Implementação para teste de integração da técnica PDMR com elementos redundantes (CORDIC) 103

Lista de Tabelas

2.1 Trabalhos sobre arquiteturas de processamento híbrido em missões espaciais. 17

3.1 Resumo dos trabalhos relacionados em comparação com a proposta . . . 24

4.1 Tabela verdade para o bit de indicação de erro no grupo i (h_i) 35
4.2 Tabela verdade para o bit de seleção de canal (s_i) e para o bit de *flag* de erro (f_i), ambos referentes ao grupo i 38
4.3 Relações para contagem do número de portas e entradas de porta para o caso geral 40
4.4 Resumo das expressões lógicas e equações para as diferentes versões da PDMR 44

5.1 Eventos possíveis para transmissão do dado $0b0$ com paridade par para a versão PDMR-B 48
5.2 Eventos possíveis para transmissão do dado $0b0$ com paridade par para a versão PDMR-G 60
5.3 Os 8 primeiros eventos possíveis para transmissão do dado $0b00$ com paridade par para a versão PDMR-G 62
5.4 Tabela-verdade e expressões booleanas para o bit de indicação de erro no grupo i (e_i) 67
5.5 Paridades e eventos associados para $p_i = 0$ 68
5.6 Paridades e eventos associados para $p_i = 1$ 69
5.7 Probabilidades de Z_0 e Z_1 para $M = 3$ 74
5.8 Equações para o cálculo de probabilidades das diferentes versões PDMR 84

6.1 Probabilidades calculadas para diferentes configurações e versões da PDMR 87
6.2 Probabilidades simuladas para diferentes configurações e versões da PDMR 89
6.3 Erro quadrático médio para 32 bits de informação e 8 de paridade 90
6.4 Erro quadrático médio para 32 bits de informação e 4 de paridade 90
6.5 Erro quadrático médio para 24 bits de informação e 8 de paridade 90
6.6 Erro quadrático médio para 28 bits de informação e 4 de paridade 91
6.7 Resultados aproximados das simulações para as configurações *32x8*, *32x4*, *24x8* e *28x4*. As linhas em que se encontra a TMR contêm informações referentes às configurações *T32* (colunas *32x8* e *32x4*), *T24* (coluna *24x8*) e *T28* (coluna *28x4*) 93

6.8 Avaliação de retransmissão em que os valores indicam o percentual da quantidade total de erros acumulados (E_A) com relação ao número total de dados transmitidos (D_T) 95
6.9 Utilização de recursos e estimativa consumo de energia para as configurações *T24*, *B24x8*, *G24x8* e *C24x8* 100
6.10 Utilização de recursos e estimativa de consumo de energia para as configurações *T28*, *B28x4*, *G28x4* e *C28x4* 101
6.11 Utilização de recursos e estimativa de consumo de energia para as configurações *T*28 e *C28x4* com *AXI Interconnect* 102
6.12 Utilização de recursos e estimativa de potência com CORDICs 104
6.13 Características de diferentes métodos 105

Lista de Símbolos e Abreviaturas

ADCS: *Attitude and Determination Control System*

AHB: *Advanced High-performance Bus*

AMBA: *Advanced Microcontroller Bus Architecture*

AXI: *Advanced eXtensible Interface*

CMOS: *Complementary Metal-Oxide-Semiconductor*

COTS: *Commercial-Off-The-Shelf*

CPU: *Central Processing Unit*

DMA: *Direct Memory Access*

DMR: *Dual Modular Redundancy*

DSP: *Digital Signal Processor*

ECC: *Error-Correcting Code*

EDAC: *Error Detection and Correction*

EPS: *Electrical Power System*

FPGA: *Field-Programmable Gate Array*

FPSoCs: *Field-Programmable Systems-on-Chip*

GCRs: *Galactic Cosmic Rays*

GPU: *Graphics Processing Unit*

IP: *Intellectual Property*

LEO: *Low Earth Orbit*

LPC: *Low Pin Counter*

LSB: *Least Significant Bit*

LUT: *Look-Up Table*

MSB: *Most Significant Bit*

MSE: *Mean Squared Error*

NRMSE: *Normalized Root Mean Squared Error*

PCB: *Printed Circuit Board*

SECDED: *Single Error Correction Double Error Detection*

SEE: *Single Event Effect*

SEFI: *Single Event Functional Interrupt*

SEL: *Single Event Latchup*

SETs: *Single Event Transients*

SEU: *Single Event Upset*

SoC: *System-on-Chip*

SRAM: *Static Random Access Memory*

TID: *Total Ionizing Dose*

TMR: *Triple Modular Redundancy*

TT&C: *Telemetry, Tracking & Command*

VBN: *Vision-Based Navigation*

Capítulo 1

Introdução

O desenvolvimento de novas tecnologias na área da microeletrônica representou um importante papel na expansão de sistemas espaciais ao permitir a miniaturização de aeronaves e a criação de padrões de satélites com tamanho reduzido, como o padrão CubeSat (Heidt et al. 2000, Woellert et al. 2011). Esse padrão foi proposto, em 1999, pelos professores Jordi Puig-Suari da Universidade Politécnica da Califórnia (CalPoly) e Bob Twiggs da Universidade de Stanford. Inicialmente, o projeto tinha como objetivo reduzir os custos e o tempo de desenvolvimento de picossatélites, aumentando, assim, a acessibilidade ao espaço por parte da comunidade acadêmica (Heidt et al. 2000, Johnstone 2022). A unidade básica do CubeSat, denominada 1U, é um cubo de 10 cm de aresta e até 2 kg, de modo que várias unidades podem ser integradas com o objetivo de formar satélites maiores (Johnstone 2022).

Ainda que o surgimento do padrão tenha colaborado para expansão da exploração espacial por parte de estudantes e pesquisadores, alguns desafios passaram a ser enfrentados pelos projetistas. Uma vez que os CubeSats possuem restrições de massa e tamanho (Johnstone 2022), configurações maiores do que a unidade básica 1U podem ser necessárias de acordo com a complexidade da missão. Isso provoca um aumento no custo do projeto, que pode inviabilizar o desenvolvimento em situações em que não se dispõe de recurso financeiro suficiente. Ademais, as condições severas do ambiente espacial oferecem riscos aos componentes eletrônicos dos nanossatélites e projetos com componentes tolerantes à radiação também impactam no custo das missões (Pignol 2010, Fortescue et al. 2011).

Como forma de minimizar os custos e o tempo de desenvolvimento de projetos, ao longo dos anos, tornou-se comum a utilização de componentes comerciais de prateleira (COTS, em inglês, *Commercial-Off-The-Shelf*), como *Field-Programmable Gate Arrays* (FPGAs) e processadores, nos subsistemas de pequenos satélites. Tais componentes tornam o processo de desenvolvimento e construção de CubeSats mais rápido, barato e simples quando comparado ao uso de *hardware* específico para aplicações espaciais (Palkovitz 2016). Além disso, processadores COTS, normalmente, encontram-se anos à frente daqueles projetados para uso em ambiente espacial (*radiation-hardened* ou *rad-hard*) em termos de desempenho (Pignol 2010, Fortescue et al. 2011, Lentaris et al. 2018).

Em face da crescente utilização de processadores COTS em CubeSats, a radiação se apresenta como outro desafio que precisa ser contornado por pesquisadores e projetistas.

Quando inseridos em ambiente espacial, os componentes eletrônicos dos satélites sofrem efeitos adversos decorrentes da radiação proveniente de diferentes fontes (LaBel et al. 1996). Podem ocorrer erros permanentes, os quais comprometem permanentemente o funcionamento do satélite, ou erros não permanentes, caracterizados por *Single Event Upsets* (SEUs) e *Single Event Transients* (SETs) (LaBel et al. 1996).

SEUs são causados pelo impacto de íons pesados ou prótons em elementos biestáveis, como *flip-flops* ou outras células de memória, o que pode gerar uma mudança de estado (*bitflip*) em um bit armazenado (Maurer et al. 2008). Enquanto que SETs ocorrem quando partículas carregadas atingem o circuito integrado causando pulsos momentâneos de tensão. Os transitórios podem se propagar no circuito e ser capturados por elementos de armazenamento (*flip-flops* ou *latches*), tonando-se um SEU e afetando, consequentemente, o comportamento do componente (Maurer et al. 2008, De Sio et al. 2019).

Dada a crescente demanda por capacidade de processamento em nanossatélites, Sistemas em Chip (SoCs, do inglês, *Systems-on-Chip*) que integram *cores* de processamento e FPGA, têm sido cada vez mais utilizados em missões com pequenos satélites (George & Wilson 2018). Esses SoCs possibilitam a implementação de processamento híbrido, em que parte do processamento pode ser realizado por processadores *hard-core* embarcados e o resto das operações computacionais pode ser realizadas por aceleração em *hardware* na área de lógica reconfigurável (FPGA). Uma vez que esses dispositivos COTS se baseiam em FPGAs SRAM (*Static Random Access Memory*), cuja memória de configuração está sujeita aos efeitos dos SEUs, vários trabalhos descrevem os efeitos da radiação e métodos para mitigá-los (Kastensmidt & Reis 2007, Siegle et al. 2015, Wirthlin 2015, Kasap et al. 2020) .

No trabalho de De Sio et al. (2020*a*), os autores destacam que módulos de interconexão (interfaces de comunicação) afetadas por SEUs podem ser fonte de erros em arquiteturas que usam aceleração de *hardware*. Benevenuti & Kastensmidt (2018) mencionam que a substituição de módulos de interconexão por versões tolerantes à falhas pode ser necessária para aumentar a confiabilidade de sistemas que incluem um módulo acelerador de *hardware* tolerante à falhas. Contudo, esses trabalhos não discutem métodos de mitigar erros na transmissão de dados causados por falhas em módulos de interconexão. Por exemplo, em Benevenuti & Kastensmidt (2018), técnicas de redundância em *hardware* são aplicadas apenas no módulo de aceleração de *hardware* e não à arquitetura de comunicação.

Diferentes técnicas de redundância podem ser empregadas para mitigar possíveis erros causados pela radiação. A redundância de informação envolve a adição de bits extras aos dados, os quais podem ser utilizados para detecção e até mesmo correção de *bitflips*. A capacidade de detecção e correção depende do código utilizado (Faraj et al. 2003). Aqueles que permitem a correção sem a necessidade de retransmissão da informação são referenciados como códigos *Forward Error Correction* (FEC).

Diversos tipos de códigos FEC são descritos na literatura (Ryan & Lin 2009). A complexidade de implementação desses códigos pode variar de acordo com o objetivo da aplicação. Desset & Fort (2003) descrevem a complexidade de códigos de correção de erros clássicos, como Hamming, Reed-Muller, Golay, Reed-Solomon, Convolutional, e códigos Turbo. A maioria desses códigos envolve o uso de circuitos sequenciais, como contadores

e registradores de deslocamento, no codificador (*encoder*) ou decodificador (*decoder*), o que pode significar um problema em sistemas com restrições de tempo. Além disso, outros códigos, como Hamming e Reed-Muller, podem envolver a implementação de um gerador de matriz no codificador, que pode ter impacto direto na complexidade lógica. O autores destacam que devido ao limite de energia de uma aplicação, apenas códigos com complexidade limitada podem ser considerados. Portanto, a seleção do código a ser implementado é crucial para sistemas com restrições de energia.

Os códigos FEC podem apresentar alta capacidade de correção, mas a quantidade de bits redundantes pode aumentar consideravelmente a depender do código utilizado. Dessa forma, sua utilização em protocolos padrões pode envolver o envio da mensagem e do *checksum* em ciclos de *clock* distintos ou uso de linhas extra de barramento para inclusão dos bits redundantes (Mach et al. 2023, ARM 2011). Considerando essas características e a complexidade dos códigos FEC, sua implementação pode impactar o tempo de desenvolvimento e consumo de energia, o que está em desacordo com o desejado para aplicações com plataformas de nanossatélites (Woellert et al. 2011, Shiroma et al. 2011, Villela et al. 2019). Nesse contexto, códigos FEC modernos podem ter aplicação limitada em sistemas embarcados e comunicação *on-chip*, embora o uso de códigos *Single Error Correction* (SEC) e *Single Error Correction Double Error Detection* sejam verificados, principalmente para a proteção de componentes de memória (Gracia-Morán et al. 2018, Mach et al. 2023). Outra observação importante sobre códigos FEC está relacionada ao fato de que a retransmissão acaba sendo mais eficiente do que a correção do ponto de vista energético (Raghunathan et al. 2003, Bertozzi et al. 2005, Mach et al. 2023, Lázaro et al. 2024).

A redundância de *hardware* apresenta uma implementação mais simples do que os esquemas com código FEC e tem sido utilizada em diversas aplicações para mitigar SETs e SEUs (Morgan et al. 2007, Mach et al. 2023). Uma das técnicas de redundância mais comuns é a Redundância Modular Tripla (TMR, *Triple Modular Redundancy*) que consiste na utilização de três componentes de *hardware* idênticos que possuem entradas iguais e executam as mesmas tarefas. As saídas desses componentes são comparadas por um circuito votador, que considera a saída do sistema como sendo o resultado da maioria. Desse modo, caso um componente tenha falhado, gerando uma saída incorreta, o votador estabelece que a saída do sistema será igual a saída dos dois componentes que operam corretamente (Shooman 2002). Contudo, se dois ou três componentes tiverem falhado, o votador irá concordar com a maioria e a saída do sistema estará errada (Shooman 2002).

Embora a TMR possa ser utilizada para mitigar os efeitos da radiação ao considerar que a falha afeta apenas um dos três componentes redundantes, as suas desvantagens dizem respeito ao aumento em mais de 200% da área e um *overhead* de energia (Siegle et al. 2015), quando observada a parte triplicada no sistema (e.g. memória, processador, acelerador de *hardware*). Outra desvantagem está relacionada com o impacto no desempenho do circuito, especialmente quando se tem muitas partições TMR (Siegle et al. 2015).

Uma alternativa à técnica TMR e à complexidade dos códigos FEC é a Redundância Modular Dupla (DMR, *Dual Modular Redundancy*), que consiste na duplicação de elementos lógicos. Nesse caso, é possível detectar erros no sistema ao comparar os bits de

cada elemento redundante. Se comparada com a TMR, a DMR apresenta uma menor área ocupada, mas permite apenas a detecção de erros, de modo que não é possível determinar qual dos dois elementos de fato apresenta falha e, consequentemente, qual a saída correta. Apesar disso, a DMR é uma abordagem de proteção mais simples em comparação com códigos FEC.

A fim de permitir que a DMR possua capacidade de mitigação de erros, neste trabalho foi desenvolvida uma abordagem DMR baseada em paridade para aplicação em interfaces de comunicação de SoCs que integram processadores *hard-core* e FPGA no mesmo *chip*. A técnica proposta visa aumentar a confiabilidade da transmissão de dados em SoCs COTS, os quais são cada vez mais utilizados em missões espaciais com nanossatélites.

A proposta, assim como a TMR, permite mascarar falhas por meio de lógicas associadas à bits de paridade. Além disso, componentes para indicação de erro possibilitam a detecção de mensagens potencialmente erradas, o que pode facilitar ações para retransmissão de dados. Para alcançar essas características, buscou-se focar em uma solução com baixa complexidade de implementação em termos de elementos lógicos.

Diferentes versões da proposta foram testadas por meio de simulações em *software* e implementações em *hardware*, em que a técnica foi aplicada à interfaces de comunicação *Advanced eXtensible Interface* (AXI) (ARM 2013) do padrão *Advanced Microcontroller Bus Architecture* (AMBA) e comparada com a TMR.

1.1 Motivação

De acordo com George & Wilson (2018), considerando as restrições de potência, tamanho, peso e custo de uma plataforma de pequeno satélite (como CubeSat), o maior desafio no desenvolvimento de missões futuras é equilibrar as demandas de processamento científico e do sensoriamento de bordo.

Ainda segundo George & Wilson (2018), atualmente, dois temas guiam a revolução da computação em pequenos satélites: a computação reconfigurável e a computação híbrida. A primeira corresponde a um conjunto de arquiteturas e metodologias de programação que permitem a personalização da lógica reconfigurável de acordo com a necessidade da aplicação. Como exemplo de dispositivo com computação reconfigurável, tem-se FPGAs. Enquanto isso, a computação híbrida é conceituada como sendo a junção de tecnologias computacionais distintas com o intuito de obter vantagens coletivas. Como exemplo de dispositivos que utilizam computação híbrida, podem ser citados SoCs que integram múltiplas arquiteturas de computação, como combinação de CPUs (*Central Processing Units*), FPGAs, GPUs (*Graphics Processing Units*) e DSPs (*Digital Signal Processors*).

A integração de núcleos de processamento, FPGAs, memórias e periféricos em SoCs torna esse tipo de tecnologia atrativa para aplicação espacial, uma vez que satisfaz a crescente demanda por capacidade de processamento e reduz a massa e o consumo energético do sistema (Estela & Montenegro 2020). Além disso, esses sistemas permitem que diferentes técnicas de tolerância a falhas sejam utilizadas por meio, por exemplo, da reconfiguração da lógica implementada no FPGA com base em comandos gerados pelo processador *hard-core*. No entanto, como apresentado nas publicações de Graham et al. (2009), Benevenuti & Kastensmidt (2018) e De Sio et al. (2020*a*), a interface de comuni-

cação entre processador *hard-core* e lógica reconfigurável (FPGA) está sujeita aos efeitos da radiação e falhas podem gerar erros, por exemplo, em aplicações que fazem uso de aceleração em *hardware*, com interrupção no funcionamento dos aceleradores ou obtenção de cálculos errados proveniente deles. Isso indica que técnicas para mitigação de falhas devem ser empregadas nas interfaces de comunicação com intuito de aumentar a confiabilidade do sistema.

1.2 Objetivos

O objetivo geral deste trabalho é propor uma técnica de DMR baseada em paridade capaz de aumentar a confiabilidade da transmissão de dados em SoCs COTS, os quais integram processadores *hard-core* e lógica reconfigurável no mesmo *chip*. Tais SoCs têm sido utilizados para processamento de bordo em nanossatélites.

Além disso, o trabalho tem como objetivos específicos:

- Analisar a capacidade de detecção de erro de diferentes versões da técnica proposta, bem como a probabilidade de mascarar/corrigir *bitflips* (erros transientes).
- Avaliar o consumo energético e a utilização de recursos de *hardware* da implementação em SoC comercial composto por processador *hard-core* e lógica reconfigurável (FPGA).

1.3 Contribuições

Este trabalho apresenta as seguintes contribuições:

- Soluções de baixa complexidade lógica para mitigação de erros na transmissão de dados em SoCs COTS aplicados no processamento de dados em nanossatélites.
- Versão simplificada da proposta DMR com paridade para aplicação em interfaces de comunicação, a qual possibilita mascarar erros.
- Técnica otimizada com inclusão de lógica para indicação de erros, permitindo elaboração de ações para detecção de erro e retransmissão de dados.
- Método DMR com paridade baseado em estruturas lógicas para comparação de bits, oferecendo uma solução com maior capacidade de detecção.
- Análise matemática com cálculos de probabilidade de correção, detecção e erros para diferentes versões da técnica DMR baseada em paridade.
- Comunicação *on-chip* tolerante a falhas para integração entre componentes de aceleração em *hardware* e processador, fundamentada no protocolo AXI do padrão AMBA.
- Estudo comparativo para avaliação da capacidade de mascarar erros da técnica proposta e da TMR, considerando diferentes taxas de erros. Além disso, comparação de implementação em *hardware* evidenciando uma menor utilização de recursos de *hardware* e um menor consumo energético da solução proposta quando comparada à TMR.

1.4 Organização do Texto

No Capítulo 2 é apresentada a fundamentação teórica utilizada como base para compreensão e elaboração do trabalho. Nele, estão dispostos conceitos relacionados a pequenos satélites, os efeitos da radiação sobre componentes eletrônicos em ambiente espacial, as técnicas de tolerância a falhas e o protocolo AMBA AXI.

No Capítulo 3 é descrito o estado da arte, onde são discutidos trabalhos relacionados aos efeitos da radiação e uso de técnicas de tolerância a falhas sobre interfaces de comunicação *on-chip*.

A abordagem proposta, com regras de formação e expressões lógicas, é descrita no Capítulo 4.

No Capítulo 5, estão dispostas as análises matemáticas para obtenção das equações que descrevem a probabilidade de correção, detecção e erros para diferentes versões da técnica proposta.

O Capítulo 6 contém os experimentos e resultado das implementações propostas.

Por fim, no Capítulo 7 estão dispostas considerações finais.

Capítulo 2

Nanossatélites e a Radiação em Sistemas Espaciais

Neste capítulo, são descritas as bases teóricas necessárias para compreensão da abordagem proposta. Nesse sentido, são apresentadas as definições sobre CubeSats, os efeitos da radiação sobre componentes eletrônicos e as técnicas de tolerância a falhas utilizada para mitigar esses efeitos. Além disso, são expostos os conceitos relacionados ao protocolo de comunicação *on-chip* AMBA AXI.

2.1 CubeSats

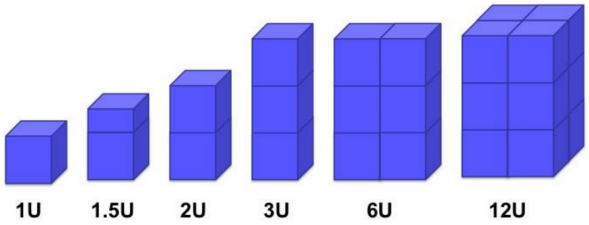
Segundo Koudelka (2016), os satélites são geralmente classificados de acordo com sua massa e, atualmente, não há um consenso internacional com relação ao termo "pequenos satélites". No entanto, parte da comunidade científica utiliza esse termo para caracterizar satélites com massa inferior a 1000 kg (Koudelka 2016, Palkovitz 2016, Jakhu & Pelton 2014). Além disso, os pequenos satélites são ainda subdivididos em: minissatélites (massa entre 100 kg e 1000 kg); microssatélites (massa entre 10 kg e 100 kg); nanossatélites (massa entre 1kg e 10 kg); picossatélites (massa entre 0,1 kg e 1kg); e femtossatélites (massa entre 1 g e 100 g) (Kramer & Cracknell 2008).

Considerando a classificação supracitada, os CubeSats são categorizados, em geral, como pico ou nanossatélites. De acordo com o documento de especificações do projeto (Johnstone 2022), um CubeSat é um cubo com 10 cm de aresta e até 2 kg, referenciado como sendo uma unidade 1U. Várias unidades podem ser agrupadas para que sejam formadas configurações de satélites maiores, como por exemplo 2U, 3U, 6U e 12U (Figura 2.1).

Não obstante o desenvolvimento inicial de CubeSats ter motivação educacional (Heidt et al. 2000, Johnstone 2022), com o passar dos anos, percebeu-se que satélites desse tipo também poderiam ser utilizados em outras aplicações, como ciência, telecomunicações, sensoriamento remoto e desenvolvimento tecnológico (Villela et al. 2019). Atualmente, graças aos CubeSats, universidades, instituições de ensino, grupos comerciais e agências governamentais ao redor do mundo possuem um programa espacial (NASA CubeSat Launch Initiative 2017).

Embora apresentem tamanho reduzido, os CubeSats são compostos por diferentes

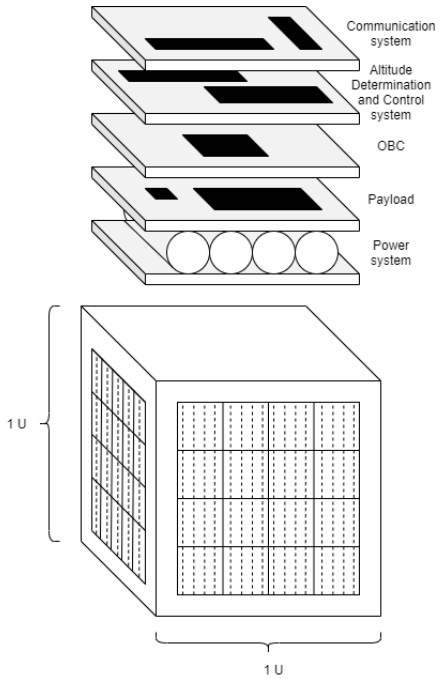
Figura 2.1: Possíveis configurações de CubeSats



Fonte: NASA (2020).

subsistemas essenciais para garantia de funcionamento em órbita e para o sucesso da missão espacial (Figura 2.2). Entres esses subsistemas, tem-se: subsistema de energia (EPS, *Electrical Power System*); subsistema de determinação e controle de atitude (ADCS, *Attitude and Determination Control System*); subsistema de telemetria e telecomando (TT&C, *Telemetry, Tracking & Command*); computador de bordo (OBC, *On-Board Computer*); e carga útil (Koudelka 2016). Em missões espaciais, objetivos específicos são alcançados por meio das funcionalidades disponibilizadas pelo sistema da carga útil (Scholz 2017). Por exemplo, em missões para observação da terra, a carga útil pode ser instrumentos de sensoriamento remoto e em missões científicas, pode-se ter um novo instrumento tecnológico (Scholz 2017).

Figura 2.2: CubeSat 1U com subsistema de comunicação (TT&C), ADCS, OBC, carga útil (*Payload*) e subsistema de energia (EPS)



Fonte: Vasudevan (2020).

Os subsistemas são construídos em placas de circuito impresso (PCB, *Printed Circuit Board*) e desempenham funções específicas (controle e distribuição de energia, medição e ajuste de orientação, comunicação com estação terrestre, gerenciamento e armazenamento de dados), essenciais para o correto funcionamento do satélite em órbita. A depender da complexidade do circuito projetado, um subsistema pode estar contido em uma ou mais PCBs. Além disso, geralmente, cada subsistema conta com um processador dedicado e a transmissão de informação com os demais componentes do CubeSat ocorre por meio de barramentos de comunicação.

2.2 Efeitos da Radiação em Componentes Eletrônicos

O ambiente espacial apresenta condições físicas severas para os componentes dos satélites. Entre as características desse ambiente, tem-se: alto vácuo; radiação solar; raios X, ultravioleta e radiação gama; partículas de alta energia (prótons, elétrons, nêutrons e partículas alfa); microgravidade e arrasto aerodinâmico; e influência de oxigênio atômico (Henjes et al. 2009).

Segundo Edmonds et al. (2000), as principais fontes de radiação espacial são os raios cósmicos galáticos (GCRs, *Galactic Cosmic Rays*), as partículas emitidas por eventos solares e as partículas presas em cinturões de radiação ao redor da Terra. GCRs têm origem fora do sistema solar e são compostos, principalmente, por prótons (83 %), embora seja observada presença de partículas alfa (13 %), elétrons (3 %) e íons pesados (1 %) (Villa 2018, Edmonds et al. 2000). Os eventos solares se caracterizam por erupções solares e ejeções de massa coronal que emitem partículas de alta energia, como prótons e íons pesados (Boudenot 2007). Adicionalmente, as partículas presas nos cinturões de radiação (cinturões de Van Allen) são, especialmente, prótons e elétrons (Boudenot 2007).

Partículas provenientes das fontes de radiação supracitadas são responsáveis por efeitos adversos sobre os componentes eletrônicos de satélites. Esses efeitos da radiação podem ser divididos em duas categorias gerais: efeitos cumulativos e efeitos de evento único (SEEs, *Single Event Effects*) (Maurer et al. 2008). Como exemplo de efeito cumulativo, tem-se a dose total ionizante (TID, *Total Ionizing Dose*), que resulta do acúmulo de energia depositada em um material por um longo período de tempo (LaBel et al. 1996). Por outro lado, os SEEs são efeitos decorrentes da passagem de uma única partícula ionizante sobre o dispositivo (LaBel et al. 1996).

De acordo com Maurer et al. (2008), os SEEs podem ser destrutivos, gerando erros permanentes, ou não destrutivos, caracterizados por erros não permanentes. Um exemplo de SEE destrutivo é o *Single Event Latchup* (SEL), que é consequência de uma alta corrente induzida pela interação de uma única partícula de energia em circuitos integrados fabricados com tecnologia CMOS (*Complementary Metal-Oxide-Semiconductor*). Danos ao circuito podem ser evitados se a alimentação for rapidamente removida (Maurer et al. 2008).

Erros não permanentes podem ser classificados da seguinte forma (Maurer et al. 2008):

- *Single Event Upset* (SEU): É causado pelo impacto de íons pesados ou prótons em elementos biestáveis, como *flip-flops* ou outras células de memória. Como con-

sequência do impacto, tem-se uma mudança de estado (*bitflip*) que pode ser corrigida por meio da reescrita de dados no elemento afetado.
- *Single Event Transients* (SETs): Ocorrem quando partículas carregadas atingem o circuito integrado causando pulsos momentâneos de tensão. Os transitórios podem se propagar no circuito e ser capturados por elementos de armazenamento (*flip-flops* ou *latches*), afetando, consequentemente, o comportamento do componente.
- *Single Event Functional Interrupt* (SEFI): É um tipo de SEE que pode levar o dispositivo a interromper sua operação normal. Esses efeitos geralmente não causam danos aos dispositivos, mas podem produzir erros de interrupção de funcionamento, de controle e de dados, que podem ser recuperados por meio da reinicialização do sistema.

Segundo Wirthlin (2015), FPGAs estão sujeitos aos mesmos problemas relacionados à radiação que outros dispositivos semicondutores. Os efeitos em FPGAs dependem, em grande parte, do tipo de tecnologia utilizada para armazenar os dados de configuração. Desse modo, são consideradas as seguintes categorias de FPGAs (Wirthlin 2015):

- Antifuse: São programados uma única vez, de modo que os estados de cada bit de configuração do FPGA são definidos permanentemente. Esses FPGAs são não voláteis e o estado da memória de configuração não se altera em caso de desligamento do dispositivo. Além disso, eles são geralmente menos sensíveis aos efeitos do tipo SEE devido à utilização de configuração estática.
- Flash: Usam células de memória flash para definição do estado da memória de configuração, são não voláteis e podem ser reconfigurados. Contudo, não são adequados para sistemas que necessitam de reconfiguração frequente, pois apresentam uma limitação no número de vezes que podem ser reprogramados. Assim como FPGAs Antifuse, possuem relativa imunidade aos efeitos do tipo SEU, mas são sensíveis à TIDs, o que faz com que possam ser utilizados em ambientes com radiação por um período de tempo limitado.
- SRAM: São dispositivos do tipo volátil e precisam de alimentação para que o estado da memória de configuração seja mantido. A configuração interna é armazenada em células de memória estática que devem ser programadas por meio de memórias externas após o FPGA ter sido energizado. Apesar de ocuparem uma maior área de silício e apresentarem um maior consumo energético em comparação com os FPGAs já citados, FPGAs SRAM podem ser reprogramados de maneira ilimitada, o que facilita reconfigurações e adaptações de maneira frequente. Por utilizarem células de memória padrão, esse FPGAs são suscetíveis ao efeitos do tipo SEU e técnicas para mitigação da radiação precisam ser empregadas para que possam operar em ambiente com radiação.

A possibilidade de reprogramação em órbita de FPGAs SRAM e o menor custo de desenvolvimento associado a esses dispositivos tornam atraente sua utilização em sistemas espaciais. Os efeitos da radiação sobre FPGAs SRAM são detalhados em Kastensmidt & Reis (2007) e em Siegle et al. (2015).

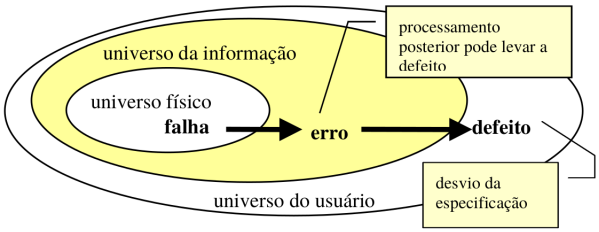
Grande parte dos pequenos satélites são implantados em Órbita Terrestre Baixa (LEO, do inglês, *Low Earth Orbit*) (Jakhu & Pelton 2014), que corresponde a uma distância entre

160 km e 2000 km acima da superfície terrestre. Os níveis de radiação em LEO são considerados modestos devido à blindagem pela magnetosfera terrestre (Woellert et al. 2011). Em vista disso, componentes COTS têm sido frequentemente aplicados em missões em LEO por apresentarem uma redução no custo de projeto e por suportarem, em parte, os efeitos da radiação quando implantados nessa órbita. Tais efeitos podem ser mitigados pela aplicação de técnicas de tolerância a falhas, as quais são descritas na próxima seção.

2.3 Tolerância a Falhas

Em sistemas computacionais, os termos "falha", "erro" e "defeito" são utilizados para caracterizar conceitos distintos: um defeito ocorre quando o sistema se comporta de modo divergente de sua especificação; o erro é definido como parte do estado total do sistema que pode levar a um defeito; por fim, falha é descrita como sendo a causa de um erro (Lee & Anderson 1990, Weber 2001, Avizienis et al. 2004). Na Figura 2.3, pode-se verificar uma simplificação desses conceitos.

Figura 2.3: Simplificação dos conceitos de falha, erro e defeito



Fonte: Weber (2001).

Para exemplificar, considera-se um circuito somador com uma de suas saídas fixa em 1, o que significa que independente dos valores de entrada o estado da saída é sempre 1 (Koren & Krishna 2007). Nesse contexto, tem-se uma falha, a qual pode causar um erro quando o somador for utilizado e sua saída tiver que ser zero ao invés de 1 (Koren & Krishna 2007). A propagação do erro no sistema, por sua vez, pode ocasionar um defeito.

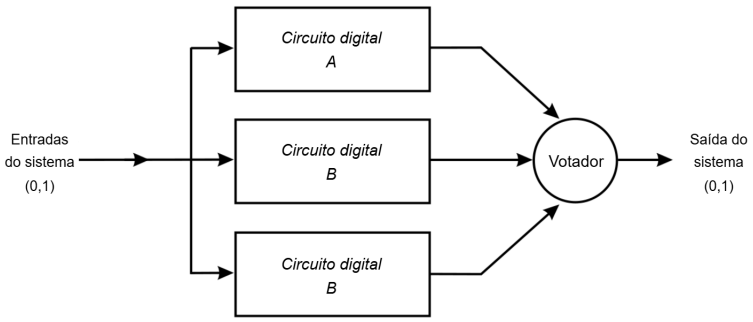
A tolerância a falhas tem como objetivo evitar que erros e falhas tornem um sistema defeituoso (Lee & Anderson 1990). Em outras palavras, o sistema deve continuar a fornecer os serviços esperados mesmo na presença de falhas (Weber 2001). Para isso, diferentes técnicas de tolerância a falhas podem ser empregadas. Essas técnicas se baseiam em alguma forma de redundância, que, de acordo com Koren & Krishna (2007), consiste em ter mais de um recurso do que o mínimo necessário para execução de uma tarefa. Koren & Krishna (2007) classificam redundância em quatro formas: *hardware*, *software*, informação e tempo.

A redundância de *hardware* é dividida em estática e dinâmica e corresponde à incorporação de *hardware* extra ao projeto com o intuito de detectar ou sobrepor os efeitos de um componente com falha (Koren & Krishna 2007). A redundância estática diz respeito ao uso de componentes redundantes de modo que os efeitos de um componente com fa-

lha seja mascarado no ambiente do sistema (Lee & Anderson 1990). Enquanto isso, a redundância dinâmica está relacionada à ativação de um componente reserva quando o componente ativo apresenta falha (Koren & Krishna 2007).

Como exemplo de redundância de *hardware* estática, tem-se a TMR que é uma técnica amplamente utilizada e baseada em um componente votador, o qual direciona o resultado da maioria dos bits de entrada para a saída (Lee & Anderson 1990, Shooman 2002). Na Figura 2.4, pode ser observado um exemplo de TMR para três circuitos digitais em paralelo.

Figura 2.4: Exemplo de Redundância Modular Tripla (TMR)



Fonte: Adaptado de Shooman (2002).

A lógica do votador clássico TMR é definida como (Shooman 2002, Balasubramanian et al. 2016):

$$V = A \cdot B + A \cdot C + B \cdot C \tag{2.1}$$

onde A, B e C são as saídas dos componentes digitais redundantes.

A técnica TMR, em geral, é aplicada em componentes que têm lógica equivalente e mesma tabela verdade, de maneira que, considerando que o votador não falha, a confiabilidade (probabilidade de não ocorrerem falhas) do sistema é dada por (Shooman 2002)

$$R_{\text{TMR}} = P(A \cdot B + A \cdot C + B \cdot C), \tag{2.2}$$

em que $P(A \cdot B + A \cdot C + B \cdot C)$ representa a probabilidade do resultado da expressão lógica em 2.1 estar correto. Segundo Shooman (2002), assumindo que a probabilidade de sucesso é dada por p, que todos os circuitos digitais são idênticos e independentes e usando o teorema binomial, tem-se:

$$\begin{aligned} R_{\text{TMR}} &= \binom{3}{3} p^3 (1-p)^0 + \binom{3}{2} p^2 (1-p)^1, \\ R_{\text{TMR}} &= 3p^2 - 2p^3 = p^2(3-2p). \end{aligned} \tag{2.3}$$

De acordo com Koren & Krishna (2007), a redundância de *software* envolve a produção de duas ou mais versões de *software* considerando que as diferentes versões não irão falhar para as mesmas entradas. A múltiplas versões de *software* podem ser executadas de maneira concorrente ou sequencial, o que requer redundância de *hardware* ou

redundância de tempo, respectivamente (Koren & Krishna 2007).

A redundância de informação permite que o erro seja detectado, mascarado e possivelmente tolerado por meio da adição de informação redundante aos dados (Goloubeva et al. 2006). Códigos de correção de erros (ECCs, *Error-Correcting Codes*) são a forma mais conhecida de redundância de informação e consistem na adição de bits extras aos bits de dados originais, de modo que um erro nos bits de dados pode ser detectado e até mesmo corrigido (Koren & Krishna 2007). Como posto por Goloubeva et al. (2006), um exemplo de ECC é o código de paridade, que consiste na adição de um bit extra para que os bits de dados resultem em uma palavra-código (*codeword*) com um número par (paridade par) ou ímpar (paridade ímpar) de 1s. Caso um erro afete uma palavra-código de paridade ímpar, trocando um de seus bits e, consequentemente, tornando a sua paridade par, é possível detectar esse erro uma vez que a paridade original era conhecida. A detecção é realizada pela simples contagem do número de 1s da palavra-código.

Segundo Vahid (2009), um bit de paridade par para um conjunto de bits de dados pode ser gerado por meio de portas *XOR*. A aplicação de operações *XOR* entre os bits de dados resulta em 0 quando o total de 1s é par e 1 quando é ímpar. Para verificar se a paridade da palavra-código está correta, podem ser realizadas operações *XOR* entre os bits dessa palavra, ou seja, bits de dados e bit de paridade. O valor binário resultante dessas operações é 0 se a paridade estiver correta e 1 em caso contrário. Para a paridade ímpar, as portas lógicas utilizadas são do tipo *XNOR*.

A repetição da computação (operação) no tempo é conceituada como redundância de tempo (ou temporal) (Weber 2001). Essa forma de redundância evita o custo de *hardware* adicional, mas aumenta o tempo necessário para realização da computação (Weber 2001). Portanto, sua utilização é adequada em sistemas em que o tempo não é crítico. A redundância temporal pode ser empregada para detecção de falhas transientes por meio da repetição de operações, onde resultados diferentes são um forte indicativo de uma falha transitória (Weber 2001, Goloubeva et al. 2006). Para detecção de falhas permanentes, codifica-se os dados durante a segunda computação e decodifica-se antes da comparação com o resultado da computação inicial (Weber 2001, Goloubeva et al. 2006).

As técnicas de tolerância a falhas podem ser utilizadas para mitigar os efeitos de radiação em sistemas para aplicação espacial. Como exemplo, tem-se: uso de processadores redundantes como forma de redundância de *hardware*; emprego de redundância de *software* no *software* de controle de bordo do OBC; uso de ECC em componentes de memória e para transferências de dados entre processadores e memórias; reexecução de programas em um mesmo OBC como forma de redundância temporal.

Além das técnicas já mencionadas, os efeitos da radiação podem ser mitigados por meio de circuitos *watchdog* e técnicas de *scrubbing*. O *watchdog* é um temporizador de *hardware* que deve ser reiniciado por *software* periodicamente quando o sistema está operando normalmente (Moore 2005). Caso o sistema entre em um modo anormal, o *watchdog* atinge o tempo limite (por não ter sido reiniciado) e, desse modo, inicia-se um processo de reinicialização do processador (Moore 2005). Por outro lado, as técnicas de *scrubbing* consistem na leitura regular de componentes de memória e na correção de eventuais erros encontrados por meio da reescrita de dados (Brosser et al. 2014).

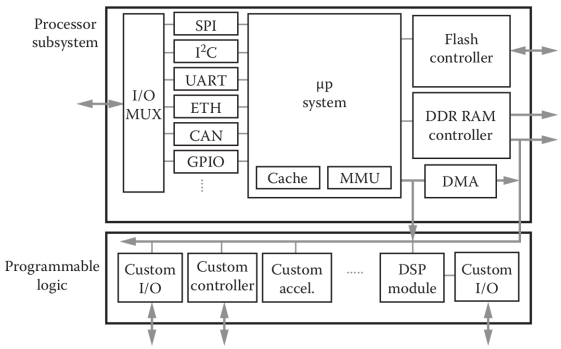
Para mitigar os efeitos de SEUs em FPGAs SRAM, podem ser empregadas as di-

versas técnicas apresentadas anteriormente. Segundo Wirthlin (2015), a redundância de *hardware* pode ser utilizada por meio da triplicação de elementos dos circuitos implementados no FPGA ou pelo uso de três FPGAs idênticos que desempenham a mesma função. Ademais, técnicas de *scrubbing* podem ser aplicadas na memória de configuração. Assim, os dados de configuração corretos são escritos na memória de configuração com o intuito de reparar SEUs. Os erros em memórias do tipo BRAM (*Block RAM*) podem ser detectados por códigos ECC, como bits de verificação e paridade.

2.4 Processamento Híbrido em Sistemas Espaciais

A computação híbrida tal qual definida por George & Wilson (2018) se baseia, por exemplo, na integração, em um mesmo chip, de um processador *hard-core* e de uma unidade lógica reconfigurável (FPGA). Esses chips são também chamados de FPSoCs (*Field-Programmable Systems-on-Chip*) e possuem uma estrutura como a apresentada na Figura 2.5.

Figura 2.5: Exemplo simplificado de uma implementação FPSoC



Fonte: Andina et al. (2017).

Essa estrutura é composta por uma região onde se encontra o subsistema de processador e uma região com o subsistema de lógica reconfigurável. O subsistema de processador, geralmente, inclui um *core* de processamento e vários periféricos, como interfaces de comunicação serial e interfaces de entrada e saída (Andina et al. 2017).

A utilização de arquiteturas de processamento híbrido em sistemas de satélites vem avançando ao longo dos anos. No final da primeira década desse século, pesquisas sobre computação de bordo passaram a analisar o uso de FPGAs e CPUs para aumento da capacidade de processamento demandada por experimentos científicos cada vez mais avançados.

No trabalho de Huber et al. (2007), observa-se um sistema de computação de bordo para microssatélites com quatro FPGAs sincronizados que recebem os mesmos sinais, executam as mesmas funções e produzem os mesmos resultados. Um esquema de votação se encarrega de comparar os resultados e determinar a saída correta. Caso os FPGAs entreguem um resultado errado ou nenhum resultado, é executada uma função de

reinicialização nesses dispositivos. Os quatro FPGAs utilizados são da família Xilinx VIRTEX-II Pro e possuem dois *cores* PowerPC. Além disso, o microcontrolador Picoblaze é implementado em FPGAs Spartan-II para desempenhar funções de comando e controle de configuração dos FPGAs VIRTEX-II.

Jacobs et al. (2008) descrevem um sistema de computação espacial composto por CPUs (PowerPC) e FPGAs COTS encarregados da execução de um algoritmo de classificação para imageamento hiperespectral. O algoritmo de detecção e classificação de imagens é dividido em três estágios e o processamento é executado na CPU ou FPGA de acordo com a possibilidade de aplicação de paralelismo às tarefas desempenhadas em cada estágio. Desse modo, a computação de natureza sequencial é realizada pela CPU e o paralelismo pelo FPGA.

Em Yu et al. (2009), os autores apontam um levantamento de missões espaciais com sistemas de compressão de imagem a bordo, descrevem algumas arquiteturas de satélites com diferentes implementações de hardware (ASIC, FPGA, DSP). Algumas das implementações fazem uso de processamento híbrido, como o satélite X-SAT que possui CPU ARM e FPGA interconectados. Além disso, os autores destacam o aumento de popularidade da utilização de FPGAs.

Ibrahim et al. (2011) apresentam um computador de bordo para uso em satélites em LEO. O sistema de processamento inclui quatro processadores, sendo um *hard-core* PowerPC e três *soft-core* Microblaze, que visam tolerância a falha por meio de TMR. O PowerPC foi utilizado como unidade de controle de configuração, encarregada do monitoramento dos processadores Microblaze e da reconfiguração do sistema em caso de falhas. A interface de comunicação usada para interconexão do PowerPC com os *soft-cores* foi a PLBv46.

Rudolph et al. (2014) apresentam um computador híbrido para uso espacial composto por componentes COTS e *radhard* e caracterizado pelo uso de diferentes técnicas de tolerância a falhas. Para os autores, a futura geração de computadores espaciais demandarão uma maior carga de processamento, mas continuarão limitados pela radiação e restrições de tamanho, peso e energia. Também destacam que apesar de um sistema com componentes COTS apresentar desempenho admirável, sua confiabilidade é menor em ambiente espacial. Desse modo, além do SoC Zynq-7020, que contém dois processadores ARM e FPGA no mesmo chip, a arquitetura de *hardware* proposta no trabalho permite que a PCB seja populada com dispositivos *radhard*. Além disso, para uma maior confiabilidade do sistema, com relação à radiação, são utilizados circuitos supervisórios, *watchdogs* e técnicas de *scrubbing* (para a memória de configuração do FPGA) e ECC.

Segundo Iturbe et al. (2015), a integração de FPGA e ARM em um único chip é extremamente conveniente em aplicações espaciais por oferecer uma redução no consumo de energia e no custo do sistema. Nesse contexto, os autores utilizam um dispositivo COTS contendo o SoC Zynq-7Z100 (ARM e FPGA em um único chip) para prototipagem de uma plataforma chamada APEX-SoC, a qual apresenta uma infraestrutura genérica e personalizável para aquisição e processamento de dados de instrumentos utilizados em ambiente espacial. Grande parte da funcionalidade dessa plataforma é implementada no FPGA, mas o processador ARM é utilizado para rotinas de *software* e para configurar e controlar o funcionamento do FPGA. Para lidar com os efeitos da radiação, várias

técnicas são utilizadas, entre as quais tem-se: ECC para elementos de memórias externos ao sistema; reconfiguração completa do FPGA para erros detectados na memória de configuração; *watchdog* para reinicialização do processador ARM.

Bruhn et al. (2015) descrevem um sistema para pequenos satélites tolerante a radiação composto por processador *multicore*, GPU e FPGA. A tolerância à radiação é justificada pela escolha de componentes COTS semelhantes a outros com comportamento já analisado em ambiente espacial. As unidades de processamento utilizadas foram o SoC AMD G-series, que inclui 4 CPUs x86 e GPU no mesmo chip, e o SoC SmartFusion2, que combina CPU (ARM Cortex-M3) e FPGA. A comunicação entre esses elementos foi realizada de maneira redundante, com utilização de um canal PCI Express e de um barramento LPC (*Low Pin Count*). Apesar da alta capacidade computacional proveniente das diversas unidades de processamento, o sistema apresenta um consumo energético elevado (6 W durante testes) para aplicação em CubeSats 1U.

Dong et al. (2016) propõem uma estrutura de computação de bordo composta por dois FPGAs e quatro DSPs. O FPGA é a unidade de processamento principal e os DSPs são utilizados no processamento de dados. A estrutura proposta utiliza redundância fria para os FPGAs e TMR para os DSPs. Nesse caso, em condições normais de funcionamento, um FPGA e três DSPs ficam ativos, enquanto os outros são elementos reserva. Quando o sistema de controle do FPGA detecta uma falha em algum dos DSPs, o dispositivo defeituoso é substituído pelo reserva, mantendo a estrutura de TMR. Em caso de falha no FPGA, uma substituição também é realizada, de modo que o componente em falha é desativado. O sistema é também composto por circuitos de *watchdog* encarregados de monitorar os estados dos FPGAs. Nesse trabalho, a estrutura proposta é apresentada sem detalhes relacionados aos componentes utilizados e a interface de comunicação entre FPGAs e DSPs.

Pérez et al. (2020) propõem uma arquitetura reconfigurável para processamento de bordo em sistemas críticos de exploração espacial. Para teste da arquitetura proposta, os autores implementaram algoritmos de navegação baseada em visão (VBN, *Vision-Based Navigation*), objetivando a aplicação em um cenário que envolve um módulo de pouso de uma espaçonave que carrega um *rover* para ser implantado em um corpo astronômico. Os autores justificam que os requisitos para algoritmos de VBN superam o desempenho de processadores qualificados para uso espacial, como o PowerPC-FT ou baseados no LEON. Sendo assim, a arquitetura proposta foi implementada no dispositivo Zynq Ultrascale+ ZU9EG, que possui uma unidade de processamento de aplicação (APU) e uma unidade de processamento em tempo real (RPU) combinadas com um FPGA. Ademais, a APU é composta por quatro processadores ARM Cortex-A53, enquanto que a RPU é constituída por dois processadores ARM Cortex-R5. Na área de lógica reconfigurável foi utilizado um *framework* de aceleração de *hardware* (Rodríguez et al. 2018) que permite que aceleradores sejam reconfigurados em tempo de execução. Desse modo, a lógica foi reconfigurada para implementação de diferentes algoritmos VBN em etapas específicas. Entre as técnicas de tolerância a falhas foram usadas ECC, *scrubbing*, redundância nos aceleradores de *hardware* e reconfiguração.

Os trabalhos descritos auxiliam na compreensão do avanço das tecnologias de processamento híbrido em aplicações espaciais ao longo dos anos. Além disso, eles evidenciam

o uso de componentes COTS para computação de bordo em satélites. Contudo, destaca-se a necessidade de aplicação de diferentes técnicas para mitigação dos efeitos da radiação, incluindo o uso de componentes *radhard* em conjunto com COTS para aumento da confiabilidade dos sistemas.

Uma análise adicional sobre o uso de processamento híbrido para satélites foi realizada por meio da busca de publicações nas bases de dados *Web of Science* e *Scopus*. Os trabalhos foram classificados de acordo com o ano, o SoC descrito, o tipo de satélite alvo e a área de aplicação. Essas informações foram utilizadas para construção da Tabela 2.1, em que as aplicações de inteligência artificial estão definidas pela sigla IA. Vale salientar que, na referida tabela, os SoCs são caracterizados por possuírem processador *hard-core* e lógica reconfigurável (FPGA) no mesmo chip.

Tabela 2.1: Trabalhos sobre arquiteturas de processamento híbrido em missões espaciais.

Ano	Trabalho	SoC	Tipo de Satélite	Aplicação
2010	(Bekker et al. 2010)	Virtex-5 FXT	CubeSat	Processamento de Imagem
2011	(Ibrahim et al. 2011)	Virtex-5 FXT	Satélites para LEO	OBC
2014	(Rudolph et al. 2014)	Zynq-7000 XC7Z020	Geral	OBC
2014	(Soh & Wu 2014)	Zynq-7000 XC7Z020	CubeSat	Determinação de Atitude
2014	(Rosário et al. 2014)	Zynq-7000 XC7Z020	Geral	Processamento de Imagem
2015	(Iturbe et al. 2015)	Zynq-7000 XC7Z100	Geral	Processamento de Dados Científicos
2016	(Nascimento & Véstias 2016)	Zynq-7000 XC7Z020	Geral	Processamento de Imagem
2016	(Ma, Ning et al. 2016)	Zynq Family[a]	Geral	Processamento de Imagem
2017	(Bezerra et al. 2017)	Zynq-7000 XC7Z030	CubeSat	OBC
2018	(Franklin et al. 2018)	Zynq-7000 XC7Z030	CubeSat	Processamento para Telecomunicação
2018	(Soh & Wu 2018)	Zynq-7000 XC7Z020 and XC7Z045	CubeSat	Determinação de Atitude
2018	(Fjeldtvedt et al. 2018)	Zynq-7000 XC7Z020	Geral	Processamento de Imagem
2018	(Vladimirova & Fayyaz 2018)	Zynq-7000 XC7Z020	Geral	Controle de Atitude e Órbita
2018	(Fuchs et al. 2018)	Kintex Ultrascale+ XCKU5P	Pequenos Satélites ($< 50 kg$)	OBC
2018	(Hernández-Calviño et al. 2018)	Zynq-7000 XC7Z020	Nanossatélite	Processamento de Imagem
2019	(Bahl et al. 2019)	Cyclone V 5CSXC6	CubeSat	IA: Segmentação de Imagem
2019	(Orlandić et al. 2019)	Zynq-7000 XC7Z035	Geral	Processamento de Imagem
2019	(Adams et al. 2019)	SmartFusion 2	CubeSat	OBC
2020	(Urban et al. 2020)	SmartFusion 2	CubeSat	Reconhecimento e Monitoramento de Radiação
2020	(Tsigkanos et al. 2020)	Zynq-7000 XC7Z045	CubeSat	Processamento de Imagem
2020	(Pérez et al. 2020)	Zynq Ultrascale+[a]	Geral	OBC
2020	(Marin & Bang 2020)	Zynq-7000 XC7Z020	CubeSat	Determinação de Atitude
2021	(Abdullah et al. 2021)	Zynq Family[a]	CubeSat	Sistema de Comunicação
2021	(Zhang et al. 2021)	Zynq-7000 XC7Z035	Geral	IA: Detecção de Objeto
2022	(Papatheofanous et al. 2022)	Zynq UltraScale+ XCZU7EV	Geral	IA: Segmentação de Imagem
2022	(Acosta et al. 2022)	Zynq Family[a]	CubeSat	OBC
2022	(Pitonak et al. 2022)	Zynq-7000 XC7Z020	Geral	IA: Classificação de Imagem
2022	(Abderrahmane et al. 2022)	Cyclone V[a]	Nanossatélite	IA: Processamento de Imagem
2023	(Coca & Datcu 2023)	Zynq Ultrascale+ XCZU9EG	Pequenos Satélites	IA: Processamento de Imagem
2023	(Lüdtke et al. 2023)	Zynq-7000 XC7Z020	CubeSat	Processamento de Bordo
2023	(Pranitha & Kavya 2023)	Zynq-7000 XC7Z020	Geral	Processamento de Imagem

[a]Numeração não descrita no trabalho

Fonte: O próprio autor.

Baseado na Tabela 2.1, percebe-se uma propensão para o uso de SoCs das famílias Zynq-7000 e Zynq UltraScale+. Isso ocorre devido ao seu vasto uso em missões ao longo dos anos e à disponibilidade de publicações que abordam técnicas de tolerância a falhas sobre os elementos desses dispositivos. Ademais, os dados da tabela destacam o uso de arquiteturas de processamento híbrido em aplicações de processamento de imagens e, mais recentemente, em soluções de IA.

Grande parte dos SoCs apresentados na Tabela 2.1 utilizam o protocolo AMBA AXI para interconexão da unidade de processamento *hard-core* e os elementos na região de lógica reconfigurável (FPGA). Desse modo, os conceitos referentes ao AMBA AXI são descritos a seguir.

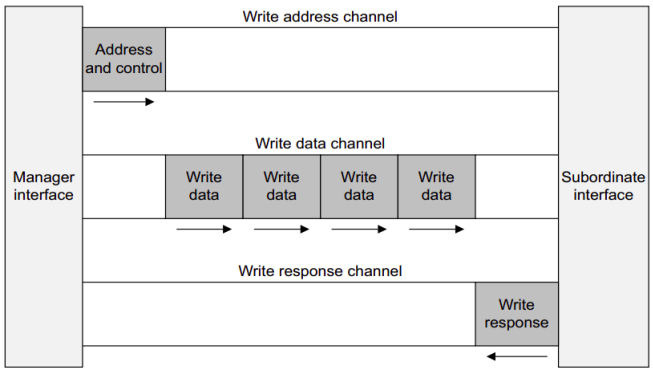
2.5 AMBA AXI

O padrão AMBA foi desenvolvido pela ARM para interconexão de blocos em sistemas em chip. Esse padrão inclui especificações para interfaces AXI e suporta comunicação de alta frequência e alto desempenho (De Sio et al. 2020*a*). De acordo com o documento Xilinx (2017), a quarta versão do protocolo AXI, chamada AXI4, possui três tipos de interfaces: AXI4, AXI4-Lite e AXI4-Stream. A interface AXI4 pode ser usada para requisitos de mapeamento de memória de alto desempenho. A AXI4-Lite é adequada para comunicação de mapeamento de memória com baixa taxa de transferência. Por fim, a AXI4-Stream é útil para dados de *streaming* de alta velocidade.

O AXI define cinco canais independentes: endereço de leitura, dado de leitura, endereço de escrita, dado de escrita e resposta de escrita (ARM 2013). Um canal de endereço contém informações sobre a natureza do dado a ser transferido (ARM 2013). A transferência de dado entre o componente mestre (gerente) e o componente escravo (subordinado) ocorre de duas maneiras: por meio do canal de escrita de dado para transferência do mestre para o escravo, em que o canal de resposta de escrita é utilizado pelo escravo para sinalizar ao mestre que a transferência foi concluída; e por meio do canal de leitura de dado para transferência do escravo para o mestre (ARM 2013, ARM 2021).

Na Figura 2.6 é apresentado o diagrama da transação de escrita.

Figura 2.6: Arquitetura dos canais de escrita para o AMBA AXI

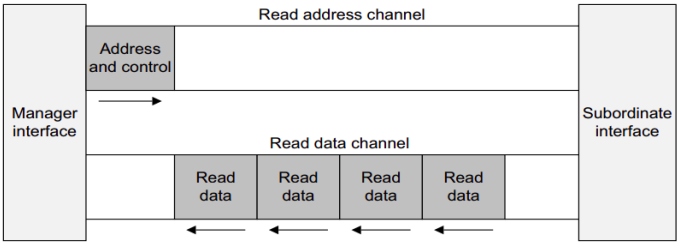


Fonte: ARM (2021).

Na Figura 2.7 é apresentado o diagrama da transação de leitura.

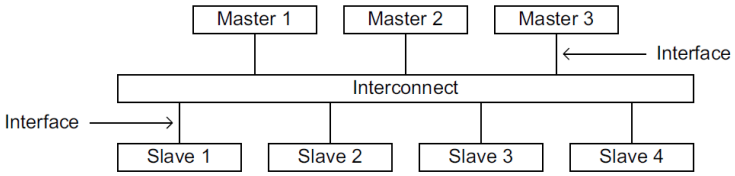
Em um sistema típico composto por vários dispositivos mestres e escravos, a conexão é realizada por meio de uma estrutura de interconexão (ARM 2013), como a da Figura 2.8. As especificações do protocolo AXI contêm a definição que serve como base para as interfaces entre um mestre e a interconexão, entre um escravo e a interconexão e entre um mestre e um escravo (ARM 2013, ARM 2021, Xilinx 2017). Geralmente, uma das seguintes topologias de conexão é utilizada: barramentos de dados e endereço compartilhados; barramentos de endereço compartilhados e múltiplos barramentos de dados; multicamada, com múltiplos barramentos de endereços e múltiplos barramentos de dados (ARM 2021).

Figura 2.7: Arquitetura dos canais de leitura para o AMBA AXI



Fonte: ARM (2021).

Figura 2.8: Interfaces e interconexão AXI para um sistema típico



Fonte: ARM (2013).

Embora a implementação de interconexões AXI em FPGA esteja sujeita aos efeitos da radiação, suas vasta documentação e o fato de ser um padrão aberto (livre de *royalties*) torna seu uso atraente para o desenvolvimento de SoCs para CubeSats.

2.6 Resumo

Neste capítulo, foram dispostos conceitos e trabalhos que fundamentam a proposta de tese. Foram descritas as características dos CubeSats, bem como os efeitos da radiação sobre componentes eletrônicos em ambiente espacial e as técnicas para mitigação de erros provenientes desses efeitos. Ademais, apresentou-se uma contextualização sobre o processamento híbrido em missões com satélites.

Notadamente, o uso de SoCs COTS que integram FPGA e CPU em um único chip (como Zynq-7000) tem se tornado comum. Esses sistemas permitem que diferentes técnicas de tolerância a falhas sejam utilizadas. A capacidade de processamento, a possibilidade de reconfiguração (incluindo reprogramação em órbita) e as características de consumo energético tornam os sistemas compostos por pela integração de CPU e FPGA atrativos no projeto de missões com satélites que apresentam restrições de tamanho e energia.

Por fim, foram expostas algumas das definições relacionadas ao protocolo AMBA AXI, uma vez que ele é utilizado na maioria das arquiteturas de processamento híbrido discutidas neste capítulo. A ampla utilização de tal protocolo também é o motivo pelo qual ele foi escolhido para os experimentos de implementação em *hardware* da abordagem DMR baseada em paridade.

Capítulo 3

Estado da Arte

Os efeitos da radiação sobre FPGAs baseados em SRAM e as possíveis técnicas de mitigação têm sido investigadas em diferentes trabalhos (Ceschia et al. 2003, Asadi & Tahoori 2005, Siegle et al. 2015). As técnicas de tolerância geralmente se baseiam na reconfiguração parcial ou total da memória de configuração ao serem detectados *bitflips*. Contudo, poucos trabalhos investigam as consequências das falhas em interfaces de comunicação que interconectam SoCs compostos por processador *hard-core* e lógica reconfigurável (FPGA).

Grande parte dos SoCs citados na Seção 2.4 utilizam o AMBA AXI como interface de comunicação. Sendo assim, em implementações de arquitetura de processamento híbrido (CPU+FPGA), o AMBA AXI tem sido empregado para interconectar a unidade de processamento *hard-core* e as aplicações desenvolvidas na região do FPGA. Como exemplo, os trabalhos de Pérez et al. (2020) e Iturbe et al. (2015), descritos na referida seção, fazem uso do AXI para comunicação de dados entre processador ARM e aplicações em FPGA. Nesses trabalhos, os eventuais erros na memória de configuração do FPGA, onde também se encontra a implementação do AXI, são corrigidos por meio do IP (*Intellectual Property*) *core* Xilinx SEM (*Soft Error Mitigation*), ou, em alguns casos, solucionados pela completa reconfiguração da memória de configuração.

Benevenuti & Kastensmidt (2018) descrevem a avaliação de confiabilidade de um acelerador de *hardware* e diferentes interfaces de comunicação, implementadas em um SoC Zynq-7000. As interfaces foram utilizadas para comunicação entre processador embarcado e o módulo de aceleração e tiveram como base o AMBA AXI. Várias configurações, consistindo de diferentes possibilidades de conexão dos módulos do sistema, foram selecionadas para a avaliação. Entre os módulos, foram consideradas versões do acelerador sem tolerância a falhas e com uso de TMR, memória compartilhada implementada no FPGA e módulo IP de acesso direto à memória (DMA, *Direct Memory Access*) com AXI. Uma área específica da memória de configuração, ocupada por um módulo de cada vez, foi isolada para injeção de falhas. Dessa maneira, os autores analisaram quais módulos contribuíram mais com a redução de confiabilidade do sistema quando falhas foram aplicadas. Como resultado, eles verificaram que o módulo de aceleração sem TMR tem um grande impacto na confiabilidade do sistema e que ao aumentar a confiabilidade com TMR há um maior impacto proveniente dos módulos que possuem interfaces AXI. Desse modo, concluem que pode ser necessária a substituição das interconexões AXI por versões tolerantes a falhas para aumento da confiabilidade do sistema. Embora esse trabalho

apresente resultados importantes, o foco da aplicação de técnicas de tolerância é o módulo de aceleração de *hardware*, de modo que o módulo de interconexão AXI não possui tolerância e é analisado em conjunto com outros elementos do sistema.

Em um trabalho mais recente, De Sio et al. (2020*a*) apresentam uma análise de confiabilidade do módulo IP AXI *Interconnect* implementado em um SoC Zynq-7000 para conexão do sistema de processamento (ARM) e da lógica reconfigurável (FPGA), que estão integrados no mesmo chip. Para isso, foram injetadas falhas nas seções da memória de configuração do FPGA ocupadas pelo módulo. Ademais, foram desenvolvidos dois aceleradores de *hardware*, os quais foram implementados no FPGA em conjunto com o AXI *Interconnect*. Durante a rotina de testes, o sistema de processamento ficou encarregado de configurar os aceleradores e estimular cada um com diferentes entradas. As falhas detectadas foram observadas e classificadas da seguinte maneira: ambos aceleradores pararam de funcionar; ambos aceleradores apresentaram cálculos defeituosos; apenas um acelerador continuou funcionando de forma correta; apenas um acelerador foi acessado pelo sistema de processamento, mas retornou resultados errados. Além disso, observou-se que os valores retornados foram os mesmos quando ambos os aceleradores produziram cálculos errados. Diante dos resultados, os autores concluem que o módulo AXI *Interconnect* pode ser fonte de erros em arquiteturas que utilizam aceleração em *hardware* e que pode impedir técnicas de mitigação baseadas em replicação de módulos de *hardware*, pois não há como detectar as falhas comparando os resultados retornados uma vez que são iguais e errôneos. Em De Sio et al. (2020*b*), os autores aprofundam a investigação das implicações de SEUs induzidos no módulo de interconexão AXI.

Esses trabalhos demonstram a necessidade da adoção de técnicas para mitigação da radiação sobre o AMBA AXI com intuito de aumentar a confiabilidade de sistemas que utilizam essa interface de comunicação. Alguns trabalhos mencionam o uso de abordagens clássicas como código de Hamming, CRC e paridade em canais de comunicação. Bertozzi et al. (2005) exploram a capacidade de detecção e o *tradeoff* entre energia/confiabilidade de diferentes esquemas de codificação para links de comunicação em chip. Os autores focam em esquemas com códigos Hamming, *Cyclic Redundancy Check* (CRC) e um simples código de bit de paridade único. Tais esquemas foram aplicados à uma implementação do AMBA AHB (*Advanced High-performance Bus*). Nesse trabalho, os códigos com correção baseados em código Hamming permitem a correção de erros únicos, enquanto aqueles baseados em CRC e bit de paridade usam retransmissão como estratégia de recuperação. Os autores apresentam um interessante resultado para o subconjunto de códigos analisados (CRC e Hamming), concluindo que, de um ponto de vista energético, a retransmissão acaba sendo mais eficiente do que a correção. Apesar da capacidade de detecção de códigos CRC, suas implementações básicas tratam apenas da detecção de erros e sua complexidade depende da escolha do polinômio gerador.

Graham et al. (2009) examinaram o impacto de falhas transientes no componente de interconexão baseado no AMBA AXI3, terceira versão do AXI que apresenta características e sinais correspondentes ao AXI4. Os autores classificaram o impacto de um erro em cada sinal AXI, considerando aqueles que causam *deadlock*, corrupção de memória e comportamento indefinido. Como forma de mitigar tais erros, os autores propuseram três técnicas de tolerância, sendo uma para os sinais de controle, uma para os sinais de ende-

reço e outra para os sinais de dados. A proteção dos sinais de controle foi realizada por meio de bits de paridade, os quais são acomodados nos sinais extras do AXI. Esses sinais, referenciados, nos documentos de especificação (ARM 2013), como sinais definidos pelo usuário (*USER*), estão disponíveis nos cinco canais AXI e são utilizados para expansão do protocolo, permitindo o transporte de informações especificadas pelos projetistas. No que se refere aos sinais de endereço, apenas parte dos bits de endereço foram protegidos por paridade (bits menos significativos) e o mapa de memória foi reorganizado para que um erro de bit nos bits mais significativos implicasse no roteamento da transação em uma região não mapeada, o que, de acordo com a especificação AXI, resulta em uma resposta de erro. A reorganização do mapa de memória implicou em uma redução do espaço de endereçamento. O último esquema apresentado pelos autores foi utilizado para a proteção dos dados transmitidos. Para isso, foi aplicado um ECC, o qual foi enviado nos sinais extras do AXI no ciclo de *clock* posterior ao do dado transmitido, o que, de maneira geral, caracteriza um atraso na transmissão. Desse modo, segundo o esquema proposto, ao ser detectado um erro, o mestre para por um ciclo de *clock* para realização da correção. Esse procedimento acaba afetando a latência do sistema. Notadamente, os esquemas apresentados pelos autores fazem uso dos sinais extras do AXI. Contudo, esses sinais não são suportados nas interfaces do tipo AXI-Lite e AXI-Stream (ARM 2013).

Mach et al. (2023) apresentam uma revisão sobre abordagens existentes para proteção contra falhas transientes (SETs) de barramentos de interconexão *on-chip*. O foco do trabalho são os processadores embarcados usados em aplicações como as do setor automotivo ou espacial, que têm características de segurança/missão-crítica. As abordagens são classificadas em três grupos: redundância de informação, temporal e espacial (*hardware*). Os autores afirmam que a implementação de redundância de informação não é simples, mas ela é viável em aplicações de alto desempenho. Eles destacam o uso de paridade e códigos do tipo *Error Detection and Correction* (EDAC), como o *Single Error Correction Double Error Detection* (SECDED). A redundância temporal requer uma análise precisa das durações de *glitches* transientes e é mais adequada para aplicações de baixo desempenho. Por outro lado, a redundância espacial (como a TMR) têm um maior impacto na área e no consumo de energia, mas apresenta uma implementação simples. Os autores concluem que a escolha da abordagem de proteção depende de fatores como caso de uso do processador, do ambiente operacional e da possibilidade de modificações em componentes externos ao processador. Em um trabalho posterior, Mach et al. (2024) descrevem a integração de proteção contra falhas transientes de barramento na interface do core RISC-V Hardisc, o qual utiliza o AMBA AHB-Lite. Os autores tomaram como base técnicas de redundância de informação em conjunto com redundância de *hardware* (espacial). Os sinais de dados são protegidos por um EDAC com características de SECDED, enquanto os sinais de controle são protegidos por TMR e bits de paridade. Os esquemas de proteção utilizam linhas extras de barramento para transmissão dos bits do EDAC e dos bits de paridade.

A redundância de barramentos internos para periféricos de baixa velocidade foi explorada por Lázaro et al. (2024). Em seu trabalho, os autores propõem uma solução composta por um IP *core* denominado *AXILiteRedundant* e um sistema redundante. O IP *core* foi utilizado para triplicar os sinais de uma interface escrava para interfaces mestres, as quais

foram utilizadas com a função de se conectar aos periféricos redundantes externos. Além disso, os valores provenientes dos múltiplos mestres para o escravo foram selecionados de um votador TMR. O IP Core também foi capaz de indicar ao processador se os três mestres apresentavam valores iguais, se um deles possuía valor diferente (passível de correção pelo votador), ou se todos apresentavam valores distintos, sinalizando um erro. Os autores descrevem as modificações realizadas no padrão AXI para a aplicação do mecanismo de redundância e os elementos que compõem o sistema redundante. Por fim, eles concluem que a proposta oferece proteção à interconexão e aos IPs periféricos, requer poucos recursos de FPGA para implementação e apresenta um consumo de energia desprezível. Embora esse trabalho se destaque como exemplo de aplicação de redundância em interfaces de comunicação, a falta de alguns elementos descritivos relacionados à implementação tornaram difícil a comparação com a abordagem proposta neste texto de tese.

No que se refere à DMR com paridade, em Borecký et al. (2016), um método de paridade (baseado em cascata) é usado com o objetivo de cobrir todas falhas possíveis no conteúdo do roteamento e *Look-Up Table* (LUT) usando um circuito *totally-self-checking* com um gerador de paridade. A estrutura é duplicada de modo que caso uma falha seja detectada em um FPGA, a saída da que não foi afetada e habilitada. Clark et al. (2011) aplicaram DMR para detectar erros no conjunto de registradores (RF, *Register File*) do estágio *write-back* do *pipeline*. Nesse trabalho, a correção de SEU é realizada por meio de paridade, a qual detecta *nibbles* (conjunto de 4 bits) de entrada do RF que estão corretos em uma cópia e errados na outra. Os conteúdos corretos são então copiados da cópia DMR com paridade correta para a outra. Esses trabalhos descrevem a aplicação de DMR com paridade em arquiteturas de componentes de memória.

3.1 Resumo

Neste capítulo, foram descritos trabalhos que destacam a necessidade de aplicação de tolerância a falhas em estruturas de comunicação em sistemas em chip. Uma vez que o protocolo AMBA AXI é amplamente utilizado em arquiteturas compostas por processador *hard-core* e lógica reconfigurável (FPGA) no mesmo chip, os trabalhos discutidos focam nesse tipo de protocolo.

Dada a suscetibilidade das interfaces de comunicação aos efeitos da radiação, diferentes estratégias têm sido propostas para aumentar a confiabilidade na transmissão de dados na comunicação *on-chip*. A Tabela 3.1 contém um resumo com as técnicas para proteção dos dados citadas nos trabalhos apresentados em comparação com a técnica proposta nesta tese.

Considerando, especificamente, a proteção na transmissão de dados, os trabalhos da Tabela 3.1, exceto Lázaro et al. (2024), utilizam redundância da informação como base de proteção para os dados transmitidos. Como discutido anteriormente, códigos fundamentados nesse tipo de redundância podem implicar em um aumento na complexidade da lógica de mitigação de erros. Além disso, linhas extras de barramento podem ser necessárias para acomodar os bits associados aos códigos utilizados. Entretanto, alguns tipos de interface, como as implementadas com base no AXI4-Lite, não suportam linhas extras,

Tabela 3.1: Resumo dos trabalhos relacionados em comparação com a proposta

Referência	**Redundância**	**Técnica**	**Protocolo Testado**
(Bertozzi et al. 2005)	Informação	Hamming, CRC e Paridade	AMBA AHB
(Graham et al. 2009)	Informação	ECC	AMBA AXI3
(Mach et al. 2024)	Informação	EDAC	AMBA AHB-Lite
(Lázaro et al. 2024)	*Hardware*	TMR	AMBA AXI4-Lite
Proposta	*Hardware* e Informação	DMR e Paridade	AMBA AXI4-Lite

Fonte: O próprio autor.

de modo que as soluções apresentadas nos trabalhos discutidos têm aplicação limitada nessas situações.

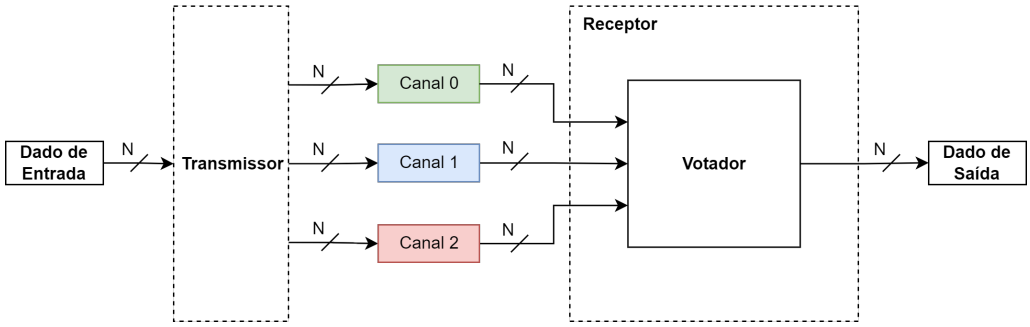
Durante a revisão do estado da arte, não foram encontrados trabalhos relacionados ao uso específico de DMR ou DMR com paridade para transmissão de dados. A abordagem apresentada nesta tese é combinacional e o dado original é dividido em grupos com paridades associadas. Desse modo, pode-se mascarar/corrigir erros em grupos diferentes ao considerar expressões lógicas desenvolvidas com base nas paridades, como pode ser visto ao longo do Capítulo 4.

Capítulo 4

PDMR: Uma Abordagem DMR Baseada em Paridade

O princípio básico da redundância de *hardware* na transmissão de dados consiste na replicação de elementos da estrutura de comunicação com o intuito de fornecer tolerância a falhas por meio de caminhos redundantes. Sendo assim, em uma estrutura com TMR, tem-se três canais por onde os dados redundantes trafegam e um circuito votador que tem por função definir a saída com base na maioria do valores binários de entrada. Um exemplo para um dado de entrada de N bits, sendo N um número inteiro, está representado na Figura 4.1.

Figura 4.1: Diagrama de transmissão de dados com TMR



Fonte: O próprio autor.

Inicialmente, o dado binário de entrada é triplicado ao passar pelo transmissor e repassado aos canais de transmissão, os quais são considerados como suscetíveis aos efeitos da radiação. No receptor, o circuito lógico votador executa operações bit a bit e se encarrega de determinar a saída com base nos bits de entrada de cada canal.

O dado de entrada a ser transmitido é composto por N bits e representado por

$$b_0 b_1 b_2 \ldots b_{L-1} b_L. \tag{4.1}$$

em que $L = N - 1$. O bit localizado na k-ésima posição do dado binário é escrito como b_k, de modo que $0 \leq k \leq L$. Assim, $k = 0$ corresponde ao bit mais significativo (MSB, do inglês, *Most Significant Bit*), enquanto L representa o bit na posição menos significativa

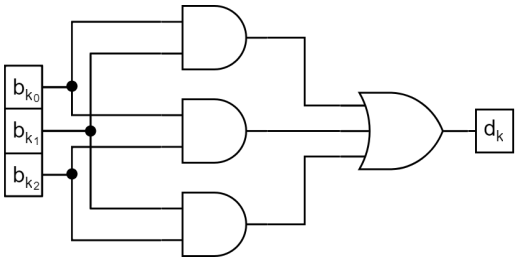
(LSB, do inglês, *Least Significant Bit*). Após a passagem pelos canais de transmissão os dados são simbolizados por

$$b_{0_j} b_{1_j} b_{2_j} \ldots b_{(L-1)_j} b_{L_j}, \tag{4.2}$$

em que j indica o canal de onde os bits são provenientes.

O circuito lógico votador para um bit de entrada está ilustrado na Figura 4.2, em que os subscritos 0, 1 e 2 indicam que os bits são provenientes dos canais 0, 1 e 2, respectivamente, enquanto d_k é o k-ésimo bit de saída resultante do votador. Observa-se que o circuito votador pode ser construído por meio da expressão lógica 2.1. Ademais, a quantidade de portas lógicas utilizadas para obtenção do dado de saída depende da quantidade de bits do dado binário de entrada (N).

Figura 4.2: Circuito lógico votador TMR para um bit de entrada



Fonte: O próprio autor.

Como exemplo, para um dado de três bits ($N = 3$), $0 \leq k \leq 2$, tem-se três circuitos como da Figura 4.2, totalizando nove portas lógicas *AND* de duas entradas (9×2 entradas de porta) e três portas lógicas *OR* de três entradas (3×3 entradas de porta). De modo geral, a quantidade de portas lógicas (Q_{TMR}) e de entradas de porta (I_{TMR}) para o votador TMR são determinadas pelas equações:

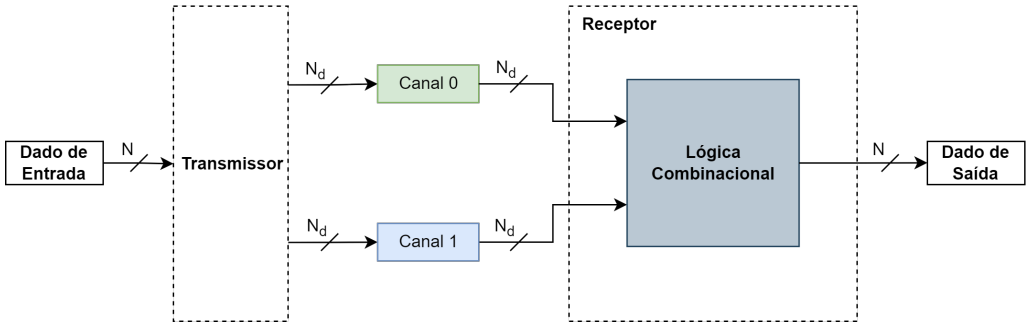
$$Q_{TMR} = 4 \times N, \tag{4.3}$$

$$I_{TMR} = 9 \times N. \tag{4.4}$$

Uma vez que a aplicação de TMR tende a aumentar os recursos de *hardware* utilizados e o consumo de energia, como já descrito anteriormente, propõe-se uma alternativa por meio da utilização de redundância de *hardware* com DMR. Nesse caso, a estrutura de transmissão com DMR tem como base o diagrama da Figura 4.3, em que estão dispostos dois canais por onde os dados são transmitidos.

Em uma aplicação de DMR, a simples duplicação de elementos da estrutura de comunicação permite apenas que sejam detectados erros entre os bits de dados provenientes dos canais. Isso pode ser feito por meio de um circuito lógico capaz de comparar os dados no receptor. Portanto, para se obter uma correção em caso de erro, a inclusão de uma lógica para retransmissão dos dados se faz necessária. Em outras palavras, ao verificar que os dados provenientes dos canais são diferentes, o sistema deve fazer com que o dado binário de entrada seja retransmitido, o que caracteriza a aplicação de redundância temporal. Como descrito na Seção 2.3, esse tipo de redundância aumenta o tempo de transmissão,

Figura 4.3: Diagrama de transmissão de dados com DMR



Fonte: O próprio autor.

o que é uma desvantagem em sistemas em que o tempo é crítico. Desse modo, a abordagem proposta neste trabalho se baseia na utilização de bits de paridade com o intuito de corrigir/mascarar erros e minimizar possíveis retransmissões de dados.

Nesse sentido, no diagrama da Figura 4.3, N_{d} é o número inteiro que corresponde ao número de bits da palavra-código formada pelo conjunto de bits do dado de entrada e pelos bits de paridade. Ou seja,

$$N_{\mathrm{d}} = N + N_{\mathrm{p}}, \tag{4.5}$$

em que N_{p} é um número inteiro que representa a quantidade de bits de paridade da palavra-código.

A abordagem proposta e sua metodologia de implementação são apresentadas nas seções seguintes deste capítulo.

4.1 Regras de Formação da Palavra-Código

A proposta de DMR baseada em paridade (PDMR) está fundamentada na duplicação de elementos da estrutura de comunicação e na aplicação de uma variação do código de paridade básico. Ao invés de associar um único bit de paridade para o dado de entrada de N bits (referenciados como bits de informação), é realizada a divisão do dado em grupos de bits de mesmo tamanho (Koren & Krishna 2007). Cada um desses grupos é composto por uma quantidade fixa de bits de informação, simbolizada por M, e um bit de paridade. Como descrito anteriormente, o número total de bits de paridade na palavra-código é representado por N_{p}.

Diferentes regras de formação da palavra-código podem ser definidas para divisão dos grupos. Neste trabalho, serão consideradas duas regras, as quais estão descritas a seguir.

4.1.1 Regra *RF1*

A regra de formação geral é direcionada às estruturas de comunicação que não apresentam restrições relacionadas à largura do barramento de transmissão de dados. Com

base no diagrama da Figura 4.3, para um dado de entrada com N bits, tem-se uma palavra-código com N_{d} bits, a qual é duplicada e transmitida por meio dos canais 0 e 1 (por simplicidade, referenciados como *Ch0* e *Ch1*). O número de bits de paridade adicionados no transmissor pode ser obtido pela equação 4.5, $N_{\mathrm{p}} = N_{\mathrm{d}} - N$.

A quantidade de bits de informação de cada grupo é dada por

$$M = N/N_{\mathrm{p}}, \tag{4.6}$$

em que N_{p} deve ser um divisor de N e $N_{\mathrm{p}} \leq N$, de modo que M é um número inteiro. Como cada grupo deve ser formado por um bit de paridade e M bits de informação, a quantidade de bits de cada grupo (N_{g}) é dada por:

$$N_{\mathrm{g}} = M + 1 \tag{4.7}$$

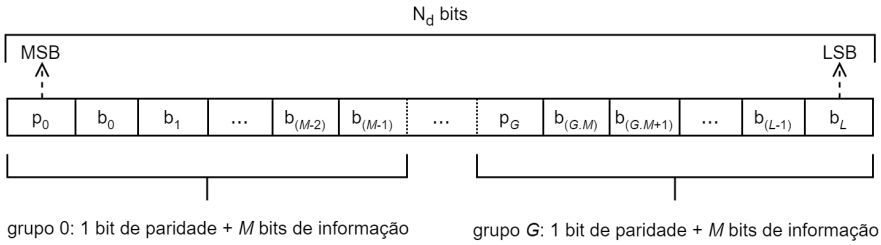
Além disso, uma vez que cada grupo possui apenas uma paridade, N_{p} também define a quantidade de grupos da palavra-código. Dessa maneira, os grupos variam do 0 ao G, com $G = N_{\mathrm{p}} - 1$.

As paridades são então obtidas com base em operações *XOR* entre os bits de informação de cada grupo, o que caracteriza um código de paridade par. Seguindo a representação binária em 4.1, tem-se:

$$\begin{aligned}
p_0 &= b_0 \oplus b_1 \oplus \cdots \oplus b_{(\mathrm{M}-2)} \oplus b_{(\mathrm{M}-1)}, \\
p_1 &= b_{\mathrm{M}} \oplus b_{(\mathrm{M}+1)} \oplus \cdots \oplus b_{(2\mathrm{M}-2)} \oplus b_{(2\mathrm{M}-1)}, \\
p_2 &= b_{2\mathrm{M}} \oplus b_{(2\mathrm{M}+1)} \oplus \cdots \oplus b_{(3\mathrm{M}-2)} \oplus b_{(3\mathrm{M}-1)}, \\
&\vdots \\
p_{\mathrm{G}} &= b_{\mathrm{G.M}} \oplus b_{(\mathrm{G.M}+1)} \oplus \cdots \oplus b_{(\mathrm{L}-1)} \oplus b_{\mathrm{L}}.
\end{aligned}$$

Na Figura 4.4, tem-se uma representação do formato de uma palavra-código de N_{d} bits para a proposta PDMR.

Figura 4.4: Formato de uma palavra-código de N_d bits com bits de paridade



Fonte: Adaptado de Alves et al. (2024).

De modo geral, utilizando o índice i para as paridades, com $0 \leq i \leq G$, a i-ésima

paridade pode ser obtida por

$$p_{\mathrm{i}} = b_{(\mathrm{i.M})} \oplus b_{(\mathrm{i.M+1})} \oplus \cdots \oplus b_{(\mathrm{i.M+M-2})} \oplus b_{(\mathrm{i.M+M-1})}. \tag{4.8}$$

Assim como destacado para 4.2, após a passagem pelos canais de transmissão, os bits de paridade são simbolizados por $p_{\mathrm{i_j}}$.

Para ilustrar a utilização dessa regra de formação, considere um dado de entrada com 12 bits (N), $b_0b_1b_2b_3b_4b_5b_6b_7b_8b_9b_{10}b_{11}$, e um barramento de dados com largura de 16 bits (N_{d}), de modo que:

$$N_{\mathrm{p}} = N_{\mathrm{d}} - N = 16 - 12 = 4.$$

Portanto,

$$M = N/N_{\mathrm{p}} = 12/4 = 3.$$

Assim, cada grupo é formado por 3 bits de informação e 1 de paridade e a quantidade total de grupos é igual 4. No que se refere aos bits de paridade da palavra-código, por meio da equação 4.8, obtêm-se:

$$\begin{aligned} p_0 &= b_0 \oplus b_1 \oplus b_2, \\ p_1 &= b_3 \oplus b_4 \oplus b_5, \\ p_2 &= b_6 \oplus b_7 \oplus p_8, \\ p_3 &= b_9 \oplus b_{10} \oplus b_{11}. \end{aligned}$$

4.1.2 Regra *RF2*

Essa regra foi formulada com o objetivo de manter a compatibilidade com a largura do barramento de dados de protocolos *on-chip* já consolidados, os quais, em geral, possuem um tamanho definido por uma potência de 2 (e.g., 8, 16, 32, 64). Nesta regra de formação, define-se N_d e N_p como potências de dois, tais que:

$$N_{\mathrm{d}} > N, \tag{4.9}$$

$$N_{\mathrm{p}} \geq N_{\mathrm{d}}. \tag{4.10}$$

As demais condições descritas para *RF1* continuam válidas, assim como a estrutura da Figura 4.4.

4.2 A Paridade como Critério de Seleção

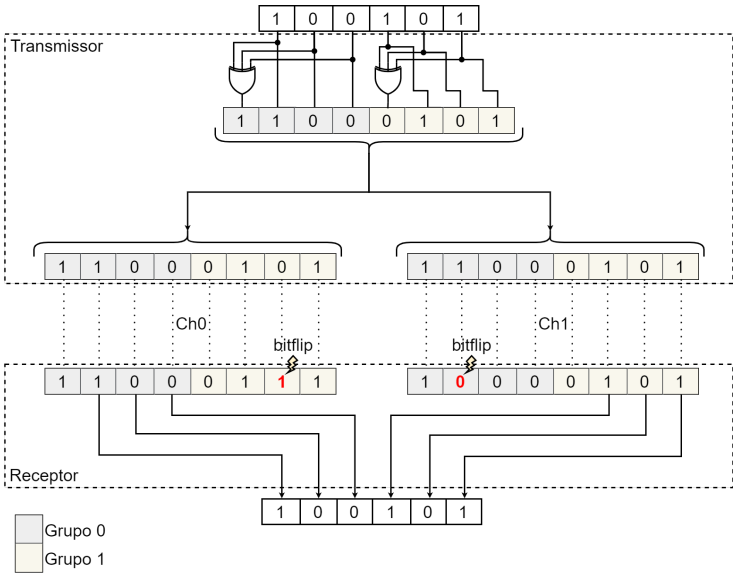
As regras de formação permitem que o dado original seja dividido em grupos e que seja associado um bit de paridade por grupo, o que caracteriza a construção da palavra-código. No caso do diagrama da Figura 4.3, a palavra-código é duplicada e cada cópia é repassada para um dos canais. Após a passagem pelos canais, o dado original deve

ser recuperado e disponibilizado na saída para o componente que deve recebê-lo. Desse modo, busca-se implementar uma solução lógica que possibilite a formação do dado de saída com base em bits de informação provenientes de grupos de canais distintos.

Como exemplo, suponha que o dado original é dividido em dois grupos, 0 e 1, com uma paridade cada. Ao passar pelo receptor, a primeira parte do dado de saída pode ser formada pelos bits de informação do grupo 0 disponibilizados pelo canal 0, enquanto a segunda parte pode ser obtida do grupo 1 do canal 1. Para que isso ocorra, os bits de paridade e de informação devem ser checados no receptor, indicando uma possível ausência de erros nos respectivos grupos. Destaca-se que, com base no código de paridade par, a ocorrência de 2 ou múltiplos de 2 erros tem como resultado a formação de um dado de saída diferente do original.

Um diagrama de fluxo de bits para o dado binário $0b100101$ está apresentado na Figura 4.5. Os bits de informação do dado $0b100101$ são divididos em duas partes. A primeira parte é composta pelos bits $0b100$ e a segunda pelos bits $0b101$. As respectivas paridades, $0b1$ e $0b0$, são adicionadas a cada uma das partes para formar os dois grupos que compõem a palavra-código. Em seguida, a palavra é duplicada e disponibilizada aos canais de transmissão. Após a passagem pelos canais, verifica-se a ocorrência de um *bitflip* no segundo grupo de *Ch0* e um outro *bitflip* no primeiro grupo de *Ch*1. Apesar da presença de falhas, é possível recuperar o dado original ao formar o dado de saída por meio dos bits de informação do grupo 0 de *Ch*0 e dos bits de informação do grupo 1 de *Ch*1.

Figura 4.5: Fluxo de bits para o dado binário $0b100101$ transmitido com a PDMR



Fonte: Adaptado de Alves et al. (2024).

A descrição anterior evidencia a necessidade da utilização de uma lógica no receptor capaz de efetuar a seleção dos bits de informação oriundos de grupos de canais distintos. Uma primeira alternativa para a implementação dessa lógica consiste na verificação da

paridade de cada grupo após passagem por um dos canais. Na ausência de erro, seleciona-se os bits de informação do grupo do canal verificado. Caso contrário, considera-se os bits de informação do grupo proveniente do outro canal. Para a checagem das paridades, os bits de paridade devem ser recalculados no receptor e comparados com os bits de paridade dos grupos (i.e., $p_{\mathrm{i_j}}$). De maneira geral, os bits de paridade calculados no receptor são dados por

$$r_{\mathrm{i_j}} = b_{(\mathrm{i.M})_\mathrm{j}} \oplus b_{(\mathrm{i.M+1})_\mathrm{j}} \oplus \cdots \oplus b_{(\mathrm{i.M+M-2})_\mathrm{j}} \oplus b_{(\mathrm{i.M+M-1})_\mathrm{j}}, \tag{4.11}$$

em que *i* indica o grupo ao qual o bit pertence e *j* sinaliza o canal.

A checagem da paridade pode ser efetuada por meio de operações *XOR* (paridade par) entre os bits de informação do grupo e a respectiva paridade, como descrito na Seção 2.3. O resultado dessas operações gera um bit que pode ser utilizado como entrada de seleção para um Multiplexador (*MUX*). Como a primeira alternativa para a implementação da lógica se baseia na verificação das paridades em apenas um dos canais, escolhendo os grupos de *Ch0* para o cálculo da paridade, o bit de seleção pode ser gerado por

$$x_\mathrm{i} = p_{\mathrm{i}_0} \oplus b_{(\mathrm{i.M})_0} \oplus b_{(\mathrm{i.M+1})_0} \oplus \cdots \oplus b_{(\mathrm{i.M+M-2})_0} \oplus b_{(\mathrm{i.M+M-1})_0}. \tag{4.12}$$

Ademais, nota-se que

$$r_{\mathrm{i}_0} = b_{(\mathrm{i.M})_0} \oplus b_{(\mathrm{i.M+1})_0} \oplus \cdots \oplus b_{(\mathrm{i.M+M-2})_0} \oplus b_{(\mathrm{i.M+M-1})_0} \tag{4.13}$$

corresponde à paridade calculada no receptor com base nos bits provenientes de *Ch0*, de maneira que

$$x_\mathrm{i} = p_{\mathrm{i}_0} \oplus r_{\mathrm{i}_0}. \tag{4.14}$$

As expressões 4.8 e 4.14 possibilitam a implementação lógica de soluções capazes de mascarar/corrigir erros na transmissão de dados. Elas correspondem à versão mais simples da proposta PDMR, na qual a verificação da paridade em apenas um dos canais é considerada como suficiente para o critério de seleção.

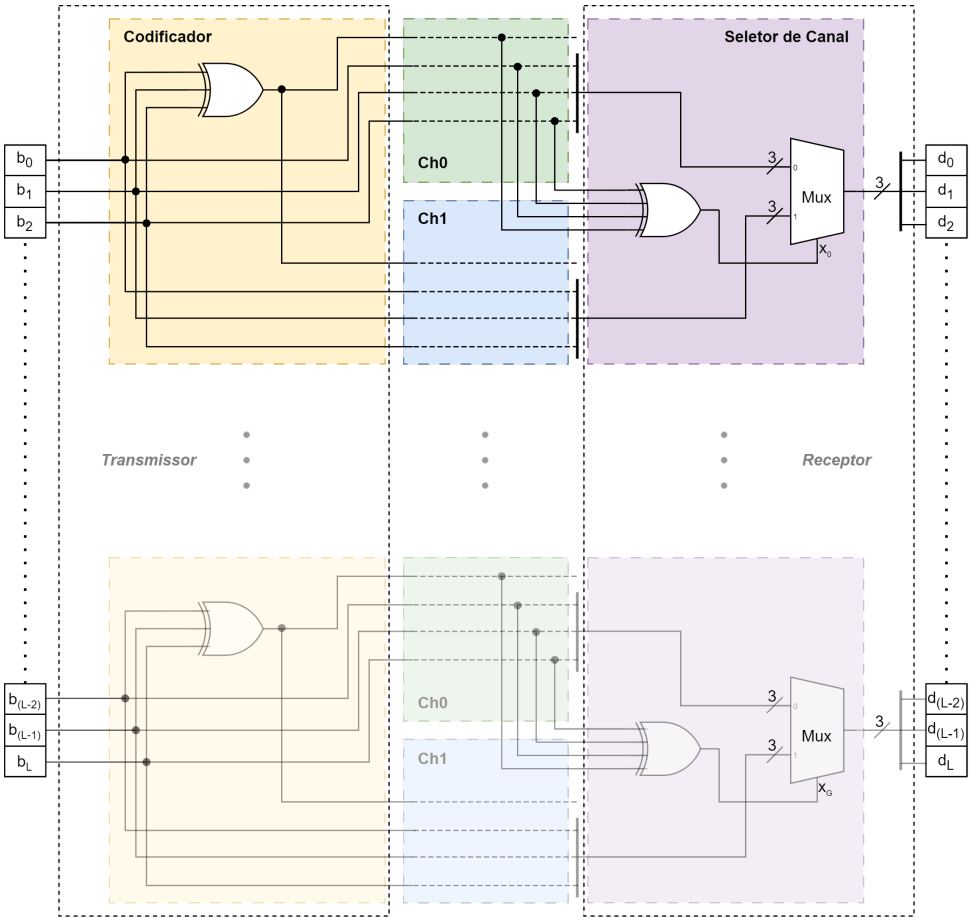
Um exemplo para um dado de entrada com N bits dividido em grupos com três bits de informação ($M = 3$) e um bit de paridade é apresentado na Figura 4.6. As partes em destaque representam os circuitos lógicos envolvidos na geração da paridade e na seleção do canal que direciona os bits de informação do grupo à saída. O módulo *Codificador* adiciona as paridades a cada grupo, formando a palavra-código. No módulo *Seletor de Canal* localizado no receptor, a paridade de cada grupo é checada e o bit gerado é usado como sinal de seleção para o *MUX*.

Com base nos elementos lógicos na Figura 4.6, para cada grupo da palavra-código, tem-se:

- Uma porta *XOR* com três entradas;
- Uma porta *XOR* com quatro entradas;
- Um *MUX* composto por seis portas *AND* com duas entradas, três portas OR com duas entradas e uma porta inversora, conforme exemplo da Figura 4.7.

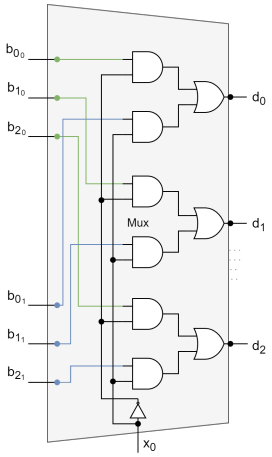
Considerando essas observações, são necessárias 17 portas lógicas e 25 entradas de porta, desconsiderando, por simplicidade, a entrada da porta inversora, para cada grupo

Figura 4.6: Exemplo de implementação da PDMR com seletor de canal simples para um dado de *N* bits dividido em grupos com três bits de informação e um bit de paridade



Fonte: O próprio autor.

Figura 4.7: MUX para seleção de canal do grupo 0 apresentado no exemplo da Figura 4.6



Fonte: O próprio autor.

de bits de informação. De maneira geral, a quantidade de portas *XOR* (2) e de portas inversoras (1) é a mesma (total igual 3) para cada grupo, independente do número de bits de informação do grupo (M), Contudo, o número de portas *AND* e *OR* do *MUX* variam com M, sendo $2 \times M$ e $1 \times M$, respectivamente. Assim, a quantidade de portas (Q_{B}) é obtida por:

$$\begin{aligned} Q_{\mathrm{B}} &= N_{\mathrm{p}} \times (3 + 2M + M), \\ &= N_{\mathrm{p}} \times (3 + 3M), \\ &= 3 \times N_{\mathrm{p}} \times (1 + M), \\ &= 3 \times N_{\mathrm{p}} \times (1 + N/N_{\mathrm{p}}), \end{aligned} \tag{4.15}$$

$$Q_{\mathrm{B}} = 3 \times (N_{\mathrm{p}} + N) = 3 \times N_{\mathrm{d}}. \tag{4.16}$$

Por outro lado, para cada grupo, a quantidade de entradas de porta para as portas *XOR* no transmissor e no receptor dependem de M e N_{g}, respectivamente. Ademais, a quantidade de entradas de porta no *MUX* também dependem de M e pode ser calculada por $6 \times M$. Dessa forma, o número total de entradas de porta (I_{B}) deriva de:

$$\begin{aligned} I_{\mathrm{B}} &= N_{\mathrm{p}} \times (M + N_{\mathrm{g}} + 6 \times M), \\ &= N_{\mathrm{p}}[M + (M + 1) + 6 \times M], \\ &= N_{\mathrm{p}} \times (8 \times M + 1), \\ &= N_{\mathrm{p}}(8 \times (N/N_{\mathrm{p}}) + 1), \end{aligned} \tag{4.17}$$

$$I_{\mathrm{B}} = 8 \times N + N_{\mathrm{p}}. \tag{4.18}$$

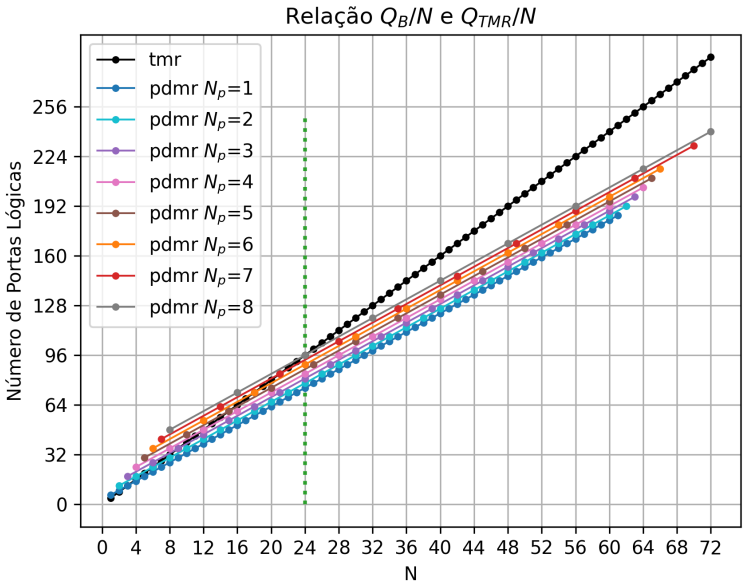
As relações entre valores de N e o número de portas necessárias para implementação com TMR e PDMR se encontram no gráfico da Figura 4.8. Esse gráfico tem como base as equações 4.3 e 4.16. Uma vez que Q_{B} depende da quantidade de bits de paridade, são considerados diferentes valores de N_{p}. Observa-se que, para valores iniciais de N, Q_{TMR} é menor que Q_{B}. Porém, para um determinado valor de N, as quantidades de portas são as mesmas. Ao igualar as equações 4.3 e 4.16, constata-se que isso ocorre quando $N = 3 \times N_{\mathrm{p}}$. Desse valor em diante, Q_{TMR} passa a ser maior do que Q_{B}. A linha tracejada na vertical indica o ponto em que $N_{\mathrm{p}} = 8$ e $N = 24$, de modo que, a partir desse ponto, $Q_{\mathrm{TMR}} > Q_{\mathrm{B}}$ para qualquer valor de N_{p}, com $1 \leq N_{\mathrm{p}} \leq 8$.

No que se refere ao número de entradas de porta para TMR e para PDMR, a relação com N pode ser determinada por meio das equações 4.4 e 4.18, respectivamente. O gráfico dessa relações está ilustrado na Figura 4.9. Nesse caso, os valores de I_{TMR} e I_{B} são iguais apenas quando $N = N_{\mathrm{p}}$ (expressão obtida ao igualar as equações 4.4 e 4.18). Desse ponto em diante, $I_{\mathrm{TMR}} > I_{\mathrm{B}}$ para N_{p} variando entre 1 e 8.

4.3 O Bit de Indicação de Erro

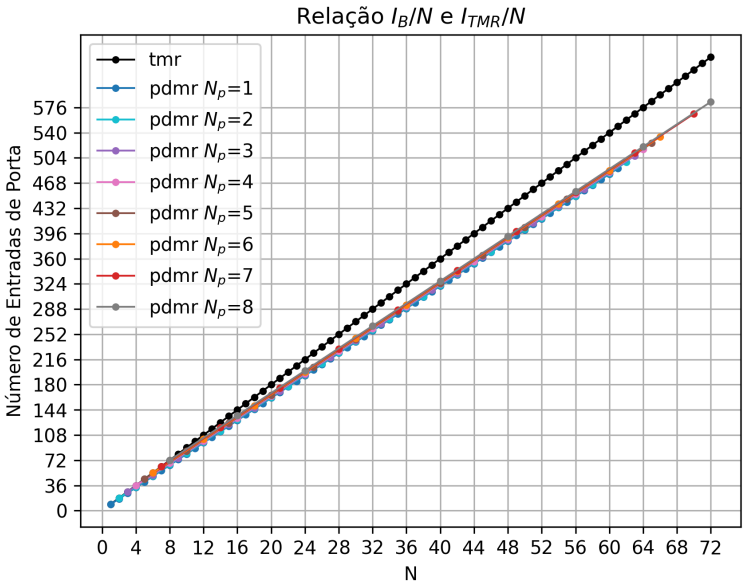
O critério de seleção apresentado na seção anterior permite que erros sejam mascarados por meio da duplicação do canal e do uso da paridade. Contudo, os bits de informação

Figura 4.8: Relação entre o número de portas lógicas e bits de informação (N) para diferentes valores de N_p



Fonte: O próprio autor.

Figura 4.9: Relação entre o número de entradas de porta e bits de informação (N) para valores distintos de N_p



Fonte: O próprio autor.

do canal selecionado para a saída também podem ter sofrido uma mudança de estado durante a transmissão. Consequentemente, o dado de saída ainda pode conter erros.

Com o intuito de minimizar a quantidade de erros na saída, propõe-se a inclusão de uma lógica capaz de detectar e indicar possíveis erros nos grupos da palavra-código. Essa lógica é construída com base nos bits de paridade transmitidos e nos calculados no receptor (expressão 4.11). Como há dois canais em que os bits são transmitidos, tem-se um bit de paridade transmitido e um calculado no receptor por canal para cada grupo, totalizando quatro bits de paridade por grupo (p_{i_0}, p_{i_1}, r_{i_0} e r_{i_1}). Desse modo, as condições para indicação de erro são:

- Os bits de paridade transmitido e calculado para *Ch0*, respectivamente p_{i_0} e r_{i_0}, possuem o mesmo valor lógico. Enquanto isso, para *Ch1* os bits p_{i_1} e r_{i_1} possuem o mesmo valor lógico, mas são o complemento do valor para *Ch0*;
- Os bits de paridade p_{i_0} e r_{i_0} possuem valores lógicos opostos, enquanto p_{i_1} e r_{i_1} também apresentam valores lógicos opostos.

A Tabela verdade 4.1 contém o valor binário do bit de indicação de erro no grupo i, h_i, com base nas condições supracitadas. Destaca-se que a indicação de erro depende da quantidade de *bitflips*, pois, uma vez que é considerado um código de paridade par, uma quantidade par de *bitflips* nos bits de informação pode gerar valores binários de r_{i_0} ou r_{i_1} cuja saída não resulta em indicação de erro. Como exemplo, para a segunda linha da Tabela verdade 4.1, caso ocorra uma quantidade par de *bitflips* nos bits de informação de *Ch0* e um número ímpar de *bitflips* nos bits de informação de *Ch1*, para um mesmo grupo, ambos os canais terão valores binários errados. Contudo, h_i produzirá valor lógico 0, o que significa que não há indicação de erro.

Tabela 4.1: Tabela verdade para o bit de indicação de erro no grupo i (h_i)

p_{i_0}	p_{i_1}	r_{i_0}	r_{i_1}	h_i	Mintermos
0	0	0	0	0	-
0	0	0	1	0	-
0	0	1	0	0	-
0	0	1	1	1	$\overline{p_{i_0}} \cdot \overline{p_{i_1}} \cdot r_{i_0} \cdot r_{i_1}$
0	1	0	0	0	-
0	1	0	1	1	$\overline{p_{i_0}} \cdot p_{i_1} \cdot \overline{r_{i_0}} \cdot r_{i_1}$
0	1	1	0	1	$\overline{p_{i_0}} \cdot p_{i_1} \cdot r_{i_0} \cdot \overline{r_{i_1}}$
0	1	1	1	0	-
1	0	0	0	0	-
1	0	0	1	1	$p_{i_0} \cdot \overline{p_{i_1}} \cdot \overline{r_{i_0}} \cdot r_{i_1}$
1	0	1	0	1	$p_{i_0} \cdot \overline{p_{i_1}} \cdot r_{i_0} \cdot \overline{r_{i_1}}$
1	0	1	1	0	-
1	1	0	0	1	$p_{i_0} \cdot p_{i_1} \cdot \overline{r_{i_0}} \cdot \overline{r_{i_1}}$
1	1	0	1	0	-
1	1	1	0	0	-
1	1	1	1	0	-

Fonte: O próprio autor.

Na Tabela verdade 4.1, estão dispostos os seis mintermos associados às combinações de variáveis de entrada que produzem a lógica 1 para h_i. Dados esses mintermos, a expressão booleana para o bit de indicação de erro (h_i) pode ser escrita como

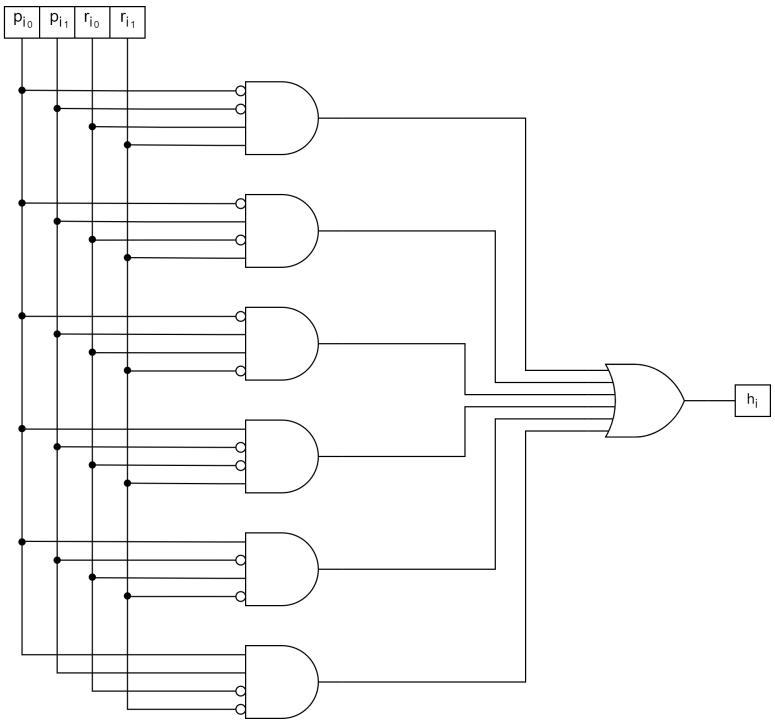
$$h_\mathrm{i} = \overline{p_{\mathrm{i}_0}} \cdot \overline{p_{\mathrm{i}_1}} \cdot r_{\mathrm{i}_0} \cdot r_{\mathrm{i}_1} + \overline{p_{\mathrm{i}_0}} \cdot p_{\mathrm{i}_1} \cdot \overline{r_{\mathrm{i}_0}} \cdot r_{\mathrm{i}_1} + \overline{p_{\mathrm{i}_0}} \cdot p_{\mathrm{i}_1} \cdot r_{\mathrm{i}_0} \cdot \overline{r_{\mathrm{i}_1}} + p_{\mathrm{i}_0} \cdot \overline{p_{\mathrm{i}_1}} \cdot \overline{r_{\mathrm{i}_0}} \cdot r_{\mathrm{i}_1} + p_{\mathrm{i}_0} \cdot \overline{p_{\mathrm{i}_1}} \cdot r_{\mathrm{i}_0} \cdot \overline{r_{\mathrm{i}_1}} + p_{\mathrm{i}_0} \cdot p_{\mathrm{i}_1} \cdot \overline{r_{\mathrm{i}_0}} \cdot \overline{r_{\mathrm{i}_1}}. \quad (4.19)$$

A expressão 4.19 caracteriza um circuito lógico composto por seis portas lógicas *AND* de quatro entradas e uma porta *OR* de seis entradas, além de doze portas inversoras. Tal circuito está representado na Figura 4.10 e indica que para cada grupo são necessárias 19 portas lógicas e 30 entradas de porta (desconsiderando, por simplicidade, os inversores). Isso significa que a inclusão da lógica de indicação de erro aumenta a quantidade de portas lógicas e entradas de porta para cada grupo, de maneira que:

$$\begin{aligned} Q'_\mathrm{B} &= Q_\mathrm{B} + 19 \times N_\mathrm{p}, = 3 \times N_\mathrm{p} + 3 \times N + 19 \times N_\mathrm{p}, \\ Q'_\mathrm{B} &= 3 \times N + 22 \times N_\mathrm{p}, \quad (4.20) \\ I'_\mathrm{B} &= I_\mathrm{B} + 30 \times N_\mathrm{p} = 8 \times N + N_\mathrm{p} + 30 \times N_\mathrm{p}, \\ I'_\mathrm{B} &= 8 \times N + 31 \times N_\mathrm{p}, \quad (4.21) \end{aligned}$$

em que Q'_B e I'_B indicam a quantidade de portas e a quantidade de entradas de porta, respectivamente, considerando a lógica de indicação de erro.

Figura 4.10: Circuito lógico para o bit de indicação de erro no grupo i (h_i)



Fonte: O próprio autor.

4.4 Otimizando a Solução

Embora a expressão 4.19 possibilite uma solução com indicação de erro, as equações 4.20 e 4.21 demonstram que há um impacto na quantidade de portas e entradas de portas para a PDMR. Uma alternativa para minimizar esse impacto se baseia em reformular as lógicas para seleção de canal e indicação de erro.

No que se refere ao bit de seleção de canal, simbolizado por s_{i}, assume-se que o seu valor lógico é igual a $0b1$ quando:

- Os bits de paridade p_{i_1} e r_{i_1} possuem o mesmo valor lógico, enquanto p_{i_0} e r_{i_0} têm valores lógicos contrários, ou seja $p_{\mathrm{i}_0} = \overline{r_{\mathrm{i}_0}}$;
- Dois dos bits de paridade possuem o mesmo valor lógico, enquanto os outros dois apresentam o valor lógico oposto. Observe que essa situação corresponde às condições descritas para h_{i}.

Em outras palavras, o bit de seleção de canal assume valor binário $0b0$ nos casos em que todos os bits de paridade possuem o mesmo valor lógico ($p_{\mathrm{i}_0} = p_{\mathrm{i}_1} = r_{\mathrm{i}_0} = r_{\mathrm{i}_1}$) e quando os bits de paridade transmitido e calculado para *Ch0* são iguais, enquanto para *Ch1* eles diferem.

Para a expressão lógica do bit de indicação de erro, inicialmente, considera-se um bit de *flag* de erro, representado por f_{i}, que tem valor binário $0b1$ quando todos os bits de paridade possuem o mesmo valor lógico ($p_{\mathrm{i}_0} = p_{\mathrm{i}_1} = r_{\mathrm{i}_0} = r_{\mathrm{i}_1}$). Além disso, f_{i} também é $0b1$ para as condições descritas para h_{i}.

A Tabela 4.2 representa a tabela verdade para s_{i} e f_{i}, de modo que as expressões lógicas podem ser escritas como:

$$s_{\mathrm{i}} = (p_{\mathrm{i}_0} \oplus r_{\mathrm{i}_0}) + (p_{\mathrm{i}_1} \cdot \overline{r_{\mathrm{i}_0}} \cdot r_{\mathrm{i}_1}) + (\overline{p_{\mathrm{i}_1}} \cdot r_{\mathrm{i}_0} \cdot \overline{r_{\mathrm{i}_1}}), \tag{4.22}$$

$$f_{\mathrm{i}} = (p_{\mathrm{i}_0} \oplus r_{\mathrm{i}_0}) \odot (p_{\mathrm{i}_1} \oplus r_{\mathrm{i}_1}). \tag{4.23}$$

A expressão 4.22 é obtida pela simplificação da função lógica que contém os mintermos de s_{i}. Ademais, a expressão 4.23 é uma representação reduzida em termos de portas *XOR* e *XNOR* (operador $\odot$) das operações *OR* entre os mintermos de f_{i}.

Com base nas expressões 4.22 e 4.23, a lógica para o bit de indicação de erro (daqui em diante referenciado por e_{i}) pode ser obtida por meio da realização de uma operação *AND* entre s_{i} e f_{i}. Portanto,

$$e_{\mathrm{i}} = s_{\mathrm{i}} \cdot f_{\mathrm{i}}, \tag{4.24}$$

de modo que $e_{\mathrm{i}} = h_{\mathrm{i}}$.

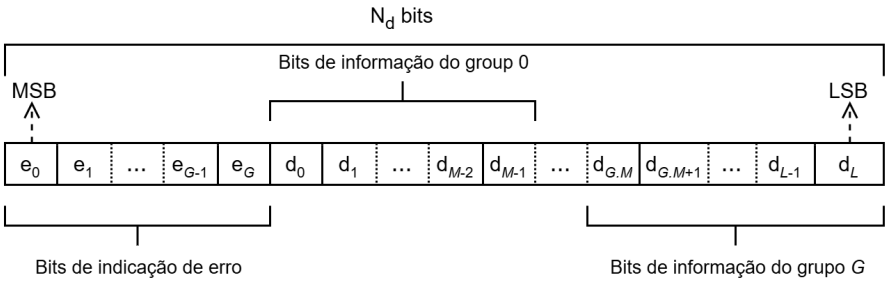
A depender da necessidade e da implementação, e_{i} pode ser utilizado como um sinal auxiliar para tomada de ações de mitigação de erro, como retransmissão ou o descarte de dados por parte da unidade de processamento. Adicionalmente, uma vez que e_{i} é gerado para cada grupo da palavra-código, o número total de bits de indicação de erro (N_{e}) para um determinado dado de entrada é igual a quantidade de grupos da palavra-código, de modo que $N_{\mathrm{e}} = N_{\mathrm{p}}$. Sendo assim, o formato do dado de saída pode ser representado pela Figura 4.11.

Tabela 4.2: Tabela verdade para o bit de seleção de canal (s_{i}) e para o bit de *flag* de erro (f_{i}), ambos referentes ao grupo i

p_{i_0}	p_{i_1}	r_{i_0}	r_{i_1}	s_{i}	f_{i}
0	0	0	0	0	1
0	0	0	1	0	0
0	0	1	0	1	0
0	0	1	1	1	1
0	1	0	0	0	0
0	1	0	1	1	1
0	1	1	0	1	1
0	1	1	1	1	0
1	0	0	0	1	0
1	0	0	1	1	1
1	0	1	0	1	1
1	0	1	1	0	0
1	1	0	0	1	1
1	1	0	1	1	0
1	1	1	0	0	0
1	1	1	1	0	1

Fonte: Alves et al. (2024).

Figura 4.11: Formato do dado de saída



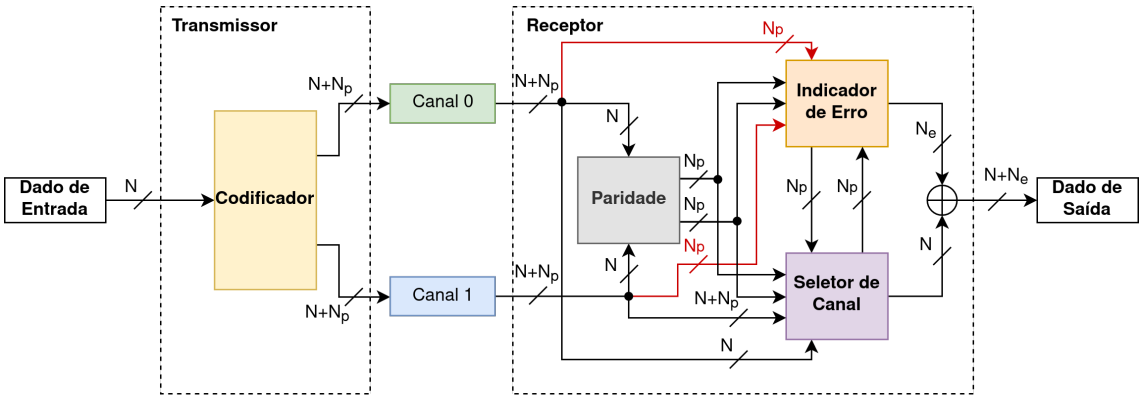
Fonte: Adaptado de Alves et al. (2024).

Diante das informações apresentadas, a estrutura de transmissão com PDMR é ilustrada pela Figura 4.12, em que as quantidades de bits são indicadas por N, N_{p} e N_{e}, sendo $N_{\mathrm{d}} = N + N_{\mathrm{p}} = N + N_{\mathrm{e}}$. Importante notar que, dependendo da implementação, o dado de saída pode ser formado apenas pelos N bits de informação, tornando a solução transparente para a unidade de processamento ou para a unidade que recebe o dado.

Na Figura 4.12, o módulo *Codificador* adiciona as paridades a cada grupo, formando a palavra-código e o módulo *Paridade* é composto pelos elementos lógicos cuja função é calcular os bits de paridade no receptor (r_{i_0} e r_{i_1}). Enquanto isso, o módulo *Indicador de Erro* contém a lógica responsável por gerar os bits de indicação de erro (expressão 4.24) e bloco *Seletor de Canal* tem como base os circuitos obtidos da expressão 4.22.

Um exemplo para um dado de entrada com três bits de informação ($N = 3$) está re-

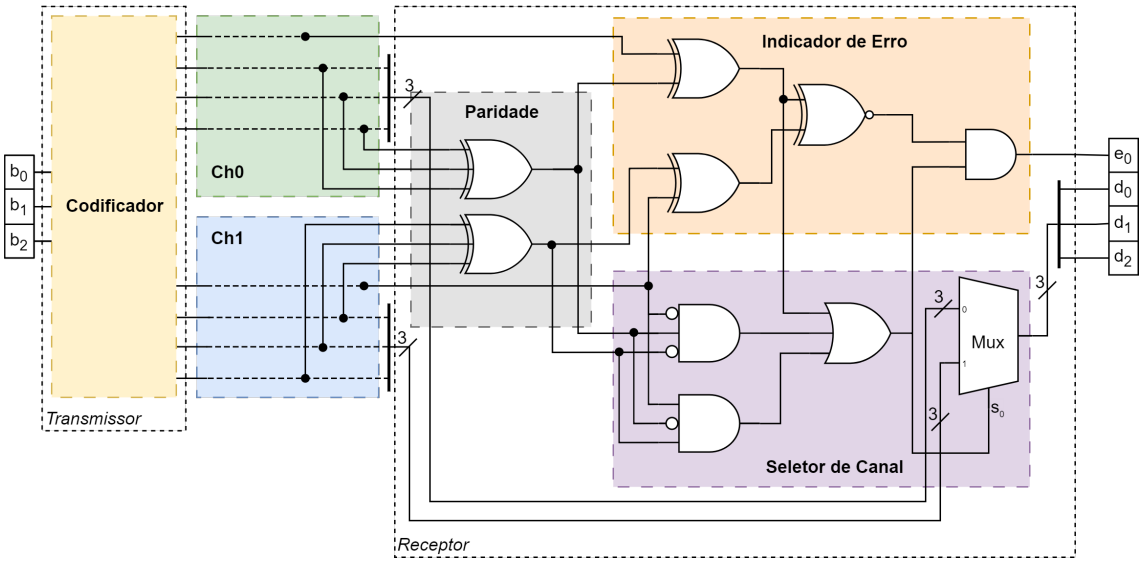
Figura 4.12: Diagrama de transmissão de dados com PDMR em que $N_p = N_e$



Fonte: Adaptado de Alves et al. (2024).

presentado na Figura 4.13. Além da quantidade de bits do dado de entrada, a principal diferença entre os exemplos das Figuras 4.6 e 4.13 diz respeito aos módulos incluídos no Receptor.

Figura 4.13: Exemplo da PDMR com indicação de erro para um dado de entrada com três bits de infomação



Fonte: O próprio autor.

No exemplo da Figura 4.13, o codificador é composto por uma porta *XOR* de três entradas, assim como no primeiro grupo do exemplo da Figura 4.6. Contudo, além do *MUX*, o módulo *Seletor de Canal* possui duas portas *AND* de três entradas, uma porta *OR* de três entradas e três portas inversoras. Ademais, como os bits de paridade no receptor são calculados para cada canal, o módulo *Paridade* possui duas portas *XOR* de três entradas. No

que se refere ao *Indicador de Erro*, observa-se que ele é composto por duas portas *XOR* de duas entradas, uma porta *XNOR* de duas entradas e uma porta *AND* de duas entradas.

De modo geral, com base no exemplo da Figura 4.13, verifica-se que os módulos que dependem da quantidade de bits de informação do grupo (M) são o *Codificador*, o *Paridade* e o *Seletor de Canal*. Nesse sentido, nos módulos *Codificador* e *Paridade*, M influencia no número de entradas de porta, enquanto no módulo *Seletor de Canal* o impacto é observado na quantidade de portas e entradas de porta dos elementos no *MUX* (conforme exposto para o exemplo das Figuras 4.6 e 4.7).

As relações para contagem do número de portas e entradas de porta para o caso geral estão dispostas na Tabela 4.3. Para o número de entradas de portas, por simplicidade, foram desconsiderados os inversores. A quantidade de portas para o caso geral, simbolizada por Q_{G}, pode então ser obtida por:

$$\begin{aligned} Q_{\mathrm{G}} &= N_{\mathrm{p}} + 2 \times N_{\mathrm{p}} + 7 \times N_{\mathrm{p}} + 3 \times N_{\mathrm{p}} \times M + 4 \times N_{\mathrm{p}}, \\ Q_{\mathrm{G}} &= 14 \times N_{\mathrm{p}} + 3 \times N_{\mathrm{p}} \times M = 14 \times N_{\mathrm{p}} + 3 \times N_{\mathrm{p}} \times N/N_{\mathrm{p}}, \end{aligned}$$

$$Q_{\mathrm{G}} = 3 \times N + 14 \times N_{\mathrm{p}}. \tag{4.25}$$

A equação 4.25 indica que uma variação de N e N_{p} para o caso geral implica em um número menor de portas lógicas quando comparado com o número dado pela equação 4.20.

Tabela 4.3: Relações para contagem do número de portas e entradas de porta para o caso geral

Módulo	**Quantidade de Portas**	**Número de Entradas de Porta**
Codificador	N_{p}	$N_{\mathrm{p}} \times M$
Paridade	$2 \times N_{\mathrm{p}}$	$2 \times N_{\mathrm{p}} \times M$
Seletor de Canal	$7 \times N_{\mathrm{p}} + 3 \times N_{\mathrm{p}} \times M$	$9 \times N_{\mathrm{p}} + 6 \times N_{\mathrm{p}} \times M$
Indicador de Erro	$4 \times N_{\mathrm{p}}$	$8 \times N_{\mathrm{p}}$

Fonte: O próprio autor.

Com base nas informações da Tabela 4.3, constata-se que o número total de entradas de porta para o caso geral (I_{G}) é dado por:

$$\begin{aligned} I_{\mathrm{G}} &= N_{\mathrm{p}} \times M + 2 \times N_{\mathrm{p}} \times M + 9 \times N_{\mathrm{p}} + 6 \times N_{\mathrm{p}} \times M + 8 \times N_{\mathrm{p}}, \\ I_{\mathrm{G}} &= 17 \times N_{\mathrm{p}} + 9 \times N_{\mathrm{p}} \times M = 17 \times N_{\mathrm{p}} + 9 \times N_{\mathrm{p}} \times N/N_{\mathrm{p}}, \end{aligned}$$

$$I_{\mathrm{G}} = 9 \times N + 17 \times N_{\mathrm{p}}. \tag{4.26}$$

Ao igualar as equações 4.21 e 4.26, tem-se que a quantidade de entradas de porta no caso geral é menor para valores de $N < 14 \times N_{\mathrm{p}}$. Portanto, com base em $N = 14 \times N_{\mathrm{p}}$ o caso geral apresenta um número de entradas de porta superior ao que se verifica para o caso mais simples, embora a quantidade de portas lógicas ainda seja menor.

4.5 Expandido a Capacidade de Detecção

A equação 4.24 permite que sejam gerados bits de indicação de erro para os casos da Tabela 4.2 em que dois dos bits de paridade possuem o mesmo valor lógico, enquanto os outros dois apresentam o valor lógico oposto. Porém, por simplicidade, assume-se que os casos em que todos os bits de paridade apresentam o mesmo valor lógico (primeira e última linhas da tabela) implicam em bits de informação corretos em ambos os canais e, por padrão, *Ch0* disponibiliza os bits para a saída ($s_\mathrm{i} = 0$). O impacto dessa suposição é verificado nos Capítulos 5 e 6.

Uma possibilidade para incrementar a capacidade de detecção está relacionada com a adição de uma lógica de comparação entre os bits de informação dos canais. Desse modo, mesmo que os bits de paridade sejam iguais para os casos da primeira e última linha da Tabela 4.2, é possível gerar o bit de indicação de erro caso os bits de informação sejam diferentes. Para isso, a comparação deve ser realizada por meio da seguinte expressão lógica

$$c_\mathrm{i} = b_{(\mathrm{Mi})_0} \oplus b_{(\mathrm{Mi})_1} + b_{(\mathrm{Mi}+1)_0} \oplus b_{(\mathrm{Mi}+1)_1} + \cdots + b_{(\mathrm{Mi}+\mathrm{M}-2)_0} \oplus b_{(\mathrm{Mi}+\mathrm{M}-2)_1} + b_{(\mathrm{Mi}+\mathrm{M}-1)_0} \oplus b_{(\mathrm{Mi}+\mathrm{M}-1)_1}, \quad (4.27)$$

em que c_i é o bit que sinaliza que há uma diferença entre os bits de informação dos canais para o grupo i.

Com base na Tabela 4.2 e na expressão 4.27, a expressão booleana para o bit de indicação de erro da solução com comparador pode ser definida por:

$$\begin{aligned} e'_\mathrm{i} &= e_\mathrm{i} + f_\mathrm{i} \cdot c_\mathrm{i}, \\ e'_\mathrm{i} &= f_\mathrm{i} \cdot s_\mathrm{i} + f_\mathrm{i} \cdot c_\mathrm{i}, \\ e'_\mathrm{i} &= f_\mathrm{i} \cdot (s_\mathrm{i} + c_\mathrm{i}). \end{aligned} \quad (4.28)$$

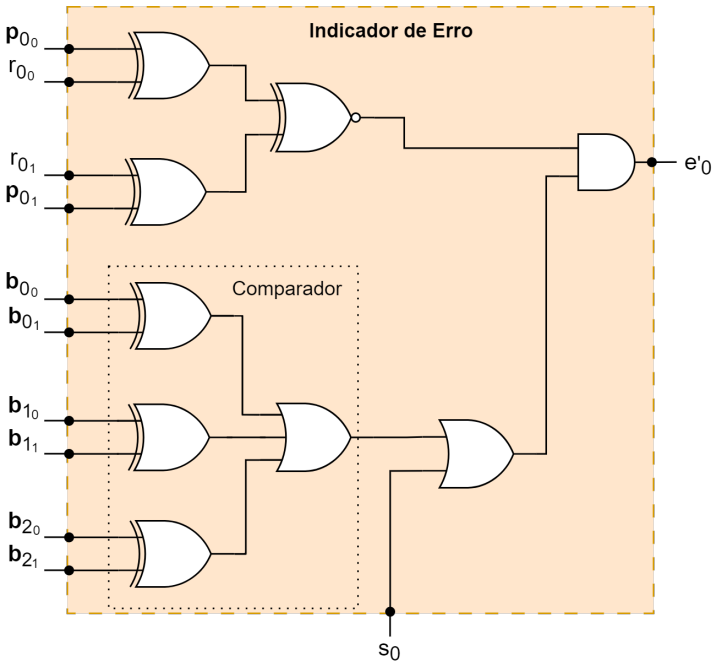
Seguindo o exemplo para três bits de informação ($N = 3$), o módulo *Indicador de Erro* após inclusão do comparador é representado pela Figura 4.14. Nesse caso, o diagrama de transmissão da Figura 4.12 continua válido desde que a largura dos barramentos em vermelho seja substituída por $N + N_\mathrm{p}$.

Por meio das expressões 4.27 e 4.28 e da Figura 4.14, pode-se verificar a quantidade de portas lógicas (Q_C) e entradas de porta (I_C) após a inclusão da lógica de comparação. Esses valores passam a depender do valor de M e há adição de uma porta *OR* de duas entradas (correspondente à $s_\mathrm{i} + c_\mathrm{i}$), outra porta *OR* de M entradas (comparador) e M portas *XOR* de duas entradas por grupo (comparador). Sendo assim, de modo geral, tem-se:

$$\begin{aligned} Q_\mathrm{C} &= Q_\mathrm{G} + 2 \times N_\mathrm{p} + N_\mathrm{p} \times M, \\ &= 3 \times N + 14 \times N_\mathrm{p} + 2 \times N_\mathrm{p} + N_\mathrm{p} \times M, \\ &= 3 \times N + 16 \times N_\mathrm{p} + N_\mathrm{p} \times N/N_\mathrm{p}, \end{aligned}$$

$$Q_\mathrm{C} = 4 \times N + 16 \times N_\mathrm{p}, \quad (4.29)$$

Figura 4.14: Exemplo de comparador para o receptor considerando três bits de informação



Fonte: O próprio autor.

e

$$\begin{aligned} I_C &= I_G + 2 \times N_p + N_p \times M + 2 \times N_p \times M, \\ &= 9 \times N + 17 \times N_p + 2 \times N_p + N_p \times M + 2 \times N_p \times M, \\ &= 9 \times N + 19 \times N_p + N_p \times N/N_p + 2 \times N_p \times N/N_p, \end{aligned}$$

$$I_C = 11 \times N + 19 \times N_p. \tag{4.30}$$

Por meio da análise das equações 4.20 e 4.29, nota-se que Q_C é menor ou igual a Q'_B para $N \leq 6 \times N_p$. Do mesmo modo, das equações 4.21 e 4.30, tem-se que I_C é menor ou igual a I'_B para $N \leq 6 \times N_p$.

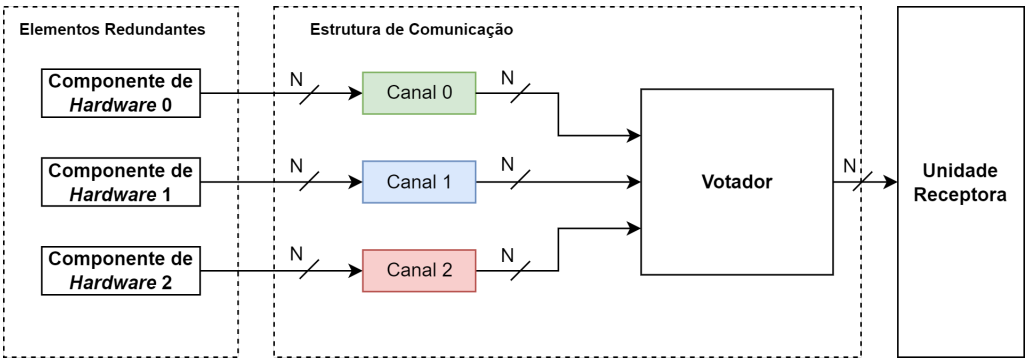
4.6 Aplicação com Elementos Redundantes

Os diagramas das Figuras 4.1 e 4.12 representam a proteção de um dado de entrada em um caminho de transmissão sujeito aos efeitos da radiação. Contudo, a aplicação de redundância modular pode se basear na utilização de cópias de componentes lógicos responsáveis por funções de processamento ou armazenamento no sistema, com o intuito de aumentar a sua confiabilidade. Isso se dá pelo fato de componentes como memórias e processadores serem suscetíveis aos efeitos da radiação quando em ambiente espacial.

Nesse contexto, a arquitetura com TMR corresponde ao diagrama da Figura 4.15, em que os componentes de hardware são elementos redundantes que desempenham a mesma

função. Tais componentes podem ser, por exemplo, elementos de memória, processadores, aceleradores de *hardware* ou unidades computacionais mais complexas. Enquanto isso, a unidade receptora é o componente do sistema que recebe o dado proveniente dos elementos redundantes, podendo ser uma unidade de processamento ou de armazenamento. Na estrutura com TMR, o votador é responsável por mitigar possíveis erros ao comparar os dados enviados pelos componentes de *hardware* e disponibilizar como resultado o valor da maioria.

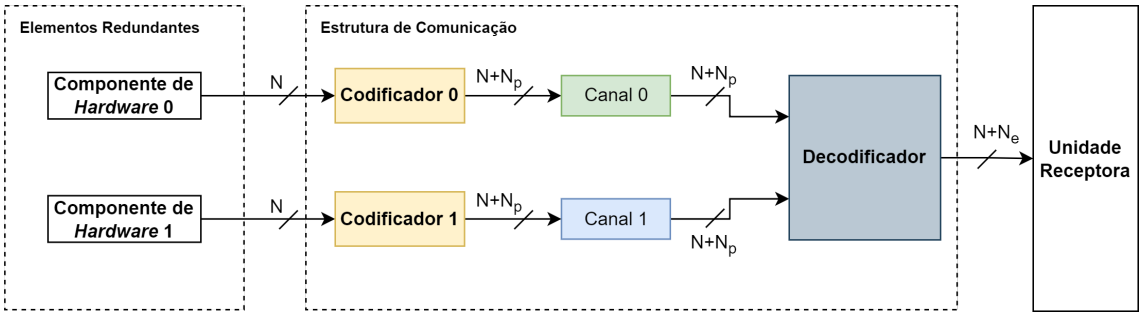
Figura 4.15: Diagrama de transmissão de dados de elementos redundantes com TMR



Fonte: O próprio autor.

Com relação à proposta PDMR, a arquitetura do sistema é ilustrada na Figura 4.16. Nesse caso, são utilizados apenas dois componentes de *hardware* e dois codificadores, os quais têm a função de gerar a palavra-código. Desse modo, os codificadores têm como elementos básicos as portas *XOR* que determinam a paridade (expressão 4.8). O módulo *Decodificador* é responsável por formar o dado de saída com base nos dados recebidos por meio dos canais. Portanto, o *Decodificador* inclui as funções dos módulos *Paridade*, *Indicador de Erro* e *Seletor de Canal*. Além disso, O dado de saída é disponibilizado à unidade receptora, que pode ser uma unidade de processamento ou armazenamento.

Figura 4.16: Diagrama de transmissão de dados de elementos redundantes com PDMR



Fonte: O próprio autor.

4.7 Resumo

Neste capítulo, foram apresentadas duas regras de formação para a palavra-código da proposta de DMR baseada em paridade (PDMR). A regra *RF1* contém definições para o número de bits de informação (M) de cada grupo, de bits da palavra-código (N_{d}) e de bits de paridade da palavra-código (N_{p}) para estruturas de comunicação que não possuem restrições relacionadas à largura do barramento de dados. Enquanto isso, a regra *RF2* apresenta definições para protocolos de comunicação cujo barramento de dados possui tamanho determinado por uma potência de 2 (e.g., 8, 16, 32, 64).

A proposta PDMR foi descrita desde a estrutura mais simples até a mais elaborada com possibilidade de indicação de erro. Para simplificar a descrição nos próximos capitulos, as diferentes versões da PDMR são definidas como:

- PDMR-B: Versão mais básica em que a seleção dos bits de informação de cada grupo é efetuada por meio da verificação da paridade após passagem por apenas um dos canais (exemplo da Figura 4.6). Nessa versão, o erro é apenas mascarado/corrigido, de modo que não é possível a detecção dados errados;
- PDMR-G: Estrutura otimizada que permite a detecção de erro por meio do bit de indicação de erro;
- PDMR-C: Versão que inclui uma lógica de comparação para aumentar a capacidade de detecção da proposta.

A Tabela 4.4 contém um resumo com as expressões booleanas e as equações para cálculo dos números de portas (Q) e entradas de porta (I) para cada uma das versões da PDMR.

Tabela 4.4: Resumo das expressões lógicas e equações para as diferentes versões da PDMR

Versão	**Expressões Lógicas**	Q	I
PDMR-B	4.8, 4.13 e 4.14	$Q_{\mathrm{B}} = 3 \times N + 3 \times N_{\mathrm{p}}$	$I_{\mathrm{B}} = 8 \times N + N_{\mathrm{p}}$
PDMR-G	4.8, 4.11, 4.22, 4.23 e 4.24	$Q_{\mathrm{G}} = 3 \times N + 14 \times N_{\mathrm{p}}$	$I_{\mathrm{G}} = 9 \times N + 17 \times N_{\mathrm{p}}$
PDMR-C	4.8, 4.11, 4.22, 4.23, 4.27 e 4.28	$Q_{\mathrm{C}} = 4 \times N + 19 \times N_{\mathrm{p}}$	$I_{\mathrm{C}} = 11 \times N + 19 \times N_{\mathrm{p}}$

Fonte: O próprio autor.

As informações na Tabela 4.4 tornam evidente que a inclusão de características como indicação de erro e comparação resultam em uma maior complexidade da solução. Desse modo, são necessárias mais portas lógicas e, consequentemente, entradas de porta se forem utilizadas as versões PDMR-G e PDMR-C. Sendo assim, no Capítulo 6, são realizadas comparações em termos de área ocupada e consumo energético das diferentes versões da PDMR e da TMR implementadas em FPGA.

No que diz respeito à aplicação da proposta PDMR com componentes de *hardware* redundantes, com base nas Figuras 4.15 e 4.16, verifica-se que a TMR inclui apenas o

votador, enquanto a aplicação de PDMR para elementos redundantes está relacionada com o uso de codificadores e do decodificador. Contudo, vale destacar que a complexidade lógica do componente de *hardware* tem impacto na área ocupada (quantidade de portas) da implementação em *chip*. Sendo assim, apesar do votador TMR apresentar uma lógica de implementação simples, os elementos redundantes podem contribuir para um consumo considerável de área. Por outro lado, por serem usados menos elementos redundantes na aplicação de PDMR, o impacto é reduzido a depender do componente de *hardware*, embora os codificadores e o decodificador possuam mais elementos lógicos do que o votador para as versões PDMR-G e PDMR-C. Por exemplo, se o componente de *hardware* a ser replicado for um processador, a quantidade de elementos lógicos que o compõem tem uma influência no consumo de área maior do que os elementos do votador TMR e dos codificadores e decodificadores da PDMR. Desse modo, a comparação em termos de área ocupada se resume à constatação de que os três componentes de *hardware* em uma solução com TMR representam um maior impacto na área do que os dois da PDMR. Um estudo de caso é apresentado no Capítulo 6.

Capítulo 5

Análise Matemática da Proposta PDMR

As diferentes versões da proposta PDMR, descritas no Capítulo 4, possuem características que contribuem para a capacidade de correção e detecção da técnica. Por meio das expressões e equações apresentadas (ver Tabela 4.4), observa-se que a inclusão das lógicas de indicação de erro e comparação aumentam a quantidade de elementos lógicos para a abordagem proposta. Desse modo, tem-se um *tradeoff* entre a capacidade de detecção de erros e a complexidade lógica das soluções.

Neste capítulo, são discutidas as análises matemáticas para as probabilidades de correção, de detecção e de erros das diferentes versões. Para as análises apresentadas, a taxa de erros nos dois canais é considerada como sendo igual, de maneira que a probabilidade de ocorrência de *bitflip* é a mesma para qualquer bit transmitido. Essa probabilidade é representada por P_{b} e pode ser aplicada a cada bit transmitido. Por conseguinte, a probabilidade de não ocorrer um *bitflip* é representada por P_{nb} e definida por

$$P_{\mathrm{nb}} = 1 - P_{\mathrm{b}}. \tag{5.1}$$

Ademais, uma vez que a proposta PDMR tem como base códigos de paridade par, as seguintes definições devem ser consideradas:

Definição 5.1. Para a transmissão de um dado de N bits, cujo i-ésimo grupo possui um número par (incluindo 0) de bits $0b1$, tem-se que o valor original (i.e., antes da transmissão) de p_{i} é $0b0$.

Definição 5.2. Para a transmissão de um dado de N bits, cujo i-ésimo grupo possui um número ímpar bits $0b1$, tem-se que o valor original de p_{i} é $0b1$.

Embora em observações reais a ocorrência de *bitflips* em uma linha de barramento do canal possa afetar as probabilidades de *bitflips* nas demais linhas, neste trabalho, assume-se que os *bitflips* entre linhas ocorrem de forma independente. Destarte, as equações obtidas a seguir levam a valores aproximados de probabilidade.

Por meio das descrições matemáticas, busca-se fornecer bases numéricas para caracterização de cada versão da técnica PDMR, as quais podem ser utilizadas para comparação com a técnica TMR, como descrito no Capítulo 6.

5.1 Cálculos de Probabilidade para PDMR-B

De acordo com a discussão da Seção 4.2, na versão mais básica da proposta PDMR (PDMR-B), a paridade é usada como critério de seleção com base na verificação de bits de paridade para apenas um dos canais. O canal escolhido para checagem foi o canal *Ch0*, de modo que são verificadas as paridades da palavra-código proveniente desse canal. Além disso, essa versão não possibilita a indicação de erro. Portanto, o dado observado na saída pode estar correto ou pode conter erros. Desse modo, para a PDMR-B a análise matemática se resume ao cálculo das probabilidades de correção e de erro.

5.1.1 Correção

A correção acontece quando o dado formado na saída do módulo *Seletor de Canal* (ver exemplo da Figura 4.6) é igual ao dado original transmitido. Isso significa que os bits de informação dos grupos devem ser selecionados por meio da verificação dos respectivos bits de paridade e não devem ter sofrido *bitflips*.

Para simplificar o cálculo da probabilidade de correção para a versão PDMR-B, são considerados os eventos possíveis para uma palavra-código com apenas um grupo. Esses eventos correspondem aos valores binários que os bits da palavra-código (do grupo) podem assumir ao sofrerem *bitflips*. Posteriormente, os resultados são generalizados considerando palavras-código com mais grupos.

A quantidade total de eventos para um grupo com N_{g} bits é dada por

$$\begin{aligned} T &= (\text{Eventos em } \mathit{Ch0}) \times (\text{Eventos em } \mathit{Ch1}), \\ T &= 2^{N_{\mathrm{g}}} \times 2^{N_{\mathrm{g}}}, \\ T &= 2^{2N_{\mathrm{g}}}. \end{aligned} \tag{5.2}$$

Ou seja, T indica a quantidade de valores binários que podem ser formados com N_{g} bits para dois canais redundantes sujeitos aos efeitos da radiação (*Ch0* e *Ch1*).

Como exemplo, suponha que o dado $0b0$ deve ser transmitido usando PDMR-B, de modo que

$$N = M = 1, N_{\mathrm{p}} = 1, N_{\mathrm{d}} = N_{\mathrm{g}} = 2.$$

A quantidade de eventos possíveis para transmissão desse dado nos dois canais é obtida pela equação 5.2. Desse modo,

$$\begin{aligned} T &= 2^{2N_{\mathrm{g}}} = 2^{2\times 2}, \\ T &= 16. \end{aligned}$$

Com base nessa informação, pode-se construir uma tabela com todas as possibilidades de eventos para transmissão do dado $0b0$. Os eventos possíveis estão dispostos na Tabela 5.1, em que a classificação na coluna *Estado* foi realizada de acordo com o resultado de saída. Essa tabela tem como referência a estrutura básica da Figura 4.3, em que a lógica combinacional é obtida das expressões da Seção 4.2, como no exemplo da Figura 4.6.

A Tabela 5.1 contém oito eventos que geram resultado correto e oito cujo resultado

Tabela 5.1: Eventos possíveis para transmissão do dado 0*b*0 com paridade par para a versão PDMR-B

Evento	*Ch0*		*Ch1*		Receptor		Bit de Saída	Estado
	p_{0_0}	b_{0_0}	p_{0_1}	b_{0_1}	r_{0_0}	x_0		
E_1	0	0	0	0	0	0	0	Correto
E_2	0	0	0	1	0	0	0	Correto
E_3	0	0	1	0	0	0	0	Correto
E_4	0	0	1	1	0	0	0	Correto
E_5	0	1	0	0	1	1	0	Correto
E_6	0	1	0	1	1	1	1	Erro
E_7	0	1	1	0	1	1	0	Correto
E_8	0	1	1	1	1	1	1	Erro
E_9	1	0	0	0	0	1	0	Correto
E_{10}	1	0	0	1	0	1	1	Erro
E_{11}	1	0	1	0	0	1	0	Correto
E_{12}	1	0	1	1	0	1	1	Erro
E_{13}	1	1	0	0	1	0	1	Erro
E_{14}	1	1	0	1	1	0	1	Erro
E_{15}	1	1	1	0	1	0	1	Erro
E_{16}	1	1	1	1	1	0	1	Erro

Fonte: O próprio autor.

corresponde a um erro. Como a verificação da paridade é efetuada apenas para *Ch0*, na coluna *Receptor* estão dispostos os valores binários para r_{0_0} e para o bit x_0, que é usado como sinal de seleção do *MUX* (expressão 4.14). Sendo assim, quando x_0 é igual a 0*b*0, o bit de informação é obtido de *Ch0*, ou seja, é o valor de b_{0_0}. Enquanto isso, quando x_0 é igual a 0*b*1, o bit de informação provém de *Ch1*, sendo o valor de b_{0_1}.

Nos quatro primeiros eventos da Tabela 5.1 (E_1, E_2, E_3 e E_4), não há *bitflips* nos bits que formam a palavra-código em *Ch0* (p_{0_0} e b_{0_0}). Consequentemente, b_{0_0} é considerado como valor de saída e o resultado está correto, embora possam ocorrer *bitflips* nos bits que compõem a palavra-código em *Ch1* (eventos E_2, E_3 e E_4). Nos eventos E_5, E_6, E_7 e E_8, tem-se um *bitflip* no bit de informação em *Ch0* (b_{0_0}), de maneira que a verificação da paridade e a lógica de seleção no receptor fazem com que o resultado seja tomado de *Ch1*. Nesse caso, o resultado está correto apenas para os eventos em que não acontece *bitflip* em b_{0_1} (eventos E_5 e E_7). No que se refere aos eventos E_9, E_{10}, E_{11} e E_{12}, a ocorrência de *bitflip* em *Ch0* é observada no bit de paridade p_{0_0}, isso faz como que o bit de paridade calculado no receptor (r_{0_0}) apresente valor binário diferente. Logo, a saída é igual ao valor de b_{0_1} e o resultado está correto nos eventos em que não há *bitflip* em b_{0_1} (eventos E_9 e E_{11}). Por fim, os eventos E_{13}, E_{14}, E_{15} e E_{16} indicam a ocorrência de *bitflips* em todos os bits da palavra-código em *Ch0*, o que significa que a paridade r_{0_0} é igual a p_{0_0} e o valor de b_{0_0} é selecionado para a saída. Contudo, como nesses eventos há um *bitflip* em b_{0_0} (valor binário muda de 0*b*0 para 0*b*1), todos geram resultados com erro.

Considerando as informações anteriores, o número total de eventos corretos ($E_{\mathrm{B_x}}$)

pode ser escrito como

$$E_{\mathrm{B_x}} = E_1 + E_2 + E_3 + E_4 + E_5 + E_7 + E_9 + E_{11}.$$

A probabilidade de correção ($P(E_{\mathrm{B_x}})$) para a transmissão de $0b0$ com PDMR-B é, então, dada por

$$P(E_{\mathrm{B_x}}) = P(E_1 \cup E_2 \cup E_3 \cup E_4 \cup E_5 \cup E_7 \cup E_9 \cup E_{11}). \tag{5.3}$$

O lado direito da equação 5.3 pode ser desenvolvido por meio do Princípio da Inclusão-Exclusão, o que resulta em somas e subtrações de probabilidades da interseção de eventos. Contudo, os eventos E_1, E_2, E_3, E_4, E_5, E_7, E_9 e E_{11} são disjuntos (mutuamente excludentes) de modo que a probabilidade de correção pode ser escrita como

$$P(E_{\mathrm{B_x}}) = P(E_1) + P(E_2) + P(E_3) + P(E_4) + P(E_5) + P(E_7) + P(E_9) + P(E_{11}), \tag{5.4}$$

em que $P(E_1)$, $P(E_2)$, $P(E_3)$, $P(E_4)$, $P(E_5)$, $P(E_7)$, $P(E_9)$ e $P(E_{11})$ são as probabilidades de ocorrência dos eventos E_1, E_2, E_3, E_4, E_5, E_7, E_9 e E_{11}, respectivamente.

Como os eventos da Tabela 5.1 estão associados à ocorrência de *bitflips* nos bits das palavras-código dos canais, as probabilidades da equação 5.4 podem ser escritas em função das probabilidades dos eventos de *bitflip*. Para essa análise, os eventos de *bitflip* para p_{0_0}, b_{0_0}, p_{0_1} e b_{0_1} são representados por X_0, Y_0, X_1 e Y_1, respectivamente. Por outro lado, os complementares X_0^c, Y_0^c, X_1^c e Y_1^c indicam os eventos de *não bitflip*. Sendo assim,

$$P(E_1) = P(X_0^c \cap Y_0^c \cap X_1^c \cap Y_1^c), \tag{5.5}$$
$$P(E_2) = P(X_0^c \cap Y_0^c \cap X_1^c \cap Y_1), \tag{5.6}$$
$$P(E_3) = P(X_0^c \cap Y_0^c \cap X_1 \cap Y_1^c), \tag{5.7}$$
$$P(E_4) = P(X_0^c \cap Y_0^c \cap X_1 \cap Y_1), \tag{5.8}$$
$$P(E_5) = P(X_0^c \cap Y_0 \cap X_1^c \cap Y_1^c), \tag{5.9}$$
$$P(E_7) = P(X_0^c \cap Y_0 \cap X_1 \cap Y_1^c), \tag{5.10}$$
$$P(E_9) = P(X_0 \cap Y_0^c \cap X_1^c \cap Y_1^c), \tag{5.11}$$
$$P(E_{11}) = P(X_0 \cap Y_0^c \cap X_1 \cap Y_1^c). \tag{5.12}$$

De acordo a descrição no início deste capítulo, uma vez que a taxa de erros nos dois canais é considerada como sendo igual, a probabilidade de ocorrência de *bitflip* é a mesma para qualquer bit durante uma transmissão. Essa probabilidade é simbolizada por P_{b}, enquanto que a probabilidade de não ocorrência de *bitflip* é representada por P_{nb} e dada pela equação 5.1. Portanto,

$$P(X_0) = P(Y_0) = P(X_1) = P(Y_1) = P_{\mathrm{b}},$$

e

$$P(X_0^c) = P(Y_0^c) = P(X_1^c) = P(Y_1^c) = P_{\mathrm{nb}}.$$

Adicionalmente, sendo X_0, Y_0, X_1 e Y_1, bem como seus complementares, eventos in-

dependentes, as equações 5.5 à 5.12 se tornam:

$$
\begin{aligned}
P(E_1) &\simeq P(X_0^c) \times P(Y_0^c) \times P(X_1^c) \times P(Y_1^c) \simeq P_{\mathrm{nb}}^4, \\
P(E_2) &\simeq P(X_0^c) \times P(Y_0^c) \times P(X_1^c) \times P(Y_1) \simeq P_{\mathrm{nb}}^3 \times P_{\mathrm{b}}, \\
P(E_3) &\simeq P(X_0^c) \times P(Y_0^c) \times P(X_1) \times P(Y_1^c) \simeq P_{\mathrm{nb}}^3 \times P_{\mathrm{b}}, \\
P(E_4) &\simeq P(X_0^c) \times P(Y_0^c) \times P(X_1) \times P(Y_1) \simeq P_{\mathrm{nb}}^2 \times P_{\mathrm{b}}^2, \\
P(E_5) &\simeq P(X_0^c) \times P(Y_0) \times P(X_1^c) \times P(Y_1^c) \simeq P_{\mathrm{nb}}^3 \times P_{\mathrm{b}}^2, \\
P(E_7) &\simeq P(X_0^c) \times P(Y_0) \times P(X_1) \times P(Y_1^c) \simeq P_{\mathrm{nb}}^2 \times P_{\mathrm{b}}^2, \\
P(E_9) &\simeq P(X_0) \times P(Y_0^c) \times P(X_1^c) \times P(Y_1^c) \simeq P_{\mathrm{nb}}^3 \times P_{\mathrm{b}}, \\
P(E_{11}) &\simeq P(X_0) \times P(Y_0^c) \times P(X_1) \times P(Y_1^c) \simeq P_{\mathrm{nb}}^2 \times P_{\mathrm{b}}^2.
\end{aligned}
$$

Logo, substituindo os valores de $P(E_1)$, $P(E_2)$, $P(E_3)$, $P(E_4)$, $P(E_5)$, $P(E_7)$, $P(E_9)$ e $P(E_{11})$ na equação 5.4, tem-se

$$
P(E_{\mathrm{B_x}}) \simeq P_{\mathrm{nb}}^4 + 4 \times P_{\mathrm{nb}}^3 \times P_{\mathrm{b}} + 3 \times P_{\mathrm{nb}}^2 \times P_{\mathrm{b}}^2 \tag{5.13}
$$

O raciocínio para a obtenção da equação 5.13 serve como base para a compreensão da técnica PDMR-B e para contextualização da aplicação das probabilidades P_{b} e P_{nb}. No entanto, fica evidente que a análise apresentada é insuficiente para a transmissão de dados com mais bits de informação. Nesse sentido, busca-se definir os casos em que a saída está correta, possibilitando a generalização do comportamento da solução e, consequentemente, o desenvolvimento da fórmula geral.

Por meio da Tabela 5.1, as condições para que a saída esteja correta na transmissão do dado $0b0$ podem ser resumidas, de maneira que os eventos corretos acontecem quando:

1. Não há *bitflips* nos bits da palavra-código em *Ch0*, enquanto podem existir *bitflips* nos bits da palavra-código em *Ch1*;
2. Há apenas um *bitflip* nos bits de *Ch0*, enquanto que o bit de informação da palavra-código em *Ch1* não deve sofrer *bitflip*, embora seja possível a ocorrência de *bitflip* no bit p_{0_1}.

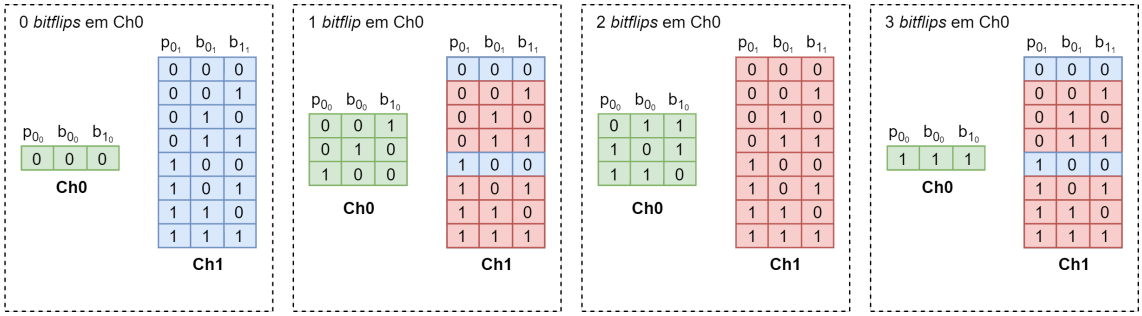
Com o intuito de analisar o comportamento da solução com PDMR-B para transmissão de dados com mais bits de informação, imagine agora que a técnica é aplicada na transmissão do dado $0b00$, de tal maneira que:

$$
N = M = 2, N_{\mathrm{p}} = 1, N_{\mathrm{d}} = N_{\mathrm{g}} = 3.
$$

Usando a equação 5.2, $T = 2^{2N_{\mathrm{g}}} = 2^6 = 64$, o que significa que há 64 eventos possíveis para a transmissão do dado $0b00$. Uma representação simplificada para os eventos possíveis está ilustrada na Figura 5.1. Nessa figura, estão dispostos quatro retângulos tracejados que destacam a quantidade de *bitflips* na palavra-código em *Ch0*. Além disso, cada retângulo inclui todos os possíveis valores binários que a palavra-código em *Ch1* pode assumir. Desse modo, cada par formado pela palavra-código em *Ch0* e *Ch1* corresponde a um evento. Como exemplo, considerando o retângulo com zero *bitflips* em

Ch0, a palavra-código 0*b*000 em *Ch0* em conjunto com a palavra-código 0*b*000 em *Ch1* constituem um evento possível, da mesma maneira que 0*b*000 em *Ch0* em conjunto com a palavra-código 0*b*001 em *Ch1* e assim por diante.

Figura 5.1: Exemplo para transmissão do dado 0*b*00 com paridade par para a PDMR-B



Fonte: O próprio autor.

Na Figura 5.1, os pares formados pelas palavras-código em *Ch0* e pelas palavras-código destacadas em azul em *Ch1* simbolizam os eventos em que o resultado de saída está correto. Por outro lado, os pares formados pelas palavras-código destacadas em vermelho estão relacionados aos eventos em que a saída está errada. Sendo assim, os eventos formados pelos pares permitem indicar que a correção acontece nas seguintes situações:

1. Não ocorre *bitflips* na palavra-código em *Ch0*, embora seja possível a ocorrência de *bitflips* na palavra-código em *Ch1*;
2. Há apenas um *bitflip* na palavra-código em *Ch0*, enquanto a palavra-código em *Ch1* pode conter *bitflip* apenas no bit de paridade p_{0_1};
3. Todos os três bits da palavra-código em *Ch0* sofrem *bitflip*, enquanto a palavra-código em *Ch1* pode conter *bitflip* apenas no bit de paridade p_{0_1}.

A primeira e a segunda situações são similares àquelas descritas para a transmissão do dado *0b0*. No que se refere a terceira situação, as palavras-código em *Ch1* estão de acordo com as da segunda situação, mas a quantidade de *bitflips* em *Ch0* difere. Contudo, é possível perceber que a segunda e a terceira situações indicam que a quantidade de *bitflips* na palavra-código em *Ch0* deve ser ímpar. Essa constatação possibilita a generalização das condições para que o resultado da transmissão de uma palavra-código composta por um grupo esteja correto.

Embora os dados 0*b*0 e 0*b*00 tenham sido utilizados como exemplos para simplificar a análise, vale destacar que exemplos de transmissão dos dados 0*b*1, 0*b*01, 0*b*10 e 0*b*11 também permitem uma verificação das situações descritas.

Fórmula Geral

Diante das análises dos exemplos da Tabela 5.1 e da Figura 5.1, os casos em que a técnica PDMR-B gera correção em um grupo são:

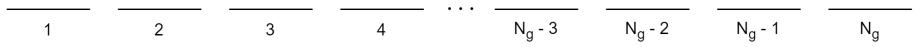
Caso 5.1.1. Os bits de informação e de paridade no canal *Ch0* não sofrem *bitflips*, embora seja possível a ocorrência de *bitflips* nos bits do canal *Ch1*;

Caso 5.1.2. O número de *bitflips* nos bits de informação e de paridade no canal *Ch0* é ímpar, enquanto apenas o bit de paridade em *Ch1* pode sofrer *bitflips*.

Em geral, para a transmissão de um dado binário com $N_g = N_d$, os casos 5.1.1 e 5.1.2 correspondem a determinar a quantidade de eventos formados pelos conjuntos de bits em *Ch0* e em *Ch1*, considerando as condições específicas. Esses eventos podem ser deduzidos da teoria de análise combinatória. Para isso, considere a Figura 5.2, em que são ilustradas as posições em que podem ocorrer *bitflips* em um dado de N_g bits em um dos canais. Nesse caso, a ordem dos *bitflips* não interessa. Sendo assim, escolhendo um subconjunto de n posições contendo *bitflips* de um conjunto de N_g posições, as posições restantes não devem conter *bitflips*. Portanto, a determinação da quantidade de palavras-código formadas por n *bitflips* em um dos canais equivale à encontrar a quantidade de combinações de N_g elementos agrupados n a n. Em outras palavras, equivale à calcular o binômio dado por

$$\binom{N_g}{n} = \frac{N_g!}{n!(N_g - n)!}. \tag{5.14}$$

Figura 5.2: Posições em que os *bitflips* podem acontecer em um dado de N_g bits em um dos canais



Fonte: O próprio autor.

Com base na equação 5.14, a quantidade de palavras-código que podem ser formadas em *Ch0* (C_0) é dada em função do número de *bitflips* no canal, de modo que

$$C_0(n_z) = \binom{N_a}{n_z}, \tag{5.15}$$

em que n_z representa o número de *bitflips* e N_a indica o total de bits que podem sofrer *bitflips* em *Ch0*, com $0 \leq N_a \leq N_g$ e $0 \leq n_z \leq N_a$. Analogamente, para *Ch1*, tem-se

$$C_1(n_u) = \binom{N_b}{n_u}, \tag{5.16}$$

em que C_1 simboliza a quantidade de palavras-código que podem ser formadas em *Ch1* em função do número de *bitflips*, n_u, no canal, sendo $0 \leq n_u \leq N_b$, em que N_b é o total de bits que podem sofrer *bitflips* em *Ch1*, de modo que $0 \leq N_b \leq N_g$.

Os eventos para os casos 5.1.1 e 5.1.2 podem, então, ser definidos pelos conjuntos de bits formados pelas possíveis palavras-código em *Ch0* e em *Ch1*. Desse modo, a

quantidade total de eventos formados por uma palavra-código em *Ch0* contendo n_z *bitflips* e por uma palavra-código em *Ch1* contendo n_u *bitflips* pode ser encontrada por

$$T_C(n_z, n_u) = C_0(n_z) \times C_1(n_u)$$

$$T_C(n_z, n_u) = \binom{N_a}{n_z} \times \binom{N_b}{n_u} \tag{5.17}$$

Como indicado no desenvolvimento da equação 5.3, os eventos estão relacionados à ocorrência de *bitflips* e a probabilidade de ocorrência de *bitflips* (P_b) é a mesma para qualquer bit durante a transmissão. Dessa maneira, para o cálculo da probabilidade, encontrar todos os eventos com valores binários formados pelos bits em *Ch0* e em *Ch1* é uma tarefa desnecessária. A análise deve se basear na determinação da quantidade de eventos contendo uma quantidade específica de *bitflips*.

Como exemplo, para um dado com N_g bits, um evento possível, simbolizado por E_x, é aquele em que não há *bitflips* em *Ch0* e há um *bitflip* no bit de posição 1 da palavra-código em *Ch1*. Contudo, um outro evento, representado por E_y, pode apresentar zero *bitflips* em *Ch0*, mas o *bitflip* em *Ch1* pode estar localizado no bit de posição 2. Observa-se que a quantidade total de *bitflips* é a mesma, mas os eventos são diferentes. Porém, como os eventos de *bitflip* e seus complementares são considerados independentes (como no exemplo da Tabela 5.1) e a probabilidade de ocorrência de *bitflips* é igual, as probabilidades associadas aos eventos podem ser obtidas por

$$P(E_x) \simeq P(E_y) \simeq P(B_0 = 0) \times P(B_1 = 1), \tag{5.18}$$

em que $P(B_0 = 0)$ e $P(B_1 = 1)$ representam as probabilidades de ocorrerem zero *bitflips* em *Ch0* e um bitflip em *Ch1*, respectivamente. Usando P_b e P_{nb}, essas probabilidades são dadas por:

$$P(B_0 = 0) \simeq P_{nb}^{N_g}, \tag{5.19}$$

$$P(B_1 = 1) \simeq P_{nb}^{(N_g-1)} \times P_b. \tag{5.20}$$

A equação 5.19 deriva do fato de que a não ocorrência de *bitflips* em *Ch0* está relacionada com a probabilidade de não ocorrência de *bitflip* (P_{nb}) em cada um dos N_g bits que formam a palavra-código em *Ch0*. Como os eventos são independentes, essas probabilidades se multiplicam. No que se refere à equação 5.20, como é considerada a ocorrência de *bitflip* em um dos bits da palavra-código em *Ch1*, os bits restantes não devem apresentar *bitflips*, ou seja, $N_g - 1$ bits. Uma vez que a probabilidade de ocorrência de *bitflip* é dada por P_b, a multiplicação dessa probabilidade com cada uma das probabilidades de não ocorrência de *bitflip* resulta na equação 5.20.

De maneira geral,

$$P(B_0 = n_z) \simeq P_{nb}^{(N_g-n_z)} \times P_b^{n_z},$$
$$P(B_1 = n_u) \simeq P_{nb}^{(N_g-n_u)} \times P_b^{n_u}.$$

em que $P(B_0 = 0)$ e $P(B_1 = n_u)$ representam as probabilidades de ocorrerem n_z *bitflips* em *Ch0* e n_u bitflips em *Ch1*, respectivamente.

Substituindo as equações 5.19 e 5.20 em 5.18, tem-se

$$P(E_{ex}) \simeq P(E_x) \simeq P(E_y) \simeq (P_{nb}^{N_g}) \times (P_{nb}^{(N_g-1)} \times P_b). \tag{5.21}$$

Importante notar que os eventos E_x e E_y não são os únicos em que se observa apenas um *bitflip* na palavra-código em *Ch1*. A quantidade de eventos com essa característica pode ser obtida por meio da equação 5.17, considerando $N_a = N_b = N_g$, de modo que:

$$\begin{aligned} T_C(n_z, n_u) &= \binom{N_g}{n_z} \times \binom{N_g}{n_u}, \\ T_C(0,1) &= \binom{N_g}{0} \times \binom{N_g}{1}, \\ T_C(0,1) &= 1 \times N_g, \\ T_C(0,1) &= N_g. \end{aligned}$$

Além disso, todos esses eventos possuem a mesma probabilidade, a qual é dada pela equação 5.21, e eles são mutuamente excludentes, pois não acontecem de maneira simultânea. Desse modo, a probabilidade de não ocorrência de *bitflips* em *Ch0* e ocorrência de apenas um *bitflip* em *Ch1*, simbolizada por $P(E_{et})$, é dada por:

$$\begin{aligned} P(E_{et}) &= T_C(0,1) \times P(E_{ex}), \\ P(E_{et}) &\simeq N_g \times P_{nb}^{N_g} \times P_{nb}^{(N_g-1)} \times P_b \end{aligned}$$

O exemplo descrito para os eventos E_x e E_y auxilia na compreensão de que a análise de probabilidade deve ter como base a determinação da quantidade de eventos contendo um número específico de *bitflips*. Nesse sentido, para o caso 5.1.1, a quantidade de eventos para cada um dos possíveis conjuntos formados pelas palavras-código em *Ch0* e *Ch1*, sendo $N_a = N_b = N_g$, pode ser obtida por:

$$\begin{aligned} T_C(n_z, n_u) &= \binom{N_g}{n_z} \times \binom{N_g}{n_u}, \\ T_C(0, n_u) &= \binom{N_g}{0} \times \binom{N_g}{n_u}, \\ T_C(0, n_u) &= \binom{N_g}{n_u}, \end{aligned}$$

com $0 \leq n_u \leq N_g$. Ademais, a probabilidade associada aos eventos do caso 5.1.1 com n_u *bitflips* em *Ch1* é dada por:

$$P(E_u) \simeq T_C(0, n_u) \times P(B_0 = 0) \times P(B_1 = n_u),$$

e os eventos relacionados a esse caso são mutuamente excludentes, de maneira que:

$$
\begin{aligned}
P(E_{\mathrm{C_1}}) &\simeq \sum_{n_\mathrm{u}=0}^{N_\mathrm{g}} P(E_\mathrm{u}),\\
P(E_{\mathrm{C_1}}) &\simeq \sum_{n_\mathrm{u}=0}^{N_\mathrm{g}} \left(T_\mathrm{C}(0,n_\mathrm{u}) \times P(B_0=0) \times P(B_1=n_\mathrm{u})\right),\\
P(E_{\mathrm{C_1}}) &\simeq P_\mathrm{nb}^{N_\mathrm{g}} \times \sum_{n_\mathrm{u}=0}^{N_\mathrm{g}} \left(\binom{N_\mathrm{g}}{n_\mathrm{u}} P_\mathrm{nb}^{(N_\mathrm{g}-n_\mathrm{u})} P_\mathrm{b}^{n_\mathrm{u}} \right),
\end{aligned}
\tag{5.22}
$$

em que $P(E_{\mathrm{C_1}})$ simboliza a probabilidade de ocorrência dos eventos do caso 5.1.1.

A probabilidade de ocorrência dos eventos do caso 5.1.2 é determinada de maneira análoga ao que foi apresentado para o caso 5.1.1. Contudo, deve-se observar que as palavras-código que podem ser formadas em *Ch0* contêm um número ímpar de *bitflips*, enquanto as palavras-código do canal *Ch1* podem conter zero *bitflips* ou apenas um *bitflip* no bit de paridade.

A quantidade de palavras-código com número ímpar de *bitflips* que podem ser formadas em *Ch0* é obtida por

$$C_0(n_\mathrm{i}) = \binom{N_\mathrm{g}}{n_\mathrm{i}},$$

em que $n_\mathrm{i} \in \mathbb{N}_\mathrm{i}$, com $\mathbb{N}_\mathrm{i}$ correspondendo ao conjunto dos números naturais ímpares, ou seja, $\mathbb{N}_\mathrm{i} = \{1,3,5,7,\dots\}$, de maneira que n_i indica uma quantidade ímpar de *bitflips*, e $n_\mathrm{i} < N_\mathrm{g}$, se N_g for par ou $n_\mathrm{i} \leq N_\mathrm{g}$, caso contrário. Para *Ch1*, as quantidades de palavras-código que podem ser formadas com zero e com um *bitflip* são

$$
\begin{aligned}
C_1(0) &= \binom{1}{0} = 1,\\
C_1(1) &= \binom{1}{1} = 1,
\end{aligned}
$$

respectivamente. Como em *Ch1* apenas o bit de paridade pode sofrer *bitflip*, o valor N_b na equação 5.14 foi substituído por 1.

A quantidade de eventos para cada um dos possíveis conjuntos formados pelas palavras-código em *Ch0* e *Ch1* para o caso 5.1.2, pode então ser obtida por:

$$
\begin{aligned}
T_\mathrm{C}(n_\mathrm{i},0) &= C_0(n_\mathrm{i}) \times C_1(0) = C_0(n_\mathrm{i})\\
T_\mathrm{C}(n_\mathrm{i},1) &= C_0(n_\mathrm{i}) \times C_1(1) = C_0(n_\mathrm{i})
\end{aligned}
$$

em que $T_\mathrm{C}(n_\mathrm{i},0)$ representa o número de eventos formados pelos bits uma palavra-código em *Ch0* com número ímpar de *bitflips* e pelos bits de outra palavra-código em *Ch1* sem *bitflips*. Analogamente, $T_\mathrm{C}(n_\mathrm{i},1)$ representa o número de eventos formados pelos bits de uma palavra-código em *Ch0* com número ímpar de *bitflips* e pelos bits de outra palavra-código em *Ch1* com apenas um *bitflip* no bit de paridade.

Em termos de probabilidade, tem-se:

$$\begin{aligned} P(B_0 = n_\mathrm{i}) &\simeq P_\mathrm{nb}^{(N_\mathrm{g}-n_\mathrm{i})} \times P_\mathrm{b}^{n_\mathrm{i}}, \\ P(B_1 = 0) &\simeq P_\mathrm{nb}^{N_\mathrm{g}}, \\ P(B_1 = 1) &\simeq P_\mathrm{nb}^{(N_\mathrm{g}-1)} \times P_\mathrm{b} \end{aligned}$$

em que $P(B_0 = n_\mathrm{i})$, $P(B_1 = 0)$ e $P(B_1 = 1)$ são as probabilidades de ocorrerem n_i *bitflips* em *Ch0*, 0 *bitflips* em *Ch*1 e 1 *bitflip* em *Ch1*, respectivamente. Além disso, a probabilidade associada aos eventos do caso 5.1.2 com n_i *bitflips* em *Ch0* e zero *bitflips* em *Ch1* é encontrada por

$$P(E_{\mathrm{u}_0}) \simeq T_\mathrm{C}(n_\mathrm{i}, 0) \times P(B_0 = n_\mathrm{i}) \times P(B_1 = 0).$$

Por outro lado, a probabilidade associada aos eventos do caso 5.1.2 com n_i *bitflips Ch0* e um *bitflip* em *Ch1* é dada por

$$P(E_{\mathrm{u}_1}) \simeq T_\mathrm{C}(n_\mathrm{i}, 1) \times P(B_0 = n_\mathrm{i}) \times P(B_1 = 1).$$

Considerando que os eventos relacionados ao caso 5.1.2 são mutuamente excludentes, as probabilidades podem ser obtidas por:

$$\begin{aligned} P(E_{\mathrm{C}_2}) &\simeq \sum_{\substack{n_\mathrm{i}=1 \\ n_\mathrm{i}\in\mathbb{N}_\mathrm{i}}}^{N_\mathrm{g}} P(E_{\mathrm{u}_0}), \\ P(E_{\mathrm{C}_2}) &\simeq \sum_{\substack{n_\mathrm{i}=1 \\ n_\mathrm{i}\in\mathbb{N}_\mathrm{i}}}^{N_\mathrm{g}} \left(T_\mathrm{C}(n_\mathrm{i}, 0) \times P(B_0 = n_\mathrm{i}) \times P(B_1 = 0)\right), \\ P(E_{\mathrm{C}_2}) &\simeq P_\mathrm{nb}^{N_\mathrm{g}} \times \sum_{\substack{n_\mathrm{i}=1 \\ n_\mathrm{i}\in\mathbb{N}_\mathrm{i}}}^{N_\mathrm{g}} \left(\binom{N_\mathrm{g}}{n_\mathrm{i}} P_\mathrm{nb}^{(N_\mathrm{g}-n_\mathrm{i})} P_\mathrm{b}^{n_\mathrm{i}} \right), \end{aligned} \tag{5.23}$$

e

$$\begin{aligned} P(E_{\mathrm{C}_3}) &\simeq \sum_{\substack{n_\mathrm{i}=1 \\ n_\mathrm{i}\in\mathbb{N}_\mathrm{i}}}^{N_\mathrm{g}} P(E_{\mathrm{u}_1}), \\ P(E_{\mathrm{C}_3}) &\simeq \sum_{\substack{n_\mathrm{i}=1 \\ n_\mathrm{i}\in\mathbb{N}_\mathrm{i}}}^{N_\mathrm{g}} \left(T_\mathrm{C}(n_\mathrm{i}, 1) \times P(B_0 = n_\mathrm{i}) \times P(B_1 = 1)\right), \\ P(E_{\mathrm{C}_3}) &\simeq P_\mathrm{nb}^{(N_\mathrm{g}-1)} P_\mathrm{b} \times \sum_{\substack{n_\mathrm{i}=1 \\ n_\mathrm{i}\in\mathbb{N}_\mathrm{i}}}^{N_\mathrm{g}} \left(\binom{N_\mathrm{g}}{n_\mathrm{i}} P_\mathrm{nb}^{(N_\mathrm{g}-n_\mathrm{i})} P_\mathrm{b}^{n_\mathrm{i}} \right), \end{aligned} \tag{5.24}$$

em que $P(E_{\mathrm{C}_2})$ é a probabilidade de ocorrência dos eventos formados pela palavra-código

em *Ch0* com número ímpar de *bitflips* e pela palavra-código em *Ch1* com zero *bitflips*. Enquanto isso, $P(E_{C_3})$ simboliza a probabilidade de ocorrência dos eventos formados pela palavra-código em *Ch1* contendo um *bitflip* e pela palavra-código em *Ch0* com um número ímpar de *bitflips*. Ademais, uma vez que n_i é ímpar, $n_i < N_g$, se N_g for par ou $n_i \leq N_g$, caso contrário.

A união dos eventos dos casos 5.1.1 e 5.1.2 inclui todos os eventos que geram um resultado de saída correto. Desse modo, uma vez que esses eventos são mutuamente excludentes, a probabilidade de correção ($P(E_B)$) é dada por

$$P(E_B) \simeq P(E_{C_1}) + P(E_{C_2}) + P(E_{C_3}). \tag{5.25}$$

Substituindo as equações 5.22, 5.23 e 5.24 em 5.25, tem-se

$$P(E_B) \simeq P_{nb}^{N_g} \times \sum_{n_u=0}^{N_g} \left(\binom{N_g}{n_u} P_{nb}^{(N_g - n_u)} P_b^{n_u} \right) + (P_{nb}^{N_g} + P_{nb}^{(N_g-1)} P_b) \times \sum_{\substack{n_i=1 \\ n_i \in \mathbb{N}_i}}^{N_g} \left(\binom{N_g}{n_i} P_{nb}^{(N_g - n_i)} P_b^{n_i} \right), \tag{5.26}$$

em que N_g é o número total de bits em um grupo; P_b é a probabilidade de ocorrer um *bitflip*; P_{nb} é a probabilidade de não ocorrer um *bitflip*; n_u é o número de *bitflips* em *Ch*1 e $0 \leq n_u \leq N_g$; n_i é um número ímpar ($n_i \in \mathbb{N}_i$) de *bitflips* em *Ch0* e $n_i < N_g$, se N_g for par ou $n_i \leq N_g$, caso contrário.

A equação 5.26 pode ser utilizada para determinar a probabilidade de correção em um dado de N_g bits. Isso significa que ela deve ser aplicada ao conjunto de bits que formam um único grupo. Contudo, a proposta apresentada para a técnica PDMR se baseia na divisão de um dado de tamanho N em grupos compostos por uma paridade e M bits de informação, sendo a quantidade de grupos definida por N_p. Desse modo, é necessária uma equação que possa ser aplicada ao conjunto inteiro de bits transmitidos usando a técnica proposta.

Nesse contexto, suponha que um dado de entrada de N bits é dividido em dois grupos com N_g bits cada, formando uma palavra-código de N_d bits. Para que ocorra uma correção, os casos 5.1.1 e 5.1.2 devem ser verificados em ambos os grupos. Em outras palavras, a condição para a correção é que haja uma correção no primeiro grupo e uma correção no segundo grupo. Desse modo, sendo os eventos de correção eventos independentes, a probabilidade de correção $P(C_{pdmrb})$ é dada por

$$P(C_{pdmrb}) = P(E_{B_1} \cap E_{B_2}) \simeq P(E_{B_1}) \times P(E_{B_2}),$$

em que $P(E_{B_1})$ e $P(E_{B_2})$ representam as probabilidades de correção no primeiro e no segundo grupos, respectivamente. Visto que os grupos possuem a mesma quantidade de bits de informação M e de paridade, $P(E_{B_1}) = P(E_{B_2}) = P(E_B)$. Portanto,

$$P(C_{pdmrb}) \simeq P(E_B)^2. \tag{5.27}$$

O expoente de $P(E_{\mathrm{B}})$ na equação 5.27 corresponde à quantidade de grupos que formam a palavra-código. Sendo assim, de modo geral,

$$P(C_{\mathrm{pdmrb}}) \simeq P(E_{\mathrm{B}})^{N_{\mathrm{p}}}, \tag{5.28}$$

em que N_{p} é o número de total de bits de paridade, equivalente também ao número de grupos.

Para verificar a aplicação das equações 5.26 e 5.28, considere novamente o exemplo da Tabela 5.1 em que:

$$N = M = 1, N_{\mathrm{p}} = 1, N_{\mathrm{d}} = N_{\mathrm{g}} = 2.$$

Logo, $n_{\mathrm{i}} \in \{1\}$ e, utilizando a equação 5.26, tem-se:

$$P(E_{\mathrm{B_{ex1}}}) \simeq P_{\mathrm{nb}}^2 \times \sum_{n_{\mathrm{u}}=0}^{2} \left(\binom{2}{n_{\mathrm{u}}} P_{\mathrm{nb}}^{(2-n_{\mathrm{u}})} P_{\mathrm{b}}^{n_{\mathrm{u}}} \right) + (P_{\mathrm{nb}}^2 + P_{\mathrm{nb}}^{(2-1)} P_{\mathrm{b}}) \times \sum_{\substack{n_{\mathrm{i}}=1 \\ n_{\mathrm{i}} \in \mathbb{N}_{\mathrm{i}}}}^{2} \left(\binom{2}{n_{\mathrm{i}}} P_{\mathrm{nb}}^{(2-n_{\mathrm{i}})} P_{\mathrm{b}}^{n_{\mathrm{i}}} \right),$$

$$\begin{aligned} P(E_{\mathrm{B_{ex1}}}) &\simeq P_{\mathrm{nb}}^2 \times (P_{\mathrm{nb}}^2 + 2 \times P_{\mathrm{nb}}^1 P_{\mathrm{b}}^1 + P_{\mathrm{b}}^2) + (P_{\mathrm{nb}}^2 + P_{\mathrm{nb}} P_{\mathrm{b}}) \times (2 \times P_{\mathrm{nb}}^1 P_{\mathrm{b}}^1), \\ &\simeq P_{\mathrm{nb}}^4 + 2 \times P_{\mathrm{nb}}^3 P_{\mathrm{b}}^1 + P_{\mathrm{nb}}^2 P_{\mathrm{b}}^2 + 2 \times P_{\mathrm{nb}}^3 P_{\mathrm{b}}^1 + 2 \times P_{\mathrm{nb}}^2 P_{\mathrm{b}}^2, \\ P(E_{\mathrm{B_{ex1}}}) &\simeq P_{\mathrm{nb}}^4 + 4 \times P_{\mathrm{nb}}^3 P_{\mathrm{b}}^1 + 3 \times P_{nb}^2 P_{b}^2. \end{aligned} \tag{5.29}$$

Ao comparar as equações 5.13 e 5.29, observa-se que elas são iguais e, consequentemente, a equação 5.13 é um caso particular da equação 5.26. Ademais, a probabilidade de correção para o exemplo da Tabela 5.1 é determinado pela equação 5.28, sendo

$$\begin{aligned} P(C_{\mathrm{pdmrb}}) &\simeq (P_{\mathrm{nb}}^4 + 4 \times P_{\mathrm{nb}}^3 P_{\mathrm{b}}^1 + 3 \times P_{\mathrm{nb}}^2 P_{\mathrm{b}}^2)^{N_{\mathrm{p}}}, \\ P(C_{\mathrm{pdmrb}}) &\simeq P_{\mathrm{nb}}^4 + 4 \times P_{\mathrm{nb}}^3 P_{\mathrm{b}}^1 + 3 \times P_{\mathrm{nb}}^2 P_{\mathrm{b}}^2. \end{aligned}$$

Por fim, a utilização da equação 5.26 no exemplo da Figura 5.1, com $N = M = 2$, $N_{\mathrm{p}} = 1$, $N_{\mathrm{d}} = N_{\mathrm{g}} = 3$ e $n_{\mathrm{i}} \in \{1, 3\}$. tem como resultado:

$$P(E_{\mathrm{B_{ex2}}}) \simeq P_{\mathrm{nb}}^3 \times \sum_{n_{\mathrm{u}}=0}^{3} \left(\binom{3}{n_{\mathrm{u}}} P_{\mathrm{nb}}^{(3-n_{\mathrm{u}})} P_{\mathrm{b}}^{n_{\mathrm{u}}} \right) + (P_{\mathrm{nb}}^3 + P_{\mathrm{nb}}^{(3-1)} P_{\mathrm{b}}) \times \sum_{\substack{n_{\mathrm{i}}=1 \\ n_{\mathrm{i}} \in \mathbb{N}_{\mathrm{i}}}}^{3} \left(\binom{3}{n_{\mathrm{i}}} P_{\mathrm{nb}}^{(3-n_{\mathrm{i}})} P_{\mathrm{b}}^{n_{\mathrm{i}}} \right),$$

$$\begin{aligned} P(E_{\mathrm{B_{ex2}}}) &\simeq P_{\mathrm{nb}}^3 \times (P_{\mathrm{nb}}^3 + 3 \times P_{\mathrm{nb}}^2 P_{\mathrm{b}}^1 + 3 \times P_{\mathrm{nb}}^1 P_{\mathrm{b}}^2 + P_{\mathrm{b}}^3) + (P_{\mathrm{nb}}^3 + P_{\mathrm{nb}}^2 P_{\mathrm{b}}) \times (3 \times P_{\mathrm{nb}}^2 P_{\mathrm{b}}^1 + P_{\mathrm{b}}^3), \\ &\simeq P_{\mathrm{nb}}^6 + 3 \times (P_{\mathrm{nb}}^5 P_{\mathrm{b}}^1 + P_{\mathrm{nb}}^4 P_{\mathrm{b}}^2 + P_{\mathrm{nb}}^5 P_{\mathrm{b}}^1 + P_{\mathrm{nb}}^4 P_{\mathrm{b}}^2) + P_{\mathrm{nb}}^3 P_{\mathrm{b}}^3 + P_{\mathrm{nb}}^3 P_{\mathrm{b}}^3 + P_{\mathrm{nb}}^2 P_{\mathrm{b}}^4, \end{aligned}$$

$$P(E_{\mathrm{B_{ex2}}}) \simeq P_{\mathrm{nb}}^{6} + 6 \times P_{\mathrm{nb}}^{5} P_{\mathrm{b}}^{1} + 6 \times P_{\mathrm{nb}}^{4} P_{\mathrm{b}}^{2} + 2 \times P_{\mathrm{nb}}^{3} P_{\mathrm{b}}^{3} + P_{\mathrm{nb}}^{2} P_{\mathrm{b}}^{4}. \tag{5.30}$$

A análise dos eventos formados pelos pares ilustrados na Figura 5.1, com base no raciocínio descrito para o exemplo da Tabela 5.1, permite que uma equação igual à equação 5.30 seja obtida.

5.1.2 Erro

A aplicação da versão PDMR-B tem como resultado dados que que podem estar corretos ou errados. Desse modo, uma vez calculada a probabilidade de correção, a probabilidade de erro pode ser encontrada por

$$P(E_{\mathrm{pdmrb}}) \simeq 1 - P(C_{\mathrm{pdmrb}}). \tag{5.31}$$

5.2 Cálculos de Probabilidade para PDMR-G

A versão PDMR-G se caracteriza, principalmente, pela inclusão de uma lógica capaz de detectar possíveis erros na transmissão dos dados. A indicação de erro é realizada por meio de operações *AND* entre os bits de seleção de canal, cuja expressão lógica difere da PDMR-B, e os respectivos bits de *flag* de erro. Desse modo, para a versão PDMR-G, o dado observado na saída pode estar correto, pode conter possíveis erros detectáveis ou pode estar errado. Nesse sentido, as próximas seções contêm a descrição detalhada dos cálculos para obtenção das probabilidades de correção, de detecção e de erro para a PDMR-G.

5.2.1 Correção

Para a análise da probabilidade de correção, inicialmente, assim como foi realizado no caso da técnica PDMR-B, suponha que o dado $0b0$ deve ser transmitido usando PDMR-G, de modo que

$$N = M = 1, N_{\mathrm{p}} = 1, N_{\mathrm{d}} = N_{\mathrm{g}} = 2.$$

A quantidade de eventos possíveis para transmissão desse dado nos dois canais é obtida pela equação 5.2. Desse modo,

$$\begin{aligned} T &= 2^{2N_{\mathrm{g}}}, \\ T &= 2^{2\times 2}, \\ T &= 16. \end{aligned}$$

Com base nessa informação, pode-se construir uma tabela com todas as possibilidades de eventos para transmissão do dado $0b0$. Os eventos possíveis estão dispostos na Tabela 5.2, em que a classificação na coluna *Estado* foi realizada de acordo com as equações 4.22 à 4.24, assim como, levando-se em consideração o resultado de saída.

Considerando todos os eventos apresentados na Tabela 5.2, observa-se que, como não há indicação de erro quando os bits de paridade transmitidos e calculados no receptor são

Tabela 5.2: Eventos possíveis para transmissão do dado $0b0$ com paridade par para a versão PDMR-G

Evento	*Ch0*		*Ch1*		Receptor		Bit de Saída	Estado
	p_{0_0}	b_{0_0}	p_{0_1}	b_{0_1}	r_{0_0}	r_{0_1}		
E_1	0	0	0	0	0	0	0	Correto
E_2	0	0	0	1	0	1	0	Correto
E_3	0	0	1	0	0	0	0	Correto
E_4	0	0	1	1	0	1	1	Detectado
E_5	0	1	0	0	1	0	0	Correto
E_6	0	1	0	1	1	1	1	Detectado
E_7	0	1	1	0	1	0	0	Detectado
E_8	0	1	1	1	1	1	1	Erro
E_9	1	0	0	0	0	0	0	Correto
E_{10}	1	0	0	1	0	1	1	Detectado
E_{11}	1	0	1	0	0	0	0	Detectado
E_{12}	1	0	1	1	0	1	1	Erro
E_{13}	1	1	0	0	1	0	0	Detectado
E_{14}	1	1	0	1	1	1	1	Erro
E_{15}	1	1	1	0	1	0	1	Erro
E_{16}	1	1	1	1	1	1	1	Erro

Fonte: O próprio autor.

iguais (E_1 e E_{16}), ocorrem uma correção quando não há *bitflips* nos dois canais (E_1) e um erro quando todos bits dos canais sofrem *bitflip* (E_{16}). Esses eventos são considerados especiais, pois só acontecem uma vez no universo de eventos possíveis. Além disso, para os casos em que ocorrem dois *bitflips* (E_4, E_6, E_7, E_{10}, E_{11} e E_{13}), tem-se uma sinalização para detecção de erro conforme equação 4.24. Os erros acontecem quando a quantidade total de *bitflips* é igual a três (E_8, E_{12}, E_{14} e E_{15}). No que se refere aos demais resultados de correção (eventos E_2, E_3, E_5 e E_9), verifica-se que há duas condições para que ocorram:

- Ao menos um dos canais deve conter o bit de informação e de paridade corretos;
- O outro canal deve sofrer apenas um *bitflip*.

Assim, o número total de eventos corretos pode ser escrito como

$$E_{\mathrm{G_x}} = E_1 + E_2 + E_3 + E_5 + E_9,$$

em que $E_{\mathrm{G_x}}$ representa a quantidade total de eventos corretos. A probabilidade de correção ($P(E_{\mathrm{G_x}})$) para a transmissão de $0b0$ com PDMR é, então, dada por

$$P(E_{\mathrm{G_x}}) = P(E_1 \cup E_2 \cup E_3 \cup E_5 \cup E_9)$$

e sendo os eventos disjuntos, tem-se

$$P(E_{\mathrm{G_x}}) = P(E_1) + P(E_2) + P(E_3) + P(E_5) + P(E_9) \tag{5.32}$$

em que $P(E_1)$, $P(E_2)$, $P(E_3)$, $P(E_5)$ e $P(E_9)$ são as probabilidades de ocorrência dos eventos E_1, E_2, E_3, E_5 e E_9, respectivamente.

Seguindo o raciocínio apresentado para o exemplo da Tabela 5.1, os eventos de *bitflip* para p_{0_0}, b_{0_0}, p_{0_1} e b_{0_1} são representados por X_0, Y_0, X_1 e Y_1, respectivamente. Por outro lado, os complementares X_0^c, Y_0^c, X_1^c e Y_1^c indicam os eventos de *não bitflip*. Sendo assim, observando E_1, E_2, E_3, E_5 e E_9 na Tabela 5.2, obtém-se:

$$P(E_1) = P(X_0^c \cap Y_0^c \cap X_1^c \cap Y_1^c), \tag{5.33}$$
$$P(E_2) = P(X_0^c \cap Y_0^c \cap X_1^c \cap Y_1), \tag{5.34}$$
$$P(E_3) = P(X_0^c \cap Y_0^c \cap X_1 \cap Y_1^c), \tag{5.35}$$
$$P(E_5) = P(X_0^c \cap Y_0 \cap X_1^c \cap Y_1^c), \tag{5.36}$$
$$P(E_9) = P(X_0 \cap Y_0^c \cap X_1^c \cap Y_1^c), \tag{5.37}$$

Usando P_b e P_nb para as probabilidade de *bitflip* e *não bitflip*, respectivamente, tem-se

$$P(X_0) = P(Y_0) = P(X_1) = P(Y_1) = P_\mathrm{b}$$

e

$$P(X_0^c) = P(Y_0^c) = P(X_1^c) = P(Y_1^c) = P_\mathrm{nb}.$$

Adicionalmente, assumindo que os eventos X_0, Y_0, X_1 e Y_1 são independentes, as equações 5.33 à 5.37 se tornam:

$$\begin{aligned}
P(E_1) &\simeq P(X_0^c) \times P(Y_0^c) \times P(X_1^c) \times P(Y_1^c) \simeq P_\mathrm{nb}^4,\\
P(E_2) &\simeq P(X_0^c) \times P(Y_0^c) \times P(X_1^c) \times P(Y_1) \simeq P_\mathrm{nb}^3 \times P_\mathrm{b},\\
P(E_3) &\simeq P(X_0^c) \times P(Y_0^c) \times P(X_1) \times P(Y_1^c) \simeq P_\mathrm{nb}^3 \times P_\mathrm{b},\\
P(E_5) &\simeq P(X_0^c) \times P(Y_0) \times P(X_1^c) \times P(Y_1^c) \simeq P_\mathrm{nb}^3 \times P_\mathrm{b},\\
P(E_9) &\simeq P(X_0) \times P(Y_0^c) \times P(X_1^c) \times P(Y_1^c) \simeq P_\mathrm{nb}^3 \times P_\mathrm{b}.
\end{aligned}$$

Logo, substituindo os valores de $P(E_1)$, $P(E_2)$, $P(E_3)$, $P(E_5)$ e $P(E_9)$ na equação 5.32, obtém-se

$$P(E_{\mathrm{G_x}}) \simeq P_\mathrm{nb}^4 + 4 \times P_\mathrm{nb}^3 \times P_\mathrm{b}. \tag{5.38}$$

Diante de uma observação mais detalhada da Tabela 5.2, percebe-se que a quantidade total de eventos corretos pode ser obtida com base nos quatro primeiros eventos. Nesse sentido, o estado é correto para o caso especial (E_1) e para E_2 e E_3, os quais satisfazem as condições supracitadas. Os valores binários em E_2 são $0b00$ para *Ch0* e $0b01$ para *Ch1*, enquanto que em E_3 são $0b00$ e $0b10$. No entanto, os valores $0b01$ para *Ch0* e $0b00$ para *Ch1*, bem como $0b10$ para *Ch0* e $0b00$ para *Ch1* também fazem parte do universo de eventos (E_5 e E_9). Ou seja, são os casos em que se tem uma troca entre os valores de $Ch0$ e $Ch1$ no que se refere ao observado em E_2 e E_3. Uma vez que a probabilidade de ocorrência de *bitflips* é a mesma, $P(E_2) = P(E_5)$ e $P(E_3) = P(E_9)$. Desse modo, a

equação 5.32 pode também ser escrita como:

$$P(E_{\mathrm{G_x}}) \simeq P(E_1) + 2 \times P(E_2) + 2 \times P(E_3).$$

Partindo dessa equação, a expressão 5.38 pode ser encontrada, levando em consideração que os eventos de *bitflip* são independentes.

Destaca-se que os eventos observados (E_{O}) nas quatro primeiras linhas da Tabela 5.2 são aqueles em que *Ch*0 possui o valor 0*b*00, o que equivale a

$$E_{\mathrm{O}} = 2^{N_{\mathrm{g}}}. \tag{5.39}$$

Com o intuito de compreender o comportamento da PDMR-G na transmissão de dados com mais bits de informação, imagine agora que a técnica é aplicada na transmissão do dado 0*b*00, de tal maneira que:

$$N = M = 2, N_{\mathrm{p}} = 1, N_{\mathrm{d}} = N_{\mathrm{g}} = 3.$$

Usando a equação 5.2, $T = 2^{2N_{\mathrm{g}}} = 2^6 = 64$, o que significa que há 64 eventos possíveis para a transmissão do dado 0*b*00. Desse total, com base na equação 5.39, $E_{\mathrm{O}} = 2^{N_{\mathrm{g}}} = 2^3 = 8$. Portanto, os 8 primeiros eventos podem ser observados para obtenção do número de eventos com resultado correto. Esses eventos são apresentados na Tabela 5.3.

Tabela 5.3: Os 8 primeiros eventos possíveis para transmissão do dado 0*b*00 com paridade par para a versão PDMR-G

Evento	*Ch0*			*Ch1*			Receptor		Saída	Estado
	p_{0_0}	b_{0_0}	b_{1_0}	p_{0_1}	b_{0_1}	b_{1_1}	r_{0_0}	r_{0_1}		
E_1	0	0	0	0	0	0	0	0	00	Correto
E_2				0	0	1	0	1	00	Correto
E_3				0	1	0	0	1	00	Correto
E_4				0	1	1	0	0	00	Correto
E_5				1	0	0	0	0	00	Correto
E_6				1	0	1	0	1	01	Detectado
E_7				1	1	0	0	1	10	Detectado
E_8				1	1	1	0	0	00	Correto

Fonte: O próprio autor.

Na Tabela 5.3, assim como para o caso de transmissão do dado 0*b*0, E_1 é o evento especial cuja combinação de bits acontece um única vez no universo de eventos. Os eventos que geram resultados corretos E_3, E_4 e E_5 são aqueles em que se observa apenas um *bitflip* no canal *Ch1*. A combinação de bits desses eventos, do mesmo modo que acontece com os eventos que apresentam um único *bitflip* na Tabela 5.2, repete-se quando *Ch0* possui valores 0*b*001, 0*b*010 e 0*b*100, enquanto o valor de *Ch1* é 0*b*000.

No caso de E_4, apesar do resultado ser correto dada a combinação de bits apresentada na Tabela 5.3, no evento em que os bits de *Ch0* e *Ch1* são trocados, ou seja, *Ch0* e *Ch1* possuem valores binários 0*b*011 e 0*b*000, respectivamente, há um resultado errado. Para

entender essa afirmação, observe que os valores das paridades calculadas no receptor (r_{0_0} e r_{0_1}), junto às paridades de p_{0_0} e p_{0_1}, indicam que o resultado é tomado de *Ch0*, conforme Tabela 4.2, bem como as equações 4.22 à 4.24. Consequentemente, quando *Ch0* é $0b011$ e *Ch1* é $0b000$, o valor $0b011$ é colocado na saída, o que significa uma mensagem errada.

O evento E_8 na Tabela 5.3 também implica em um resultado correto. Esse evento representa a ocorrência de três *bitflips* no canal *Ch1*, enquanto o canal *Ch0* mantém os bits originais. O caso oposto, em que os bits de *Ch0* sofrem *bitflip* e os de *Ch1* permanecem os mesmos, também representa um resultado correto. Isso pode ser verificado com base nos bits de paridade p_{0_0} e p_{0_1} e os bits de paridade calculados no receptor (r_{0_0} e r_{0_1}), os quais indicam que a saída deve ser tomada de $Ch1$.

Os eventos E_6 e E_7 demandam uma observação detalhada. Para esses eventos, a quantidade de *bitflips* em *Ch1* é idêntica a do evento E_4 e totaliza 2. Contudo, as lógicas baseadas nas paridades (p_{0_0}, p_{0_1}, r_{0_0} e r_{0_1}) indicam uma detecção. Isso se dá pelo fato dos *bitflips* ocorrerem na paridade (p_{0_1}) e em um dos bits de informação.

No que se refere à probabilidade de correção, seguindo o raciocínio realizado para os eventos da Tabela 5.2 e com base na Tabela 5.3, tem-se

$$P(E_{\mathrm{G_y}}) \simeq P(E_1) + 2 \times P(E_2) + 2 \times P(E_3) + P(E_4) + 2 \times P(E_5) + 2 \times P(E_8). \quad (5.40)$$

Como o evento oposto à E_2 (i.e., $Ch0 : \{0b001\}$ e $Ch1 : \{0b000\}$), também faz parte do universo de eventos, sua probabilidade é tida como a mesma que de E_2. Isso acontece pelo fato de que a probabilidade de ocorrência de *bitflips* é considerada igual para qualquer bit dos canais. Dessa maneira, a soma das probabilidades de E_2 e de seu evento oposto resulta em $2 \times P(E_2)$. O mesmo ocorre para os eventos E_3 e E_5 e seus eventos opostos. Como discutido anteriormente, E_1 é um evento cujo valor binário é único e o evento oposto à E_4 resulta em um erro. Em consequência, os coeficientes associados à $P(E_1)$ e $P(E_4)$ na equação 5.40, têm valor 1.

Usando as probabilidades de *bitflip* (P_{b}) e *não bitflip* (P_{nb}), as probabilidades da equação 5.40 podem ser escritas como:

$$\begin{aligned}
P(E_1) &\simeq P_{\mathrm{nb}}^6, \\
P(E_2) &\simeq P_{\mathrm{nb}}^5 \times P_{\mathrm{b}}, \\
P(E_3) &\simeq P_{\mathrm{nb}}^5 \times P_{\mathrm{b}}, \\
P(E_4) &\simeq P_{\mathrm{nb}}^4 \times P_{\mathrm{b}}^2, \\
P(E_5) &\simeq P_{\mathrm{nb}}^5 \times P_{\mathrm{b}}, \\
P(E_8) &\simeq P_{\mathrm{nb}}^3 \times P_{\mathrm{b}}^3.
\end{aligned}$$

Substituindo em 5.40,

$$P(E_{\mathrm{G_y}}) \simeq P_{\mathrm{nb}}^6 + 6 \times P_{\mathrm{nb}}^5 \times P_{\mathrm{b}} + P_{\mathrm{nb}}^4 \times P_{\mathrm{b}}^2 + 2 \times P_{\mathrm{nb}}^3 \times P_{\mathrm{b}}^3.$$

Fórmula Geral

Os exemplos apresentados servem como base para a compreensão do comportamento da versão PDMR-G. Nesse contexto, dadas as observações descritas anteriormente, os casos em que a técnica PDMR-G gera correção podem ser generalizados, sendo os seguintes:

Caso 5.2.1. Os bits de informação e paridade dos dois canais não sofrem *bitflip*;

Caso 5.2.2. Ao menos um dos canais contém os bits de informação e de paridade corretos. Enquanto isso, o número total de *bitflips* no outro canal é ímpar;

Caso 5.2.3. O canal *Ch0* possui os bits de informação e paridade corretos e o canal *Ch1* apresenta um número par de *bitflips*. Contudo, esses *bitflips* acontecem apenas nos bits de informação.

De forma geral, para a transmissão de um dado binário com $N_{\mathrm{g}} = N_{\mathrm{d}}$, o caso 5.2.1 corresponde ao evento em que não há ocorrência de *bitflips*, cuja probabilidade é dada por

$$\begin{aligned} P(E_{\mathrm{Cg_1}}) &\simeq T_{\mathrm{C}}(0,0) \times P(B_0 = 0) \times P(B_1 = 0), \\ P(E_{\mathrm{Cg_1}}) &\simeq P_{\mathrm{nb}}^{2N_{\mathrm{g}}}. \end{aligned} \tag{5.41}$$

Os eventos para o caso 5.2.2 podem ser deduzidos da teoria de análise combinatória. Sendo assim, escolhendo um subconjunto de n_{i} posições contendo *bitflips* de um conjunto de N_{g} posições, as posições restantes não devem conter *bitflips*. Portanto, a determinação do número de eventos equivale à encontrar a quantidade de combinações de N_{g} elementos agrupados n_{i} a n_{i}, em que n_{i} é um número ímpar de *bitflips*.

Considerando o total de bits dos dois canais, a ocorrência de n_{i} *bitflips* implica na não ocorrência de $2N_{\mathrm{g}} - n_{\mathrm{i}}$ *bitflips*. Além disso, como já discutido, o caso 5.2.2 inclui os eventos em que *Ch0* possui bits corretos e *Ch1* possui um número ímpar de *bitflips*, bem como os eventos em que se observa o oposto. Esses eventos são disjuntos, consequentemente, a probabilidade de ocorrência é dada por

$$\begin{aligned} P(E_{\mathrm{Cg_2}}) \simeq \sum_{\substack{n_{\mathrm{i}}=1 \\ n_{\mathrm{i}} \in \mathbb{N}_{\mathrm{i}}}}^{N_{\mathrm{g}}} & \left(T_{\mathrm{C}}(0,n_{\mathrm{i}}) \times P(B_0 = 0) \times P(B_1 = n_{\mathrm{i}})\right) \\ & + \sum_{\substack{n_{\mathrm{i}}=1 \\ n_{\mathrm{i}} \in \mathbb{N}_{\mathrm{i}}}}^{N_{\mathrm{g}}} \left(T_{\mathrm{C}}(n_{\mathrm{i}},0) \times P(B_0 = n_{\mathrm{i}}) \times P(B_1 = 0)\right), \end{aligned}$$

$$P(E_{\mathrm{Cg_2}}) \simeq \sum_{\substack{n_{\mathrm{i}}=1 \\ n_{\mathrm{i}} \in \mathbb{N}_{\mathrm{i}}}}^{N_{\mathrm{g}}} \left(\binom{N_{\mathrm{g}}}{n_{\mathrm{i}}} P_{\mathrm{nb}}^{N_{\mathrm{g}}} P_{\mathrm{nb}}^{(N_{\mathrm{g}}-n_{\mathrm{i}})} P_{\mathrm{b}}^{n_{\mathrm{i}}} \right) + \sum_{\substack{n_{\mathrm{i}}=1 \\ n_{\mathrm{i}} \in \mathbb{N}_{\mathrm{i}}}}^{N_{\mathrm{g}}} \left(\binom{N_{\mathrm{g}}}{n_{\mathrm{i}}} P_{\mathrm{nb}}^{(N_{\mathrm{g}}-n_{\mathrm{i}})} P_{\mathrm{b}}^{n_{\mathrm{i}}} P_{\mathrm{nb}}^{N_{\mathrm{g}}} \right),$$

$$P(E_{\mathrm{Cg_2}}) \simeq 2 \times P_{\mathrm{nb}}^{N_{\mathrm{g}}} \times \sum_{\substack{n_{\mathrm{i}}=1 \\ n_{\mathrm{i}} \in \mathbb{N}_{\mathrm{i}}}}^{N_{\mathrm{g}}} \left(\binom{N_{\mathrm{g}}}{n_{\mathrm{i}}} \times P_{\mathrm{nb}}^{(N_{\mathrm{g}}-n_{\mathrm{i}})} \times P_{\mathrm{b}}^{n_{\mathrm{i}}} \right), \tag{5.42}$$

em que $n_\mathrm{i} \in \mathbb{N}_\mathrm{i}$, ou seja, é um número natural ímpar e $n_\mathrm{i} < N_\mathrm{g}$, se N_g for par ou $n_\mathrm{i} \leq N_\mathrm{g}$, caso contrário. Se $n_\mathrm{i} > N_\mathrm{g}$ os dois canais possuem *bitflips*, o que indica um evento sem correção.

O caso 5.2.3 requer atenção, uma vez que um canal não deve conter *bitflips*, enquanto eles devem ocorrer apenas em bits de informação do outro canal. Nesse sentido, de acordo com o raciocínio apresentado para a Figura 5.2, os eventos devem corresponder ao número de combinações de $N_\mathrm{g} - 1$ (o bit de paridade é desconsiderado) elementos agrupados n_p a n_p, em que n_p é o número par de *bitflips* e $n_\mathrm{p} > 0$. Como os eventos para o caso 5.2.3 são mutuamente excludentes, os eventos opostos não resultam em correção e $N_\mathrm{a} = N_\mathrm{g} - 1$, a probabilidade de ocorrência dos eventos é obtida por

$$
\begin{aligned}
P(E_{\mathrm{Cg}_3}) &\simeq \sum_{\substack{n_\mathrm{p}=2 \\ n_\mathrm{p}\in\mathbb{N}_\mathrm{p}^*}}^{(N_\mathrm{g}-1)} \left(T_\mathrm{C}(0,n_\mathrm{p}) \times P(B_0 = 0) \times P(B_1 = n_\mathrm{p})\right), \\
P(E_{\mathrm{Cg}_3}) &\simeq \sum_{\substack{n_\mathrm{p}=2 \\ n_\mathrm{p}\in\mathbb{N}_\mathrm{p}^*}}^{(N_\mathrm{g}-1)} \left(\binom{N_\mathrm{g}-1}{n_\mathrm{p}} P_\mathrm{nb}^{N_\mathrm{g}} P_\mathrm{nb}^{(N_\mathrm{g}-n_\mathrm{p})} P_\mathrm{b}^{n_\mathrm{p}} \right), \\
P(E_{\mathrm{Cg}_3}) &\simeq P_\mathrm{nb}^{N_\mathrm{g}} \times \sum_{\substack{n_\mathrm{p}=2 \\ n_\mathrm{p}\in\mathbb{N}_\mathrm{p}^*}}^{(N_\mathrm{g}-1)} \left(\binom{N_\mathrm{g}-1}{n_\mathrm{p}} P_\mathrm{nb}^{(N_\mathrm{g}-n_\mathrm{p})} P_\mathrm{b}^{n_\mathrm{p}} \right),
\end{aligned} \tag{5.43}
$$

ou, com base na equação 4.7,

$$
P(E_{\mathrm{Cg}_3}) \simeq P_\mathrm{nb}^{N_\mathrm{g}} \times \sum_{\substack{n_\mathrm{p}=2 \\ n_\mathrm{p}\in\mathbb{N}_\mathrm{p}^*}}^{M} \left(\binom{M}{n_\mathrm{p}} \times P_\mathrm{nb}^{(N_\mathrm{g}-n_\mathrm{p})} \times P_\mathrm{b}^{n_\mathrm{p}} \right). \tag{5.44}
$$

Nas equações 5.43 e 5.44, $n_\mathrm{p} \in \mathbb{N}_\mathrm{p}^*$, com $\mathbb{N}_\mathrm{p}^*$ correspondendo ao conjunto dos números naturais pares não-nulos, ou seja, $\mathbb{N}_\mathrm{p}^* = \{2,4,6,8,\dots\}$, de maneira que n_p indica um número par de *bitflips* e $n_\mathrm{p} \leq M$. Se N_g for par, o valor de $n_\mathrm{p} = N_\mathrm{g}$ indica que todos de $Ch1$ sofreram *bitflips*, o que significa uma detecção ao invés de correção (análogo ao evento E_4 da Tabela 5.2).

Com fundamento nas equações 5.41, 5.42 e 5.44, a probabilidade de correção para um dado com M bits de informação e 1 bit de paridade (N_g bits) pode ser escrita como

$$
P(E_\mathrm{G}) \simeq P(E_{\mathrm{Cg}_1}) + P(E_{\mathrm{Cg}_2}) + P(E_{\mathrm{Cg}_3}).
$$

Logo,

$$
P(E_\mathrm{G}) \simeq P_\mathrm{nb}^{N_\mathrm{g}} \times \left(P_\mathrm{nb}^{N_\mathrm{g}} + 2 \times \sum_{\substack{n_\mathrm{i}=1 \\ n_\mathrm{i}\in\mathbb{N}_\mathrm{i}}}^{N_\mathrm{g}} \left(\binom{N_\mathrm{g}}{n_\mathrm{i}} P_\mathrm{nb}^{(N_\mathrm{g}-n_\mathrm{i})} P_b^{n_\mathrm{i}} \right) + \sum_{\substack{n_\mathrm{p}=2 \\ n_\mathrm{p}\in\mathbb{N}_\mathrm{p}^*}}^{M} \left(\binom{M}{n_\mathrm{p}} P_\mathrm{nb}^{(N_\mathrm{g}-n_\mathrm{p})} P_\mathrm{b}^{n_\mathrm{p}} \right) \right).
$$

Uma fórmula geral para $P(E_{\mathrm{G}})$ pode ser definida com base nos coeficientes que multiplicam as probabilidades. Sendo a_{n} o enésimo coeficiente multiplicador, a fórmula geral pode ser escrita como

$$P(E_{\mathrm{G}}) \simeq P_{\mathrm{nb}}^{N_{\mathrm{g}}} \times \sum_{n=0}^{N_{\mathrm{g}}} \left(a_{\mathrm{n}} P_{\mathrm{nb}}^{(N_{\mathrm{g}}-n)} P_{\mathrm{b}}^{n} \right), \tag{5.45}$$

em que

$$a_{\mathrm{n}} = \begin{cases} 0 & \text{, se } n = N_{\mathrm{g}} \text{ para } N_{\mathrm{g}} \text{ par} \\ 1 & \text{, se } n = 0 \\ 2 \times \binom{N_{\mathrm{g}}}{n} & \text{, se } n \in \mathbb{N}_{\mathrm{i}} \\ \binom{M}{n} & \text{, se } n \in \mathbb{N}_{\mathrm{p}}^{*}. \end{cases} \tag{5.46}$$

Destaca-se que o valor de n define a quantidade de *bitflips*.

As equações 5.45 e 5.46 podem ser utilizadas para determinar a probabilidade de correção em um dado de N_{g} bits. Isso significa que ela deve ser aplicada para um conjunto de bits que formam um único grupo. Contudo, a proposta apresentada para a PDMR se baseia na divisão de um dado de tamanho N em grupos compostos por uma paridade e M bits de informação, sendo a quantidade de grupos definidas por N_{p}. Desse modo, é necessária uma equação que possa ser aplicada ao conjunto inteiro de bits transmitidos usando a técnica proposta.

Nesse contexto, suponha que um dado de entrada de N bits é dividido em dois grupos com N_{g} bits cada, formando uma palavra-código de N_{d} bits. Para que ocorra uma correção, os casos 5.2.1, 5.2.2 e 5.2.3, devem ser verificados em ambos os grupos. Em outras palavras, a condição para a correção é que haja uma correção no primeiro grupo e uma correção no segundo grupo. Desse modo, a probabilidade de correção $P(C_{\mathrm{pdmrg}})$ é dada por

$$P(C_{\mathrm{pdmrg}}) = P(E_{\mathrm{G_1}} \cap E_{\mathrm{G_2}}) \simeq P(E_{\mathrm{G_1}}) \times P(E_{\mathrm{G_2}}),$$

em que $P(E_{\mathrm{G_1}})$ e $P(E_{\mathrm{G_2}})$ representam as probabilidades de correção no primeiro e no segundo grupos, respectivamente. Visto que os grupos possuem a mesma quantidade de bits de informação M e de paridade, $P(E_{\mathrm{G_1}}) = P(E_{\mathrm{G_2}}) = P(E_{\mathrm{G}})$. Portanto,

$$P(C_{\mathrm{pdmrg}}) \simeq P(E_{\mathrm{G}})^2. \tag{5.47}$$

O expoente de $P(E_{\mathrm{G}})$ na equação 5.47 corresponde à quantidade de grupos que formam a palavra-código. Sendo assim, de modo geral,

$$P(C_{\mathrm{pdmrg}}) \simeq P(E_{\mathrm{G}})^{N_{\mathrm{p}}} \tag{5.48}$$

ou

$$P(C_{\mathrm{pdmrg}}) \simeq \left(P_{\mathrm{nb}}^{N_{\mathrm{g}}} \times \sum_{n=0}^{N_{\mathrm{g}}} \left(a_{\mathrm{n}} P_{\mathrm{nb}}^{(N_{\mathrm{g}}-n)} P_{\mathrm{b}}^{n} \right) \right)^{N_{\mathrm{p}}}, \tag{5.49}$$

em que N_{g} é o número total de bits em um grupo; N_{p} é o número de total de bits de paridade, equivalente também ao número de grupos; P_{b} é a probabilidade de ocorrer um

bitflip; P_{nb} é a probabilidade de não ocorrer um *bitflip*; a_{n} é o coeficiente calculado pela equação 5.46, o qual está relacionado a cada probabilidade e define o número de *bitflips*.

5.2.2 Detecção

Para o cálculo de probabilidade de detecção para a PDMR-G, inicialmente, considere a Tabela 5.4, a qual contém os valores de e_{i}, representado pela expressão lógica 4.24, com base nos valores binários p_{i_0}, p_{i_1}, r_{i_0} e r_{i_1}. Além disso, há uma coluna que especifica as expressões booleanas para os casos em que $e_{\mathrm{i}} = 1$.

Tabela 5.4: Tabela-verdade e expressões booleanas para o bit de indicação de erro no grupo i (e_{i})

p_{i_0}	p_{i_1}	r_{i_0}	r_{i_1}	s_{i}	f_{i}	e_{i}	Expressões booleanas
0	0	0	0	0	1	0	-
0	0	0	1	0	0	0	-
0	0	1	0	1	0	0	-
0	0	1	1	1	1	1	$o_0 = \overline{p_{\mathrm{i}_0}} \cdot \overline{p_{\mathrm{i}_1}} \cdot r_{\mathrm{i}_0} \cdot r_{\mathrm{i}_1}$
0	1	0	0	0	0	0	-
0	1	0	1	1	1	1	$o_1 = \overline{p_{\mathrm{i}_0}} \cdot p_{\mathrm{i}_1} \cdot \overline{r_{\mathrm{i}_0}} \cdot r_{\mathrm{i}_1}$
0	1	1	0	1	1	1	$o_2 = \overline{p_{\mathrm{i}_0}} \cdot p_{\mathrm{i}_1} \cdot r_{\mathrm{i}_0} \cdot \overline{r_{\mathrm{i}_1}}$
0	1	1	1	1	0	0	-
1	0	0	0	1	0	0	-
1	0	0	1	1	1	1	$o_3 = p_{\mathrm{i}_0} \cdot \overline{p_{\mathrm{i}_1}} \cdot \overline{r_{\mathrm{i}_0}} \cdot r_{\mathrm{i}_1}$
1	0	1	0	1	1	1	$o_4 = p_{\mathrm{i}_0} \cdot \overline{p_{\mathrm{i}_1}} \cdot r_{\mathrm{i}_0} \cdot \overline{r_{\mathrm{i}_1}}$
1	0	1	1	0	0	0	-
1	1	0	0	1	1	1	$o_5 = p_{\mathrm{i}_0} \cdot p_{\mathrm{i}_1} \cdot \overline{r_{\mathrm{i}_0}} \cdot \overline{r_{\mathrm{i}_1}}$
1	1	0	1	1	0	0	-
1	1	1	0	0	0	0	-
1	1	1	1	0	1	0	-

Fonte: O próprio autor.

Com base na Tabela 5.4, observa-se que e_{i} pode ser escrito como

$$e_{\mathrm{i}} = o_0 + o_1 + o_2 + o_3 + o_4 + o_5.$$

Isso significa que, para que haja uma indicação de erro e, consequentemente, uma detecção, ao menos uma das saídas (o_0 ou o_1 ou o_2 ou o_3 ou o_4 ou o_5) deve ter valor lógico $0b1$. Em termos de probabilidade, tem-se:

$$P(E_{\mathrm{i}}) = P(O_0 \cup O_1 \cup O_2 \cup O_3 \cup O_4 \cup O_5),$$

em que O_0, O_1, O_2, O_3, O_4 e O_5 são os eventos associados à ocorrência das saídas o_0, o_1, o_2, o_3, o_4 e o_5, respectivamente, enquanto E_{i} corresponde ao evento associado à ocorrência de e_{i}, ou seja, $e_{\mathrm{i}} = 1$ para o grupo i. Porém, esses eventos são mutuamente excludentes, uma vez que não podem ocorrer ao mesmo tempo para um mesmo grupo i. Sendo assim,

sua interseção é vazia, o que resulta em

$$P(E_{\mathrm{i}}) = P(O_0) + P(O_1) + P(O_2) + P(O_3) + P(O_4) + P(O_5). \tag{5.50}$$

Ou seja, a probabilidade de ocorrer uma detecção para um grupo é dada pela soma das probabilidades de ocorrência de um dos eventos associados às saídas apresentadas na Tabela 5.4.

Como as saídas dependem dos valores binários dos bits de paridade p_{i_0}, p_{i_1}, r_{i_0} e r_{i_1} e de seus complementos, os eventos O_0, O_1, O_2, O_3, O_4 e O_5 estão relacionados com os eventos atribuídos a essas paridades. De maneira simplificada, tais eventos podem ser definidos considerando a possibilidade das paridades assumirem valor binário $0b0$ ou $0b1$. Neste texto, a possibilidade do valor assumido pelas paridades ser $0b1$ é tomada como base para a análise dos eventos e probabilidades. Nesse sentido, utilizando a definição 5.1, os eventos associados às paridades p_{i_0}, p_{i_1}, r_{i_0} e r_{i_1} e seus complementos são representados de acordo com a Tabela 5.5.

Tabela 5.5: Paridades e eventos associados para $p_{\mathrm{i}} = 0$

p_{i}	Paridade	Evento	Descrição
0	p_{i_0}	X_0	Evento de *bitflip* em p_{i_0}.
	p_{i_1}	X_1	Evento de *bitflip* em p_{i_1}.
	r_{i_0}	Z_0	Evento relacionado a um número ímpar de *bitflips* nos bits de informação usados no cálculo de r_{i_0}.
	r_{i_1}	Z_1	Evento relacionado a um número ímpar de *bitflips* nos bits de informação usados no cálculo de r_{i_1}.
	$\overline{p_{\mathrm{i}_0}}$	X_0^c	Evento de *não bitflip* em p_{i_0}.
	$\overline{p_{\mathrm{i}_1}}$	X_1^c	Evento de *não bitflip* em p_{i_1}.
	$\overline{r_{\mathrm{i}_0}}$	Z_0^c	Evento relacionado a um número par de *bitflips* nos bits de informação usados no cálculo de r_{i_0}.
	$\overline{r_{\mathrm{i}_1}}$	Z_1^c	Evento relacionado a um número par de *bitflips* nos bits de informação usados no cálculo de r_{i_1}.

Fonte: O próprio autor.

Na Tabela 5.5, os eventos X_0 e X_1 estão associados à ocorrência de *bitflips* nos bits p_{i_0} e p_{i_1}, respectivamente. Essa associação acontece pelo fato de que, dado que o valor original de p_{i} é $0b0$, p_{i_0} e p_{i_1} só assumem valor $0b1$ caso ocorra *bitflips* durante a transmissão. Analogamente, $\overline{p_{\mathrm{i}_0}}$ e $\overline{p_{\mathrm{i}_1}}$ têm valor binário $0b1$ caso não aconteçam *bitflips* (eventos X_0^c e X_1^c). No que se refere aos bits r_{i_0} e r_{i_1}, uma vez que eles dependem dos bits de informação nos canais, eles assumem valor binário $0b1$ quando são verificados um número ímpar de *bitflips* nos bits de informação do i-ésimo grupo em *Ch0* e em *Ch1* (eventos Z_0 e Z_1). Desse modo, seguindo o mesmo raciocínio, $\overline{r_{\mathrm{i}_0}}$ e $\overline{r_{\mathrm{i}_1}}$ têm valor $0b1$ quando o número de *bitflips* é par (eventos Z_0^c e Z_1^c).

Para os eventos associados às paridades p_{i_0}, p_{i_1}, r_{i_0} e r_{i_1} e seus complementos, com base na definição 5.2, são utilizadas as representações da Tabela 5.6. Nesse caso, como

o valor original de p_{i} é $0b1$, para que p_{i_0} e p_{i_1} sejam iguais a $0b1$ não devem ocorrer *bitflips*, de maneira que X_0' e X_1' simbolizam os eventos de *não bitflip* em p_{i_0} e p_{i_1}, respectivamente. Portanto, $X_0^{c\prime}$ e $X_1^{c\prime}$ são, respectivamente, os eventos de *bitflip* em $\overline{p_{\mathrm{i}_0}}$ e $\overline{p_{\mathrm{i}_1}}$. Por fim, os bits r_{i_0} e r_{i_1} têm valor $0b1$ quando há um número par de *bitflips* nos bits de informação do i-ésimo grupo das palavras-código em cada um dos canais (eventos Z_0' e Z_1'). Consequentemente, $\overline{r_{\mathrm{i}_0}}$ e $\overline{r_{\mathrm{i}_1}}$ assumem valor $0b1$ caso a quantidade de *bitflips* nos bits de informação seja ímpar ($Z_0^{c\prime}$ e $Z_1^{c\prime}$).

Tabela 5.6: Paridades e eventos associados para $p_{\mathrm{i}} = 1$

p_{i}	Paridade	Evento	Descrição
1	p_{i_0}	X_0'	Evento de *não bitflip* em p_{i_0}.
	p_{i_1}	X_1'	Evento de *não bitflip* em p_{i_1}.
	r_{i_0}	Z_0'	Evento relacionado a um número par de *bitflips* nos bits de informação usados no cálculo de r_{i_0}.
	r_{i_1}	Z_1'	Evento relacionado a um número par de *bitflips* nos bits de informação usados no cálculo de r_{i_1}.
	$\overline{p_{\mathrm{i}_0}}$	$X_0^{c\prime}$	Evento de *bitflip* em p_{i_0}
	$\overline{p_{\mathrm{i}_1}}$	$X_1^{c\prime}$	Evento de *bitflip* em p_{i_1}
	$\overline{r_{\mathrm{i}_0}}$	$Z_0^{c\prime}$	Evento relacionado a um número ímpar de *bitflips* nos bits de informação usados no cálculo de r_{i_0}.
	$\overline{r_{\mathrm{i}_1}}$	$Z_1^{c\prime}$	Evento relacionado a um número ímpar de *bitflips* nos bits de informação usados no cálculo de r_{i_1}.

Fonte: O próprio autor.

Inicialmente, são consideradas as representações da Tabela 5.5 para o cálculo das probabilidades dos eventos O_0, O_1, O_2, O_3, O_4 e O_5. Logo,

$$
\begin{aligned}
P(O_0) &= P(X_0^c \cap X_1^c \cap Z_0 \cap Z_1),\\
P(O_1) &= P(X_0^c \cap X_1 \cap Z_0^c \cap Z_1),\\
P(O_2) &= P(X_0^c \cap X_1 \cap Z_0 \cap Z_1^c),\\
P(O_3) &= P(X_0 \cap X_1^c \cap Z_0^c \cap Z_1),\\
P(O_4) &= P(X_0 \cap X_1^c \cap Z_0 \cap Z_1^c),\\
P(O_5) &= P(X_0 \cap X_1 \cap Z_0^c \cap Z_1^c).
\end{aligned}
$$

Contudo, de acordo com a descrição apresentada no início do capítulo, a taxa de erros nos canais é considerada como sendo igual, de modo que a probabilidade de ocorrência de *bitflip* é a mesma para qualquer bit transmitido. Portanto, os eventos na Tabela 5.5 são independentes e:

$$P(O_0) \simeq P(X_0^c) \times P(X_1^c) \times P(Z_0) \times P(Z_1), \tag{5.51}$$

$$P(O_1) \simeq P(X_0^c) \times P(X_1) \times P(Z_0^c) \times P(Z_1), \tag{5.52}$$

$$P(O_2) \simeq P(X_0^c) \times P(X_1) \times P(Z_0) \times P(Z_1^c), \tag{5.53}$$

$$P(O_3) \simeq P(X_0) \times P(X_1^c) \times P(Z_0^c) \times P(Z_1), \tag{5.54}$$
$$P(O_4) \simeq P(X_0) \times P(X_1^c) \times P(Z_0) \times P(Z_1^c), \tag{5.55}$$
$$P(O_5) \simeq P(X_0) \times P(X_1) \times P(Z_0^c) \times P(Z_1^c). \tag{5.56}$$

Além disso, analisando a Tabela 5.5, pode-se concluir que:

$$P(X_0^c) = 1 - P(X_0), \tag{5.57}$$
$$P(X_1^c) = 1 - P(X_1), \tag{5.58}$$
$$P(Z_0^c) = 1 - P(Z_0), \tag{5.59}$$
$$P(Z_1^c) = 1 - P(Z_1). \tag{5.60}$$

Assim como descrito na seção 5.1.1, as probabilidades de eventos de *bitflip* são dadas por P_b, sendo $P(X_0) = P(X_1) = P_\mathrm{b}$, enquanto as probabilidade de *não bitflip* são definidas por P_nb, de maneira que $P(X_0^c) = P(X_1^c) = P_\mathrm{nb}$. Por outro lado, as paridades calculadas no receptor são determinadas pela equação 4.11, o que significa que têm relação com a quantidade de bits de informação (M) em cada grupo. Por meio da utilização de P_b e P_nb nas equações 5.51 à 5.56, tem-se:

$$\begin{aligned}
P(O_0) &\simeq P_\mathrm{nb}^2 \times P(Z_0) \times P(Z_1),\\
P(O_1) &\simeq P_\mathrm{nb} \times P_\mathrm{b} \times P(Z_0^c) \times P(Z_1),\\
P(O_2) &\simeq P_\mathrm{nb} \times P_\mathrm{b} \times P(Z_0) \times P(Z_1^c),\\
P(O_3) &\simeq P_\mathrm{b} \times P_\mathrm{nb} \times P(Z_0^c) \times P(Z_1),\\
P(O_4) &\simeq P_\mathrm{b} \times P_\mathrm{nb} \times P(Z_0) \times P(Z_1^c),\\
P(O_5) &\simeq P_\mathrm{b}^2 \times P(Z_0^c) \times P(Z_1^c),
\end{aligned}$$

para o caso em que p_i tem valor binário $0b0$ antes da transmissão (Tabela 5.5).

Sendo assim, a equação 5.50 é dada por

$$\begin{aligned}
P(E_\mathrm{i}) \simeq P_\mathrm{nb}^2 \times P(Z_0) \times P(Z_1) + 2 \times P_\mathrm{nb} \times P_\mathrm{b} \times P(Z_0^c) \times P(Z_1) + \\
2 \times P_\mathrm{nb} \times P_\mathrm{b} \times P(Z_0) \times P(Z_1^c) + P_\mathrm{b}^2 \times P(Z_0^c) \times P(Z_1^c),
\end{aligned} \tag{5.61}$$

quando $p_\mathrm{i} = 0$.

No caso em que o valor é de p_i é $0b1$ antes da transmissão, são utilizados os eventos da Tabela 5.6, de maneira que as equações para as probabilidades dos eventos associados à ocorrência das saídas o_0 à o_5 são:

$$\begin{aligned}
P(O_0)' &= P(X_0^{c\prime} \cap X_1^{c\prime} \cap Z_0' \cap Z_1'),\\
P(O_1)' &= P(X_0^{c\prime} \cap X_1' \cap Z_0^{c\prime} \cap Z_1'),\\
P(O_2)' &= P(X_0^{c\prime} \cap X_1' \cap Z_0' \cap Z_1^{c\prime}),\\
P(O_3)' &= P(X_0' \cap X_1^{c\prime} \cap Z_0^{c\prime} \cap Z_1'),\\
P(O_4)' &= P(X_0' \cap X_1^{c\prime} \cap Z_0' \cap Z_1^{c\prime}),
\end{aligned}$$

$$P(O_5)' = P(X_0' \cap X_1' \cap Z_0^{c\prime} \cap Z_1^{c\prime}).$$

Dada a independência dos eventos da Tabela 5.6, tem-se:

$$P(O_0)' \simeq P(X_0^{c\prime}) \times P(X_1^{c\prime}) \times P(Z_0') \times P(Z_1'), \tag{5.62}$$
$$P(O_1)' \simeq P(X_0^{c\prime}) \times P(X_1') \times P(Z_0^{c\prime}) \times P(Z_1'), \tag{5.63}$$
$$P(O_2)' \simeq P(X_0^{c\prime}) \times P(X_1') \times P(Z_0') \times P(Z_1^{c\prime}), \tag{5.64}$$
$$P(O_3)' \simeq P(X_0') \times P(X_1^{c\prime}) \times P(Z_0^{c\prime}) \times P(Z_1'), \tag{5.65}$$
$$P(O_4)' \simeq P(X_0') \times P(X_1^{c\prime}) \times P(Z_0') \times P(Z_1^{c\prime}), \tag{5.66}$$
$$P(O_5)' \simeq P(X_0') \times P(X_1') \times P(Z_0^{c\prime}) \times P(Z_1^{c\prime}). \tag{5.67}$$

Destaca-se que:

$$P(X_0^{c\prime}) = 1 - P(X_0'), \tag{5.68}$$
$$P(X_1^{c\prime}) = 1 - P(X_1'), \tag{5.69}$$
$$P(Z_0^{c\prime}) = 1 - P(Z_0'), \tag{5.70}$$
$$P(Z_1^{c\prime}) = 1 - P(Z_1'). \tag{5.71}$$

Ademais, de acordo com as descrições da Tabela 5.6, $P(X_0') = P(X_1') = P_{\mathrm{nb}}$ e $P(X_0^{c\prime}) = P(X_1^{c\prime}) = P_{\mathrm{b}}$. Substituindo as respectivas probabilidades por P_{b} e P_{nb} nas equações 5.62 à 5.67, obtém-se:

$$\begin{aligned}
P(O_0)' &\simeq P_{\mathrm{b}}^2 \times P(Z_0') \times P(Z_1'),\\
P(O_1)' &\simeq P_{\mathrm{b}} \times P_{\mathrm{nb}} \times P(Z_0^{c\prime}) \times P(Z_1'),\\
P(O_2)' &\simeq P_{\mathrm{b}} \times P_{\mathrm{nb}} \times P(Z_0') \times P(Z_1^{c\prime}),\\
P(O_3)' &\simeq P_{\mathrm{nb}} \times P_{\mathrm{b}} \times P(Z_0^{c\prime}) \times P(Z_1'),\\
P(O_4)' &\simeq P_{\mathrm{nb}} \times P_{\mathrm{b}} \times P(Z_0') \times P(Z_1^{c\prime}),\\
P(O_5)' &\simeq P_{\mathrm{nb}}^2 \times P(Z_0^{c\prime}) \times P(Z_1^{c\prime}).
\end{aligned}$$

Portanto, com base na equação 5.50, quando $p_{\mathrm{i}} = 1$, pode-se escrever

$$\begin{aligned}
P(E_\eta) \simeq P_{\mathrm{b}}^2 \times P(Z_0') \times P(Z_1') + 2 \times P_{\mathrm{b}} \times P_{\mathrm{nb}} \times P(Z_0^{c\prime}) \times P(Z_1') + \\
2 \times P_{\mathrm{b}} \times P_{\mathrm{nb}} \times P(Z_0') \times P(Z_1^{c\prime}) + P_{\mathrm{nb}}^2 \times P(Z_0^{c\prime}) \times P(Z_1^{c\prime}),
\end{aligned} \tag{5.72}$$

em que o índice η equivale a i, de modo que $0 \leq \eta \leq G$. Esse índice é utilizado para distinguir as equações 5.61 e 5.72. Essas equações evidenciam a necessidade de determinar as probabilidades das paridades calculadas no receptor assumirem ou não o valor binário $0b1$. Nesse contexto, considere um único grupo com dado $0b00$ ($M = 2$), de modo que:

$$\begin{aligned}
r_{0_0} &= b_{0_0} \oplus b_{1_0} = b_{0_0} \cdot \overline{b_{1_0}} + \overline{b_{0_0}} \cdot b_{1_0},\\
r_{0_1} &= b_{0_1} \oplus b_{1_1} = b_{0_1} \cdot \overline{b_{1_1}} + \overline{b_{0_1}} \cdot b_{1_1}.
\end{aligned}$$

Considerando que os eventos de *bitflip* em b_{0_0}, b_{1_0}, b_{0_1} e b_{1_1} são representados por Y_0, W_0, Y_1 e W_1, respectivamente, e que os eventos de *não bitflip*, equivalente aos complementares $\overline{b_{0_0}}$, $\overline{b_{1_0}}$, $\overline{b_{0_1}}$ e $\overline{b_{1_1}}$, são representados, respectivamente, por Y_0^c, W_0^c, Y_1^c e W_1^c, tem-se:

$$P(Z_0) = P((Y_0 \cap W_0^c) \cup (Y_0^c \cap W_0)),$$
$$P(Z_1) = P((Y_1 \cap W_1^c) \cup (Y_1^c \cap W_0)),$$

ou

$$P(Z_0) = P((Y_0 \cap W_0^c)) + P((Y_0^c \cap W_0)) - P((Y_0 \cap W_0^c) \cap (Y_0^c \cap W_0)),$$
$$P(Z_1) = P((Y_1 \cap W_1^c)) + P((Y_1^c \cap W_1)) - P((Y_1 \cap W_1^c) \cap (Y_1^c \cap W_1)).$$

Contudo, os eventos associados à $(Y_0 \cap W_0^c)$ e à $(Y_0^c \cap W_0)$, bem como os relacionados à $(Y_1 \cap W_1^c)$ e à $(Y_1 \cap W_1^c)$, são mutuamente excludentes. Logo,

$$P(Z_0) = P((Y_0 \cap W_0^c)) + P((Y_0^c \cap W_0)),$$
$$P(Z_1) = P((Y_1 \cap W_1^c)) + P((Y_1^c \cap W_1)).$$

Nesse caso, para canais sob a mesma taxa de erros e probabilidade de *bitflips* iguais para cada bit transmitido, sendo $P_\mathrm{b} = P(Y_0) = P(Y_1) = P(W_0) = P(W_1)$ e $P_\mathrm{nb} = P(Y_0^c) = P(Y_1^c) = P(W_0^c) = P(W_1^c)$, tem-se:

$$\begin{aligned} P(Z_0) &\simeq P_\mathrm{b} \times P_\mathrm{nb} + P_\mathrm{nb} \times P_\mathrm{b} \simeq 2 \times P_\mathrm{b} \times P_\mathrm{nb}, \\ &\simeq 2 \times P_\mathrm{b} \times (1 - P_\mathrm{b}), \end{aligned}$$

$$P(Z_0) \simeq 2 \times P_\mathrm{b} - 2 \times P_\mathrm{b}^2, \tag{5.73}$$

e

$$\begin{aligned} P(Z_1) &\simeq P_\mathrm{b} \times P_\mathrm{nb} + P_\mathrm{nb} \times P_\mathrm{b} \simeq 2 \times P_\mathrm{b} \times P_\mathrm{nb} \\ &\simeq 2 \times P_\mathrm{b} \times (1 - P_\mathrm{b}), \end{aligned}$$

$$P(Z_1) \simeq 2 \times P_\mathrm{b} - 2 \times P_\mathrm{b}^2. \tag{5.74}$$

De acordo com as equações 5.73 e 5.74, $P(Z_0) = P(Z_1)$ e, consequentemente, $P(Z_0^c) = P(Z_1^c)$. Ademais, vale salientar que o dado $0b00$ foi utilizado como referência para simplificar o raciocínio. Porém, pode-se demonstrar que para o dado $0b11$ a probabilidade é dada pela mesma equação. Contudo, os cálculos para os dados $0b01$ e $0b10$ são:

$$\begin{aligned} P(Z_0') &\simeq P_\mathrm{b} \times P_\mathrm{b} + P_\mathrm{nb} \times P_\mathrm{nb} \simeq P_\mathrm{b}^2 + P_\mathrm{nb}^2, \\ &\simeq P_\mathrm{b}^2 + (1 - P_\mathrm{b})^2, \\ &\simeq 1 + 2 \times P_\mathrm{b}^2 - 2 \times P_\mathrm{b}, \end{aligned} \tag{5.75}$$

e

$$P(Z_1') \simeq P_\mathrm{b} \times P_\mathrm{nb} + P_\mathrm{nb} \times P_\mathrm{b} \simeq P_\mathrm{b}^2 + P_\mathrm{nb}^2,$$

$$\simeq P_{\mathrm{b}}^2 + (1-P_{\mathrm{b}})^2,$$
$$\simeq 1 + 2 \times P_{\mathrm{b}}^2 - 2 \times P_{\mathrm{b}}. \tag{5.76}$$

Observa-se que $P(Z_0') = P(Z_1')$.

Diante das equações apresentadas, $P(E_{\mathrm{i}})$, para transmissão dos dados $0b00$ e $0b11$, pode ser calculada pela substituição das equações 5.73, 5.74 e dos resultados gerados pelas equações 5.59 e 5.60 em 5.61. Por outro lado, para a transmissão dos dados $0b01$ e $0b10$, o cálculo envolve a substituição das equações 5.75, 5.76 e dos resultados gerados pelas equações 5.70 e 5.71 em 5.72. Ao efetuar essas substituições, pode-se demonstrar que

$$P(E_{\mathrm{i}}) = P(E_{\eta}). \tag{5.77}$$

Apesar desse resultado ter sido obtido para a transmissão de dois bits de dados ($M = 2$), ele vale para qualquer valor de M. O que significa que independente do valor binário original dos bits de informação do grupo, pode-se obter $P(E_{\mathrm{i}})$ pelas equações 5.61 ou 5.72. Porém, por simplicidade, ao longo do texto, será considerada a equação 5.61.

Para exemplificar a validade da equação 5.77 para outros valores de M, considere a transmissão de um dado com $M = 3$. Seguindo o mesmo raciocínio, os valores de paridade calculada no receptor para a transmissão do dado, podem ser obtidas por:

$$r_{0_0} = b_{0_0} \oplus b_{1_0} \oplus b_{2_0},$$
$$r_{0_1} = b_{0_1} \oplus b_{1_1} \oplus b_{2_1}.$$

O desenvolvimento dessas equações implica nos seguintes resultados:

$$r_{0_0} = b_{0_0} \cdot b_{1_0} \cdot b_{2_0} + b_{0_0} \cdot \overline{b_{1_0}} \cdot \overline{b_{2_0}} + \overline{b_{0_0}} \cdot b_{1_0} \cdot \overline{b_{2_0}} + \overline{b_{0_0}} \cdot \overline{b_{1_0}} \cdot b_{2_0}, \tag{5.78}$$
$$r_{0_1} = b_{0_1} \cdot b_{1_1} \cdot b_{2_1} + b_{0_1} \cdot \overline{b_{1_1}} \cdot \overline{b_{2_1}} + \overline{b_{0_1}} \cdot b_{1_1} \cdot \overline{b_{2_1}} + \overline{b_{0_1}} \cdot \overline{b_{1_1}} \cdot b_{2_1}. \tag{5.79}$$

Com base nas equações 5.78 e 5.79, as probabilidades para os diferentes valores possíveis para os bits de informação podem ser obtidas. Essa probabilidades são apresentadas na Tabela 5.7, em que a coluna p_{i} contém os valores da paridade antes da transmissão e a última coluna referencia as equações que devem ser usadas para o cálculo de $P(E_{\mathrm{i}})$ ou $P(E_{\eta})$.

Por meio da Tabela 5.7, percebe-se que

$$P(Z_0)' = 1 - P(Z_0).$$

Além disso, a substituição das equações na respectiva equação de referência comprova que

$$P(E_{\mathrm{i}}) = P(E_{\eta}).$$

Fórmula Geral

Com fundamento nas definições e análises anteriores, verifica-se que o cálculo da probabilidade $P(E_{\mathrm{i}})$ tem como parte essencial a obtenção das probabilidades $P(Z_0)$ e $P(Z_1)$.

Tabela 5.7: Probabilidades de Z_0 e Z_1 para $M = 3$

p_i	Dado			Probabilidades	Equação de Ref.
0	0	0	0	$P(Z_0) = P(Z_1) = 4 \times P_{\mathrm{b}}^3 - 6 \times P_{\mathrm{b}}^2 + 3 \times P_{\mathrm{b}}$	5.61
1	0	0	1	$P(Z_0)' = P(Z_1)' = 1 - 4 \times P_{\mathrm{b}}^3 + 6 \times P_{\mathrm{b}}^2 - 3 \times P_{\mathrm{b}}$	5.72
1	0	1	0	$P(Z_0)' = P(Z_1)' = 1 - 4 \times P_{\mathrm{b}}^3 + 6 \times P_{\mathrm{b}}^2 - 3 \times P_{\mathrm{b}}$	5.72
0	0	1	1	$P(Z_0) = P(Z_1) = 4 \times P_{\mathrm{b}}^3 - 6 \times P_{\mathrm{b}}^2 + 3 \times P_{\mathrm{b}}$	5.61
1	1	0	0	$P(Z_0)' = P(Z_1)' = 1 - 4 \times P_{\mathrm{b}}^3 + 6 \times P_{\mathrm{b}}^2 - 3 \times P_{\mathrm{b}}$	5.72
0	1	0	1	$P(Z_0) = P(Z_1) = 4 \times P_{\mathrm{b}}^3 - 6 \times P_{\mathrm{b}}^2 + 3 \times P_{\mathrm{b}}$	5.61
0	1	1	0	$P(Z_0) = P(Z_1) = 4 \times P_{\mathrm{b}}^3 - 6 \times P_{\mathrm{b}}^2 + 3 \times P_{\mathrm{b}}$	5.61
1	1	1	1	$P(Z_0)' = P(Z_1)' = 1 - 4 \times P_{\mathrm{b}}^3 + 6 \times P_{\mathrm{b}}^2 - 3 \times P_{\mathrm{b}}$	5.72

Fonte: O próprio autor.

Os exemplos para um único grupo com $M = 2$ e $M = 3$ tornam evidente que essas probabilidades dependem da quantidade de bits de informação no grupo.

Definição 5.2.1. Considerando, por simplicidade, o valor binário nulo (i.e., dado com M bits iguais a zero), de maneira geral, r_{i_0} e, consequentemente, r_{i_1} assumem valor binário $0b1$ quando a quantidade de *bitflips* nos bits de informação, checados após a transmissão, é ímpar.

Dadas as equações 5.15 e 5.16, com $n_{\mathrm{z}} = n_{\mathrm{i}}$ e $n_{\mathrm{u}} = n_{\mathrm{i}}$, em que $n_{\mathrm{i}} \in \mathbb{N}_{\mathrm{i}}$, ou seja, é um número ímpar de *bitflips*, e sabendo que a equação 5.77 é válida para qualquer valor de M, a definição 5.2.1 implica em:

$$
\begin{aligned}
C_0(n_i) &= \binom{N_{\mathrm{a}}}{n_{\mathrm{i}}}, \\
C_1(n_{\mathrm{i}}) &= \binom{N_{\mathrm{b}}}{n_{\mathrm{i}}}, \\
P(B_{\mathrm{r}} = n_{\mathrm{i}}) &\simeq P_{\mathrm{nb}}^{(M-n_{\mathrm{i}})} P_{\mathrm{b}}^{n_{\mathrm{i}}},
\end{aligned}
\tag{5.80}
$$

em que $P(B_{\mathrm{r}} = n_{\mathrm{i}})$ é a probabilidade de ocorrer n_{i} *bitflips* nos bits de informação. Importante notar que a equação 5.80 pode ser utilizada tanto para r_{i_0} quanto para r_{i_1}. Além disso, sendo $N_{\mathrm{a}} = N_{\mathrm{b}} = M$, tem-se

$$
C_{\mathrm{g}}(n_{\mathrm{i}}) = C_0(n_{\mathrm{i}}) = C_1(n_{\mathrm{i}}) = \binom{M}{n_{\mathrm{i}}}. \tag{5.81}
$$

Logo, uma vez $P(Z_0) = P(Z_1)$ e usando P_{r} para representar $P(Z_0)$ e $P(Z_1)$, a probabilidade

é calculada por:

$$P_{\mathrm{r}} \simeq \sum_{\substack{n_{\mathrm{i}}=1 \\ n_{\mathrm{i}}\in\mathbb{N}_{\mathrm{i}}}}^{M} \left(C_{\mathrm{g}}(n_{\mathrm{i}}) \times P(B_{\mathrm{r}} = n_{\mathrm{i}})\right),$$

$$P_{\mathrm{r}} \simeq \sum_{\substack{n_{\mathrm{i}}=1 \\ n_{\mathrm{i}}\in\mathbb{N}_{\mathrm{i}}}}^{M} \left(\binom{M}{n_{\mathrm{i}}} P_{\mathrm{nb}}^{(M-n_{\mathrm{i}})} P_{\mathrm{b}}^{n_{\mathrm{i}}}\right). \tag{5.82}$$

Uma vez que $P(Z_0^c) = P(Z_1^c)$ e usando P_{nr} para simbolizar $P(Z_0^c)$ e $P(Z_1^c)$, a equação 5.61 pode então ser definida como:

$$P(E_{\mathrm{i}}) \simeq P_{\mathrm{nb}}^2 \times P_{\mathrm{r}}^2 + 4 \times P_{\mathrm{nb}} \times P_{\mathrm{b}} \times P_{\mathrm{nr}} \times P_{\mathrm{r}} + P_{\mathrm{b}}^2 \times P_{\mathrm{nr}}^2. \tag{5.83}$$

Importante notar que

$$P_{\mathrm{nr}} = 1 - P_{\mathrm{r}}. \tag{5.84}$$

A equação 5.83 pode ser usada como fórmula geral para o cálculo da probabilidade de detecção em cada grupo da palavra-código. Todavia, a probabilidade de detecção para a PDMR-G na transmissão de um dado de N bits dividido em N_{p} grupos envolve uma análise adicional. Nesse sentido, para que a detecção aconteça, basta que haja detecção em pelo menos um dos grupos. Ou seja,

$$P(D_{\mathrm{pdmrg}}) = P(E_0 \cup E_1 \cup \cdots \cup E_{(\mathrm{G}-1)} \cup E_{\mathrm{G}}), \tag{5.85}$$

em que $P(D_{\mathrm{pdmrg}})$ é a probabilidade de detecção na transmissão de um dado de N bits para a PDMR-G.

Os eventos E_0, E_1, $\cdots$, $E_{(\mathrm{G}-1)}$, E_{G} não são disjuntos, pois podem ocorrer simultaneamente e o cálculo da equação 5.85 se baseia no Princípio da Inclusão-Exclusão. Desse modo:

$$\begin{aligned} P(D_{\mathrm{pdmrg}}) = \sum_{i=0}^{G} P(E_{\mathrm{i}}) - \sum_{i=0}^{G}\left(\sum_{\alpha=1}^{G} P(E_{\mathrm{i}} \cap E_\alpha)\right) + \sum_{i=0}^{G}\left(\sum_{\alpha=1}^{G}\left(\sum_{\beta=2}^{G} P(E_{\mathrm{i}} \cap E_\alpha \cap E_\beta)\right)\right) \\ - \sum_{i=0}^{G}\left(\sum_{\alpha=1}^{G}\left(\sum_{\beta=2}^{G}\left(\sum_{\gamma=3}^{G} P(E_{\mathrm{i}} \cap E_\alpha \cap E_\beta \cap E_\gamma)\right)\right)\right) + \cdots \\ + (-1)^G (P(E_0 \cap E_1 \cap E_2 \cap \cdots \cap E_{(\mathrm{G}-1)} \cap E_{\mathrm{G}}). \end{aligned}$$

Contudo, como os eventos de detecção em grupos diferentes são independentes:

$$\begin{aligned} P(D_{\mathrm{pdmrg}}) \simeq \sum_{i=0}^{G} P(E_{\mathrm{i}}) - \sum_{i=0}^{G}\left(\sum_{\alpha=1}^{G} (P(E_{\mathrm{i}}) \times P(E_\alpha))\right) \\ + \sum_{i=0}^{G}\left(\sum_{\alpha=1}^{G}\left(\sum_{\beta=2}^{G} (P(E_{\mathrm{i}}) \times P(E_\alpha) \times P(E_\beta))\right)\right) \end{aligned}$$

$$
\begin{aligned}
&- \sum_{i=0}^{G} \left(\sum_{\alpha=1}^{G} \left(\sum_{\beta=2}^{G} \left(\sum_{\gamma=3}^{G} (P(E_{\mathrm{i}}) \times P(E_{\alpha}) \times P(E_{\beta}) \times P(E_{\gamma})) \right) \right) \right) \\
&+ \cdots \\
&+ (-1)^{G} (P(E_0) \times P(E_1) \times P(E_2) \times \cdots \times P(E_{(\mathrm{G}-1)}) \times P(E_{\mathrm{G}}))
\end{aligned} \tag{5.86}
$$

Verifica-se que os termos nos somatórios à direita na equação 5.86 correspondem à N_{p} elementos agrupados 1 a 1, 2 a 2, 3 a 3, 4 a 4, $\cdots$, N_{p} a N_{p}. Além disso, uma vez que $P(E_{\mathrm{i}})$ para qualquer valor de i é dado pela equação 5.83, a sua utilização na equação 5.86 resulta em

$$
\begin{aligned}
P(D_{\mathrm{pdmrg}}) \simeq \binom{N_{\mathrm{p}}}{1} P(E_{\mathrm{i}}) &- \binom{N_{\mathrm{p}}}{2} (P(E_{\mathrm{i}}) \times P(E_{\mathrm{i}})) + \binom{N_{\mathrm{p}}}{3} (P(E_{\mathrm{i}}) \times P(E_{\mathrm{i}}) \times P(E_{\mathrm{i}})) \\
&- \binom{N_{\mathrm{p}}}{4} (P(E_{\mathrm{i}}) \times P(E_{\mathrm{i}}) \times P(E_{\mathrm{i}}) \times P(E_{\mathrm{i}})) + \cdots \\
&+ (-1)^{G} \binom{N_{\mathrm{p}}}{N_{\mathrm{p}}} (P(E_{\mathrm{i}}) \times P(E_{\mathrm{i}}) \times P(E_{\mathrm{i}}) \times \cdots \times P(E_{\mathrm{i}}) \times P(E_{\mathrm{i}})).
\end{aligned}
$$

Logo,

$$
\begin{aligned}
P(D_{\mathrm{pdmrg}}) \simeq \binom{N_{\mathrm{p}}}{1} P(E_{\mathrm{i}}) - \binom{N_{\mathrm{p}}}{2} P(E_{\mathrm{i}})^2 + \binom{N_{\mathrm{p}}}{3} P(E_{\mathrm{i}})^3 - \binom{N_{\mathrm{p}}}{4} P(E_{\mathrm{i}})^4 \\
+ \cdots + (-1)^{G} \binom{N_{\mathrm{p}}}{N_{\mathrm{p}}} P(E_{\mathrm{i}})^{N_{\mathrm{p}}}
\end{aligned} \tag{5.87}
$$

A equação 5.87 pode ser escrita de forma simplificada, sendo

$$
P(D_{\mathrm{pdmrg}}) \simeq \sum_{n=1}^{N_{\mathrm{p}}} \left((-1)^{(n-1)} \times \binom{N_{\mathrm{p}}}{n} \times P(E_{\mathrm{i}})^n \right). \tag{5.88}
$$

Essa equação é a fórmula geral para a probabilidade de detecção da PDMR-G.

5.2.3 Erro

A técnica PDMR-G gera dados que podem ser classificados de três maneiras distintas, sendo correção, detecção e erro. Desse modo, uma vez calculadas as probabilidades de correção e de detecção, a probabilidade de erro pode ser encontrada por

$$
P(E_{\mathrm{pdmrg}}) \simeq 1 - P(C_{\mathrm{pdmrg}}) - P(D_{\mathrm{pdmrg}}). \tag{5.89}
$$

5.3 Cálculos de Probabilidade para PDMR-C

A versão PDMR-C expande a capacidade de detecção da proposta PDMR-G. Desse modo, as três possibilidades para o dado de saída são correção, detecção e erro. A seguir, são apresentados os desenvolvimentos matemáticos para obtenção das probabilidades relacionadas às três possibilidades citadas.

5.3.1 Correção

A lógica de comparação na PDMR tem impacto na detecção de possíveis erros nos canais. Desse modo, os eventos de correção são em grande parte os mesmos apresentados na seção 5.2.1. As exceções são os eventos em que o número de *bitflips* é par (caso 5.2.3), pois nesses casos a lógica de comparação gera um detecção. Isso acontece pelo fato de que, apesar do dado estar correto em *Ch0*, a comparação com o dado de *Ch1* indica uma diferença nos valores binários de cada canal.

Sendo assim, a probabilidade de correção para a PDMR-C é dada por

$$P(C_{\text{pdmrc}}) \simeq \left(P_{\text{nb}}^{N_\text{g}} \times \sum_{n=0}^{N_\text{g}} \left(a'_\text{n} P_{\text{nb}}^{(N_\text{g}-n)} P_\text{b}^{n} \right) \right)^{N_\text{p}}, \tag{5.90}$$

em que

$$a'_\text{n} = \begin{cases} 0 & \text{, se } n = N_\text{g} \text{ para } N_\text{g} \text{ par} \\ 1 & \text{, se } n = 0 \\ 2 \times \binom{N_\text{g}}{n} & \text{, se } n \in \{1,3,5,7,\dots\} \\ 0 & \text{, se } n \in \{2,4,6,8,\dots\}. \end{cases} \tag{5.91}$$

5.3.2 Detecção

O desenvolvimento matemático para o cálculo da probabilidade de detecção para a técnica PDMR-C tem como fundamento a expressão lógica 4.28, a qual pode ser escrita como:

$$\begin{aligned} e'_\text{i} &= f_\text{i} \cdot (s_\text{i} + c_\text{i}), \\ e'_\text{i} &= f_\text{i} \cdot s_\text{i} + f_\text{i} \cdot c_\text{i}, \\ e'_\text{i} &= e_\text{i} + f_\text{i} \cdot c_\text{i}. \end{aligned} \tag{5.92}$$

Além disso, por meio da Tabela 5.4, verifica-se que f_i pode ser expresso por

$$f_\text{i} = e_\text{i} + \overline{p_{\text{i}_0}} \cdot \overline{p_{\text{i}_1}} \cdot \overline{r_{\text{i}_0}} \cdot \overline{r_{\text{i}_1}} + p_{\text{i}_0} \cdot p_{\text{i}_1} \cdot r_{\text{i}_0} \cdot r_{\text{i}_1}. \tag{5.93}$$

Usando a expressão 5.93 em 5.92, pode-se escrever

$$\begin{aligned} e'_\text{i} &= e_\text{i} + (e_\text{i} + \overline{p_{\text{i}_0}} \cdot \overline{p_{\text{i}_1}} \cdot \overline{r_{\text{i}_0}} \cdot \overline{r_{\text{i}_1}} + p_{\text{i}_0} \cdot p_{\text{i}_1} \cdot r_{\text{i}_0} \cdot r_{\text{i}_1}) \cdot c_\text{i}, \\ e'_\text{i} &= e_\text{i} + e_\text{i} \cdot c_\text{i} + \overline{p_{\text{i}_0}} \cdot \overline{p_{\text{i}_1}} \cdot \overline{r_{\text{i}_0}} \cdot \overline{r_{\text{i}_1}} \cdot c_\text{i} + p_{\text{i}_0} \cdot p_{\text{i}_1} \cdot r_{\text{i}_0} \cdot r_{\text{i}_1} \cdot c_\text{i}. \end{aligned}$$

Fazendo

$$\theta_{\mathrm{i}} = \overline{p_{\mathrm{i}_0}} \cdot \overline{p_{\mathrm{i}_1}} \cdot \overline{r_{\mathrm{i}_0}} \cdot \overline{r_{\mathrm{i}_1}} \cdot c_{\mathrm{i}} \tag{5.94}$$

e

$$\phi_{\mathrm{i}} = p_{\mathrm{i}_0} \cdot p_{\mathrm{i}_1} \cdot r_{\mathrm{i}_0} \cdot r_{\mathrm{i}_1} \cdot c_{\mathrm{i}}, \tag{5.95}$$

tem-se:

$$e'_{\mathrm{i}} = e_{\mathrm{i}} + e_{\mathrm{i}} \cdot c_{\mathrm{i}} + \theta_{\mathrm{i}} + \phi_{\mathrm{i}}.$$

Assumindo, inicialmente, que o dado transmitido contém uma número par de 1s, o que significa que o valor binário de p_{i} é originalmente $0b0$, a probabilidade pode, então, ser determinada por

$$P(E'_{\mathrm{i}}) = P(E_{\mathrm{i}} \cup (E_{\mathrm{i}} \cap C_{\mathrm{i}}) \cup \Theta_{\mathrm{i}} \cup \Phi_{\mathrm{i}}), \tag{5.96}$$

em que C_{i} simboliza o evento caracterizado por $c_{\mathrm{i}} = 1$. Aplicando o Princípio da Inclusão-Exclusão, tem-se:

$$\begin{aligned} P(E'_{\mathrm{i}}) = P(E_{\mathrm{i}}) + P(E_{\mathrm{i}} \cap C_{\mathrm{i}}) + P(\Theta_{\mathrm{i}}) + P(\Phi_{\mathrm{i}}) \\ - P(E_{\mathrm{i}} \cap (E_{\mathrm{i}} \cap C_{\mathrm{i}})) - P(E_{\mathrm{i}} \cap \Theta_{\mathrm{i}}) - P(E_{\mathrm{i}} \cap \Phi_{\mathrm{i}}) \\ - P((E_{\mathrm{i}} \cap C_{\mathrm{i}}) \cap \Theta_{\mathrm{i}}) - P((E_{\mathrm{i}} \cap C_{\mathrm{i}}) \cap \Phi_{\mathrm{i}}) - P(\Theta_{\mathrm{i}} \cap \Phi_{\mathrm{i}}) \\ + P(E_{\mathrm{i}} \cap (E_{\mathrm{i}} \cap C_{\mathrm{i}}) \cap \Theta_{\mathrm{i}}) + P(E_{\mathrm{i}} \cap (E_{\mathrm{i}} \cap C_{\mathrm{i}}) \cap \Phi_{\mathrm{i}}) \\ + P(E_{\mathrm{i}} \cap \Theta_{\mathrm{i}} \cap \Phi_{\mathrm{i}}) + P((E_{\mathrm{i}} \cap C_{\mathrm{i}}) \cap \Theta_{\mathrm{i}} \cap \Phi_{\mathrm{i}}) \\ - P(E_{\mathrm{i}} \cap (E_{\mathrm{i}} \cap C_{\mathrm{i}}) \cap \Theta_{\mathrm{i}} \cap \Phi_{\mathrm{i}}). \end{aligned} \tag{5.97}$$

Pode-se verificar, com auxílio da Tabela 5.4, que os eventos representados por E_{i}, Θ_{i} e Φ_{i} são mutuamente excludentes, de maneira que:

$$P(E'_{\mathrm{i}}) = P(E_{\mathrm{i}}) + P(E_{\mathrm{i}} \cap C_{\mathrm{i}}) + P(\Theta_{\mathrm{i}}) + P(\Phi_{\mathrm{i}}) - P(E_{\mathrm{i}} \cap (E_{\mathrm{i}} \cap C_{\mathrm{i}})). \tag{5.98}$$

Como a interseção de E_{i} com E_{i} é o próprio E_{i},

$$P(E'_{\mathrm{i}}) = P(E_{\mathrm{i}}) + P(E_{\mathrm{i}} \cap C_{\mathrm{i}}) + P(\Theta_{\mathrm{i}}) + P(\Phi_{\mathrm{i}}) - P(E_{\mathrm{i}} \cap C_{\mathrm{i}}). \tag{5.99}$$

Portanto,

$$P(E'_{\mathrm{i}}) = P(E_{\mathrm{i}}) + P(\Theta_{\mathrm{i}}) + P(\Phi_{\mathrm{i}}). \tag{5.100}$$

As probabilidades relacionadas aos eventos Θ_{i} e Φ_{i} podem ser obtidas das expressões 5.94 e 5.95. Para isso, uma vez que p_{i} é originalmente $0b0$, são considerados os eventos da Tabela 5.5. Portanto,

$$P(\Theta_{\mathrm{i}}) = P(X_0^c \cap X_1^c \cap Z_0^c \cap Z_1^c \cap C_{\mathrm{i}}), \tag{5.101}$$

$$P(\Phi_{\mathrm{i}}) = P(X_0 \cap X_1 \cap Z_0 \cap Z_1 \cap C_{\mathrm{i}}). \tag{5.102}$$

Contudo, os eventos X_0^c e X_1^c são independentes dos eventos Z_0, Z_1 e C_{i}, pois a ocorrência

dos eventos associados à $\overline{p_{\mathrm{i}_0}}$ e $\overline{p_{\mathrm{i}_1}}$ não interfere na probabilidade dos eventos relacionados com $\overline{r_{\mathrm{i}_0}}$ e $\overline{r_{\mathrm{i}_1}}$, bem como com c_{i}. Essa independência deriva do fato de que a taxa de erros nos canais é considerada como sendo a mesma, de modo que a probabilidade de *bitflips* é igual para qualquer bit transmitido. Analogamente, os eventos X_0 e X_1 são independentes dos eventos Z_0, Z_1 e C_{i}. Desse modo,

$$P(\Theta_{\mathrm{i}}) \simeq P(X_0^c) \times P(X_1^c) \times P(Z_0^c \cap Z_1^c \cap C_{\mathrm{i}}), \tag{5.103}$$

$$P(\Phi_{\mathrm{i}}) \simeq P(X_0) \times P(X_1) \times P(Z_0 \cap Z_1 \cap C_{\mathrm{i}}). \tag{5.104}$$

Como discutido anteriormente, $P(X_0^c) = P(X_1^c) = P_{\mathrm{nb}}$ e $P(X_0) = P(X_1) = P_{\mathrm{b}}$. Consequentemente,

$$P(\Theta_{\mathrm{i}}) \simeq P_{\mathrm{nb}}^2 \times P(Z_0^c \cap Z_1^c \cap C_{\mathrm{i}}), \tag{5.105}$$

$$P(\Phi_{\mathrm{i}}) \simeq P_{\mathrm{b}}^2 \times P(Z_0 \cap Z_1 \cap C_{\mathrm{i}}). \tag{5.106}$$

Logo,

$$P(E_{\mathrm{i}}') = P(E_{\mathrm{i}}) + P(\Theta_{\mathrm{i}}) + P(\Phi_{\mathrm{i}}),$$

$$P(E_{\mathrm{i}}') \simeq P(E_{\mathrm{i}}) + P_{\mathrm{nb}}^2 \times P(Z_0^c \cap Z_1^c \cap C_{\mathrm{i}}) + P_{\mathrm{b}}^2 \times P(Z_0 \cap Z_1 \cap C_{\mathrm{i}}). \tag{5.107}$$

Os eventos Z_0^c, Z_1^c e C_{i} apresentam dependência, pois estão associados aos bits $\overline{r_{\mathrm{i}_0}}$, $\overline{r_{\mathrm{i}_1}}$ e c_{i}, os quais são dependentes dos bits de informação. O mesmo vale para os eventos Z_0, Z_1 e C_{i}, que estão associados aos bits r_{i_0}, r_{i_1} e c_{i}. Sendo assim, para a determinação das probabilidades $P(Z_0^c \cap Z_1^c \cap C_{\mathrm{i}})$ e $P(Z_0 \cap Z_1 \cap C_{\mathrm{i}})$, deve-se levar em consideração as expressões 5.94 e 5.95, mais especificamente os casos em que $\overline{r_{\mathrm{i}_0}} \cdot \overline{r_{\mathrm{i}_1}} \cdot c_{\mathrm{i}}$ e $r_{\mathrm{i}_0} \cdot r_{\mathrm{i}_1} \cdot c_{\mathrm{i}}$ assumem valor binário $0b1$.

Definição 5.3.1. De modo geral, assumindo, por simplicidade, um dado de M bits composto por um número par de 1s ($p_{\mathrm{i}} = 0$), os casos em que $\overline{r_{\mathrm{i}_0}} \cdot \overline{r_{\mathrm{i}_1}} \cdot c_{\mathrm{i}}$ é igual ao valor binário $0b1$ são aqueles em que o valor binário em *Ch0* é diferente do valor em *Ch1*, mesmo que aconteça *bitflips*. Além disso, pode ocorrer um número par de *bitflips* nos bits de informação em *Ch0* (associados à $\overline{r_{\mathrm{i}_0}}$) e um número par de *bitflips* nos bits de informação em *Ch1* (associados à $\overline{r_{\mathrm{i}_1}}$). Nesse caso, o zero é considerado como valor par, embora na prática signifique que não ocorreram *bitflips*.

Os casos em que os *bitflips* em *Ch0* e em *Ch1* ocorrem nos bits de mesma posição, fazendo com que o valor binário em *Ch0* seja igual ao valor em *Ch1*, podem ser calculados por meio de

$$C_{\mathrm{m}}(n_{\mathrm{p}}) = \binom{M}{n_{\mathrm{p}}}, \tag{5.108}$$

em que $n_{\mathrm{p}} \in \mathbb{N}_{\mathrm{p}}$, com $\mathbb{N}_{\mathrm{p}}$ correspondendo ao conjunto dos números naturais pares, ou seja, $\mathbb{N}_{\mathrm{p}} = \{0, 2, 4, 6, 8, \ldots\}$. Dessa maneira, o número de casos com essa característica corresponde à quantidade de combinações de M elementos agrupados n_{p} a n_{p}, em que n_{p} é um número par de *bitflips*. Enquanto isso, o número total de conjuntos de bits que podem ser formados pelas palavras-código em *Ch0* e em *Ch1*, considerando um número

par de *bitflips*, pode ser encontrado pela equação 5.17, de modo que:

$$T_{\mathrm{C}}(n_{\mathrm{p}_0}, n_{\mathrm{p}_1}) = \binom{M}{n_{\mathrm{p}_0}} \binom{M}{n_{\mathrm{p}_1}}, \tag{5.109}$$

em que $n_{\mathrm{p}_0} \in \mathbb{N}_{\mathrm{p}}$ e $n_{\mathrm{p}_1} \in \mathbb{N}_{\mathrm{p}}$ representam o número de *bitflips* que podem ser observados nas palavras-código em *Ch0* e em *Ch1*, respectivamente. Nota-se que $N_{\mathrm{a}} = N_{\mathrm{b}} = M$.

De acordo com a definição 5.3.2, os casos em que valores binários em *Ch0* e *Ch1* são iguais não devem ser considerados na contagem. Esses valores podem ser observados apenas quando a quantidade de *bitflips* em *Ch0* é igual a de *Ch1* ou quando não acontecem *bitflips* na palavras-código de ambos canais. Ou seja, usando as equações 5.109 e 5.108,

$$T_{\mathrm{C_p}}(n_{\mathrm{p}}) = T_{\mathrm{C}}(n_{\mathrm{p}_0}, n_{\mathrm{p}_1}) - C_{\mathrm{m}}(n_{\mathrm{p}}),$$

com $n_{\mathrm{p}_0} = n_{\mathrm{p}_1} = n_{\mathrm{p}}$, tem-se

$$\begin{aligned} T_{\mathrm{C_p}}(n_{\mathrm{p}}) &= T_{\mathrm{C}}(n_{\mathrm{p}}, n_{\mathrm{p}}) - C_{\mathrm{m}}(n_{\mathrm{p}}), \\ T_{C_{\mathrm{p}}}(n_{\mathrm{p}}) &= \binom{M}{n_{\mathrm{p}}} \binom{M}{n_{\mathrm{p}}} - \binom{M}{n_{\mathrm{p}}}, \end{aligned} \tag{5.110}$$

em que $T_{\mathrm{C_p}}(n_{\mathrm{p}})$ simboliza o total de casos em que o número de *bitflips* em *Ch0* e *Ch1* é par e igual a n_{p}, excluindo-se os casos em que os *bitflips* acontecem em bits de mesma posição.

Com base nas equações 5.109 e 5.110, pode-se escrever:

$$C_{\mathrm{p}}(n_{\mathrm{p}_0}, n_{\mathrm{p}_1}) = \begin{cases} T_{\mathrm{C}}(n_{\mathrm{p}_0}, n_{\mathrm{p}_1}) & \text{, se } n_{\mathrm{p}_0} \neq n_{\mathrm{p}_1} \\ T_{\mathrm{C_p}}(n_{\mathrm{p}}) & \text{, se } n_{\mathrm{p}_0} = n_{\mathrm{p}_1} = n_{\mathrm{p}}. \end{cases}$$

Por fim, as probabilidades de ocorrerem n_{p_0} *bitflips* nos bits de informação em *Ch0* e n_{p_1} *bitflips* nos bits de informação em *Ch1*, simbolizadas por $P(B_{\mathrm{r}_0} = n_{\mathrm{p}_0})$ e $P(B_{\mathrm{r}_1} = n_{\mathrm{p}_1})$, respectivamente, são obtidas por:

$$\begin{aligned} P(B_{\mathrm{r}_0} = n_{\mathrm{p}_0}) &\simeq P_{\mathrm{nb}}^{(M-n_{\mathrm{p}_0})} \times P_{\mathrm{b}}^{n_{\mathrm{p}_0}}, \\ P(B_{\mathrm{r}_1} = n_{p_1}) &\simeq P_{\mathrm{nb}}^{(M-n_{\mathrm{p}_1})} \times P_{\mathrm{b}}^{n_{\mathrm{p}_1}}. \end{aligned}$$

Desse modo, a probabilidade associada aos casos em que $\overline{r_{\mathrm{i}_0}} \cdot \overline{r_{\mathrm{i}_1}} \cdot c_{\mathrm{i}} = 1$ é dada por

$$P(Z_0^c \cap Z_1^c \cap C_{\mathrm{i}}) \simeq \sum_{\substack{n_{\mathrm{p}_0}=0 \\ n_{\mathrm{p}_0} \in \mathbb{N}_{\mathrm{p}}}}^{M} \left(\sum_{\substack{n_{\mathrm{p}_1}=0 \\ n_{\mathrm{p}_1} \in \mathbb{N}_{\mathrm{p}}}}^{M} \left(C_{\mathrm{p}}(n_{\mathrm{p}_0}, n_{\mathrm{p}_1}) P(B_{\mathrm{r}_0} = n_{\mathrm{p}_0}) P(B_{\mathrm{r}_1} = n_{\mathrm{p}_1}) \right) \right),$$

$$P(Z_0^c \cap Z_1^c \cap C_\mathrm{i}) \simeq \sum_{\substack{n_{\mathrm{p}_0}=0 \\ n_{\mathrm{p}_0}\in\mathbb{N}_\mathrm{p}}}^{M} \left(\sum_{\substack{n_{\mathrm{p}_1}=0 \\ n_{\mathrm{p}_1}\in\mathbb{N}_\mathrm{p}}}^{M} \left(C_\mathrm{p}(n_{\mathrm{p}_0}, n_{\mathrm{p}_1}) P_\mathrm{nb}^{(2M-(n_{\mathrm{p}_0}+n_{\mathrm{p}_1}))} P_\mathrm{b}^{(n_{\mathrm{p}_0}+n_{\mathrm{p}_1})} \right) \right). \quad (5.111)$$

Definição 5.3.2. Assumindo, por simplicidade, um dado de M bits composto por um número par de 1s ($p_\mathrm{i} = 0$), os casos em que $r_{\mathrm{i}_0} \cdot r_{\mathrm{i}_1} \cdot c_\mathrm{i}$ é igual ao valor binário $0b1$ são aqueles em que o valor binário em *Ch0* é diferente do valor em *Ch1*, mesmo que aconteça *bitflips*. Além disso, pode ocorrer um número ímpar de *bitflips* nos bits de informação em *Ch0* (associados à r_{i_0}) e um número ímpar de *bitflips* nos bits de informação em *Ch1* (associados à r_{i_1}).

Analisando a definição 5.3.2, verifica-se que a equação 5.81 pode ser utilizada para o cálculo do número de casos em que os *bitflips* em *Ch0* e em *Ch1* ocorrem nos bits de mesma posição, fazendo com que o valor binário em *Ch0* seja igual ao valor em *Ch1*. Enquanto isso, o número total de conjuntos de bits que podem ser formados pelas palavras-código em *Ch0* e em *Ch1*, considerando um número ímpar de *bitflips*, pode ser encontrado pela equação 5.17, de modo que:

$$T_\mathrm{C}(n_{\mathrm{i}_0}, n_{\mathrm{i}_1}) = \binom{M}{n_{\mathrm{i}_0}} \binom{M}{n_{\mathrm{i}_1}}, \quad (5.112)$$

em que $n_{\mathrm{i}_0} \in \mathbb{N}_\mathrm{i}$ e $n_{\mathrm{i}_1} \in \mathbb{N}_\mathrm{i}$ representam o número de *bitflips* que podem ser observados nas palavras-código em *Ch0* e em *Ch1*, respectivamente. Nota-se que $N_\mathrm{a} = N_\mathrm{b} = M$.

Desconsiderando, com base na definição 5.3.2, os casos em que valores binários em *Ch0* e *Ch1* são iguais e sabendo que eles acontecem apenas quando a quantidade de *bitflips* em *Ch0* é igual a de *Ch1*, usando 5.112 e 5.81, obtém-se

$$T_{\mathrm{C}_\mathrm{c}}(n_\mathrm{i}) = T_\mathrm{C}(n_{\mathrm{i}_0}, n_{\mathrm{i}_1}) - C_\mathrm{g}(n_\mathrm{i}),$$

com $n_{\mathrm{i}_0} = n_{\mathrm{i}_1} = n_\mathrm{i}$,

$$\begin{aligned} T_{\mathrm{C}_\mathrm{c}}(n_\mathrm{i}) &= T_\mathrm{C}(n_\mathrm{i}, n_\mathrm{i}) - C_\mathrm{g}(n_\mathrm{i}), \\ T_{\mathrm{C}_\mathrm{c}}(n_\mathrm{i}) &= \binom{M}{n_\mathrm{i}} \binom{M}{n_\mathrm{i}} - \binom{M}{n_\mathrm{i}}, \end{aligned} \quad (5.113)$$

em que $T_{\mathrm{C}_\mathrm{c}}(n_\mathrm{i})$ simboliza o total de casos em que o número de *bitflips* em *Ch0* e *Ch1* é ímpar e igual a n_i, excluindo-se os casos em que os *bitflips* acontecem em bits de mesma posição.

Com base nas equações 5.112 e 5.113, pode-se escrever:

$$C_\mathrm{c}(n_{\mathrm{i}_0}, n_{\mathrm{i}_1}) = \begin{cases} T_\mathrm{C}(n_{\mathrm{i}_0}, n_{\mathrm{i}_1}) & \text{, se } n_{\mathrm{i}_0} \neq n_{\mathrm{i}_1} \\ T_{\mathrm{C}_\mathrm{c}}(n_\mathrm{i}) & \text{, se } n_{\mathrm{i}_0} = n_{\mathrm{i}_1} = n_\mathrm{i}. \end{cases} \quad (5.114)$$

Desse modo, a probabilidade associada aos casos em que $r_{i_0} \cdot r_{i_1} \cdot c_i = 1$ é dada por

$$P(Z_0 \cap Z_1 \cap C_i) \simeq \sum_{\substack{n_{i_0}=1 \\ n_{i_0} \in \mathbb{N}_i}}^{M} \left(\sum_{\substack{n_{i_1}=1 \\ n_{i_1} \in \mathbb{N}_i}}^{M} \left(C_c(n_{i_0}, n_{i_1}) P_{nb}^{(2M-(n_{i_0}+n_{i_1}))} P_b^{(n_{i_0}+n_{i_1})} \right) \right). \quad (5.115)$$

As equações 5.111 e 5.115, em conjunto com a equação 5.83, são aplicadas na equação 5.107 para determinar a probabilidade $P(E_i')$. Além disso, importante notar que as equações 5.96 à 5.107 foram desenvolvidas supondo que o dado a ser transmitido contém um número par de 1s e, consequentemente, $(p_i = 0)$. Sendo assim, deve-se verificar o comportamento dos valores binários de paridade para os casos em que o dado é composto por um número ímpar de bits 1s. Nesses casos o valor original de p_i é igual a $0b1$. Por conseguinte,

$$P(E_\eta') = P(E_\eta) + P(\Theta_i)' + P(\Phi_i)', \quad (5.116)$$

em que $P(E_\eta)$ é dada pela equação 5.72 e o índice η em E_η' equivale a i, sendo $0 \leq \eta \leq G$. Esse índice é utilizado para distinção entre as equações 5.100 e 5.116. Além disso, baseando-se nos eventos apresentados na Tabela 5.6, tem-se:

$$P(\Theta_i)' = P(X_0^{c\prime} \cap X_1^{c\prime} \cap Z_0^{c\prime} \cap Z_1^{c\prime} \cap C_i),$$
$$P(\Phi_i)' = P(X_0' \cap X_1' \cap Z_0' \cap Z_1' \cap C_i).$$

Dada a independência entre os eventos, de acordo com descrição anterior,

$$P(\Theta_i)' \simeq P(X_0^{c\prime}) \times P(X_1^{c\prime}) \times P(Z_0^{c\prime} \cap Z_1^{c\prime} \cap C_i),$$
$$P(\Phi_i)' \simeq P(X_0') \times P(X_1') \times P(Z_0' \cap Z_1' \cap C_i).$$

como $P(X_0^{c\prime}) = P(X_1^{c\prime}) = P_b$ e $P(X_0') = P(X_1') = P_{nb}$,

$$P(\Theta_i)' \simeq P_b^2 \times P(Z_0^{c\prime} \cap Z_1^{c\prime} \cap C_i), \quad (5.117)$$
$$P(\Phi_i)' \simeq P_{nb}^2 \times P(Z_0' \cap Z_1' \cap C_i). \quad (5.118)$$

As fórmulas para as probabilidades $P(Z_0^{c\prime} \cap Z_1^{c\prime} \cap C_i)$ e $P(Z_0' \cap Z_1' \cap C_i)$ podem ser desenvolvidas de maneira análoga àquela realizada para $P(Z_0^c \cap Z_1^c \cap C_i)$ e $P(Z_0 \cap Z_1 \cap C_i)$. Contudo, a condição para que $\overline{r_{i_0}} \cdot \overline{r_{i_1}} \cdot c_i$ seja $0b1$ é que o valor binário em *Ch0* seja diferente do valor em *Ch1*, considerando a possibilidade de ocorrerem *bitflips*. Ademais, pode ocorrer um número ímpar de *bitflips* nos bits de informação em *Ch0* e em *Ch1*. Observa-se que essa condição é equivalente àquela descrita na definição 5.6. Sendo assim, o desenvolvimento matemático resulta em uma equação como a 5.115, de maneira que:

$$P(Z_0^{c\prime} \cap Z_1^{c\prime} \cap C_i) = P(Z_0 \cap Z_1 \cap C_i). \quad (5.119)$$

Analogamente, para que $r_{i_0} \cdot r_{i_1} \cdot c_i = 1$, tem-se uma condição equivalente àquela da defi-

nição 5.6. Portanto,

$$P(Z'_0 \cap Z'_1 \cap C_\mathrm{i}) = P(Z^c_0 \cap Z^c_1 \cap C_\mathrm{i}). \tag{5.120}$$

Utilizando as igualdades das equações 5.119 e 5.120 em 5.117 e 5.118, respectivamente, obtém-se:

$$P(\Theta_\mathrm{i})' \simeq P_\mathrm{b}^2 \times P(Z_0 \cap Z_1 \cap C_\mathrm{i}),$$
$$P(\Phi_\mathrm{i})' \simeq P_\mathrm{nb}^2 \times P(Z^c_0 \cap Z^c_1 \cap C_\mathrm{i}).$$

Logo,

$$P(\Theta_\mathrm{i})' = P(\Phi_\mathrm{i}),$$
$$P(\Phi_\mathrm{i})' = P(\Theta_\mathrm{i}).$$

Consequentemente,

$$P(E'_\eta) = P(E_\eta) + P(\Phi_\mathrm{i}) + P(\Theta_\mathrm{i}).$$

Contudo, de acordo com a equação 5.77, $P(E_\eta) = P(E_\mathrm{i})$, de maneira que

$$P(E'_\eta) = P(E_\mathrm{i}) + P(\Phi_\mathrm{i}) + P(\Theta_\mathrm{i}).$$

Ou seja,

$$P(E'_\eta) = P(E'_\mathrm{i}). \tag{5.121}$$

A igualdade da equação 5.121 torna evidente que a equação 5.107 é suficiente para o cálculo da probabilidade de detecção em cada grupo da palavra-código. Todavia, na transmissão de um dado de N bits dividido em N_p grupos, a probabilidade de detecção para a PDMR-C pode ser encontrada com fundamento no raciocínio apresentado para $P(D_\mathrm{pdmrg})$, considerando E'_i ao invés de E_i. Ou seja,

$$P(D_\mathrm{pdmrc}) \simeq \sum_{n=1}^{N_\mathrm{p}} \left((-1)^{(n-1)} \times \binom{N_\mathrm{p}}{n} \times P(E'_\mathrm{i})^n \right). \tag{5.122}$$

5.3.3 Erro

Assim como descrito para a PDMR-G, a probabilidade de erro para a PDMR-C é obtida com base nas probabilidades de correção e de detecção, sendo

$$P(E_\mathrm{pdmrc}) \simeq 1 - P(C_\mathrm{pdmrc}) - P(D_\mathrm{pdmrc}). \tag{5.123}$$

5.4 Resumo

Neste capítulo, foram apresentadas as análises matemáticas para as probabilidades de correção, de detecção e de erros das diferentes versões da proposta PDMR. De forma geral, a taxa de erros nos dois canais foi considerada como sendo a mesma, de modo que qualquer bit transmitido tem a mesma probabilidade de sofrer *bitflip*.

Diante das diversas fórmulas desenvolvidas ao longo do capítulo, na Tabela 5.8 estão dispostas as equações que devem ser utilizadas no cálculo das diferentes probabilidade para cada versão PDMR.

Tabela 5.8: Equações para o cálculo de probabilidades das diferentes versões PDMR

Versão	**Probabilidade**	**Equação Principal**	**Equações Auxiliares**
PDMR-B	Correção	5.28	5.26
	Erro	5.31	5.28
PDMR-G	Correção	5.48	5.45, 5.46
	Detecção	5.88	5.82, 5.83, 5.84
	Erro	5.89	5.48, 5.88
PDMR-C	Correção	5.90	5.91
	Detecção	5.122	5.83, 5.108, 5.109, 5.110, 5.111, 5.112, 5.113, 5.114, 5.115, 5.107
	Erro	5.123	5.90, 5.122

Fonte: O próprio autor.

As equações da Tabela 5.8 são utilizadas no Capítulo 6 para comparação dos resultados com experimentos de simulação em *software*, considerando diferentes taxas de erro, para as diversas versões PDMR.

Capítulo 6

Experimentos e Resultados

Dadas as definições da PDMR e suas expressões lógicas (Capítulo 4), bem como a base matemática para os cálculos das probabilidades (Capítulo 5), diferentes experimentos foram desenvolvidos. Com isso, buscou-se validar as equações formuladas e verificar aspectos de implementação da técnica proposta relacionados à utilização de recursos de *hardware*, como área ocupada e consumo energético.

Os experimentos foram fundamentados em simulações em *software* e implementações em *hardware*. Os resultados foram então comparados com aqueles obtidos para a técnica TMR. Dessa maneira, foram levantadas as características da PDMR referentes à capacidade de mascarar/corrigir falhas e detectar possíveis erros na transmissão de dados.

Os cálculos das probabilidades para as diferentes versões da PDMR foram baseados nas equações referenciadas na Tabela 5.8. Enquanto isso, para o desenvolvimento das simulações e a implementação em *hardware*, foram consideradas as expressões referenciadas na Tabela 4.4. Ademais, tomou-se como base as seguintes configurações para a PDMR:

- **32x8*: Configuração com $N = 32$ e $N_p = 8$ corresponde à 32 bits de informação e 8 bits de paridade;
- **32x4*: Configuração com $N = 32$ e $N_p = 4$ corresponde à 32 bits de informação e 4 bits de paridade;
- **24x8*: Configuração com $N = 24$ e $N_p = 8$ corresponde à 24 bits de informação e 8 bits de paridade;
- **28x4*: Configuração com $N = 28$ e $N_p = 4$ corresponde à 28 bits de informação e 4 bits de paridade.

Vale salientar que o símbolo * foi utilizado para generalizar as configurações. Entretanto, ao longo do capítulo, são utilizadas as letras *B*, *G* e *C*, para indicar, respectivamente, as versões PDMR-B, PDMR-G e PDMR-C. Ou seja, por exemplo, a implementação *B32x8*, corresponde à aplicação da técnica PDMR-B em uma entrada com 32 bits, formando uma palavra-código composta por 8 bits de paridade, enquanto *G32x8* representa a aplicação da técnica PDMR-G para a mesma quantidade de bits de entrada e de bits de paridade. Observa-se ainda que as configurações **32x8* e **32x4* se enquadram na regra de formação *RF1* (4.1.1). Por outro lado, as configurações **24x8* e **28x4* têm relação com a regra de formação *RF2* (4.1.2) e caracterizam um barramento de 32 bits.

No caso da TMR, uma vez que não são adicionado bits extras, considera-se apenas a transmissão dos bits de informação. Desse modo, as transmissões de 24, 28 e 32 bits com a TMR são simbolizada por *T24*, *T28* e *T32*, respectivamente. Além disso, durante as simulações em *software* a quantidade de dados transmitidos foi considerada como sendo igual a $D_{\mathrm{T}} = 10^6$.

6.1 Cálculos Matemáticos

As equações referenciadas na Tabela 5.8 podem ser utilizadas para o cálculo das probabilidades de correção, detecção e erro das diferentes versões da PDMR. Os valores calculados dependem, principalmente, da quantidade de bits do dado de entrada (N), do número de bits de paridade (N_{p}, que também define a quantidade de grupos) e da probabilidade de *bitflips* (P_{b}).

Como destacado no início do capítulo, quatro configurações distintas foram consideradas para cada uma das versões da PDMR. Os cálculos foram realizados com auxílio de *scripts* Python elaborados com base nas equações referentes a cada versão. Esses *scripts* recebem como entrada a largura do barramento (N_{d}), a quantidade de bits de informação (N) e a probabilidade de *bitflips* (P_{b}). Como saída, para a versão PDMR-B, tem-se as probabilidades de correção e de erro. Por outro lado, para as versões PDMR-G e PDMR-C, obtém-se também a probabilidade de detecção.

Os valores obtidos por meio das fórmulas desenvolvidas estão dispostos na Tabela 6.1. Esses valores são representados com duas casas decimais de maneira que simbolizam aproximações dos valores absolutos. Observa-se que as configurações com capacidade de detecção apresentam uma menor probabilidade de erros em comparação com a versão PDMR-B. Ademais, a adição do comparador torna a probabilidade de erros ainda menor.

6.2 Simulações em *Software*

A verificação da abordagem proposta foi realizada, inicialmente, por meio da comparação entre as probabilidades calculadas e os valores obtidos por simulação. Posteriormente, realizou-se a comparação da capacidade de correção da técnica TMR e das diferentes versões da PDMR. Em ambos os casos, os testes de comparação foram realizados via *software* por meio da linguagem de programação Python. Para isso, foram criados alguns *scripts* com diferentes funcionalidades, sendo os principais:

- Geração de dados: Responsável por gerar os dados a serem utilizados para transmissão. Esse *script* recebe como argumentos de entrada a quantidade de dados a ser gerada (D_{T}) e o tamanho de bits de cada dado (N). Como saída, o *script* gera um arquivo *CSV* em que cada linha representa um dado com N bits de informação e o número de linhas corresponde ao total de dados (D_{T});
- Geração de falhas: Esse *script* foi utilizado para gerar um vetor de falhas ao receber como argumentos de entradas a quantidade de dados e o tamanho dos bits (nesse caso, definido por N_{d}) de cada dado. Além disso, o *script* recebe como entrada a

Tabela 6.1: Probabilidades calculadas para diferentes configurações e versões da PDMR

Configuração	**Probabilidade**	P_b 10^{-3}	2×10^{-3}	5×10^{-3}	10^{-2}	2×10^{-2}
B32x8	$P(C_\mathrm{pdmrb})$	$9{,}99\times10^{-1}$	$9{,}99\times10^{-1}$	$9{,}94\times10^{-1}$	$9{,}77\times10^{-1}$	$9{,}16\times10^{-1}$
	$P(E_\mathrm{pdmrb})$	$2{,}39\times10^{-4}$	$9{,}51\times10^{-4}$	$5{,}85\times10^{-3}$	$2{,}27\times10^{-2}$	$8{,}42\times10^{-2}$
G32x8	$P(C_\mathrm{pdmrg})$	$9{,}99\times10^{-1}$	$9{,}99\times10^{-1}$	$9{,}92\times10^{-1}$	$9{,}71\times10^{-1}$	$8{,}93\times10^{-1}$
	$P(D_\mathrm{pdmrg})$	$2{,}62\times10^{-4}$	$1{,}04\times10^{-3}$	$6{,}32\times10^{-3}$	$2{,}41\times10^{-2}$	$8{,}66\times10^{-2}$
	$P(E_\mathrm{pdmrg})$	$4{,}84\times10^{-5}$	$1{,}95\times10^{-4}$	$1{,}24\times10^{-3}$	$5{,}06\times10^{-3}$	$2{,}01\times10^{-2}$
C32x8	$P(C_\mathrm{pdmrc})$	$9{,}99\times10^{-1}$	$9{,}99\times10^{-1}$	$9{,}91\times10^{-1}$	$9{,}67\times10^{-1}$	$8{,}79\times10^{-1}$
	$P(D_\mathrm{pdmrc})$	$3{,}57\times10^{-4}$	$1{,}42\times10^{-3}$	$8{,}61\times10^{-3}$	$3{,}28\times10^{-2}$	$1{,}16\times10^{-1}$
	$P(E_\mathrm{pdmrc})$	$7{,}90\times10^{-7}$	$6{,}30\times10^{-6}$	$9{,}59\times10^{-5}$	$7{,}25\times10^{-4}$	$4{,}99\times10^{-3}$
B32x4	$P(C_\mathrm{pdmrb})$	$9{,}99\times10^{-1}$	$9{,}98\times10^{-1}$	$9{,}90\times10^{-1}$	$9{,}61\times10^{-1}$	$8{,}66\times10^{-1}$
	$P(E_\mathrm{pdmrb})$	$4{,}28\times10^{-4}$	$1{,}69\times10^{-3}$	$1{,}02\times10^{-2}$	$3{,}85\times10^{-2}$	$1{,}34\times10^{-1}$
G32x4	$P(C_\mathrm{pdmrg})$	$9{,}99\times10^{-1}$	$9{,}98\times10^{-1}$	$9{,}88\times10^{-1}$	$9{,}56\times10^{-1}$	$8{,}48\times10^{-1}$
	$P(D_\mathrm{pdmrg})$	$3{,}82\times10^{-4}$	$1{,}50\times10^{-3}$	$8{,}93\times10^{-3}$	$3{,}27\times10^{-2}$	$1{,}09\times10^{-1}$
	$P(E_\mathrm{pdmrg})$	$1{,}13\times10^{-4}$	$4{,}53\times10^{-4}$	$2{,}86\times10^{-3}$	$1{,}15\times10^{-2}$	$4{,}35\times10^{-2}$
C32x4	$P(C_\mathrm{pdmrc})$	$9{,}99\times10^{-1}$	$9{,}98\times10^{-1}$	$9{,}86\times10^{-1}$	$9{,}47\times10^{-1}$	$8{,}20\times10^{-1}$
	$P(D_\mathrm{pdmrc})$	$6{,}02\times10^{-4}$	$2{,}37\times10^{-3}$	$1{,}41\times10^{-2}$	$5{,}12\times10^{-2}$	$1{,}67\times10^{-1}$
	$P(E_\mathrm{pdmrc})$	$2{,}55\times10^{-6}$	$2{,}01\times10^{-5}$	$2{,}98\times10^{-4}$	$2{,}15\times10^{-3}$	$1{,}34\times10^{-2}$
B24x8	$P(C_\mathrm{pdmrb})$	$9{,}99\times10^{-1}$	$9{,}99\times10^{-1}$	$9{,}96\times10^{-1}$	$9{,}86\times10^{-1}$	$9{,}47\times10^{-1}$
	$P(E_\mathrm{pdmrb})$	$1{,}44\times10^{-4}$	$5{,}72\times10^{-4}$	$3{,}53\times10^{-3}$	$1{,}38\times10^{-2}$	$5{,}26\times10^{-2}$
G24x8	$P(C_\mathrm{pdmrg})$	$9{,}99\times10^{-1}$	$9{,}99\times10^{-1}$	$9{,}95\times10^{-1}$	$9{,}81\times10^{-1}$	$9{,}29\times10^{-1}$
	$P(D_\mathrm{pdmrg})$	$1{,}75\times10^{-4}$	$6{,}95\times10^{-4}$	$4{,}26\times10^{-3}$	$1{,}65\times10^{-2}$	$6{,}07\times10^{-2}$
	$P(E_\mathrm{pdmrg})$	$2{,}42\times10^{-5}$	$9{,}78\times10^{-5}$	$6{,}27\times10^{-4}$	$2{,}59\times10^{-3}$	$1{,}06\times10^{-2}$
C24x8	$P(C_\mathrm{pdmrc})$	$9{,}99\times10^{-1}$	$9{,}99\times10^{-1}$	$9{,}95\times10^{-1}$	$9{,}79\times10^{-1}$	$9{,}21\times10^{-1}$
	$P(D_\mathrm{pdmrc})$	$2{,}23\times10^{-4}$	$8{,}85\times10^{-4}$	$5{,}42\times10^{-3}$	$2{,}09\times10^{-2}$	$7{,}67\times10^{-2}$
	$P(E_\mathrm{pdmrc})$	$3{,}80\times10^{-7}$	$3{,}04\times10^{-6}$	$4{,}66\times10^{-5}$	$3{,}59\times10^{-4}$	$2{,}59\times10^{-3}$
B28x4	$P(C_\mathrm{pdmrb})$	$9{,}99\times10^{-1}$	$9{,}99\times10^{-1}$	$9{,}92\times10^{-1}$	$9{,}70\times10^{-1}$	$8{,}92\times10^{-1}$
	$P(E_\mathrm{pdmrb})$	$3{,}33\times10^{-4}$	$1{,}32\times10^{-3}$	$8{,}02\times10^{-3}$	$3{,}05\times10^{-2}$	$1{,}08\times10^{-1}$
G28x4	$P(C_\mathrm{pdmrg})$	$9{,}99\times10^{-1}$	$9{,}98\times10^{-1}$	$9{,}91\times10^{-1}$	$9{,}64\times10^{-1}$	$8{,}75\times10^{-1}$
	$P(D_\mathrm{pdmrg})$	$3{,}08\times10^{-4}$	$1{,}21\times10^{-3}$	$7{,}25\times10^{-3}$	$2{,}69\times10^{-2}$	$9{,}14\times10^{-2}$
	$P(E_\mathrm{pdmrg})$	$8{,}46\times10^{-5}$	$3{,}40\times10^{-4}$	$2{,}16\times10^{-3}$	$8{,}68\times10^{-3}$	$3{,}36\times10^{-2}$
C28x4	$P(C_\mathrm{pdmrc})$	$9{,}99\times10^{-1}$	$9{,}98\times10^{-1}$	$9{,}89\times10^{-1}$	$9{,}57\times10^{-1}$	$8{,}52\times10^{-1}$
	$P(D_\mathrm{pdmrc})$	$4{,}73\times10^{-4}$	$1{,}87\times10^{-3}$	$1{,}11\times10^{-2}$	$4{,}11\times10^{-2}$	$1{,}38\times10^{-1}$
	$P(E_\mathrm{pdmrc})$	$1{,}76\times10^{-6}$	$1{,}39\times10^{-5}$	$2{,}08\times10^{-4}$	$1{,}53\times10^{-3}$	$9{,}87\times10^{-3}$

Fonte: O próprio autor.

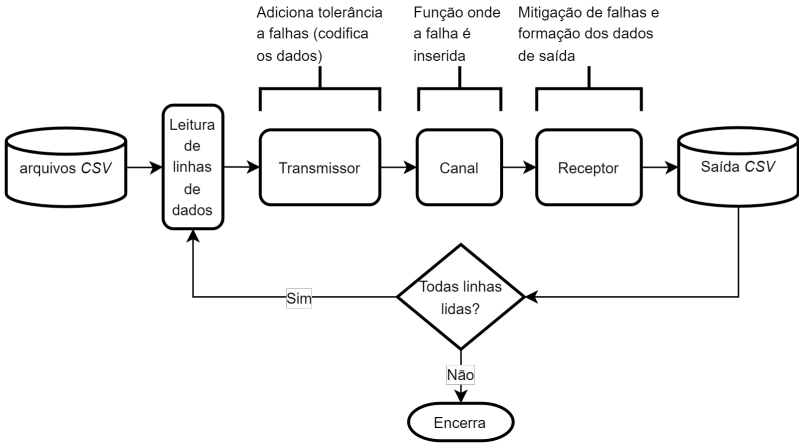
taxa de erros a ser utilizada para o vetor de falhas. Essa taxa de erros é definida pela probabilidade de *bitflips* e denotada por P_b. Um evento aleatório de ocorrência de um *bitflip* é simulado por meio de uma distribuição de probabilidade uniforme. Um número aleatório é gerado e caso seu valor seja menor que o valor da taxa, um *bitflip* ocorre. O procedimento é aplicado para cada bit da mensagem transmitida. Como saída, o *script* gera um arquivo *CSV* com três colunas com dados de N_d bits cada, em que cada uma deve alimentar um canal para o teste com TMR, enquanto apenas duas delas devem ser utilizadas para os testes com PDMR. O número de linhas no arquivo *CSV* também corresponde ao total de dados (D_T);

- Controle da simulação e análise dos dados: Efetua a leitura do arquivos *CSV* gerados pelos dois *scripts* com funcionalidade descrita anteriormente. Entre as entradas

recebidas por esse *script*, tem-se o tamanho do barramento (N_d), a quantidade de bits de informação (N) e um argumento que indica a localização dos bits de paridade na palavra de dados, o que está de acordo com o diagrama apresentado na Figura 4.4. Ademais, esse *script* gera um arquivo de saída *CSV*, que contém os dados transmitidos originalmente, os dados após passarem por cada canal e os dados recebidos pelo receptor após aplicação dos métodos de tolerância TMR, PDMR.

Na Figura 6.1, está apresentado um diagrama da simulação. Inicialmente, os arquivos gerados pelos *scripts* de geração de dados e de falhas são lidos e cada linha é percorrida. O transmissor se encarrega da duplicação ou triplicação dos canais, bem como da inserção da paridade para a PDMR. Em seguida, são injetadas falhas aos canais por meio de uma função que recebe como argumento o dado do canal e realiza uma operação *XOR* com uma das colunas (de acordo com o canal) do arquivo que contém os vetores de falhas. Nos testes com PDMR, apenas dois canais são utilizados, de modo que apenas as colunas 0 e 1 do arquivo são consideradas, já para a TMR as colunas 0, 1 e 2 são utilizadas. No receptor, esses dados são tratados de modo a mitigar os erros, de maneira que, para a TMR, utiliza-se o votador lógico clássico 2.1, enquanto que, para a PDMR, foram utilizadas as lógicas descritas nas seções 4.2, 4.4 e 4.5, de acordo com a versão implementada.

Figura 6.1: Diagrama da simulação em *software* baseada em *scripts* Python



Fonte: Adaptado de Alves et al. (2024).

6.2.1 Comparação Entre Valores Simulados e Calculados

A Tabela 6.2 contém os resultados aproximados das simulações para as diferentes versões e configurações da PDMR. Esses resultados foram obtidos considerando uma média da execução de 100 simulações. Em outras palavras, cem arquivos de vetores de falhas, com um total de D_T linhas, foram gerados para cada valor de P_b e cada arquivo foi aplicado no conjunto de dados correspondente (arquivo *CSV* com D_T linhas). Os valores da tabela são aproximados uma vez que são representados com duas casas decimais.

Tabela 6.2: Probabilidades simuladas para diferentes configurações e versões da PDMR

Configuração	**Probabilidade**	P_b 10^{-3}	2×10^{-3}	5×10^{-3}	10^{-2}	2×10^{-2}
B32x8	$P(C_\mathrm{pdmrb})$	$9,99\times10^{-1}$	$9,99\times10^{-1}$	$9,94\times10^{-1}$	$9,77\times10^{-1}$	$9,16\times10^{-1}$
	$P(E_\mathrm{pdmrb})$	$2,37\times10^{-4}$	$9,53\times10^{-4}$	$5,85\times10^{-3}$	$2,27\times10^{-2}$	$8,42\times10^{-2}$
G32x8	$P(C_\mathrm{pdmrg})$	$9,99\times10^{-1}$	$9,99\times10^{-1}$	$9,92\times10^{-1}$	$9,71\times10^{-1}$	$8,93\times10^{-1}$
	$P(D_\mathrm{pdmrg})$	$2,61\times10^{-4}$	$1,04\times10^{-3}$	$6,33\times10^{-3}$	$2,41\times10^{-2}$	$8,66\times10^{-2}$
	$P(E_\mathrm{pdmrg})$	$4,85\times10^{-5}$	$1,95\times10^{-4}$	$1,24\times10^{-3}$	$5,05\times10^{-3}$	$2,01\times10^{-2}$
C32x8	$P(C_\mathrm{pdmrc})$	$9,99\times10^{-1}$	$9,99\times10^{-1}$	$9,91\times10^{-1}$	$9,67\times10^{-1}$	$8,79\times10^{-1}$
	$P(D_\mathrm{pdmrc})$	$3,56\times10^{-4}$	$1,42\times10^{-3}$	$8,62\times10^{-3}$	$3,28\times10^{-2}$	$1,16\times10^{-1}$
	$P(E_\mathrm{pdmrc})$	$6,80\times10^{-7}$	$6,66\times10^{-6}$	$9,60\times10^{-5}$	$7,27\times10^{-4}$	$4,99\times10^{-3}$
B32x4	$P(C_\mathrm{pdmrb})$	$9,99\times10^{-1}$	$9,98\times10^{-1}$	$9,90\times10^{-1}$	$9,61\times10^{-1}$	$8,65\times10^{-1}$
	$P(E_\mathrm{pdmrb})$	$4,25\times10^{-4}$	$1,69\times10^{-3}$	$1,02\times10^{-2}$	$3,86\times10^{-2}$	$1,35\times10^{-1}$
G32x4	$P(C_\mathrm{pdmrg})$	$9,99\times10^{-1}$	$9,98\times10^{-1}$	$9,88\times10^{-1}$	$9,56\times10^{-1}$	$8,48\times10^{-1}$
	$P(D_\mathrm{pdmrg})$	$3,83\times10^{-4}$	$1,50\times10^{-3}$	$8,91\times10^{-3}$	$3,27\times10^{-2}$	$1,09\times10^{-1}$
	$P(E_\mathrm{pdmrg})$	$1,12\times10^{-4}$	$4,53\times10^{-4}$	$2,87\times10^{-3}$	$1,15\times10^{-2}$	$4,35\times10^{-2}$
C32x4	$P(C_\mathrm{pdmrc})$	$9,99\times10^{-1}$	$9,98\times10^{-1}$	$9,86\times10^{-1}$	$9,47\times10^{-1}$	$8,20\times10^{-1}$
	$P(D_\mathrm{pdmrc})$	$6,02\times10^{-4}$	$2,37\times10^{-3}$	$1,40\times10^{-2}$	$5,12\times10^{-2}$	$1,67\times10^{-1}$
	$P(E_\mathrm{pdmrc})$	$2,79\times10^{-6}$	$1,98\times10^{-5}$	$3,02\times10^{-4}$	$2,15\times10^{-3}$	$1,34\times10^{-2}$
B24x8	$P(C_\mathrm{pdmrb})$	$9,99\times10^{-1}$	$9,99\times10^{-1}$	$9,96\times10^{-1}$	$9,86\times10^{-1}$	$9,47\times10^{-1}$
	$P(E_\mathrm{pdmrb})$	$1,43\times10^{-4}$	$5,77\times10^{-4}$	$3,55\times10^{-3}$	$1,38\times10^{-2}$	$5,26\times10^{-2}$
G24x8	$P(C_\mathrm{pdmrg})$	$9,99\times10^{-1}$	$9,99\times10^{-1}$	$9,95\times10^{-1}$	$9,81\times10^{-1}$	$9,29\times10^{-1}$
	$P(D_\mathrm{pdmrg})$	$1,75\times10^{-4}$	$6,97\times10^{-4}$	$4,26\times10^{-3}$	$1,64\times10^{-2}$	$6,07\times10^{-2}$
	$P(E_\mathrm{pdmrg})$	$2,35\times10^{-5}$	$9,81\times10^{-5}$	$6,26\times10^{-4}$	$2,59\times10^{-3}$	$1,06\times10^{-2}$
C24x8	$P(C_\mathrm{pdmrc})$	$9,99\times10^{-1}$	$9,99\times10^{-1}$	$9,95\times10^{-1}$	$9,79\times10^{-1}$	$9,21\times10^{-1}$
	$P(D_\mathrm{pdmrc})$	$2,23\times10^{-4}$	$8,86\times10^{-4}$	$5,41\times10^{-3}$	$2,09\times10^{-2}$	$7,67\times10^{-2}$
	$P(E_\mathrm{pdmrc})$	$3,70\times10^{-7}$	$3,02\times10^{-6}$	$4,68\times10^{-5}$	$3,55\times10^{-4}$	$2,59\times10^{-3}$
B28x4	$P(C_\mathrm{pdmrb})$	$9,99\times10^{-1}$	$9,99\times10^{-1}$	$9,92\times10^{-1}$	$9,70\times10^{-1}$	$8,91\times10^{-1}$
	$P(E_\mathrm{pdmrb})$	$3,29\times10^{-4}$	$1,33\times10^{-3}$	$8,03\times10^{-3}$	$3,05\times10^{-2}$	$1,09\times10^{-1}$
G28x4	$P(C_\mathrm{pdmrg})$	$9,99\times10^{-1}$	$9,98\times10^{-1}$	$9,91\times10^{-1}$	$9,64\times10^{-1}$	$8,75\times10^{-1}$
	$P(D_\mathrm{pdmrg})$	$3,09\times10^{-4}$	$1,22\times10^{-3}$	$7,25\times10^{-3}$	$2,69\times10^{-2}$	$9,14\times10^{-2}$
	$P(E_\mathrm{pdmrg})$	$8,47\times10^{-5}$	$3,39\times10^{-4}$	$2,15\times10^{-3}$	$8,67\times10^{-3}$	$3,36\times10^{-2}$
C28x4	$P(C_\mathrm{pdmrc})$	$9,99\times10^{-1}$	$9,98\times10^{-1}$	$9,89\times10^{-1}$	$9,57\times10^{-1}$	$8,52\times10^{-1}$
	$P(D_\mathrm{pdmrc})$	$4,75\times10^{-4}$	$1,87\times10^{-3}$	$1,11\times10^{-2}$	$4,11\times10^{-2}$	$1,38\times10^{-1}$
	$P(E_\mathrm{pdmrc})$	$1,92\times10^{-6}$	$1,35\times10^{-5}$	$2,11\times10^{-4}$	$1,52\times10^{-3}$	$9,85\times10^{-3}$

Fonte: O próprio autor.

Com base nas Tabelas 6.1 e 6.2, pode-se observar que os valores calculados estão próximos aos valores simulados, de maneira que as diferenças ocorrem em ordem de grandeza pequenas. Desse modo, com o intuito de avaliar os modelos calculados foi utilizada a métrica do erro quadrático médio (MSE, *Mean Squared Error*). Para isso, aplicou-se a seguinte equação

$$\mathrm{MSE} = \frac{1}{N_\mathrm{S}} \sum_{\kappa=1}^{N_\mathrm{S}} \left(V_{\mathrm{S}_\kappa} - V_\mathrm{C}\right)^2, \tag{6.1}$$

em que N_S indica a quantidade de simulações, V_{S_κ} simboliza o valor absoluto (sem arredondamentos) para a κ-ésima simulação e V_C representa o valor absoluto calculado. Por

representar a diferença entre o valor simulado e calculado elevada ao quadrado, o MSE penaliza os valores que são muito diferentes entre o simulado e o calculado. Ou seja, um valor de MSE alto indica que o modelo obtido pelos cálculos está distante do verificado nas simulações.

Os resultados do cálculo do MSE para as diferentes versões da PDMR podem ser observados nas Tabelas 6.3, 6.4, 6.5 e 6.6. Destaca-se que alguns valores foram omitidos nas tabelas pelo fato de serem valores imprecisos em função do número de pontos simulados ser insuficiente para caracterizar a ocorrência dos eventos.

Tabela 6.3: Erro quadrático médio para 32 bits de informação e 8 de paridade

P_b	*B32x8*	*G32x8*			*C32x8*		
	Correção e Erro	Correção	Detecção	Erro	Correção	Detecção	Erro
10^{-3}	$2,51 \times 10^{-10}$	$2,41 \times 10^{-10}$	$2,60 \times 10^{-10}$	-	$2,53 \times 10^{-10}$	$2,53 \times 10^{-10}$	-
2×10^{-3}	$8,15 \times 10^{-10}$	$1,36 \times 10^{-09}$	$1,05 \times 10^{-09}$	$1,68 \times 10^{-10}$	$1,60 \times 10^{-09}$	$1,55 \times 10^{-09}$	-
5×10^{-3}	$5,35 \times 10^{-09}$	$5,34 \times 10^{-09}$	$4,64 \times 10^{-09}$	$1,18 \times 10^{-09}$	$6,92 \times 10^{-09}$	$6,84 \times 10^{-09}$	$1,04 \times 10^{-10}$
10^{-2}	$2,32 \times 10^{-08}$	$3,01 \times 10^{-08}$	$2,24 \times 10^{-08}$	$5,38 \times 10^{-09}$	$3,65 \times 10^{-08}$	$3,58 \times 10^{-08}$	$7,23 \times 10^{-10}$
2×10^{-2}	$8,42 \times 10^{-08}$	$8,24 \times 10^{-08}$	$7,47 \times 10^{-08}$	$2,57 \times 10^{-08}$	$8,12 \times 10^{-08}$	$8,12 \times 10^{-08}$	$6,38 \times 10^{-09}$

Fonte: O próprio autor.

Tabela 6.4: Erro quadrático médio para 32 bits de informação e 4 de paridade

P_b	*B32x4*	*G32x4*			*C32x4*		
	Correção e Erro	Correção	Detecção	Erro	Correção	Detecção	Erro
10^{-3}	$4,74 \times 10^{-10}$	$4,76 \times 10^{-10}$	$3,92 \times 10^{-10}$	$1,13 \times 10^{-10}$	$6,30 \times 10^{-10}$	$6,26 \times 10^{-10}$	-
2×10^{-3}	$1,72 \times 10^{-09}$	$1,46 \times 10^{-09}$	$1,35 \times 10^{-09}$	$5,24 \times 10^{-10}$	$1,91 \times 10^{-09}$	$1,88 \times 10^{-09}$	-
5×10^{-3}	$8,93 \times 10^{-09}$	$1,24 \times 10^{-08}$	$8,54 \times 10^{-09}$	$2,98 \times 10^{-09}$	$1,39 \times 10^{-08}$	$1,34 \times 10^{-08}$	$3,24 \times 10^{-10}$
10^{-2}	$3,89 \times 10^{-08}$	$3,18 \times 10^{-08}$	$2,74 \times 10^{-08}$	$1,02 \times 10^{-08}$	$4,19 \times 10^{-08}$	$4,56 \times 10^{-08}$	$2,52 \times 10^{-09}$
2×10^{-2}	$1,46 \times 10^{-07}$	$1,24 \times 10^{-07}$	$9,56 \times 10^{-08}$	$4,99 \times 10^{-08}$	$1,58 \times 10^{-07}$	$1,43 \times 10^{-07}$	$1,16 \times 10^{-08}$

Fonte: O próprio autor.

Tabela 6.5: Erro quadrático médio para 24 bits de informação e 8 de paridade

P_b	*B24x8*	*G24x8*			*C24x8*		
	Correção e Erro	Correção	Detecção	Erro	Correção	Detecção	Erro
10^{-3}	$1,50 \times 10^{-10}$	$1,92 \times 10^{-10}$	$1,82 \times 10^{-10}$	-	$2,16 \times 10^{-10}$	$2,16 \times 10^{-10}$	-
2×10^{-3}	$4,67 \times 10^{-10}$	$8,78 \times 10^{-10}$	$8,38 \times 10^{-10}$	$9,98 \times 10^{-11}$	$9,29 \times 10^{-10}$	$9,42 \times 10^{-10}$	-
5×10^{-3}	$4,02 \times 10^{-09}$	$3,99 \times 10^{-09}$	$3,32 \times 10^{-09}$	$5,32 \times 10^{-10}$	$3,99 \times 10^{-09}$	$3,90 \times 10^{-09}$	-
10^{-2}	$1,34 \times 10^{-08}$	$2,01 \times 10^{-08}$	$1,66 \times 10^{-08}$	$2,24 \times 10^{-09}$	$1,91 \times 10^{-08}$	$1,84 \times 10^{-08}$	$5,20 \times 10^{-10}$
2×10^{-2}	$4,86 \times 10^{-08}$	$7,01 \times 10^{-08}$	$6,33 \times 10^{-08}$	$1,21 \times 10^{-08}$	$7,31 \times 10^{-08}$	$7,00 \times 10^{-08}$	$2,57 \times 10^{-09}$

Fonte: O próprio autor.

Os valores de MSE nas Tabelas 6.3, 6.4, 6.5 e 6.6 evidenciam que as fórmulas desenvolvidas no Capítulo 5 geram probabilidades muito próximas ao que se obtém por simulação. Sendo assim, essas fórmulas caracterizam o comportamento das diferentes versões

Tabela 6.6: Erro quadrático médio para 28 bits de informação e 4 de paridade

P_b	*B28x4*	*G28x4*			*C28x4*		
	Correção e Erro	Correção	Detecção	Erro	Correção	Detecção	Erro
10^{-3}	$3,17 \times 10^{-10}$	$3,83 \times 10^{-10}$	$2,93 \times 10^{-10}$	-	$4,00 \times 10^{-10}$	$3,93 \times 10^{-10}$	-
2×10^{-3}	$1,36 \times 10^{-09}$	$1,43 \times 10^{-09}$	$1,25 \times 10^{-09}$	$3,96 \times 10^{-10}$	$1,61 \times 10^{-09}$	$1,54 \times 10^{-09}$	-
5×10^{-3}	$9,27 \times 10^{-09}$	$7,92 \times 10^{-09}$	$6,17 \times 10^{-09}$	$1,84 \times 10^{-09}$	$9,50 \times 10^{-09}$	$9,07 \times 10^{-09}$	$2,08 \times 10^{-10}$
10^{-2}	$3,05 \times 10^{-08}$	$3,86 \times 10^{-08}$	$3,07 \times 10^{-08}$	$8,42 \times 10^{-09}$	$4,71 \times 10^{-08}$	$4,54 \times 10^{-08}$	$1,46 \times 10^{-09}$
2×10^{-2}	$1,01 \times 10^{-07}$	$9,56 \times 10^{-08}$	$6,19 \times 10^{-08}$	$2,68 \times 10^{-08}$	$1,24 \times 10^{-07}$	$1,16 \times 10^{-07}$	$1,08 \times 10^{-08}$

Fonte: O próprio autor.

PDMR de maneira satisfatória e podem ser utilizadas para o cálculo de probabilidades para configurações PDMR diferentes das apresentadas neste trabalho.

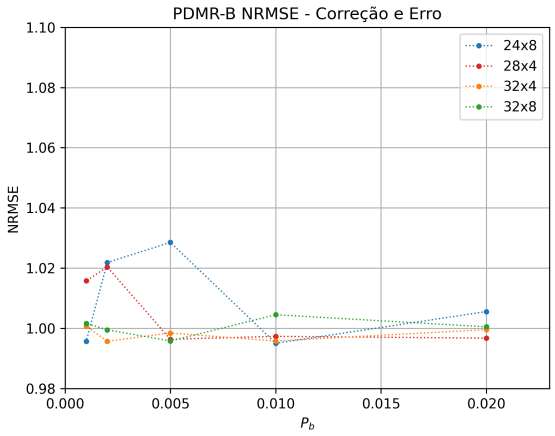
A raiz quadrada do erro quadrático médio normalizado (NRMSE, *Normalized Root Mean Squared Error*) foi considerada como métrica adicional para caracterização do erro entre os valores simulados e calculados. Tal normalização foi obtida por

$$\text{NRMSE} = \frac{1}{\sigma}\sqrt{\text{MSE}}, \tag{6.2}$$

em que σ é o desvio padrão dos valores simulados. Desse modo, valores de NRMSE próximos a 1 indicam pouca variação dos valores simulados com relação aos calculados.

As Figuras 6.2 e 6.3 contêm os valores NRMSE para diferentes configurações com relação ao valores de P_b. Observa-se que os pontos nos gráficos estão próximos à 1. Ademais, alguns valores para as Figuras 6.3e e 6.3f foram desconsiderados por representarem imprecisões relacionadas ao número de pontos simulados. Uma vez que a inclusão do comparador implica em uma redução na probabilidade de erros, a quantidade de dados com erros após as simulações é muito pequena para o cálculo do NRMSE. Isso explica uma maior variação dos pontos na Figura 6.3f.

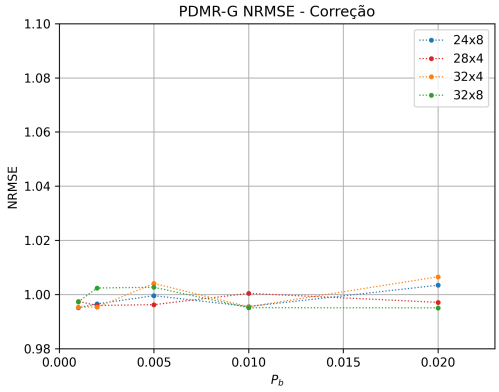
Figura 6.2: NRMSE para a versão PDMR-B



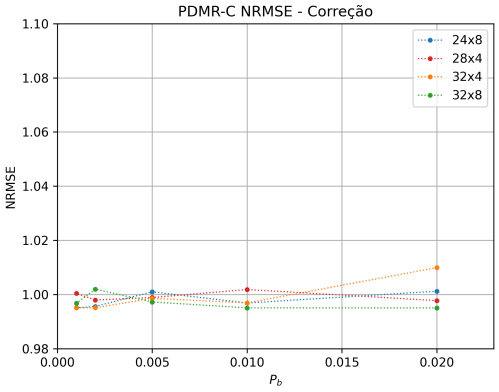
Fonte: O próprio autor.

Figura 6.3: NRMSE para as versões PDMR-G e PDMR-C

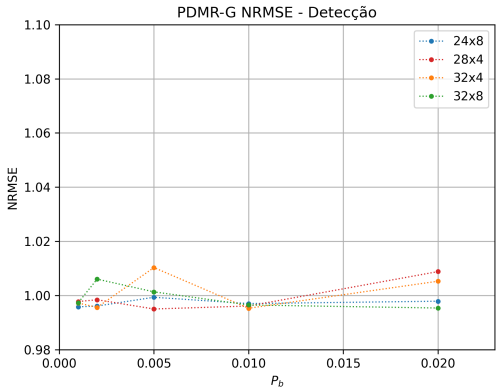
(a)



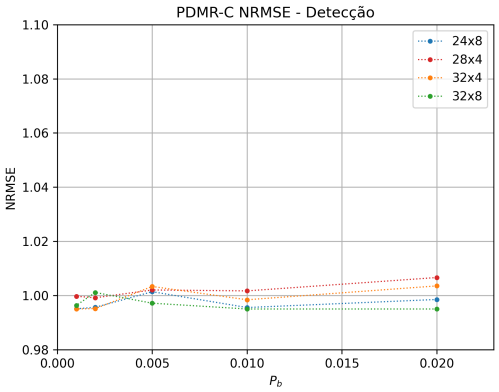
(b)



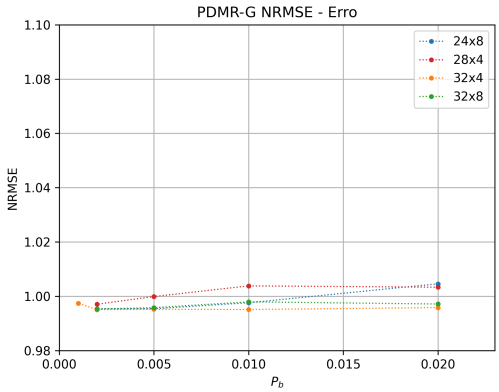
(c)



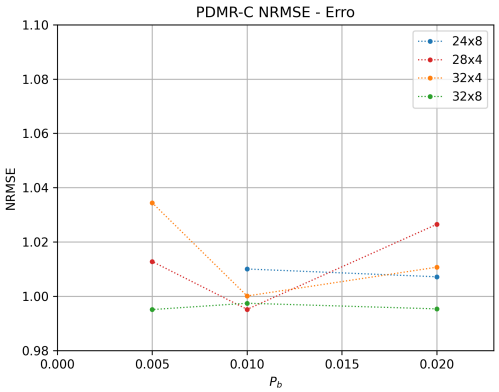
(d)



(e)



(f)



Fonte: O próprio autor.

6.2.2 Capacidade de Correção da PDMR e da TMR

Diante da verificação de correspondência entre os valores calculados e simulados, as simulações foram tomadas como base para comparação da PDMR com a técnica TMR. Nesse contexto, a Tabela 6.7 contém os resultados aproximados das simulações para as configurações *32x8*, *32x4*, *24x8* e *28x4*, bem como das transmissões TMR correspondentes (*T32*, *T24* e *T28*). Nessa tabela, P_{D} representa a porcentagem de erros detectados, enquanto P_{U} e P_{E} representam as porcentagens de erros não detectados e erros totais, respectivamente. Importante notar que o valor de P_{E} está relacionado com o número total de dados (mensagens) transmitidos (D_{T}). Por outro lado, P_{D} e P_{U} são as porcentagens associadas com o número de dados classificados como errados ($E = D_{\mathrm{T}} \times P_{\mathrm{E}}$). Como exemplo, para a configuração *C32x8* com $P_{\mathrm{b}} = 2 \times 10^{-2}$ e tomando $D_{\mathrm{T}} = 10^6$, a quantidade de dados errados é igual a $E = D_{\mathrm{T}} \times P_{\mathrm{E}} = 10^6 \times 12,13/100 \simeq 121300$. Desse total, a quantidade de dados detectados com erro (E_{D}) é obtida por $E_{\mathrm{D}} = 121300 \times P_{\mathrm{D}} = 121300 \times 95,89/100 \simeq 116315$, enquanto a não detectada é dada por $E_{\mathrm{U}} = 121300 \times P_{\mathrm{U}} = 121300 \times 4,11/100 \simeq 4985$.

Tabela 6.7: Resultados aproximados das simulações para as configurações *32x8*, *32x4*, *24x8* e *28x4*. As linhas em que se encontra a TMR contêm informações referentes às configurações *T32* (colunas *32x8* e *32x4*), *T24* (coluna *24x8*) e *T28* (coluna *28x4*)

P_{b}	**Método**	*32x8*			*32x4*			*24x8*			*28x4*		
		P_{D}	P_{U}	P_{E}	P_{D}	P_{U}	P_{E}	P_{D}	P_{U}	P_{E}	P_{D}	P_{U}	P_{E}
10^{-3}	TMR	0	100	0,01	0	100	0,01	0	100	0,01	0	100	0,01
	PDMR-B	0	100	0,02	0	100	0,04	0	100	0,01	0	100	0,03
	PDMR-G	84,33	15,67	0,03	77,38	22,62	0,05	88,18	11,82	0,02	78,48	21,52	0,04
	PDMR-C	99,81	0,19	0,04	99,54	0,46	0,06	99,83	0,17	0,02	99,60	0,40	0,05
2×10^{-3}	TMR	0	100	0,04	0	100	0,04	0	100	0,03	0	100	0,03
	PDMR-B	0	100	0,10	0	100	0,17	0	100	0,06	0	100	0,13
	PDMR-G	84,27	15,73	0,12	76,84	23,16	0,20	87,66	12,34	0,08	78,19	21,81	0,16
	PDMR-C	99,53	0,47	0,14	99,17	0,83	0,24	99,66	0,34	0,09	99,28	0,72	0,19
5×10^{-3}	TMR	0	100	0,24	0	100	0,24	0	100	0,18	0	100	0,21
	PDMR-B	0	100	0,58	0	100	1,02	0	100	0,36	0	100	0,80
	PDMR-G	83,58	16,42	0,76	75,67	24,33	1,18	87,18	12,82	0,49	77,13	22,87	0,94
	PDMR-C	98,90	1,10	0,87	97,89	2,11	1,43	99,14	0,86	0,55	98,14	1,86	1,13
10^{-2}	TMR	0	100	0,95	0	100	0,95	0	100	0,71	0	100	0,83
	PDMR-B	0	100	2,27	0	100	3,86	0	100	1,38	0	100	3,05
	PDMR-G	82,68	17,32	2,92	74,05	25,95	4,42	86,40	13,60	1,90	75,60	24,40	3,55
	PDMR-C	97,83	2,17	3,35	95,97	4,03	5,33	98,33	1,67	2,13	96,42	3,58	4,26
2×10^{-2}	TMR	0	100	3,72	0	100	3,72	0	100	2,81	0	100	3,26
	PDMR-B	0	100	8,42	0	100	13,45	0	100	5,26	0	100	10,85
	PDMR-G	81,17	18,83	10,66	71,42	28,58	15,21	85,10	14,90	7,14	73,12	26,88	12,50
	PDMR-C	95,89	4,11	12,13	92,59	7,41	18,05	96,74	3,26	7,93	93,33	6,67	14,77

Fonte: O próprio autor.

Com base nos resultados da Tabela 6.7, pode-se notar que os valores tendem a se aproximar com o decréscimo do valor de P_{b}. Além disso, a TMR e a PDMR-B não possuem capacidade de detecção, de maneira que a mitigação de erros fica a cargo do esquema votador e da seleção de canal, respectivamente. Os resultados para PDMR-C sugerem que a adição do comparador ao projeto lógico impacta significativamente a

porcentagem de erros detectados. Consequentemente, há uma diminuição na quantidade de erros que não são detectados pela PDMR. Desse modo, pode-se concluir que apesar do aumento na quantidade elementos lógicos para a PDMR-C, sua capacidade de diminuir P_{U} e aumentar P_{D} é vantajosa a depender da aplicação, o que pode vir a ser explorado em trabalhos futuros.

Para uma análise comparativa entre a PDMR e a TMR em termos de valores calculados, a equação 2.3 pode ser expandida para caracterizar a probabilidade de sucesso, assumindo a independência entre os eventos de *bitflip*, na transmissão de N bits de informação, de maneira que:

$$R_{\mathrm{TMR_N}} = (p^2(3-2p))^N. \tag{6.3}$$

Desse modo, a equação 6.3, em conjunto com as equações apresentadas no Capítulo 5, pode ser utilizada para comparação entre a TMR e a PDMR no que se refere a configurações não apresentadas neste texto.

6.2.3 Análise de Retransmissão

Com o intuito de comparar os resultados percentuais, levando-se em consideração a possibilidade de retransmissão das mensagens detectadas com erro, as configurações *G32x8*, *C32x8*, *C32x4*, *G24x8*, *C24x8* e *C24x8* foram escolhidas para uma análise mais detalhada. As configurações *G32x4* e *G28x4* foram desconsideradas pelo fato de gerarem uma quantidade de erros não detectados maior que a quantidade correspondente para a *T32* e *T28*.

Os percentuais apresentados na Tabela 6.7 foram usados matematicamente de modo a obter a quantidade de erros totais acumulados a cada transmissão. Para isso, foi utilizada a equação $E_{\mathrm{A}} = T_{\mathrm{r}} \times P_{\mathrm{E}} + U_{\mathrm{T}}$, em que E_{A} é o total de erros acumulados e representa a quantidade de erros totais na transmissão atual somada a quantidade de erros não detectados acumulados ao longo das transmissões; T_{r} é a quantidade de mensagens transmitidas; P_{E} é a percentagem de erros e U_{T} corresponde ao total de mensagens erradas não detectadas ao longo das transmissões. Na primeira transmissão, T_{r} é igual ao número total de mensagens transmitidas (e.g., $T_{\mathrm{r}} = D_{\mathrm{T}} = 10^6$) e $U_{\mathrm{T}} = 0$. Após a primeira transmissão, $T_{\mathrm{r}} = D'$, em que $D' = E' \times P_{\mathrm{D}}$ é a quantidade de mensagens erradas detectadas, obtida da quantidade total de mensagens com erros na transmissão anterior ($E' = T'_{\mathrm{r}} \times P_{\mathrm{E}}$, em que T'_{r} é a quantidade de mensagens transmitidas na transmissão anterior). Nesse caso, $U_{\mathrm{T}} = E' \times P_{\mathrm{U}} + U'_{\mathrm{T}}$, onde U'_{T} é o valor de U_{T} obtido na transmissão anterior. Os resultados dessa avaliação estão dispostos na Tabela 6.8, em que as colunas contêm os valores de $\%E_{\mathrm{A}} = E_{\mathrm{A}} \times 100/D_{\mathrm{T}}$.

Analisando a Tabela 6.8, percebe-se que para as configurações *G32x8*, *C32x8*, *G24x8*, *C24x8* e *C28x4*, uma retransmissão é suficiente para que os percentuais de erro acumulado sejam menores do que uma única transmissão do *T32*, *T24* e *T28*, respectivamente. Uma retransmissão para a TMR não é considerada uma vez que não há capacidade de detecção. Avaliando numericamente, tem-se que uma transmissão única do *T32* implica em 96×10^6 bits transmitidos ($3\,canais \times 32\,bits/canal \times 10^6\,dados$). Enquanto isso, o *G32x8* com $P_{\mathrm{b}} = 2 \times 10^{-2}$ implica em aproximadamente $86,922 \times 10^6$ bits transmitidos

Tabela 6.8: Avaliação de retransmissão em que os valores indicam o percentual da quantidade total de erros acumulados (E_A) com relação ao número total de dados transmitidos (D_T)

Método	**Transmissão**	P_b				
		10^{-3}	2×10^{-3}	5×10^{-3}	10^{-2}	2×10^{-2}
T32	1	0,01	0,04	0,24	0,95	3,72
G32x8	1	0,03	0,12	0,76	2,92	10,66
	2	0	0,02	0,13	0,58	2,93
C32x8	1	0,04	0,14	0,87	3,35	12,13
	2	0	0	0,02	0,18	1,91
C32x4	1	0.06	0,24	1,43	5,33	18,05
	2	0	0	0,05	0,49	4,35
	3	0	0	0,03	0,24	2,07
T24	1	0,01	0,03	0,18	0,71	2,81
G24x8	1	0,02	0,08	0,49	1,90	7,14
	2	0	0,01	0,06	0,29	1,50
C24x8	1	0,02	0,09	0,55	2,13	7,94
	2	0	0	0,01	0,08	0,87
T28	1	0,01	0,03	0,21	0,83	3,26
C28x4	1	0,05	0,19	1,13	4,26	14,77
	2	0	0	0,03	0,33	3,02

Fonte: O próprio autor.

$(2\,canais \times 40\,bits/canal \times (10^6\,dados + P_E \times P_D \times 10^6))$. Seguindo esse mesmo raciocínio, para *C32x8*, tem-se aproximadamente $89,305 \times 10^6$ bits transmitidos considerando os dois canais e a retransmissão. Para a configuração *C32x4*, verifica-se que é necessária uma segunda retransmissão para que o percentual de erro acumulado seja menor do que o da *T32* no caso de $P_b = 2\times10^{-2}$. Sendo assim, a quantidade de bits transmitidos deve ser igual a aproximadamente $86,044 \times 10^6$. Já para o *T24*, tem-se 72×10^6 bits transmitidos e para *G24x8* com $P_b = 2\times10^{-2}$, tem-se aproximadamente $67,889 \times 10^6$ bits transmitidos. No que se refere a *C24x8* para o mesmo valor de P_b, a quantidade de bits transmitidos é igual a aproximadamente $68,910 \times 10^6$. Por fim, para *T28* e para a configuração *C28x4*, são transmitidos 84×10^6 e aproximadamente $72,822 \times 10^6$, respectivamente. Para valores menores de P_b, a diferença entre os números de bits transmitidos é ainda maior, visto que menos bits são retransmitidos pelas implementações PDMR. Esses resultados demonstram, que a TMR implica uma maior ocupação dos canais de transmissão digital.

Uma vez que para *G32x8* e para *G24x8* uma retransmissão é suficiente para obtenção de um percentual de erros menor do que o verificado para *T32* e *T24*, a utilização das configurações *C32x8* e *C24x8* pode ser desconsiderada a depender do projeto, visto que essas configurações apresentam lógica mais complexa caracterizada pela inclusão do comparador. Desse modo, a escolha do esquema de detecção depende da aplicação em que se deseja integrar a PDMR.

6.3 Implementações em *Hardware*

Após as simulações em *software*, a TMR e as diferentes versões da PDMR foram implementadas em *hardware* usando linguagem de descrição de *hardware* (HDL) e incorporadas à interfaces de comunicação AMBA AXI4-Lite. Desse modo, buscou-se comparar as técnicas em termos de consumo de área e energia.

Importante notar que cada canal independente AXI é composto por um conjunto de sinais de informação e de controle para o fluxo de transferência (mecanismo *handshake*) (ARM 2013). A proteção de sinais de controle e *handshake* pode ser efetuada por meio de bits de paridade (Mach et al. 2023), o que simplifica a lógica de implementação, mas requer linhas extra de barramento.

Uma outra opção de proteção, uma vez que na abordagem PDMR são considerados duas interfaces de comunicação redundantes, é permitir a transmissão apenas quando ambos sinais de controle das interfaces têm os mesmos valores lógicos necessários para realização da transferência. Assim, um evento transiente na linha de controle de um dos canais não compromete toda a transmissão. Como o principal objetivo da implementação em *hardware* foi validar a proposta e comparar com a TMR, por simplicidade, a comparação das técnicas foi realizada por meio da aplicação de cada uma delas apenas no barramento de dados do canal de leitura AXI, o qual foi tomado como sendo o caminho de transmissão de dados para a CPU.

Como a AXI4-Lite é uma versão simplificada da AXI4 com menos sinais e sua especificação indica que o barramento de dados deve ter uma largura de 8, 16, 32, 64, 128, 256, 512 ou 1024 bits, as configurações **24x8* e **28x4* foram utilizadas para as implementações em *hardware*. Contudo, as configurações **32x8* e **32x4* podem ser implementadas em interfaces do tipo AXI4, as quais possuem sinais extras (sinais *USER* citados no Capítulo 3) que podem ser utilizados para transmissão dos bits adicionais.

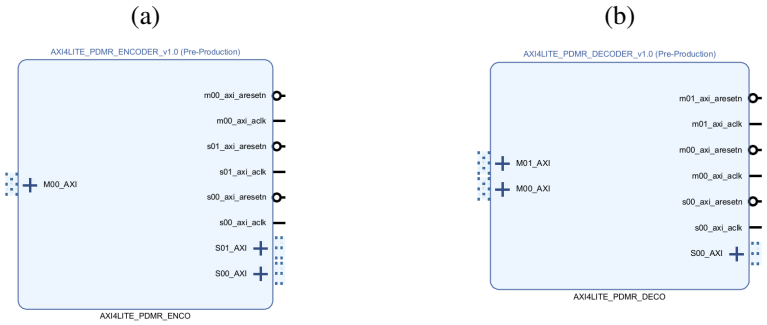
6.3.1 Plataforma de *Hardware* e Sistemas Projetados

As diferentes versões da PDMR foram implementadas e testadas na placa de desenvolvimento Zedboard, que é composta pelo SoC Xilinx Zynq-7000 XC7Z020 que possui um sistema de processamento ARM e lógica reconfigurável integrados no mesmo chip. Alem disso, o *software* Vivado foi utilizado para a síntese e implementação dos módulos de *hardware*.

A ferramenta para criação de IP *core* customizado do Vivado foi usada para criar os blocos para as configurações de TMR e PDMR (AMD-Xilinx Inc. 2023). Essa ferramenta constrói IPs com interfaces AXI4 (Figura 6.4) e gera códigos Verilog para essas interfaces. Assim, os códigos podem ser modificados de acordo com a aplicação. Neste trabalho, foram realizadas apenas pequenas alterações, de modo a manter as características de funcionamento das interfaces geradas.

Para cada configuração, foram gerados dois IPs chamados de *encoder* e *decoder*. O *encoder* incorpora a função apresentada anteriormente para o transmissor, enquanto que o *decoder* funciona como o receptor. No caso da PDMR-B, PDMR-G e PDMR-C, os *encoders* são o mesmo e adicionam as paridades dadas pela lógica da expressão 4.8 aos

Figura 6.4: IPs gerados com auxílio do Vivado (a) codificador e (b) decodificador.



Fonte: Alves et al. (2024).

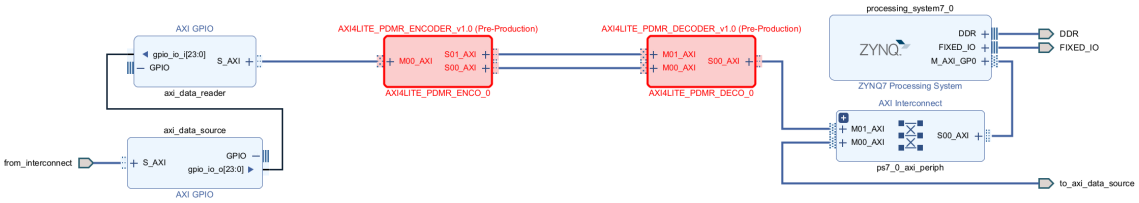
dados lidos por meio da interface mestre. A palavra-código resultante é então duplicada e disponibilizada para as interfaces escravas. Por outro lado, o *encoder* da TMR apenas triplica os dados provenientes da interface mestre.

O IP do *decoder* da PDMR recebe os dados por meio das interfaces mestres, determina a palavra de saída com base nas expressões das seções 4.2, 4.4 e 4.5, correspondendo, respectivamente, às versões PDMR-B, PDMR-G e PDMR-C, e mantém o resultado em um registrador que pode ser lido pela CPU através da interface escrava. O *decoder* da TMR apresenta comportamento similar, mas a palavra de saída é obtida usando votadores baseados na expressão lógica 2.1.

Com base nos *encoders* e *decoders* desenvolvidos, diferentes sistemas foram projetados com o objetivo de comparar a área ocupada e o consumo de energia para as implementações em *hardware* da TMR e da PDMR. Na Figura 6.5, é apresentado o design de blocos para o sistema mais básico, em que se tem uma comunicação entre CPU e periféricos (representado como *axi_data_reader*). Nesse caso, com o auxílio do *AXI Interconnect*, um dado de 24 bits é transferido da CPU para o bloco *axi_data_source*, que por sua vez os envia para o bloco *axi_data_reader*. Os blocos *AXI GPIO* fazem parte do repositório de IP *cores* da Xilinx e foram utilizados por tornarem o desenvolvimento dos testes mais simples e rápido. Posteriormente, a leitura é realizada por meio do *decoder*. Como o AXI4-Lite permite comunicação de mapeamento de memória, os endereços das interfaces *S_AXI* do bloco *axi_data_source* e *S00_AXI* do bloco *decoder*, podem ser acessados pela CPU. Para esse teste, foi desenvolvido um código em linguagem de programação C, o qual foi incorporado ao processador Zynq. O *encoder* e *decoder* foram marcados em vermelho para indicar que são os blocos que foram desenvolvidos. Enquanto isso, os outros blocos, em azul, representam os componentes que emulam a estrutura de transmissão de dados no teste, como CPU e periféricos.

Com o objetivo de verificar o funcionamento do sistema diante de falhas nos dados dos canais de comunicação, foi construído o sistema da Figura 6.6. Nessa figura, alguns blocos foram marcados em vermelho para destacar que são os blocos PDMR desenvolvidos, enquanto os outros, em azul, foram usados para emular a estrutura de injeção de falhas e de transmissão de dados, como a CPU e os periféricos. Os blocos *fault_injector_ch0* e *fault_injector_ch1* foram utilizados para inserir falhas no barramento de leitura de dados

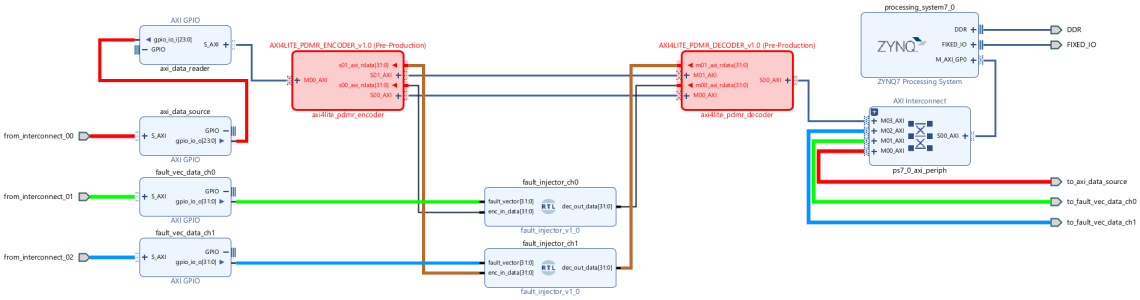
Figura 6.5: Sistema para emulação de transmissão de dados com implementações PDMR (blocos em vermelho)



Fonte: Alves et al. (2024).

dos canais entre os blocos *encoder* e *decoder*. Os vetores de falhas de 32 bits a serem inseridos nos canais, foram enviados pela CPU para os blocos *fault_vec_data_ch0* (linha verde) e *fault_vec_data_ch1* (linha azul). Ademais, o dado de 24 bits a ser lido através do *decoder* também foi repassado para o bloco *axi_data_source*, o qual o transmitiu para o bloco *axi_data_reader* (linha vermelha). Assim como descrito para a Figura 6.5, essa rotina de transferência de dados e falhas, bem como a leitura dos dados via *decoder*, foi realizada por meio de um programa escrito em C e embarcado no processador Zynq.

Figura 6.6: *Design* de blocos para emulação de injeção de falhas usando os blocos PDMR desenvolvidos (marcados em vermelho)

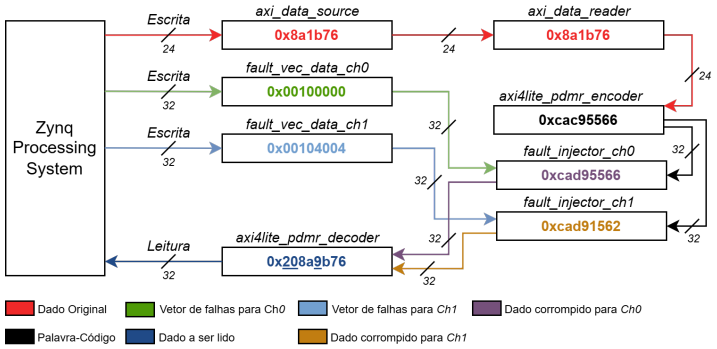


Fonte: Adaptado de Alves et al. (2024).

Um exemplo do fluxo de dados no sistema da Figura 6.6 está ilustrado na Figura 6.7. Inicialmente, o dado 0x8*a*1*b*76, representado em hexadecimal, foi transferido pelo sistema de processamento Zynq para o *axi_data_source*, que o repassou para o bloco *axi_data_reader*. Além disso, os vetores de falhas 0x00100000 e 0x00104004 foram armazenados nos blocos *fault_vec_data_ch0* e *fault_vec_data_ch1* por meio da CPU. Em seguida, a informação foi lida do *axi_data_reader* e ao passar pelo *encoder* foi adicionada a paridade seguindo a regra de formação da palavra código descrita na Seção 4.1. As palavras código duplicadas, ao passarem nos blocos *fault_injector_ch0* e *fault_injector_ch1*, foram corrompidas por meio de uma operação *bitwise XOR* com os respectivos vetores de falhas. Os dados corrompidos provenientes dos blocos *fault_injector_ch0* e *fault_injector_ch1* foram tratados para mitigação de falhas no *decoder*. Observa-se que, ao fim do processo, o valor lido pela CPU foi 0x208*a*9*b*6 e que os dois dígitos hexadecimais mais

significativos, sublinhados na Figura 6.7, correspondem aos bits de indicação de erro 0*b*00100000. Esses bits indicam um erro em algum dos bits de informação no grupo 2. O hexadecimal 0x9, também sublinhado, destaca o erro na mensagem.

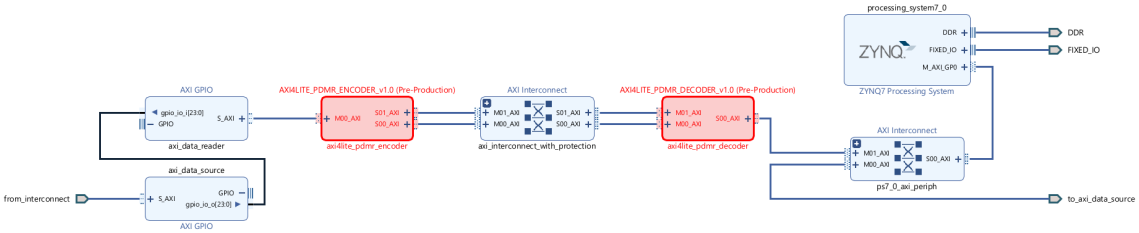
Figura 6.7: Exemplo de fluxo de dados para o teste de injeção de falhas



Fonte: Adaptado de Alves et al. (2024).

O último sistema implementado pode ser observado na Figura 6.8. Nesse sistema, buscou-se adicionar tolerância a falhas ao *AXI Interconnect*, uma vez que, como descrito em (De Sio et al. 2020*b*) e (De Sio et al. 2020*a*), ele pode ser fonte de erros em arquiteturas que utilizam aceleração em *hardware*. O comportamento desse sistema é similar ao que foi descrito para o da Figura 6.5. Contudo, assumiu-se que as falhas podem ocorrer nos caminhos entre as interfaces mestre e escravo do *AXI Interconnect* e do *encoder*, e entre as interfaces mestre e escravo do *AXI Interconnect* e do *decoder*.

Figura 6.8: Ambiente de teste de emulação incluindo redundância na interconexão com blocos PDMR (em vermelho)



Fonte: Adaptado de Alves et al. (2024).

Para os testes da TMR, foram desenvolvidos sistemas como os das Figuras 6.5 e 6.8. As características e modificações das interfaces de comunicação geradas pelo Vivado foram mantidas exatamente iguais para fins de comparação. As diferenças dizem respeito à adição de uma interface mestre a mais para o IP *decoder* da TMR e uma interface escrava a mais para o seu IP *encoder*. Esses IPs foram usados no lugar dos blocos PDMR (marcados em vermelho nas figura) e o *axi_interconnect_with_protection* (Figura 6.8) foi configurado com três interfaces mestre e três interfaces escravas.

6.3.2 Resultados de Consumo de Área e Potência

A Tabela 6.9 contém os resultados da utilização de recursos e estimativa de consumo de energia obtidos da pós-implementação do Vivado. Nessa tabela, estão dispostos os valores de *Slice LUTs*, *Slice Registers* e potência em milliwatts (mW). As colunas % correspondem à relação de redução entre os valores das configurações $*24x8$ em relação à *T24*, obtidas por

$$V_{\%} = 100 \times (1 - V_{\text{PDMR}}/V_{\text{TMR}}), \tag{6.4}$$

em que $V_{\%}$ representa o valor percentual, V_{PDMR} é o valor para as configurações *B24x8*, *G24x8* ou *C24x8*, e V_{TMR} é o valor para a *T24*. Além disso, os valores negativos indicam um aumento de recursos para a *B24x8*, *G24x8* ou *C24x8* com relação à *T24*.

Tabela 6.9: Utilização de recursos e estimativa consumo de energia para as configurações *T24*, *B24x8*, *G24x8* e *C24x8*

Módulo	**Método**	**Recurso**				**Potência**	
		Slice LUTs	%	*Slice Registers*	%	mW	%
Encoder	*T24*	202	–	443	–	2	–
	B24x8	157	22,28	343	22,53	2	0
	G24x8 e *C24x8*	174	13,86	383	13,54	2	0
Decoder	*T24*	162	–	349	–	2	–
	B24x8	145	10,49	288	17,48	1	50
	G24x8	161	0,62	296	15,19	1	50
	C24x8	181	-11,73	296	15,19	2	0
Total [a]	*T24*	364	–	792	–	4	–
	B24x8	302	17,03	631	20,33	3	25
	G24x8	335	7,97	679	14,27	3	25
	C24x8	355	2,47	679	14,27	4	0

[a]Total corresponde à soma dos valores do Encoder e Decoder.

Fonte: O próprio autor.

Com base na Tabela 6.9, percebe-se que o *encoder* utilizado para *B24x8* apresentou uma redução de $22,28\%$ em *Slice LUTs* e uma redução de $22,53\%$ em *Slice Registers* quando comparado com o *T24*. Para as configurações *G24x8* e *C24x8* a redução foi de $13,86\%$ em *Slice LUTs* e de $13,54\%$ em *Slice Registers*. Enquanto isso, para as três configurações PDMR não foi observada redução com relação ao consumo de energia. Considerando os valores para o *decoder*, houve uma diminuição de $10,49\%$ para *B24x8*, $0,62\%$ para *G24x8* e um aumento de $11,73\%$ para *C24x8* no que se refere aos *Slice LUTs*. Esse aumento em *C24x8* pode ser explicado pela inserção da lógica comparadora. Por outro lado, houve uma redução de $17,48\%$ nos *Slices Registers* para a *B*24*x*8 e $15,19\%$ para ambas as configurações *G*24*x*8 e *C*24*x*8. Os valores de potência mostram que, em *B*24*x*8 e em *G*24*x*8, tem-se um consumo 50% menor do que para o *decoder* do T24 e que para *C*24*x*8 o consumo é o mesmo. Contudo, nota-se que a mitigação de erros com PDMR pode envolver a retransmissão de dados detectados com erros, o que significa que o consumo deve sofrer um impacto que depende diretamente da taxa de erros do sistema. Uma análise mais detalhada nesse contexto deve ser considerada em trabalhos futuros.

Dados os valores totais como sendo a soma do encoder e do decoder, verifica-se $17,03\%$, $7,97\%$ e $2,47\%$ *Slice LUTs* a menos para as configurações *B24x8*, *G24x8* e *C24x8*, respectivamente, quando comparados com o total para a *T24*. No caso dos *Slice Registers*, tem-se uma diminuição de $20,33\%$ para a *B24x8*, enquanto ambas as configurações *G24x8* e *C24x8* apresentaram uma diminuição de $14,27\%$ com relação à *T24*. Para o consumo de energia, não houve diferença entre *C24x8* e a *T24*, enquanto *B24x8* e *G24x8* apresentaram um consumo estimado de energia 25% menor.

No que se refere às configurações *T28*, *B28x4*, *G28x4* e *C24x4*, as informações de utilização de recursos de *hardware* e estimativa de consumo energético podem ser encontradas na Tabela 6.10. Nesse caso, observa-se que os valores estimados de consumo energético foram os mesmos para a *T28* e as diferentes versões da PDMR. Ademais, os valores percentuais de redução de *Slice LUTs* e *Slice Registers* são maiores do que aqueles apresentados na Tabela 6.9. Apesar disso, considerando as características de consumo energético e a quantidade total de recursos utilizados pela configuração *C28x4*, optou-se por considerar essa configuração para as demais implementações realizadas em *hardware*.

Tabela 6.10: Utilização de recursos e estimativa de consumo de energia para as configurações *T28*, *B28x4*, *G28x4* e *C28x4*

Módulo	**Método**	**Recurso**				**Potência**	
		Slice LUTs	*%*	*Slice Registers*	*%*	mW	*%*
Encoder	*T28*	226	–	503	–	2	–
	B28x4	170	24,78	367	27,04	2	0
	G28x4 e *C28x4*	178	21,24	387	23,06	2	0
Decoder	*T28*	162	–	361	–	2	–
	B28x4	150	7,41	292	19,11	2	0
	G28x4	162	0	296	18,01	2	0
	C28x4	178	-9,88	296	18,01	2	0
Total [a]	*T28*	388	–	864	–	4	–
	B28x4	320	17,53	659	23,73	4	0
	G28x4	340	12,37	683	20,95	4	0
	C28x4	356	8,25	683	20,95	4	0

[a]Total corresponde à soma dos valores do Encoder e Decoder.

Fonte: O próprio autor.

Os resultados de utilização de recursos e consumo de energia para o teste da Figura 6.8 pode ser encontrado na Tabela 6.11. Nesse caso, o *Interconnect* foi considerado como parte do caminho de transmissão de dados. Desse modo, os seus valores de *Slice LUTs*, *Slice Registers* e potência foram incluídos como parte da soma total. Os valores na Tabela 6.11 mostram que as porcentagens de *Slice Registers* para o *encoder*, *decoder* e sua soma se mantiveram iguais da Tabela 6.10. Contudo, para o *Interconnect*, houve um aumento de $1,72\%$ nos *Slice Registers*. Tomando o total como sendo a soma dos valores do *encoder*, *decoder* e *Interconnect*, houve uma diminuição de $18,27\%$ nos *Slice Registers*. No que se refere aos *Slice LUTs*, verifica-se percentuais próximos aos da Tabela 6.10 e destaca-se uma maior redução quando considerado o valor total. Além disso, foi observada uma redução no consumo estimado de energia, chegando a um valor total de 25%.

Tabela 6.11: Utilização de recursos e estimativa de consumo de energia para as configurações *T*28 e *C28x4* com *AXI Interconnect*

Módulo	**Método**	**Recurso**				**Potência**	
		Slice LUTs	*%*	*Slice Registers*	*%*	mW	*%*
Encoder	*T28*	244	–	503	–	2	–
	C28x4	178	27,05	387	23,06	1	50
Decoder	*T8*	161	–	361	–	1	–
	C28x4	178	-10,56	296	18,01	1	0
Interconnect	*T28*	169	–	116	–	1	–
	C28x4	131	22,49	118	-1,72	1	0
Encoder + Decoder	*T28*	405	–	864	–	3	–
	C28x4	356	12,10	683	20,10	2	33,33
Total [a]	*T28*	574	–	980	–	4	–
	C28x4	487	15,16	801	18,27	3	25

[a]Total corresponde à soma dos valores do Encoder, Decoder, e Interconnect.

Fonte: O próprio autor.

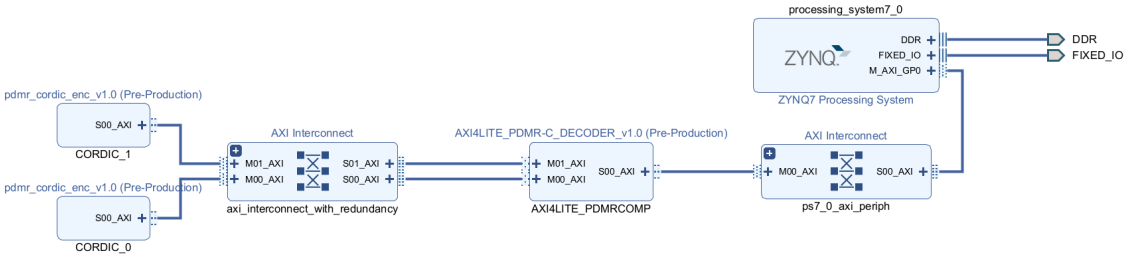
Diante dos resultados apresentados nessa subseção, destaca-se que as interfaces AXI4-Lite geradas pelo Vivado são compostas por diversos sinais da especificação do AXI4 cujos comportamentos são implementados pela ferramenta, principalmente, com registradores. Em outras palavras, as interfaces básicas, sem aplicação de método de tolerância, possuem mais registradores do que LUTs. Isso pode explicar o maior impacto da proposta nos *Slice Registers*. Enquanto isso, as lógicas combinacionais da PDMR-G e PDMR-C são mais complexas do que o votador clássico (2.1), o que justifica um uso maior de LUTs, embora o bloco *decoder* da TMR possua uma interface mestre a mais. Contudo, ainda foi possível chegar a valores totais menores para ambas configurações PDMR. Considerando o consumo de energia e uma menor utilização de recursos para o *decoder*, as configuração *G24x8* é uma boa opção de implementação, embora representem uma redução na capacidade de detecção. Ademais, com base na Tabela 6.11, assume-se que a inclusão do método proposto em conjunto com estruturas mais elaboradas como o *Interconnect* pode significar uma maior redução no uso de recursos e no consumo de energia.

Vale salientar que a técnica proposta foi comparada com a TMR em interfaces básicas como forma de verificar a área ocupada e o consumo de energia em um nível mais básico de aplicação. Contudo, técnicas de redundância envolvem, geralmente, a replicação de componentes de aceleração em *hardware* com tarefas mais robustas e que ocupam parte considerável da lógica reconfigurável do FPGA. Por exemplo, um acelerador de hardware para compressão de imagens pode ser triplicado com TMR e os dados resultantes das operações podem ser votados com o circuito votador. No caso da PDMR, haveria a necessidade de duplicar a aplicação, o que já representaria uma redução na área ocupada, e de adicionar a lógica combinacional do *encoder* (formação da palavra-código), que pode ser embutido na interface de comunicação de cada elemento redundante. Na próxima seção são representados os resultados de implementação para um estudo de caso que corrobora com a afirmação apresentada nesse parágrafo.

6.4 Estudo de Caso com Elementos Redundantes

Com o intuito de analisar a integração da técnica PDMR em elementos redundantes, foi desenvolvido um IP *core* baseado no algoritmo CORDIC (*COordinate Rotation DIgital Computer*). O código Verilog de implementação do CORDIC pode ser encontrado em VLSI Universe (2021). Essa implementação permite o cálculo do cosseno e seno com 16 bits de precisão. Desse modo, utilizou-se a ferramenta do Vivado para criação de IP *cores* com interfaces AXI4-Lite. O IP *core* criado contém registradores que são acessados pelo sistema de processamento Zynq (CPU) por meio do mapeamento de memória do AXI4-Lite. Os registradores são utilizados na configuração do CORDIC, como, por exemplo, para a definição dos valores das coordenadas do ponto inicial e do ângulo para o cálculo do cosseno e do seno. Os resultados dos cálculos são armazenados em registradores que podem então ser lidos pela CPU. Ademais, os resultados são codificados com a lógica da PDMR, de maneira que o esquema de proteção se encontra incorporado ao CORDIC. Ou seja, tomando como base a arquitetura da Figura 4.16, os codificadores se encontram encapsulados nos blocos CORDIC (componentes de *hardware*). O diagrama do sistema projetado para o teste com elementos redundantes está ilustrado na Figura 6.9.

Figura 6.9: Implementação para teste de integração da técnica PDMR com elementos redundantes (CORDIC)



Fonte: O próprio autor.

Na Figura 6.9, pode-se observar que a interconexão entre os componentes redundantes (*CORDIC_0* e *CORDIC_1*) foi realizada por um bloco *AXI Interconnect*, de modo que as interfaces de comunicação replicadas representam caminhos redundantes para os dados. Durante os testes, foi utilizado o *decoder* com comparador. Sendo assim, buscou-se comparar a *T28* com a configuração PDMR *C28x4*. Os resultados de utilização de recursos de *hardware* e estimativa de consumo energético pode ser encontrados na Tabela 6.12. Importante notar que para a *T28* foram utilizados três IP *cores* CORDIC, sendo o *AXI Interconnect* configurado com três interfaces mestre e três interfaces escravas, enquanto foi utilizado o *decoder* para a TMR, o qual tem como base votadores elaborados com a lógica da expressão 2.1.

Os valores da Tabela 6.12 indicam que o uso de dois módulos CORDICs para a PDMR em comparação com os três necessários para a TMR gerou uma redução de $33,13\%$ e $33,10\%$ na utilização de *Slice LUTS* e *Slice Registers*, respectivamente. Com relação

Tabela 6.12: Utilização de recursos e estimativa de potência com CORDICs

Módulo	**Método**	**Recurso**				**Potência**	
		Slice LUTs	*%*	*Slice Registers*	*%*	mW	*%*
CORDICs	*T28*	3287	–	3363	–	15	–
	C28x4	2198	33,13	2250	33,10	10	33,33
Decoder	*T28*	161	–	361	–	2	–
	C28x4	176	-9,32	296	18,01	2	0
Interconnect	*T28*	168	–	116	–	1	–
	C28x4	131	22,02	118	-1,72	1	0
CORDICs + Decoder	*T28*	3448	–	3724	–	17	–
	C28x4	2374	31,15	2546	31,63	12	29,41
Total [a]	*T28*	3616	–	3840	–	18	–
	C28x4	2505	30,72	2664	30,62	13	27,78

[a]Total corresponde à soma dos valores dos CORDICs, Decoder, e Interconnect.

Fonte: O próprio autor.

ao consumo de energia, houve uma diminuição de $33,33\%$. Já para o *decoder* e para o *Interconnect* os valores ficaram próximos dos observados nas Tabelas 6.10 e 6.11. Considerando o total como a soma dos valores dos CORDICs, do *decoder* e do *Interconnect*, a redução total na utilização de recursos foi de $30,72\%$ para os *Slice LUTs* e $30,62\%$ para os *Slice Register*, com consumo energético $27,78\%$ menor em comparação com a técnica TMR.

Em última análise, com base nos relatórios de pós-implementação gerados pelo Vivado, foi verificado o tempo de atraso para os sistemas CORDIC com TMR e com PDMR. Para o sistema projetado com *T28*, foi reportado caminho crítico com 5 níveis e atraso total de $7,467\,ns$. Por outro lado, para o sistema com *C28x4* foi observado caminho crítico com 4 níveis e atraso total de $6,853\,ns$. Desse modo, ambos os sistemas cumpriram o requisito de período de *clock* de $10\,ns$. Contudo, a implementação com PDMR apresentou um atraso total menor.

6.5 Comparações Adicionais

Embora este trabalho não tenha como objetivo principal uma comparação com códigos FEC, uma vez que a abordagem proposta é considerada como mais próxima à métodos que envolvem mascarar falhas e a possibilidade de retransmissão, a análise realizada por Mach et al. (2023) foi tomada como base para resumir as propriedades da PDMR, da TMR e de FECs. Esse resumo é apresentado na Tabela 6.13. Vale salientar que os autores fazem a análise considerando técnicas de redundância de informação, como paridade e SECDED. Considerando que os FEC fazem parte desse tipo de redundância, a tabela foi adaptada para designar suas características.

A primeira propriedade na Tabela 6.13 está relacionada com a ocorrência de eventos transientes em múltiplas linhas do barramento. No caso dos esquemas FEC, como o SEC e SECDED, como há a identificação da posição em que ocorreu um erro por meio da síndrome, múltiplas falhas (*glitches*) concorrentes podem acabar não detectados pela lógica do decodificador, como por exemplo quando as falhas transientes ocorrem em vários fios

Tabela 6.13: Características de diferentes métodos

Propriedade/Característica	**Método**		
	PDMR	TMR	FEC
Falhas concorrentes em múltiplas linhas do barramento podem não ser detectadas	Parcialmente	Parcialmente	Sim
Impacta a frequência de *clock* do barramento	Parcialmente	Parcialmente	Parcialmente
Aumenta a área da interconexão	Sim	Sim	Parcialmente
Requer alterações no gerente (mestre)	Parcialmente	Parcialmente	Sim
Requer alterações nos subordinados (escravos)	Parcialmente	Não	Sim
Implementação Simples	Sim	Sim	Não
Algumas transações podem necessitar de mais transferências	Sim	Não	Sim

Fonte: Adaptado de Mach et al. (2023)

protegidos pelo mesmo bit de paridade. Para a TMR, o efeito é parcial, pois a replicação dos canais faz com que não ocorra correção apenas se as falhas atingirem a mesma linha de barramento de dois canais distintos. Caso um canal sofra múltiplas falhas, o votador de maioria garante a correção a partir dos bits dos outros canais. A lógica da PDMR tem comportamento parecido com a da TMR no sentido da replicação dos canais. Contudo, a divisão da palavra-código em grupos também faz com que as falhas (*glitches*) em grupos distintos possam ser mascaradas ou detectadas.

Todas as técnicas podem impactar a frequência de *clock*, principalmente devido aos codificadores e decodificadores adicionados nos canais para a PDMR e para o FEC. No caso da TMR, o impacto é proveniente do circuito votador. Ademais, tanto a TMR quanto a PDMR têm um maior impacto na área da interconexão (interfaces de comunicação) quando comparados com códigos FEC simples. Contudo, a PDMR apresenta um impacto reduzido com relação à TMR, o qual pode ser ainda menor se considerada a aplicação em elementos redundantes, como discutido na Seção 6.3.

Neste ponto, considera-se gerente como sendo o componente responsável por iniciar a transferência dos dados, como por exemplo o processador. Enquanto isso, os subordinados são os componentes que recebem ou mandam informações para o gerente, como por exemplo os periféricos. No que se refere à execução de mudanças no gerente (mestre) e nos subordinados (escravos), a redundância de informação (códigos FEC) requer que elas sejam realizadas pelo fato de serem necessárias as implementações das lógicas do codificador no mestre e do decodificador nos escravos. Os códigos FEC podem necessitar, a depender da largura do dado de entrada, de linhas extras de barramento para que o dado e os bits redundantes possam ser transferidos, de modo que essas linhas devem ser incluídas no projeto das interfaces de comunicação. Já a TMR requer uma alteração parcial no mestre, caso o votador seja utilizado ao seu lado para votar dados provenientes de escravos replicados. Nesse caso, os escravos não precisam sofrer mudanças. Na PDMR, são necessárias alterações no mestre e no escravo para inclusão das lógicas de codificação e decodificação. Contudo, dependendo do tamanho do dado de entrada, a inclusão de linhas extras de barramento é descartada.

Os codificadores e decodificadores da PDMR envolvem, principalmente, o uso de ope-

rações *XOR* para o cálculo da paridade e bit comparador (Capítulo 4), caso seja usado, e de *MUX* para seleção do canal. Por outro lado, apesar do codificador do SEC e SECDED ser relativamente simples (usa operações lógicas *XOR*), seu decodificador pode ser composto por operações *XOR* e *flip-flops* para armazenar resultados (Desset & Fort 2003), o que torna sua implementação mais complexa. Considerando as diferenças de complexidade, os códigos FEC foram classificados como de implementação mais complexa e a PDMR e TMR como de simples implementação. Com base nisso, as mudanças no gerente e nos subordinados foram classificadas como *Parcialmente* para a PDMR.

Por fim, verifica-se que a PDMR, assim como códigos FEC, podem requerer várias transferências para uma transação. Isso pode ocorrer no caso em que o processador (sistema) só pode enviar uma quantidade limitada de bits, de modo que para uma única transação são necessárias várias transferências para que dado e os bits redundantes sejam enviados. Como na TMR não há bits extras adicionados ao dado, a transação pode ocorrer de maneira simples.

Importante notar que apesar a TMR apresenta melhores características na comparação da Tabela 6.13. Contudo, seu alto *overhead* de área e energia pode ser um problema em aplicações com restrições de projeto, como os sistemas de bordo em nanossatélites. Como demonstrado, a PDMR apresenta um menor consumo de energia e uma menor área, que pode ser ainda menor se for desconsiderada a lógica de detecção/comparação (versão PDMR-B), o que pode acontecer em aplicações em que a taxa de erro é muito pequena. A possibilidade de detecção também é uma propriedade interessante da PDMR com relação à TMR. Essa propriedade permite que possíveis erros sejam indicados nos grupos do dado transmitido, de modo que ações de retransmissão podem ser efetuadas por implementações via *hardware* ou *software*. Além disso, ressalta-se que a divisão em grupos dá margem para implementações em que apenas os grupos com indicação de erro são retransmitidos, cujo impacto deve ser explorado em trabalhos futuros. No que se refere aos códigos FEC, a PDMR leva vantagem com relação à facilidade de implementação e à possibilidade de mascarar falhas em múltiplos fios, enquanto outras características podem ser dependentes da aplicação.

6.6 Resumo

Neste capítulo, foram apresentados os experimentos de *software* e *hardware* desenvolvidos para validar a abordagem proposta. Inicialmente, foram descritos os resultados das simulações em *software* utilizadas para comparação com os valores de probabilidades obtidos por meio das equações desenvolvidas no Capítulo 5. A análise comparativa permite concluir que os valores calculados estão próximos daqueles observados nas simulações, uma vez que o erro quadrático médio (MSE) no pior dos casos é da ordem de 10^{-7} e a normalização da raiz do erro quadrático médio (NRMSE) pelo desvio padrão dos valores simulados gera resultados próximos à 1.

Dada a proximidade entre os valores calculados e simulados, tomou-se como base os arquivos gerados nas simulações para comparação entre a capacidade de correção da TMR e da PDMR. Os resultados mostram que, com a diminuição da probabilidade de *bitflips* há uma redução nas porcentagens de erros com relação ao total de dados transmitidos. Sendo

assim, para valores de P_b muito pequenos, a utilização da versão PDMR-B, pode ser atrativa, pois ela apresenta uma lógica de implementação menos complexa. Nesse sentido, a versão PDMR-G pode ser interessante em situações em que se deseja aplicar detecção de erros e se deseja evitar a complexidade lógica imposta pela versão PDMR-C. A característica de detecção da PDMR permite que sejam realizadas retransmissões dos erros detectados, o que minimiza a quantidade de erros acumulados e, para algumas configurações, gera percentuais de erros menores do que a TMR com apenas uma retransmissão. Nesses casos, as configurações com comparador são mais eficazes, embora representem um aumento na lógica implementada. Na verdade, para configurações com menos bits de paridade, como **32x4* e **28x4* é essencial a aplicação da versão com comparador para que se tenha um percentual de erros menor do que a TMR, considerando as retransmissões.

Vale salientar que as implementações em *hardware* foram aplicadas na placa de desenvolvimento Zedboard e fundamentadas no AMBA AXI4-Lite. Para as configurações testadas, observou-se uma redução na utilização de recursos de *hardware* quando analisados os valores totais. Enquanto isso, o consumo estimado de energia foi menor para as configurações *B24x8* e *G24x8*, de maneira que não houve redução para as demais configurações implementadas. Ademais, verificou-se uma diminuição na utilização de recursos e no consumo de energia para o sistema com redundância no *AXI Interconnect*.

Implementações em hardware podem depender da tecnologia e protocolos de comunicação escolhidos. Nesse sentido, assim como destacado por Lazaro *et al.* (Lázaro et al. 2024), outras abordagens são específicas para definição de outras interfaces, o que a torna difícil a realização de comparações. A abordagem baseada em TMR descrita por (Lázaro et al. 2024) difere da proposta apresentada nesta tese, principalmente, nos seguintes aspectos: os autores realizaram modificações nos cinco canais AXI para implementação do mecanismo de redundância; foi desenvolvido um único módulo, que seria equivalente às implementações de *decoder*. A comparação em termos de ocupação se torna difícil por não serem apresentados detalhes relacionados à minimização dos recursos das interfaces AXI. Além disso, um dos focos dos autores foi a minimização dos recursos do FPGA, enquanto as implementações de *hardware* da PDMR foram desenvolvidas com objetivo de verificar a viabilidade da abordagem proposta com relação à TMR, considerando modificações mínimas na estrutura das interfaces geradas pelo Vivado. Sendo assim, uma comparação direta envolveria uma maior minimização dos modelos e uma ampliação da aplicação da técnica nos demais canais AXI.

Em suma, os resultados da integração da PDMR com elementos redundantes confirmam uma diminuição considerável no uso dos recursos em comparação com a TMR. Além disso, nas implementações com CORDIC, o sistema com TMR apresentou caminho crítico maior e, consequentemente, um atraso total maior comparado ao sistema com PDMR.

Capítulo 7

Conclusão

Considerando o uso de SoCs COTS em missões com nanossatélites e a necessidade de aumentar a confiabilidade na transmissão de dados nesses dispositivos, neste trabalho foi proposta uma abordagem DMR baseada em paridade (PDMR). Desse maneira, buscou-se uma solução de baixa complexidade lógica capaz de mitigar os efeitos provenientes da radiação em ambiente espacial.

A técnica PDMR tem como fundamento a duplicação da estrutura de comunicação e a aplicação de bits de paridade. Os bits de informação do dado a ser transmitido são divididos em grupos de tamanho fixo e a cada grupo é associado um bit de paridade. Com isso, o dado de saída é formado com base em bits de informação provenientes de grupos distintos.

Diferentes versões da técnica proposta foram apresentadas neste texto. A versão mais básica (PDMR-B) permite a seleção dos bits de informação de cada grupo por meio da verificação da paridade após passagem em apenas um dos canais, de modo que não há detecção de erro. Na versão PDMR-G, tem-se uma estrutura lógica com bit de indicação de erro, o qual pode ser utilizado para detecção de possíveis erros nos dados transmitidos. Enquanto isso, a versão mais complexa, chamada PDMR-C, inclui uma lógica de comparação para incrementar a capacidade de detecção da técnica proposta.

Para cada uma das versões PDMR, foram formuladas equações que descrevem suas probabilidades de correção e de erro. Ademais, para as versões PDMR-G e PDMR-C, também foram demonstradas equações referentes à probabilidade de detecção. Os valores calculados com a utilização das equações desenvolvidas foram comparados com resultados de simulações, sendo observado um erro quadrático médio na ordem de 10^{-7} para o pior dos casos.

Os experimentos de comparação da abordagem proposta com a TMR foram, inicialmente, realizados por meio de simulações desenvolvidas em *scripts* Python. Os resultados mostram que o percentual de erros totais para as configurações PDMR se aproximam da TMR com a diminuição da probabilidade de *bitflips* (P_b). Adicionalmente, considerando a possibilidade de detecção nas configurações PDMR, é possível reduzir o número de erros acumulados por meio de retransmissões, mantendo a quantidade total de bits transmitidos menor do que implementações com a TMR.

Após as simulações em *software*, implementações em *hardware* foram desenvolvidas com auxílio do SoC Xilinx Zynq-7000 XCTZ020 e comparada com a TMR em termos de utilização de recurso de *hardware* e consumo energético. Os testes se basearam na

aplicação das diferentes versões PDMR no barramento de leitura de dados de interfaces AXI-Lite simples. Nesse contexto, a abordagem proposta apresentou uma redução na utilização de *Slice LUTs* e *Slice Registers* quando considerados os valores totais das implementações. Além disso, foi possível observar um menor consumo de energia para algumas das configurações PDMR, enquanto outras apresentaram o mesmo consumo da TMR. Notadamente, a redução nos valores de utilização de recursos de hardware se mostrou ainda maior em estruturas de comunicação mais complexas, como aquelas com *AXI Interconnect* e com elementos redundantes.

As análises realizadas para a caracterização da técnica proposta, desde a capacidade de correção, da possibilidade de retransmissão e da aplicação em componentes de aceleração redundantes, tornam a PDMR uma alternativa para mascarar/corrigir falhas na transmissão de dados em sistemas de processamento de nanossatélites. Tais sistemas apresentam restrições potência, peso, tamanho e custo, de maneira que é possível aumentar a confiabilidade do sistema com uma diminuição na utilização de recursos e aplicação de soluções lógicas de baixa complexidade, quando comparadas com códigos de correção modernos.

7.1 Trabalhos Futuros

Como proposições de trabalhos futuros, tem-se:

- Analisar a integração da proposta PDMR em interfaces AXI e AXI-Stream, assim como em outros protocolos já estabelecidos;
- Desenvolver uma solução em *hardware* para retransmissão automática dos dados detectados com erro;
- Avaliar o impacto da retransmissão considerando apenas os grupos com indicação de erro;
- Caracterizar a técnica PDMR no que se refere à ocorrência de *bitflips* na área ocupada por componentes de aceleração em *hardware*.

7.2 Publicação Relacionada

A abordagem proposta neste texto de tese e os resultados obtidos nos experimentos de *software* e *hardware* foram utilizados como base para elaboração do seguinte artigo de revista:

- Alves, Alex C. R., Silveira, Luiz F. Q., Kreutz, Márcio E., Dias, Samaherni M. *A Parity-Based Dual Modular Redundancy Approach for the Reliability of Data Transmission in Nanosatellite's Onboard Processing*. IEEE Access, vol. 12, pp. 90815-90828, 2024, doi: 10.1109/ACCESS.2024.3421608.

Referências Bibliográficas

Abderrahmane, Nassim, Benoît Miramond, Erwann Kervennic & Adrien Girard (2022), Spleat: Spiking low-power event-based architecture for in-orbit processing of satellite imagery, *em* '2022 International Joint Conference on Neural Networks (IJCNN)', pp. 1–10.

Abdullah, Haythem Hussein, Ayman Elboushi, Ahmed Essam Gohar & Esmat A. Abdallah (2021), 'An improved s-band cubesat communication subsystem design and implementation', *IEEE Access* **9**, 45123–45136.

Acosta, Braulio Peraza, Eduardo Angel Guzmán, Jorge Alberto Osuna Valdez, Luis Rizo Domínguez, Mario Mendoza Barcenas & M. I. Rafael Prieto Meléndez (2022), Adaptable o-board computer for nanosatellites, *em* '2022 IEEE 13th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)', pp. 0421–0425.

Adams, Caleb, Allen Spain, Jackson Parker, Matthew Hevert, James Roach & David Cotten (2019), Towards an integrated gpu accelerated soc as a flight computer for small satellites, *em* '2019 IEEE Aerospace Conference', pp. 1–7.

Alves, Alex C. R., Luiz F. Q. Silveira, Márcio E. Kreutz & Samaherni M. Dias (2024), 'A parity-based dual modular redundancy approach for the reliability of data transmission in nanosatellite's onboard processing', *IEEE Access* **12**, 90815–90828.

AMD-Xilinx Inc. (2023), 'Vivado design suite user guide: Creating and packaging custom ip', `https://docs.amd.com/r/en-US/ug1118-vivado-creating-packaging-custom-ip/Creating-and-Packaging-Custom-IP`. Acesso em: 17 ago. 2024.

Andina, J.J.R., E. de la Torre-Arnanz & M.D.V. Peña (2017), *FPGAs: Fundamentals, Advanced Features, and Applications in Industrial Electronics*, CRC Press.

ARM (2011), 'Cortex^{TM}-R5 Technical Reference Manual, revision: r1p2', `https://developer.arm.com/documentation/ddi0460/latest/`. Acesso em: 05 ago. 2024.

ARM (2013), 'Amba axi and ace protocol specification axi3, axi4, and axi4-lite ace and ace-lite', `https://developer.arm.com/documentation/ihi0022/e`. Acesso em: 06 ago. 2024.

ARM (2021), 'Amba axi and ace protocol specification', `https://documentation-service.arm.com/static/602a9df190ee6824a1e02b98?token=`. Acesso em: 03 jun. 2021.

Asadi, G.-H. & M.B. Tahoori (2005), Soft error mitigation for sram-based fpgas, *em* '23rd IEEE VLSI Test Symposium (VTS'05)', pp. 207–212.

Avizienis, A., J.-C. Laprie, B. Randell & C. Landwehr (2004), 'Basic concepts and taxonomy of dependable and secure computing', *IEEE Transactions on Dependable and Secure Computing* **1**(1), 11–33.

Bahl, Gaetan, Lionel Daniel, Matthieu Moretti & Florent Lafarge (2019), Low-power neural networks for semantic segmentation of satellite images, *em* '2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)', pp. 2469–2476.

Balasubramanian, P., K. Prasad & Nikos E. Mastorakis (2016), 'A fault tolerance improved majority voter for TMR system architectures', *WSEAS Trans. Circuits Syst.* **15**, 108–122.

Bekker, Dmitriy L., Thomas A. Werne, Thor O. Wilson, Paula J. Pingree, Kiril Dontchev, Michael Heywood, Rafael Ramos, Brad Freyberg, Fernando Saca, Brian Gilchrist, Alec Gallimore & James Cutler (2010), A cubesat design to validate the virtex-5 fpga for spaceborne image processing, *em* '2010 IEEE Aerospace Conference', pp. 1–9.

Benevenuti, Fabio & Fernanda Lima Kastensmidt (2018), Reliability evaluation on interfacing with axi and axi-s on xilinx zynq-7000 ap-soc, *em* '2018 IEEE 19th Latin-American Test Symposium (LATS)', pp. 1–6.

Bertozzi, D., L. Benini & G. De Micheli (2005), 'Error control schemes for on-chip communication links: the energy-reliability tradeoff', *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* **24**(6), 818–831.

Bezerra, F., D. Dangla, F. Manni, J. Mekki, D. Standarovski, R. G. Alia, M. Brugger & S. Danzeca (2017), Evaluation of an alternative low cost approach for see assessment of a soc, *em* '2017 17th European Conference on Radiation and Its Effects on Components and Systems (RADECS)', pp. 1–5.

Borecký, Jaroslav, Martin Kohlík, Pavel Vít & Hana Kubátová (2016), Enhanced duplication method with tmr-like masking abilities, *em* '2016 Euromicro Conference on Digital System Design (DSD)', pp. 690–693.

Boudenot, Jean Claude (2007), Radiation space environment, *em* R.Velazco, P.Fouillat & R.Reis, eds., 'Radiation Effects on Embedded Systems', Springer Netherlands, Dordrecht, pp. 1–9.

Brosser, Fredrik, Emil Milh, Vilhelm Geijer & Per Larsson-Edefors (2014), Assessing scrubbing techniques for xilinx sram-based fpgas in space applications, *em* '2014 International Conference on Field-Programmable Technology (FPT)', pp. 296–299.

Bruhn, Fredrik, Kjell Brunberg, John Hines, Lars Asplund & Magnus Norgren (2015), Introducing radiation tolerant heterogeneous computers for small satellites, *em* '2015 IEEE Aerospace Conference', pp. 1–10.

Ceschia, M., M. Violante, M.S. Reorda, A. Paccagnella, P. Bernardi, M. Rebaudengo, D. Bortolato, M. Bellato, P. Zambolin & A. Candelori (2003), 'Identification and classification of single-event upsets in the configuration memory of sram-based fpgas', *IEEE Transactions on Nuclear Science* **50**(6), 2088–2094.

Clark, Lawrence T., Dan W. Patterson, Nathan D. Hindman, Keith E. Holbert, Satendra Maurya & Steven M. Guertin (2011), 'A dual mode redundant approach for microprocessor soft error hardness', *IEEE Transactions on Nuclear Science* **58**(6), 3018–3025.

Coca, Mihai & Mihai Datcu (2023), 'Fpga accelerator for meta-recognition anomaly detection: Case of burned area detection', *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* **16**, 5247–5259.

De Sio, C., S. Azimi & L. Sterpone (2020*a*), On the evaluation of seu effects on axi interconnect within ap-socs, *em* A.Brinkmann, W.Karl, S.Lankes, S.Tomforde, T.Pionteck & C.Trinitis, eds., 'Architecture of Computing Systems – ARCS 2020', Springer International Publishing, Cham, pp. 215–227.

De Sio, C., S. Azimi & L. Sterpone (2020*b*), 'On the analysis of radiation-induced failures in the axi interconnect module', *Microelectronics Reliability* **114**, 113733. 31st European Symposium on Reliability of Electron Devices, Failure Physics and Analysis, ESREF 2020.

De Sio, Corrado, Sarah Azimi, Luca Sterpone & Boyang Du (2019), 'Analyzing radiation-induced transient errors on sram-based fpgas by propagation of broadening effect', *IEEE Access* **7**, 140182–140189.

Desset, C. & A. Fort (2003), Selection of channel coding for low-power wireless systems, *em* 'The 57th IEEE Semiannual Vehicular Technology Conference, 2003. VTC 2003-Spring.', Vol. 3, pp. 1920–1924 vol.3.

Dong, Zhen, Yanning Guo, Youmin Gong & Chuanjiang Li (2016), A high reliability radiation hardened on-board computer system for space application, *em* '2016 Sixth International Conference on Instrumentation Measurement, Computer, Communication and Control (IMCCC)', pp. 671–674.

Edmonds, LD, CE Barnes & LZ Scheick (2000), An introduction to space radiation effects on microelectronics, Relatório Técnico JPL00-06, Jet Propulsion Laboratory, National Aeronautics and Space Administration, Washington, DC, USA.

Estela, Jaime & Sergio Montenegro (2020), On-board computer and data handling, *em* R. A.de Carvalho, J.Estela & M.Langer, eds., 'Nanosatellites: Space and Ground

Technologies, Operations and Economics', John Wiley & Sons, Ltd, capítulo 2, pp. 31–50.

Faraj, P., J. Leibrich & W. Rosenkranz (2003), Coding gain of basic fec block-codes in the presence of ase noise, *em* 'Proceedings of 2003 5th International Conference on Transparent Optical Networks, 2003.', Vol. 2, pp. 80–83 vol.2.

Fjeldtvedt, Johan, Milica Orlandić & Tor Arne Johansen (2018), 'An efficient real-time fpga implementation of the ccsds-123 compression standard for hyperspectral images', *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* **11**(10), 3841–3852.

Fortescue, Peter, Graham Swinerd & John Stark (2011), *Spacecraft Systems Engineering*, 4ª edição, John Wiley & Sons, Ltd.

Franklin, Garth, Stephan Esterhuizen, Chad Galley, Byron Iijima, Kameron Larsen, Myron Lee, Jehhal Liu, Thomas Meehan & Larry Young (2018), A GNSS receiver for small-sats enabling precision POD, radio occultations, and reflections, *em* T. S.Pagano & C. D.Norton, eds., 'CubeSats and NanoSats for Remote Sensing II', Vol. 10769, International Society for Optics and Photonics, SPIE, p. 1076905.

Fuchs, Christian M., Nadia M. Murillo, Aske Plaat, Erik van der Kouwe, Daniel Harsono & Todor P. Stefanov (2018), Fault-tolerant nanosatellite computing on a budget, *em* '2018 18th European Conference on Radiation and Its Effects on Components and Systems (RADECS)', pp. 1–8.

George, Alan D. & Christopher M. Wilson (2018), 'Onboard processing with hybrid and reconfigurable computing on small satellites', *Proceedings of the IEEE* **106**(3), 458–470.

Goloubeva, Olga, Maurizio Rebaudengo, Matteo Sonza Reorda & Massimo Violante (2006), *Software-implemented hardware fault tolerance*, Springer US.

Gracia-Morán, Joaquín, Luis J. Saiz-Adalid, Daniel Gil-Tomás & Pedro J. Gil-Vicente (2018), 'Improving error correction codes for multiple-cell upsets in space applications', *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* **26**(10), 2132–2142.

Graham, Derek, Per Strid, Scott Roy & Fernando Rodriguez (2009), A low-tech solution to avoid the severe impact of transient errors on the ip interconnect, *em* '2009 IEEE/IFIP International Conference on Dependable Systems & Networks', pp. 478–483.

Heidt, H., J. Puig-Suari, A. Moore, S. Nakasuka & R. Twiggs (2000), Cubesat: A new generation of picosatellite for education and industry low-cost space experimentation, *em* 'Proceedings of the Thirteenth Annual AIAA/USU SmallSatellite Conference'.

Henjes, Christian, Holger Kügler, Wilfried Ley, Manfred Magg & Steffen Scharfenberg (2009), The space environment, *em* W.Ley, K.Wittmann & W.Hallmann, eds., 'Handbook of Space Technology', John Wiley & Sons, Ltd.

Hernández-Calviño, Manuel, Salvador Ibarra-Delgado, Remberto Sandoval-Aréchiga, Jorge Flores-Troncoso & Laura García-Luciano (2018), Image compressor ip-core based on loco algorithm for space photography application, *em* '2018 IEEE International Autumn Meeting on Power, Electronics and Computing (ROPEC)', pp. 1–4.

Huber, Felix, Peter Behr, Hans-Peter Röser & Samuel Pletner (2007), Fpga based onboard computer system for the flying laptop micro-satellite, *em* 'Proceedings of Data System in Aerospace Conference'.

Ibrahim, Mohamed Mahmoud, Ahmed Mahmoud Tobal, Mohamed Youssri El Nahas & Mohamed. K. Refai (2011), Fpga based on board computer for leo satellites, *em* 'Proceeding of the 2011 IEEE International Conference on Space Science and Communication (IconSpace)', pp. 314–319.

Iturbe, Xabier, Didier Keymeulen, Emre Ozer, Patrick Yiu, Daniel Berisford, Kevin Hand & Robert Carlson (2015), An integrated soc for science data processing in next-generation space flight instruments avionics, *em* '2015 IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC)', pp. 134–141.

Jacobs, Adam, Chris Conger & Alan D. George (2008), Multiparadigm space processing for hyperspectral imaging, *em* '2008 IEEE Aerospace Conference', pp. 1–11.

Jakhu, R. S. & J. N. Pelton (2014), *Small satellites and their regulation*, SpringerBriefs in Space Development, Springer.

Johnstone, Alicia (2022), CubeSat Design Specification rev.14.1, Relatório técnico, The CubeSat Program, Cal Poly, SLO.

Kasap, Server, Eduardo Weber Wächter, Xiaojun Zhai, Shoaib Ehsan & Klaus Mcdonald-Maier (2020), 'Survey of soft error mitigation techniques applied to leon3 soft processors on sram-based fpgas', *IEEE Access* **8**, 28646–28658.

Kastensmidt, Fernanda Lima & Ricardo Reis (2007), Fault tolerance in programmable circuits, *em* R.Velazco, P.Fouillat & R.Reis, eds., 'Radiation Effects on Embedded Systems', Springer Netherlands, Dordrecht, pp. 161–181.

Koren, Israel & C.M. Krishna (2007), *Fault-Tolerant Systems*, Elsevier.

Koudelka, O. (2016), Micro/nano/picosatellite-activities: Challenges towards space education and utilisation, *em* I.Marboe, ed., 'Small Satellites: Regulatory Challenges and Chances', Studies in Space Law, Brill.

Kramer, Herbert J & Arthur P Cracknell (2008), 'An overview of small satellites in remote sensing', *International journal of remote Sensing* **29**(15), 4285–4337.

LaBel, K.A., M.M. Gates, A.K. Moran, P.W. Marshall, J. Barth, E.G. Stassinopoulos, C.M. Seidleck & C.J. Dale (1996), Commercial microelectronics technologies for applications in the satellite radiation environment, *em* '1996 IEEE Aerospace Applications Conference. Proceedings', Vol. 1, pp. 375–390.

Lee, P. A. & T. Anderson (1990), *Fault Tolerance: Principles and Practice*, 2ª edição, Springer-Verlag, Berlin, Heidelberg.

Lentaris, George, Konstantinos Maragos, Ioannis Stratakos, Lazaros Papadopoulos, Odysseas Papanikolaou, Dimitrios Soudris, Manolis Lourakis, Xenophon Zabulis, David Gonzalez-Arjona & Gianluca Furano (2018), 'High-performance embedded computing in space: Evaluation of platforms for vision-based navigation', *Journal of Aerospace Information Systems* **15**(4), 178–192.

Lázaro, Jesús, Armando Astarloa, Aitzol Zuloaga, José Ángel Araujo & Jaime Jiménez (2024), 'Axi lite redundant on-chip bus interconnect for high reliability systems', *IEEE Transactions on Reliability* **73**(1), 602–607.

Lüdtke, Daniel, Thomas Firchau, Carlos Gonzalez Cortes, Andreas Lund, Ayush Mani Nepal, Mahmoud M. Elbarrawy, Zain Haj Hammadeh, Jan-Gerd Meß, Patrick Kenny, Fiona Brömer, Michael Mirzaagha, George Saleip, Hannah Kirstein, Christoph Kirchhefer & Andreas Gerndt (2023), Scosa on the way to orbit: Reconfigurable high-performance computing for spacecraft, *em* '2023 IEEE Space Computing Conference (SCC)', pp. 34–44.

Ma, Ning, Wang, Shaojun, Ali, Syed Mohsin, Cui, Xiuhai & Peng, Yu (2016), 'High efficiency on-board hyperspectral image classification with zynq soc', *MATEC Web of Conferences* **45**, 05001.

Mach, Ján, Lukáš Kohútka & Pavel Čičák (2023), 'On-chip bus protection against soft errors', *Electronics* **12**(22).

Mach, Ján, Lukáš Kohútka & Pavel Čičák (2024), Interface protection against transient faults, *em* '2024 27th International Symposium on Design & Diagnostics of Electronic Circuits & Systems (DDECS)', pp. 136–141.

Marin, Mikaël & Hyochoong Bang (2020), 'Design and simulation of a high-speed star tracker for direct optical feedback control in adcs', *Sensors* **20**(8).

Maurer, Richard H, Martin E Fraeman, Mark N Martin & David R Roth (2008), 'Harsh environments: Space radiation', *Johns Hopkins APL technical digest* **28**(1), 17.

Moore, Robert C. (2005), Spacecraft computer systems, *em* V.Pisacane, ed., 'Fundamentals of Space Systems', Applied Physics Laboratory series in science and engineering, Oxford University Press.

Morgan, Keith S., Daniel L. McMurtrey, Brian H. Pratt & Michael J. Wirthlin (2007), 'A comparison of tmr with alternative fault-tolerant design techniques for fpgas', *IEEE Transactions on Nuclear Science* **54**(6), 2065–2072.

NASA (2020), Small Spacecraft Technology State of the Art, Relatório técnico, NASA.

NASA CubeSat Launch Initiative (2017), *CubeSat 101: Basic Concepts and Processes for First-Time CubeSat Developers*, NASA.

Nascimento, José M. P. & Mário Véstias (2016), 'System-on-chip field-programmable gate array design for onboard real-time hyperspectral unmixing', *Journal of Applied Remote Sensing* **10**(1), 015004.

Orlandić, Milica, Johan Fjeldtvedt & Tor Arne Johansen (2019), 'A parallel fpga implementation of the ccsds-123 compression algorithm', *Remote Sensing* **11**(6).

Palkovitz, N. (2016), Small satellites: Innovative activities, tranditional laws, and the industry perspective, *em* I.Marboe, ed., 'Small Satellites: Regulatory Challenges and Chances', Studies in Space Law, Brill.

Papatheofanous, E. A., Ph. Tziolos, V. Kalekis, Tz. Amrou, G. Konstantoulakis, G. Venitourakis & D. Reisis (2022), Soc fpga acceleration for semantic segmentation of clouds in satellite images, *em* '2022 IFIP/IEEE 30th International Conference on Very Large Scale Integration (VLSI-SoC)', pp. 1–4.

Pignol, Michel (2010), Cots-based applications in space avionics, *em* '2010 Design, Automation Test in Europe Conference Exhibition (DATE 2010)', pp. 1213–1219.

Pitonak, Radoslav, Jan Mucha, Lukas Dobis, Martin Javorka & Marek Marusin (2022), 'Cloudsatnet-1: Fpga-based hardware-accelerated quantized cnn for satellite on-board cloud coverage classification', *Remote Sensing* **14**(13).

Pranitha, K & G Kavya (2023), 'An efficient image compression architecture based on optimized 9/7 wavelet transform with hybrid post processing and entropy encoder module', *Microprocessors and Microsystems* **98**, 104821.

Pérez, Arturo, Alfonso Rodríguez, Andrés Otero, David González Arjona, Álvaro Jiménez-Peralo, Miguel Ángel Verdugo & Eduardo De La Torre (2020), 'Run-time reconfigurable mpsoc-based on-board processor for vision-based space navigation', *IEEE Access* **8**, 59891–59905.

Raghunathan, V., M.B. Srivastava & R.K. Gupta (2003), A survey of techniques for energy efficient on-chip communication, *em* 'Proceedings 2003. Design Automation Conference (IEEE Cat. No.03CH37451)', pp. 900–905.

Rodríguez, Alfonso, Juan Valverde, Jorge Portilla, Andrés Otero, Teresa Riesgo & Eduardo De la Torre (2018), 'Fpga-based high-performance embedded systems for adaptive edge computing in cyber-physical systems: The artico3 framework', *Sensors* **18**(6).

Rosário, João, José M. P. Nascimento & Mário Véstias (2014), FPGA-based architecture for hyperspectral endmember extraction, *em* B.Huang, S.López & Z.Wu, eds.,

'High-Performance Computing in Remote Sensing IV', Vol. 9247, International Society for Optics and Photonics, SPIE, p. 924703.

Rudolph, Dylan, Christopher Wilson, Jacob Stewart, Patrick Gauvin, Alan George, Herman Lam, Gary Crum, Mike Wirthlin, Alex Wilson & Aaron Stoddard (2014), Csp: A multifaceted hybrid architecture for space computing, *em* 'Proc. 28th Annu. AIAA/USU Conf. Small Satellites'.

Ryan, William & Shu Lin (2009), *Channel Codes: Classical and Modern*, Cambridge University Press.

Scholz, Artur (2017), *CubeSat Standards Handbook: A Survey of International Space Standards with Application for CubeSat Missions*, The LibreCube Initiative.

Shiroma, Wayne, Larry Martin, Justin Akagi, Jason Akagi, Byron Wolfe, Bryan Fewell & Aaron Ohta (2011), 'Cubesats: A bright future for nanosatellites', *Open Engineering* **1**(1), 9–15.

Shooman, Martin L. (2002), *Reliability of Computer Systems and Networks: Fault Tolerance,Analysis,and Design*, John Wiley & Sons, Inc., USA.

Siegle, Felix, Tanya Vladimirova, Jørgen Ilstad & Omar Emam (2015), 'Mitigation of radiation effects in sram-based fpgas for space applications', *ACM Comput. Surv.* **47**(2).

Soh, Jeremy & Xiaofeng Wu (2014), A modular fpga-based implementation of the unscented kalman filter, *em* '2014 NASA/ESA Conference on Adaptive Hardware and Systems (AHS)', pp. 127–134.

Soh, Jeremy & Xiaofeng Wu (2018), 'A five-stage pipeline architecture of the unscented kalman filter for system-on-chip applications', *IEEE Transactions on Industrial Electronics* **65**(3), 2785–2794.

Tsigkanos, Antonis, Nektarios Kranitis, Dimitris Theodoropoulos & Antonios Paschalis (2020), 'High-performance cots fpga soc for parallel hyperspectral image compression with ccsds-123.0-b-1', *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* **28**(11), 2397–2409.

Urban, Ondřej, Ondřej Vavroch, Libor Poláček, Vjačeslav Georgiev, Petr Burian, Pavel Turjanica, Pavel Fiala, Pavel Broulím & Benedikt Bergmann (2020), 'Hodoscope with timepix detectors for pilsencube2 cubesat', *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* **980**, 164462.

Vahid, F. (2009), *Sistemas Digitais*, Bookman.

Vasudevan, Siddarth (2020), Design and development of a cubesat hardware architecture with cots mpsoc using radiation mitigation techniques, Dissertação de mestrado, School of Electrical Engineering and Computer Science, KTH.

Villa, Paulo Ricardo Cechelero (2018), Reliability enhanced microprocessor architecture for the on-board computer of future satellites, Tese de doutorado, Universidade Federal de Santa Catarina, Florianópolis, SC.

Villela, Thyrso, César Costa, Alessandra Brandão, Fernando Bueno & Rodrigo Leonardi (2019), 'Towards the thousandth cubesat: A statistical overview', *International Journal of Aerospace Engineering* **2019**, 1–13.

Vladimirova, Tanya & Muhammad Fayyaz (2018), Fault-tolerant distributed attitude and orbit control system for space applications, *em* '2018 NASA/ESA Conference on Adaptive Hardware and Systems (AHS)', pp. 43–50.

VLSI Universe (2021), 'Verilog code for sine cos and arctan using cordic algorithm', `https://vlsiuniverse.com/verilog-code-for-sine-cos-and-tan-cordic/`. Acesso em: 19 set. 2024.

Weber, Taisy Silva (2001), 'Tolerância a falhas: conceitos e exemplos', `http://www.inf.ufrgs.br/~taisy/disciplinas/textos/ConceitosDependabilidade.PDF`. Acesso em: 06 jul. 2021.

Wirthlin, M. (2015), 'High-reliability fpga-based systems: Space, high-energy physics, and beyond', *Proceedings of the IEEE* **103**(3), 379–389.

Woellert, Kirk, Pascale Ehrenfreund, Antonio J. Ricco & Henry Hertzfeld (2011), 'Cubesats: Cost-effective science and technology platforms for emerging and developing nations"', *Advances in Space Research* **47**(4), 663 – 684.

Xilinx (2017), 'Vivado design suite: Axi reference guide', `https://docs.amd.com/v/u/en-US/ug1037-vivado-axi-reference-guide`. Acesso em: 06 ago. 2024.

Yu, Guoxia, Tanya Vladimirova & Martin N. Sweeting (2009), 'Image compression systems on board satellites', *Acta Astronautica* **64**(9), 988–1005.

Zhang, Ning, Xin Wei, He Chen & Wenchao Liu (2021), 'Fpga implementation for cnn-based optical remote sensing object detection', *Electronics* **10**(3).