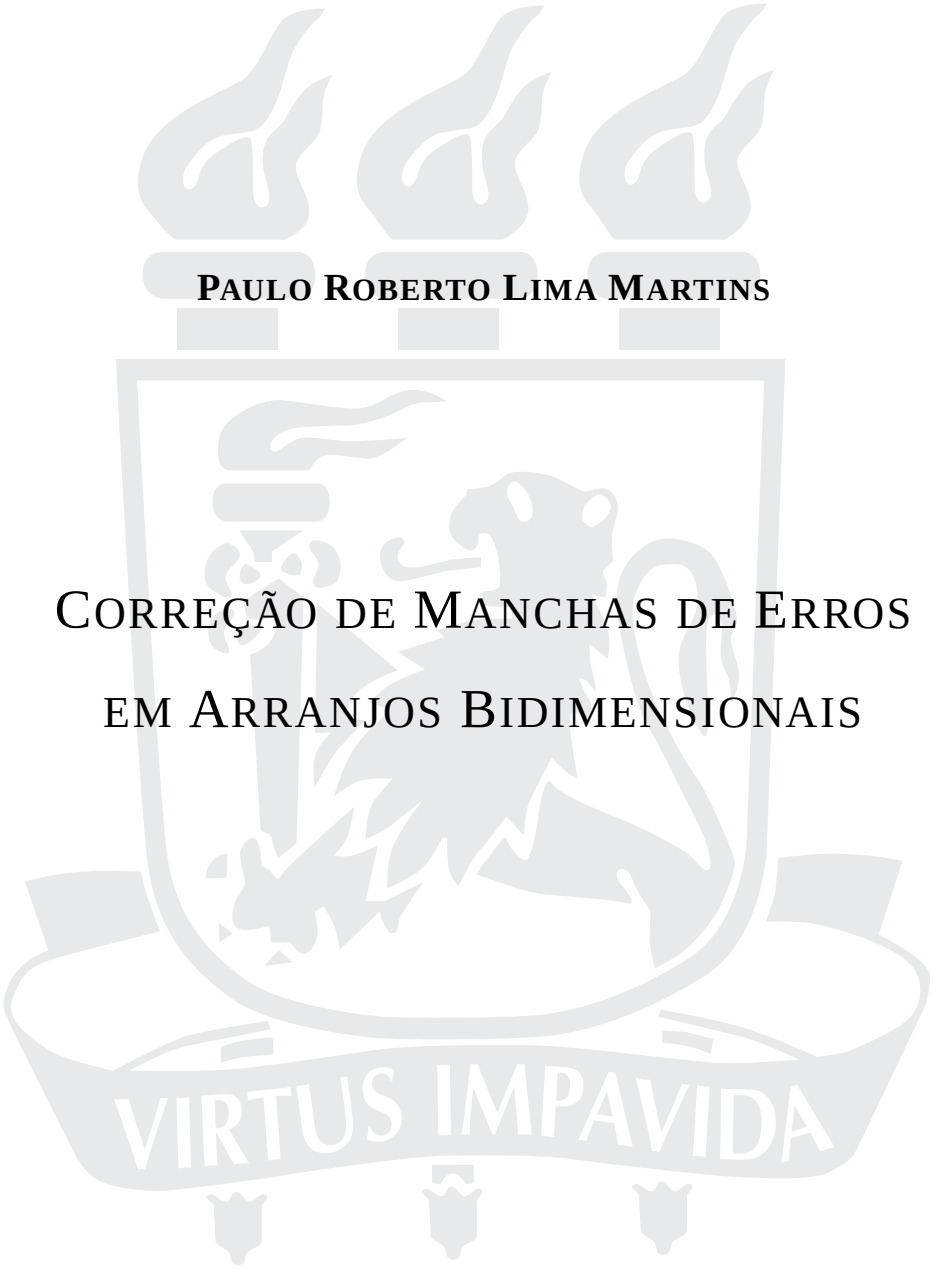
UNIVERSIDADE FEDERAL DE PERNAMBUCO

CENTRO DE TECNOLOGIA E GEOCIÊNCIAS

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA



PAULO ROBERTO LIMA MARTINS

CORREÇÃO DE MANCHAS DE ERROS EM ARRANJOS BIDIMENSIONAIS

RECIFE, JANEIRO DE 2012.

PAULO ROBERTO LIMA MARTINS

CORREÇÃO DE MANCHAS DE ERROS EM ARRANJOS BIDIMENSIONAIS

Dissertação submetida ao Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Pernambuco como parte dos requisitos para obtenção do grau de **Mestre em Engenharia Elétrica**

ORIENTADOR: PROF. VALDEMAR CARDOSO DA ROCHA JÚNIOR, PH.D.

Recife, janeiro de 2012.

Catalogação na fonte
Bibliotecária Margareth Malta, CRB-4 / 1198

M386c Martins, Paulo Roberto Lima.
Correção de manchas de erros em arranjos bidimensionais / Paulo Roberto Lima Martins. - Recife: O Autor, 2012.
94 folhas, il., gráfs., tabs.

Orientador: Prof. Dr. Valdemar Cardoso da Rocha Júnior.
Dissertação (Mestrado) – Universidade Federal de Pernambuco. CTG. Programa de Pós-Graduação em Engenharia Elétrica, 2012.
Inclui Referências Bibliográficas.

1. Engenharia Elétrica. 2. Códigos corretores de erros em surto. 3. Códigos cíclicos. 4. Entrelaçador. 5. Erros bidimensionais. I. Rocha Júnior, Valdemar Cardoso. (Orientador). II. Título.

UFPE

621.3 CDD (22. ed.) BCTG/2013-008

Universidade Federal de Pernambuco
Pós-Graduação em Engenharia Elétrica

PARECER DA COMISSÃO EXAMINADORA DE DEFESA DE DISSERTAÇÃO DO MESTRADO ACADÊMICO DE

PAULO ROBERTO LIMA MARTINS

TÍTULO

"CORREÇÃO DE MANCHAS DE ERROS EM ARRANJOS BIDIMENSIONAIS"

A comissão examinadora composta pelos professores: VALDEMAR CARDOSO DA ROCHA JÚNIOR, DES/UFPE, CECÍLIO JOSÉ LINS PIMENTEL, DES/UFPE e MARCELO SAMPAIO DE ALENCAR, DEE/UFCG sob a presidência do primeiro, consideram o candidato **PAULO ROBERTO LIMA MARTINS APROVADO.**

Recife, 31 de janeiro de 2012.

RAFAEL DUEIRE LINS Coordenador do PPGEE	**VALDEMAR CARDOSO DA ROCHA JÚNIOR** Orientador e Membro Titular Interno
MARCELO SAMPAIO DE ALENCAR Membro Titular Externo	**CECÍLIO JOSÉ LINS PIMENTEL** Membro Titular Interno

Aos meus pais,

Francisca de Sousa Lima Martins e

José Paulo Martins da Silva,

a minha irmã,

Grazielle de Lima Martins,

e a todos os **meus amigos**.

AGRADECIMENTOS

Agradeço em primeiro lugar a Deus por me dar capacidade e ferramentas para realizar este trabalho, pois sem Ele sei que não seria possível concluí-lo.

À minha família pelo carinho, compreensão e apoio incondicional nas diversas etapas e dificuldades deste trabalho. Em especial à minha mãe, por quem tenho profunda admiração não apenas pelo seu exemplo de vida, mas por sempre se preocupar em oferecer o melhor possível para seus filhos. À minha irmã, pelas conversas e pelo eterno apoio.

Ao meu orientador, professor Valdemar Cardoso da Rocha Jr, por sua orientação, paciência, dedicação e esforço em sempre me guiar pelo melhor caminho. Suas conversas descontraídas e apoio foram fundamentais no meu aprendizado.

Aos meus amigos, pelo apoio nos momentos de dificuldades. Em especial gostaria de agradecer a Paulo Freitas, meu grande amigo, pelo apoio espiritual e por sempre que possível se dispor a me ajudar. À Elda Lizandra pelos momentos bons, por sempre me ouvir e me dar conselhos diretos e precisos. A José Sampaio por me ajudar nos diversos assuntos téoricos, pelas observações ao longo do desenvolvimento do trabalho, além de sempre me dar força. A Paulo Hugo pelos conselhos e dicas na etapa final deste trabalho. A todos os meus colegas do LACRI e da graduação.

Aos professores do DES com os quais formei a base do meu aprendizado em engenharia eletrônica por meio de diversas cadeiras. Em especial aos professores do grupo de Telecomunicações, Cecilio José Lins Pimentel, Hélio Magalhães de Oliveira e Márcia Mahon Campello de Souza pelas conversas construtivas e pelas disciplinas cursadas que foram de fundamental importância para minha formação. Aos funcionários do DES, em especial à Andrea Tenório, secretária do Programa de Pós Graduação em Engenharia Elétrica (PPGEE) por sua competência, organização e sempre disponibilidade em me ajudar nos diversos momentos.

Por fim à Coordenação do PPGEE e à Coordenação de Aperfeiçoamento de Pessoa de Nível Superior (Capes) pelo apoio financeiro ao longo do desenvolvimento desta dissertação.

PAULO ROBERTO LIMA MARTINS

Universidade Federal de Pernambuco

31 de janeiro de 2012

Resumo da Dissertação apresentada à UFPE como parte dos requisitos necessários para a obtenção do grau de Mestre em Engenharia Elétrica

CORREÇÃO DE MANCHAS DE ERROS EM ARRANJOS BIDIMENSIONAIS

Paulo Roberto Lima Martins

janeiro/2012

Orientador: Prof. Valdemar Cardoso da Rocha Júnior, Ph.D.
Área de Concentração: Comunicações
Palavras-chaves: Códigos corretores de erros em surto, códigos cíclicos, entrelaçador, erros bidimensionais.
Número de páginas: 94

A correção de manchas de erros em arranjos bidimensionais é analisada por meio de simulação computacional de um sistema de comunicação digital simplificado. Nesse sistema é feito o uso de códigos cíclicos lineares binários em apenas uma das dimensões do arranjo. Por escolha adequada dos parâmetros do código e do arranjo bidimensional, manchas de erros com moldura na forma de quadrado, retângulo ou cruz, quando desentrelaçadas, aparecem como surtos de erros corrigíveis nas linhas do arranjo. Utilizando a capacidade de correção de surtos de erros de códigos cíclicos lineares binários, tais manchas de erros são então tratadas como surtos de erros em uma dimensão e corrigidas com a técnica de decodificação de surtos por armadilha. É considerado nas simulações também o decodificador adaptativo de surtos por armadilha proposto por Gallager, que produz melhores resultados.

Abstract of Dissertation presented to UFPE as a partial fulfillment of the requirements for the degree of Master in Electrical Engineering

CORRECTION OF PATCHS OF ERRORS IN TWO-DIMENSIONAL ARRAYS

Paulo Roberto Lima Martins

january/2012

Supervisor: Prof. Valdemar Cardoso da Rocha Júnior, Ph.D.
Area of Concentration: Communications
Keywords: Burst Error correcting codes, cyclic codes, interleaver, two-dimensional errors.
Number of pages: 94

Correction of patches of errors in two-dimensional arrays is analyzed by means of computer simulation of a simplified digital communication system. This system employs binary linear cyclic codes in only one dimension of the two-dimensional array. By suitable choice of the code and two-dimensional array parameters, patches of errors in the form of square, rectangle or cross, when deinterleaved, appear as burst correctable errors in the lines of the two-dimensional array. Using the ability of cyclic linear binary codes to correct errors in bursts, such patches are then treated as error bursts in one dimension and corrected using the technique of burst trapping decoding. It is also considered in the simulations the adaptive burst trapping decoder proposed by Gallager, which leads to better results.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

CAPÍTULO 1

INTRODUÇÃO

ESTE capítulo tem como objetivo apresentar uma visão geral sobre o trabalho desenvolvido nesta dissertação. Inicia-se com a apresentação dos sistemas de transmissão de dados e sua importância nas tecnologias que circundam o cotidiano. Em seguida, os blocos que compõem o sistema desenvolvido são apresentados. Logo após, são apresentados os objetivos deste trabalho bem como está organizada a estrutura da dissertação com um pequeno resumo sobre cada capítulo.

1.1 SISTEMAS DE TRANSMISSÃO DE DADOS

A sobrevivência do ser humano até hoje se deve, entre outros fatores, à capacidade de formar grupos e viver em sociedade com seus semelhantes. Sem uma boa comunicação entre os membros dessa sociedade não seria possível avançar muito além. Hoje em dia, diversos avanços tecnológicos tornam a comunicação mais fácil e ágil à medida que o tempo passa. A transmissão de informação analógica tem sido gradativamente trocada pela comunicação digital que vem sendo empregada em diversos sistemas do nosso dia-a-dia. Algumas vantagens dessa transmissão apresentadas em [1] podem ser destacadas:

- Facilidade de regeneração do sinal digital. O uso de repetidores regeneradores permite a recuperação perfeita do sinal transmitido, exceto por alguns erros que podem ser controlados no projeto do sistema.
- Incorporação de técnicas de processamento digital de sinais: códigos corretores de erro, codificação de fonte.

- ▷ Flexibilidade. Diferentes tipos de sinais (voz, vídeo, dados), com diferentes taxas podem ser tratados e manipulados como símbolos digitais e podem ser combinados usando técnicas de multiplexação.

Dentre os diversos exemplos pode-se citar conexões de Internet cabeadas ou sem fio como *Wi-Fi* e *WiMax*, rádios digitais, telefonia fixa e móvel e nas redes de televisão com a televisão digital, que oferece alta qualidade de imagem e som. Dando continuidade, tem-se o armazenamento de dados digitais por meio magnético como nos discos rígidos dos computadores e nas fitas magnéticas e um armazenamento óptico como o presente em *CDs, DVDs* e *Blu-rays*.

É função dos sistemas de transmissão de dados reproduzir fielmente em um destino a informação enviada por uma fonte de informação. Infelizmente, a informação pode sofrer danos ao ser enviada pelo canal ao destino, ou por armazenamento indevido como no caso dos armazenamentos magnéticos e ópticos. Os trabalhos desenvolvidos por Claude Shannon [2] [3] originaram a fundamentação teórica da Teoria da Informação, base para a comunicação digital. Com o avanço da teoria, desenvolveram-se estratégias para a deteção e correção desses danos, os chamados códigos corretores de erros.

Com o passar do tempo foram desenvolvidos dispositivos que correspondem a teoria criada anos antes. No entanto, a evolução não pára e ainda há muito a ser descoberto e inventado para facilitar a vida e melhorar o processo de comunicação.

1.1.1 SISTEMA DESENVOLVIDO

O sistema de comunicação desenvolvido consiste nos blocos ilustrados na Figura 1.1. Trata-se da representação de um sistema de comunicação simplificado entre um usuário A (Fonte) e um usuário B (Destino). Considera-se que o usuário A deseja mandar informação ao usuário B. A seguir cada bloco é apresentado com sua função.

O bloco *Fonte de Informação* gera os dados a serem enviados ao usuário B. Devido à natureza dos sistemas considerados neste trabalho, a informação gerada pela fonte de informação está na forma de matrizes binárias com dimensões pré-determinadas. É possível por meio da distribuição de probabilidade de 1’s e 0’s gerados pela fonte construir seu modelo estatístico.

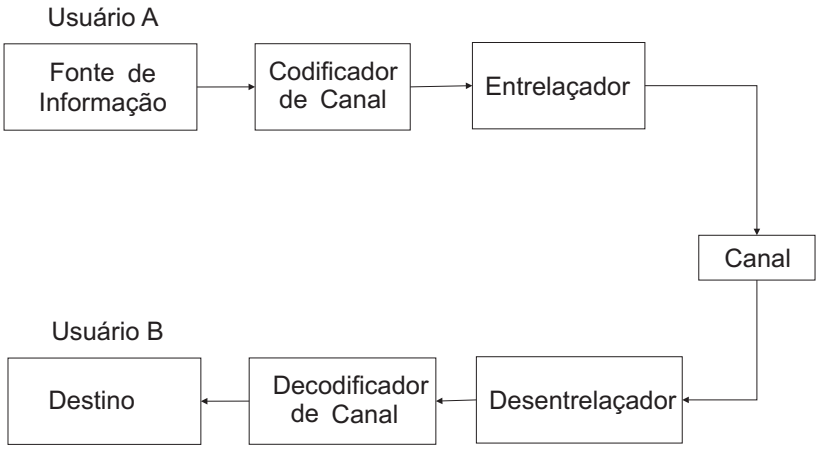


Figura 1.1: *Sistema desenvolvido - diagrama de blocos.*

O bloco *Codificador de Canal* é responsável por introduzir redundância na matriz de informação originada pela fonte de informação por meio da codificação de cada linha da matriz de informação, obtendo assim a matriz codificada. O codificador de canal utiliza os códigos corretores de erros. O primeiro codificador de canal inventado deve-se a Hamming [4], capaz de corrigir um erro por palavra-código. Tal redundância é necessária para proteger a informação aos ataques que ela pode sofrer ao ser enviada pelo bloco *Canal*. Tais ataques podem alterar o valor orginal e causar a perda de dados ao usuário B. Por meio desse processo é possível detectar e até corrigir alterações inseridas pelo canal físico. Pode-se definir como k/n, em que $n > k$, a taxa de codificação e n/k como a medida de controle sobre a redundância introduzida pelo processo de codificação. A variável n representa o comprimento da palavra-código e k a quantidade de *bits* de informação. A teoria para uma classe destes códigos é abordada no Capítulo 2.

O bloco *Entrelaçador* é usado neste trabalho para entrelaçar os *bits* da matriz codificada por meio da transformação linear apresentada em [5], originando a matriz entrelaçada. Dessa maneira, um surto de erro poderá não afetar *bits* consecutivos de determinada palavra código. Ele pode ser utilizado para aumentar a capacidade de correção de erros em surtos de determinado código cíclico por meio da forma de envio e organização do bloco de informação a ser transmitido [6].

O bloco *Canal* é uma representação do meio físico no qual é feita a troca de informação entre os usuários A e B. Ao passar pelo canal, a informação pode ser afetada por ruídos e inteferências. Nesta dissertação os ruídos e interferências são representados pela adição módulo 2 de manchas de erros binárias com molduras pré-estabelicidas à matriz entrelaçada. As linhas dessa matriz gerada por distribuição aleatórias de 1s e 0s representam surtos de erros que afetam a matriz codificada e entrelaçada.

O bloco *Desentrelaçador* desfaz o que foi realizado pelo entrelaçador aplicando sua transformação inversa. Dessa forma, a interferência, na forma de surtos binários adicionada à matriz entrelaçada, é distribuída. Os processos de entrelaçamento, adição de erro e desentrelaçamento são apresentados em detalhes no Capítulo 4.

O bloco *Decodificador de Canal* recebe a matriz desentrelaçada com toda a interferência adicionada pelo canal de comunicação. Com o conhecimento do código corretor de erros e utilizando a reduntância inserida pelo processo de codificação, o decodificador tenta recuperar a informação que o usuário A enviou, de modo a repassar para o usuário B a sequência que mais se aproxime da enviada por A. É desejado que todos os erros inseridos pelo canal sejam elimados, no entanto, características do código e tipo do ruído adicionado são fatores que afetam a probabilidade de erro no processo de decodificação.

1.2 OBJETIVOS

Os objetivos desta dissertação incluem realizar um estudo sobre a decodificação de erros em surtos pela técnica de armadilha e utilizá-la com o auxílio do entrelaçamento na correção de manchas de erros bidimensionais em determinado arranjo. O entrelaçamento e a codificação realizados em apenas uma dimensão do arranjo torna possível a correção de manchas de erros bidimensionais. Esta é uma outra possibilidade, ao invés de realizar codificação nas duas dimensões. As técnicas de decodificação por armadilha estudadas são a com armadilha fixa e a com armadilha adaptativa ou variável, sendo a segunda proposta por Gallager em [7]. O tamanho da armadilha é determinado pelos parâmetros do código corretor escolhido. Como os nomes sugerem, o tamanho da armadilha para decodificação fixa possui um valor pré-determinado e o da armadilha adaptativa (ou variável) se adequa durante a execução do algoritmo de decodificação. Com o uso do entrelaçador proposto em [5] manchas de erros são convertidas em surtos nas linhas da matriz de informação, podendo assim serem corrigidas pelo decodificador. Essa dissertação tem como objetivo mostrar o melhor desempenho na correção das manchas de erros quando se utilizada a técnica de decodificação por armadilha variável em relação à fixa. Utilizando imagens como fonte de informação, construi-se tabelas com contagem de *pixels* diferentes em relação à imagem original enviada por determinado usuário. A visualização de exemplos com imagens comprova esta melhora.

1.3 ORGANIZAÇÃO DA DISSERTAÇÃO

O conteúdo desta dissertação está dividido em cinco capítulos. As referências encontram-se nas páginas finais da dissertação e são ordenadas de acordo com a sequência em que foram citadas no texto. A seguir é apresentado um resumo dos capítulos.

Capítulo 2. O objetivo principal deste capítulo é apresentar os fundamentos dos códigos de bloco lineares, suas representações por matrizes e deteção de erros. Em seguida, apresentar a teoria dos códigos cíclicos com exemplos de codificação e deteção de erro por meio de registradores de deslocamento. Tais códigos foram de fundamental importância, pois pela sua capacidade de correção de erros em surto é que se pôde obter os resultados desejados.

Capítulo 3. Neste capítulo a teoria do códigos corretores de erros em surtos é apresentada. Os algoritmos das técnicas de decodificação por armadilha fixa e por armadilha adaptativa são mostrados e exemplificados para um mesmo surto.

Capítulo 4. Este capítulo é dedicado a mostrar as etapas do trabalho desenvolvido. É apresentado o sistema usado nas simulações computacionais. Exemplos ilustram o entrelaçamento, a geração e adição das manchas, desentrelaçamento e decodificação dos blocos de informação que percorem o sistema. É feita a análise do desempenho das técnicas utilizadas por cada decodificador.

Capítulo 5. Neste capítulo é feita a conclusão e comentários finais da dissertação bem como sugestões para trabalhos futuros.

CAPÍTULO 2

CÓDIGOS DE BLOCO LINEARES

EXISTEM diferentes estruturas para os códigos corretores de erros. Neste capítulo, os fundamentos de códigos de blocos lineares são introduzidos. Em seguida, o foco é voltado para uma classe desses códigos, os códigos cíclicos, devido a sua importância no desenvolvimento deste trabalho. É apresentada a especificação destes códigos por meio de suas matrizes geradora e de verificação de paridade. Também são apresentados, codificação e decodificação com o uso de registradores de deslocamento e detecção e correção de erros aleatórios com a síndrome. Em toda a dissertação considera-se apenas códigos de bloco lineares binários.

2.1 CONCEITOS BÁSICOS DOS CÓDIGOS DE BLOCOS LINEARES

Considere uma fonte de informação gerando símbolos binários na forma de uma sequência de *bits*. Na codificação por bloco, essa sequência é dividida em blocos de comprimento fixo. Esses blocos com *k* dígitos de informação são denominados *mensagens* que se representam por um vetor $\mathbf{u} = (u_0, u_1, \ldots, u_{k-1})$,totalizando um número de 2^k possíveis mensagens. A codificação de cada mensagem resulta em um novo bloco com *n* dígitos, em que $n > k$. Esse novo bloco é denominado *vetor código*, representado por $\mathbf{v} = (v_0, v_1, \ldots, v_{n-1})$. Cada vetor $\mathbf{u}$ possui um correspondente único $\mathbf{v}$, para ser possível o processo de decodificação. Esse conjunto de 2^k palavras códigos constitui o dicionário do código de bloco binário $C(n, k)$.

Códigos são utilizados na deteção e correção de erros [6]. Esses erros são inseridos na transmis-

são, seja por ruído do canal ou interferências e podem ser corrigidos pelo acréscimo dos $n-k$ dígitos de redundância no bloco de informação realizada pela codificação. Uma característica importante existente em alguns códigos de bloco é a linearidade [8]. A linearidade proporciona uma estrutura matemática aos códigos de bloco que os permite a obter simpliplicações em relação a propriedades como capacidade de deteção e correção de erros. Esses códigos são os mais estudados e utilizados em aplicações práticas [9]. Um código de bloco linear é definido como segue.

Definição 2.1 – Código de Bloco Linear

Um código de bloco de comprimento n e 2^k palavras código é linear binário $C(n,k)$ se e somente se suas 2^k palavras código formam um subespaço de dimensão k do espaço vetorial de todas n-uplas sobre GF(2). Em que, GF denota o Corpo de Galois. □

2.1.1 DESCRIÇÃO DOS CÓDIGOS DE BLOCO LINEARES POR MATRIZES

Duas matrizes caracterizam um código de bloco, a matriz geradora e a matriz de verificação de paridade. Com base na Definição 2.1, tem-se que as palavras do código de bloco linear $C(n,k)$ geram um subespaço de dimensão k de todas as n-uplas sobre GF(2). Sendo assim, existem k palavras-código linearmente independentes,$\{\mathbf{g}_0, \mathbf{g}_1, \ldots, \mathbf{g}_{k-1}\}$, que formam a base desse subespaço e toda palavra do código $\mathbf{v}$ é formada pela combinação linear dessas k palavras-código linearmente independentes. Considerando $\mathbf{u} = (u_0, u_1, \ldots, u_{k-1})$ o vetor informação a ser codificado, o vetor codificado $\mathbf{v} = (v_0, v_1, \ldots, v_{n-1})$ é obtido por combinação linear de $\{\mathbf{g}_0, \mathbf{g}_1, \ldots, \mathbf{g}_{k-1}\}$ resultando $\mathbf{v} = u_0\mathbf{g}_0 + u_1\mathbf{g}_1 + \ldots + u_{k-1}\mathbf{g}_{k-1}$, em que u_i, são os dígitos de informação para $0 \leq i \leq k-1$.

Organizando matricialmente e definindo os vetores-base $\mathbf{g}_i$ como linhas de uma *matriz geradora* denotada por $\mathbf{G_{k\times n}}$, têm-se

$$\mathbf{G} = \begin{bmatrix} \mathbf{g}_0 \\ \mathbf{g}_1 \\ \vdots \\ \mathbf{g}_{k-1} \end{bmatrix} = \begin{bmatrix} g_{0,0} & g_{0,1} & \cdots & g_{0,n-1} \\ g_{1,0} & g_{1,1} & \cdots & g_{1,n-1} \\ \vdots & \vdots & \ddots & \vdots \\ g_{k-1,0} & g_{k-1,1} & \cdots & g_{k-1,n-1} \end{bmatrix}. \tag{2.1}$$

Dado o vetor $\mathbf{u} = (u_0, u_1, \ldots, u_{k-1})$ é possivel encontrar seu vetor codificado correspondente por meio da multiplicação do vetor $\mathbf{u}$ pela matriz $\mathbf{G}$, sendo assim: $\mathbf{v} = \mathbf{u} \cdot \mathbf{G}$. Pelo fato das linhas de $\mathbf{G}$ gerarem as palavras do código ela é considerada *matriz geradora* do código linear $C(n,k)$.

Com base no visto até o momento, é possivel a existência de várias matrizes $\mathbf{G}$, que codificam um vetor $\mathbf{u}$. Por exemplo, ao trocar uma linha da matriz pelo resultado da adição de outras duas linhas, obtém-se uma nova matriz $\mathbf{G}^*$, ainda formada de k linhas linearmente independentes. A

multiplicação do mesmo vetor $\mathbf{u}$ para cada matriz gerará um vetor $\mathbf{v}$ diferente. Considera-se deste ponto em diante a codificação na forma sistemática.

Definição 2.2 – Codificação Sistemática

Após o processo de codificação sistemática, toda palavra código do código linear $C(n,k)$ é dividida em duas partes. Os primeiros $n-k$ dígitos do vetor $\mathbf{v}$, são considerados dígitos de verificação de paridade e os demais k últimos dígitos, são considerados dígitos de informação.□

Códigos de bloco lineares que obedecem à Definição 2.2 são chamados de códigos de bloco lineares sistemáticos. A Figura 2.1 ilustra a estrutura da palavra código após a codificação sistemática.

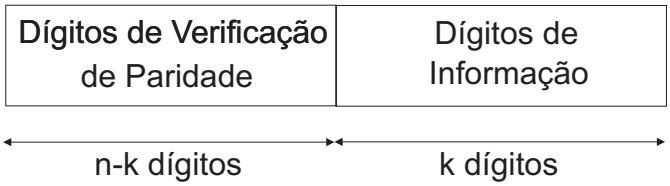


Figura 2.1: *Palavra código na forma sistemática.*

A matriz geradora de um código sistemático possui a seguinte forma

$$\mathbf{G} = [\mathbf{P}|\mathbf{I}_k] = \begin{bmatrix} p_{0,0} & p_{0,1} & \cdots & p_{0,n-k-1} & 1 & 0 & 0 & \ldots & 0 \\ p_{1,0} & p_{1,1} & \cdots & p_{1,n-k-1} & 0 & 1 & 0 & \ldots & 0 \\ p_{2,0} & p_{2,1} & \cdots & p_{2,n-k-1} & 0 & 0 & 1 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ p_{k-1,0} & p_{k-1,1} & \cdots & p_{k-1,n-k-1} & 0 & 0 & 0 & \ldots & 1 \end{bmatrix}. \tag{2.2}$$

em que $p_{ij} = 0$ ou 1 para $0 \leq i \leq k-1$ e $0 \leq j \leq n-k-1$, $\mathbf{I_k}$ é a matriz identidade $k \times k$ e $\mathbf{P}$ é uma matriz $k \times (n-k)$ que gera os dígitos de paridade.

A segunda matriz associada com os códigos de bloco lineares é a matriz de verificação de paridade $\mathbf{H_{(n-k)\times n}}$. Com suas $n-k$ linhas linearmente independentes, essa matriz gera o espaço dual, de dimensão $n-k$, do espaço vetorial V gerado pelas k linhas linearmente independentes de $\mathbf{G}$. A matriz $\mathbf{H}$ é a matriz geradora do chamado código dual $C_d(n, n-k)$ de $C(n,k)$. Logo, toda palavra em C_d é obtida pela combinação linear dos vetores que compõem as linhas de $\mathbf{H}$. Sendo assim forma-se a matriz $\mathbf{H_{(n-k)\times n}}$ sobre GF(2).

$$\mathbf{H} = \begin{bmatrix} \mathbf{h}_0 \\ \mathbf{h}_1 \\ \vdots \\ \mathbf{h}_{n-k-1} \end{bmatrix} = \begin{bmatrix} h_{0,0} & h_{0,1} & \dots & h_{0,n-1} \\ h_{1,0} & h_{1,1} & \dots & h_{1,n-1} \\ \vdots & \vdots & \ddots & \vdots \\ h_{k-1,0} & h_{k-1,1} & \dots & h_{k-1,n-1} \end{bmatrix}. \tag{2.3}$$

O espaço vetorial gerado por $\mathbf{H}$ é ortogonal ao espaço vetorial gerado por $\mathbf{G}$. Dessa maneira é possível definir um código de bloco linear gerado por $\mathbf{G}$ em função da matriz $\mathbf{H}$.

Definição 2.3 – Código de Bloco Linear

Uma n-upla $\mathbf{v}$ *é uma palavra do código gerado por* $\mathbf{G}$ *se e somente se* $\mathbf{v} \cdot \mathbf{H^T} = 0$*, em que* $\mathbf{H^T}$ *denota a matriz transposta de* $\mathbf{H}$. □

De maneira análoga a $\mathbf{G}$, representa-se $\mathbf{H}$ na forma sistemática que pode ser obtida diretamente da matriz $\mathbf{G}$ na forma sistemática, apresentada na Equação (2.2).

$$\mathbf{H} = [\mathbf{I}_{n-k}|\mathbf{P}^T] = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 & p_{0,0} & p_{1,0} & \dots & p_{k-1,0} \\ 0 & 1 & 0 & \dots & 0 & p_{0,1} & p_{1,1} & \dots & p_{k-1,1} \\ 0 & 0 & 1 & \dots & 0 & p_{0,2} & p_{1,2} & \dots & p_{k-1,2} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 1 & p_{0,n-k-1} & p_{1,n-k-1} & \dots & p_{k-1,n-k-1} \end{bmatrix}. \tag{2.4}$$

em que $p_{ij} = 0$ ou 1 para $0 \leq i \leq k-1$ e $0 \leq j \leq n-k-1$, $\mathbf{P}^T$ é a matriz transposta de $\mathbf{P}$ e $\mathbf{I}_{n-k}$ é a matriz identidade $(n-k) \times (n-k)$.

2.1.2 Síndrome e Detecção de Erro

Seja $C(n,k)$ um código linear binário com matrizes $\mathbf{G}$ e $\mathbf{H}$. Supondo a transmissão da palavra código $\mathbf{v} = (v_0, v_1, \dots, v_{n-1})$ por meio de um canal com decisão abrupta, por exemplo, o Canal Binário Simétrico - BSC [10]. Seja $\mathbf{r} = (r_0, r_1, \dots, r_{n-1})$, o vetor recebido. Devido a interferências e a ruídos do canal, algumas posições de $\mathbf{r}$ podem ser distintas das de $\mathbf{v}$. A mudança dessas posições no vetor $\mathbf{v}$ é ocasionada pela adição módulo 2 do vetor erro $\mathbf{e}$. As posições não-nulas de $\mathbf{e}$ alteram o valor das respectivas posições em $\mathbf{v}$. Logo, o vetor $\mathbf{r}$ pode ser escrito como

$$\mathbf{r} \quad = \quad (\mathbf{v} + \mathbf{e}) \bmod 2. \tag{2.5}$$

Como o receptor não conhece os vetores $\mathbf{e}$ e $\mathbf{v}$, ele primeiro necessita descobrir se o vetor recebido $\mathbf{r}$ contém erros ou não e em seguida tentar corrigí-los ou solicitar uma retransmissão.

É possível detectar os erros após o recebimento de $\mathbf{r}$ realizando o cálculo do vetor *síndrome*.

Definição 2.4 – Síndrome

Seja $\mathbf{r}$ *um n-upla binária e* $\mathbf{H}$ *a matriz de verificação de paridade de um código de bloco linear* $C(n,k)$. *O vetor* $\mathbf{s} = \mathbf{r} \cdot \mathbf{H^T}$ *é denominado vetor síndrome.* □

De acordo com a Definição 2.3 sabe-se que $\mathbf{s} = \mathbf{0}$, se e somente se, $\mathbf{r}$ for uma palavra código de $C(n,k)$. Seguindo o mesmo pensamento, se $\mathbf{s} \neq \mathbf{0}$, então $\mathbf{r}$ não é uma palavra do código de $C(n,k)$. De maneira geral, quando $\mathbf{s} \neq \mathbf{0}$ é sabido que houve erro, no entanto, não há conhecimento de sua localização. Para o caso de $\mathbf{s} = \mathbf{0}$ o receptor considera o vetor recebido $\mathbf{r}$ como sendo o vetor transmitido. No entanto, ainda pode ocorrer o caso em que o erro inserido pelo canal torne o vetor transmitido em outra palavra código, neste caso temos um erro indetectável. Existe um total de $2^k - 1$ possivéis palavras-código que podem gerar esses erros indetectáveis.

2.2 CÓDIGOS CÍCLICOS

Formando uma classe dos códigos de bloco lineares, os códigos cíclicos inicialmente estudados por Prange em 1957 [11] são utilizados na correção de erros aleatórios e em surto. Na prática já ganharam bastante destaque no seu uso em *Compact Disc* (CD) [12]. A construção do decodificador adequado torna esse código capaz de corrigir erros aleatórios, em surto ou ambos em uma mesma palavra código. Os códigos cíclicos têm a vantagem da simplificidade de codificação, cálculo da síndrome e decodificação por meio do uso de registradores de deslocamento realimentados.

Seja $\mathbf{v} = (v_0, v_1, \ldots, v_{n-2}, v_{n-1})$ uma n-upla binária sobre GF(2), em que se representa um deslocamento para a direita desse vetor por $\mathbf{v^{(1)}} = (v_{n-1}, v_0, v_1 \ldots, v_{n-2})$. Seguindo o mesmo raciocínio, pode-se representar o i-ésimo deslocamento de $\mathbf{v}$, para $1 \leq i \leq n$, como $\mathbf{v^{(i)}} = (v_{i+n-1}, v_i, v_{i+1} \ldots, v_{i+n-2})$. É baseado neste conjunto de deslocamentos que é construído o dicionário do código cíclico.

Definição 2.5 – Código Cíclico

Seja $\mathbf{v} = (v_0, v_1, \ldots, v_{n-2}, v_{n-1})$ *uma palavra do código binário linear* $C(n,k)$, *então todos os seus* i*-ésimos deslocamentos, para* $1 \leq i \leq n$, *representados por* $\mathbf{v^{(i)}}$ *também serão palavras-código do código* $C(n,k)$. □

As palavras-código do código cíclico $C(n,k)$ podem ser representadas na sua forma polinomial. A análise das propriedades desses códigos torna-se assim mais fácil, pois realizar operações com polinômios em um corpo já estabelecido é mais simples. Sendo assim, dada a n-$upla$ $\mathbf{v} = (v_0, v_1, \ldots, v_{n-2}, v_{n-1})$ sobre $GF(2)$, têm-se sua representação polinomial da seguinte forma

$$v(x) \quad = \quad v_0 + v_1 x + \ldots + v_{n-2}x^{n-2} + v_{n-1}x^{n-1}. \tag{2.6}$$

Considere esse polinômio, como polinômio código do código cíclico linear binário $C(n,k)$. Os 2^k diferentes polinômios código que podem ser originados das 2^k k-uplas binárias de um código binário linear $C(n,k)$ formam o dicionário do código cíclico linear $C(n,k)$. Sendo assim, com base em 2.6, toda palavra código é representada por um polinômio de grau r para $n-k \leq r \leq n-1$.

Existe uma relação entre o i-ésimo deslocamento à direita, $v(x)^{(i)}$ e o polinômio original $v(x)$ dada por

$$\begin{aligned} v(x)^{(i)} \quad &= \quad a(x)(x^n - 1) + x^i v(x) \\ \\ &= \quad x^i v(x) \bmod (x^n - 1). \end{aligned} \tag{2.7}$$

em que $a(x) = v_{n-i} + v_{n-i+1}x + \ldots + v_{n-1}x^{i-1}$. De 2.7 percebe-se que o polinômio código $v^i(x)$ é igual ao resto da divisão de de $x^i v(x)$ por $x^n + 1$.

Em códigos cíclicos existe um polinômio, o *polinômio gerador*, que é responsável por especificar determinado código cíclico $C(n,k)$. Este polinômio, aqui representado por $g(x)$ possui a seguinte forma

$$g(x) \quad = \quad 1 + g_1 x + \ldots + g_{n-k-1}x^{n-k-1} + x^{n-k}, \tag{2.8}$$

e obedece às seguintes propriedades:

- $g(x)$ é não-nulo e o único de grau igual a n-k;
- $g(x)$ é fator de x^n-1;
- Todo polinômio código é múltiplo de $g(x)$.

Vale resaltar que todo $g(x)$ utilizado nesta dissertação obedece as propriedades citadas.

2.2.1 CODIFICAÇÃO

Como mencionado, todo polinômio código é um múltiplo de $g(x)$. Dada a representação polinomial de um vetor informação $\mathbf{u}$, $u(x)$, então a representação polinonimal do vetor código $\mathbf{v}$, $v(x)$, se

dá pela multiplicação de $u(x)$ por $g(x)$.

$$v(x) \quad = \quad u(x)g(x). \tag{2.9}$$

Exemplo 2.1

Seja o código cíclico binário $C(15,5)$ *gerado por* $g(x) = 1 + x^2 + x^5 + x^6 + x^8 + x^9 + x^{10}$ *e o polinômio informação* $u(x) = 1 + x^2$. *Logo* $v(x) = u(x) \cdot g(x) = 1 + x^4 + x^5 + x^6 + x^7 + x^9 + x^{11} + x^{12}$. □

Da mesma forma, conforme descrito na Seção 2.1.1, pode-se representar matrizes $\mathbf{G}$ e $\mathbf{H}$ de um código cíclico. As linhas da matriz $\mathbf{G}$ podem ser formadas por k delocamentos da representação vetorial de $g(x)$ com n *bits*. Assim, há k linhas linearmente independentes formando a matriz $\mathbf{G_{kxn}}$.

$$\mathbf{G} = \begin{bmatrix} \mathbf{g}^{(0)} \\ \mathbf{g}^{(1)} \\ \vdots \\ \mathbf{g}^{(k-1)} \end{bmatrix} = \begin{bmatrix} 1 & g_1 & \dots & g_{n-k-1} & 1 & 0 & 0 & \dots & 0 & 0 \\ 0 & 1 & \dots & g_{n-k-2} & g_{n-k-1} & 1 & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \\ 0 & 0 & \dots & 1 & g_1 & g_2 & g_3 & \cdots & g_{0,n-k-1} & 1 \end{bmatrix}. \tag{2.10}$$

O processo de codificação exemplificado no Exemplo 2.1 utilizando a multiplicação da matriz $\mathbf{G}$ descrita em 2.10, resulta no processo de codificação **não** sistemática.

Sabendo que $g(x)$ é fator de x^n-1, logo

$$x^n - 1 \quad = \quad g(x)a(x), \tag{2.11}$$

em que $a(x)$ é um polinômio de grau k sobre $GF(2)$. O polinômio de verificação de paridade, $h(x)$, é considerado como o polinômio recíproco de $a(x)$ sendo encontrado conforme segue:

$$\begin{aligned} h(x) &= x^k a(x^{-1}) \\ &= 1 + h_1 x + \ldots + h_{k-1} x^{k-1} + x^k. \end{aligned} \tag{2.12}$$

A partir de $h(x)$ pode-se construir a matriz $\mathbf{H}$ do código cíclico $C(n,k)$. As linhas da matriz $\mathbf{H}$ podem ser formadas por n-k delocamentos da representação vetorial de $h(x)$ com n *bits*. Assim, obtem-se n-k linhas linearmente independentes formando a matriz de verificação de paridade, $\mathbf{H_{(n-k)\times n}}$. O polinômio $h(x)$, bem como a matriz $\mathbf{H}$ geram o código cíclico dual de $C(n,k)$, $C_d(n-k,k)$

$$\mathbf{H} = \begin{bmatrix} \mathbf{h}^{(0)} \\ \mathbf{h}^{(1)} \\ \vdots \\ \mathbf{h}^{(k-1)} \end{bmatrix} = \begin{bmatrix} 1 & h_1 & \dots & h_{k-1} & 1 & 0 & 0 & \dots & 0 & 0 \\ 0 & 1 & \dots & h_{k-2} & h_{k-1} & 1 & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \\ 0 & 0 & \dots & 1 & h_1 & h_2 & h_3 & \dots & h_{0,k-1} & 1 \end{bmatrix}. \tag{2.13}$$

A matriz $\mathbf{G}$ também pode ser construída na forma sistemática. Obtendo o resto da divisão, $r_i(x)$, de x^{n-k-i}, para $0 \leq i \leq k-1$, pelo polinômio gerador $g(x)$, monta-se um conjunto de k polinômios de grau máximo igual a n-k-1. Organizando cada polinômio $g_i(x)$, em que $g_i(x) = r_i(x) + x^{n-k-i}$, como linhas da matriz $\mathbf{G_{k \times n}}$ obtem-se a sua representação na forma sistemática.

$$\mathbf{G} = \begin{bmatrix} r_{0,0} & r_{0,1} & \dots & r_{0,n-k-1} & 1 & 0 & 0 & \dots & 0 \\ r_{1,0} & r_{1,1} & \dots & r_{1,n-k-1} & 0 & 1 & 0 & \dots & 0 \\ r_{2,0} & r_{2,1} & \dots & r_{2,n-k-1} & 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ r_{k-1,0} & r_{k-1,1} & \dots & r_{k-1,n-k-1} & 0 & 0 & 0 & \dots & 1 \end{bmatrix}. \tag{2.14}$$

Seguindo o mesmo raciocínio da Seção 2.1.1 tem-se a representação da matriz $\mathbf{H_{(n-k) \times k}}$ na forma sistemática para códigos cíclicos

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 & r_{0,0} & r_{1,0} & \dots & r_{k-1,0} \\ 0 & 1 & 0 & \dots & 0 & r_{0,1} & r_{1,1} & \dots & r_{k-1,1} \\ 0 & 0 & 1 & \dots & 0 & r_{0,2} & r_{1,2} & \dots & r_{k-1,2} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 1 & r_{0,n-k-1} & r_{1,n-k-1} & \dots & r_{k-1,n-k-1} \end{bmatrix}. \tag{2.15}$$

O procedimento usado para montar as linhas da matriz apresentada em 2.14 pode ser generalizado para qualquer polinômio informação, resultando na codificação sistemática. A codificação consiste em três passos:

1. Primeiro, multiplicar o polinômio $u(x)$ por x^{n-k};
2. Em seguida, obter o polinômio resto da divisão de $u(x)x^{n-k}$ por $g(x)$, $p(x)$. Este polinômio representa os dígitos de paridade;
3. Por último obter o polinômio código $v(x)$, em que $v(x) = p(x) + u(x)x^{n-k}$.

O Exemplo 2.2 ilustra essa codificação.

Exemplo 2.2

Seja o código cíclico binário $C(15,5)$ *gerado por* $g(x) = 1 + x^2 + x^5 + x^6 + x^8 + x^9 + x^{10}$ *e o polinômio informação* $u(x) = 1 + x^2$*. Seguindo os passos da codificação sistemática:*

$$\begin{aligned} u(x)x^{15-5} &= (1+x^2)x^{10} = x^{10} + x^{12}. \\ p(x) &= (x^{10} + x^{12})(\ \mathit{mod}(g(x)) = 1 + x + x^3 + x^4 + x^5. \\ v(x) &= u(x)x^{10} + p(x) = 1 + x + x^3 + x^4 + x^5 + x^{10} + x^{12}. \end{aligned} \tag{2.16}$$

Logo, $v(x) = 1 + x + x^3 + x^4 + x^5 + x^{10} + x^{12}$. □

Observando as representações vetoriais de $u(x)$, $\mathbf{u} = [10100]$, e $v(x)$, $\mathbf{v} = [110111000010100]$, percebe-se que os $k = 5$ *bits* finais de $\mathbf{v}$ correspondem ao vetor $\mathbf{u}$.

Uma das vantagens citadas no início desta seção é a facilidade na implementação dos codificadores para os códigos cíclicos por meio do uso de registradores de deslocamento realimentados. O registrador deve realizar a divisão de $x^{n-k}u(x)$ por $g(x)$ e armazenar uma sequência em seus $n-k$ estágios. A sequência binária armazenada no registrador após o carregamento completo do vetor $x^{n-k}u(x)$ irá formar os dígitos de paridade do polinômio código. Os ramos de realimentação do registrador são baseados no polinômio gerador $g(x)$. A Figura 2.2 ilustra um codificador genérico para códigos cíclicos.

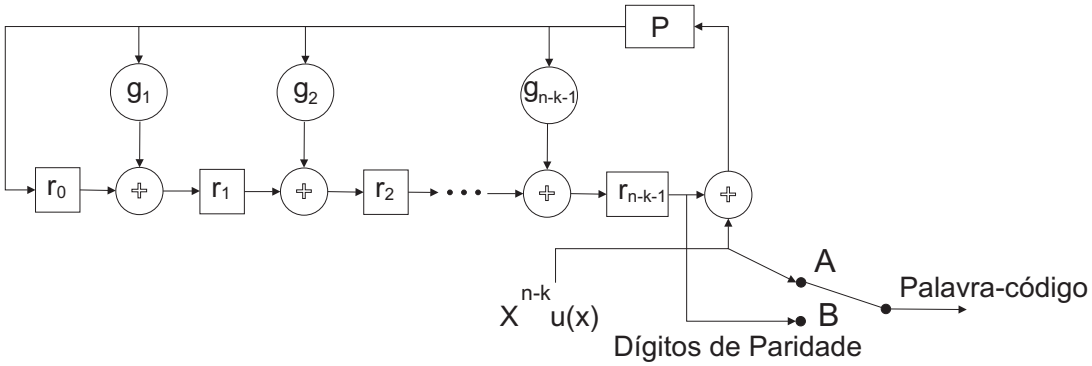


Figura 2.2: *Circuito codificador genérico para um código cíclico C(n,k).*

Considera-se que os estágios de memória do circuito codificador são iniciados com zeros. O procedimento de codificação com registradores inicia pelo carregamento do vetor informação $\mathbf{u}$, com a chave P ativada. Um carregamento por A, equivale à pré-multiplicação por x^{n-k}. Ao mesmo tempo os *bits* são enviados pelo canal. Finalizado o carregamento de $\mathbf{u}$, os *bits* de paridade estão

contidos no registrador, então a porta P é desativada e após uma comutação de A para B da chave seletora os *bits* de paridade são enviados ao canal finalizando o envio da palavra código.

O Exemplo 2.3 ilustra essa codificação

Exemplo 2.3

Seja o código cíclico binário $C(15,5)$ gerado por $g(x) = 1 + x^2 + x^5 + x^6 + x^8 + x^9 + x^{10}$ e o polinômio informação $u(x) = 1 + x^2$. A Figura 2.3 ilustra um codificador para $g(x) = 1 + x^2 + x^5 + x^6 + x^8 + x^9 + x^{10}$.

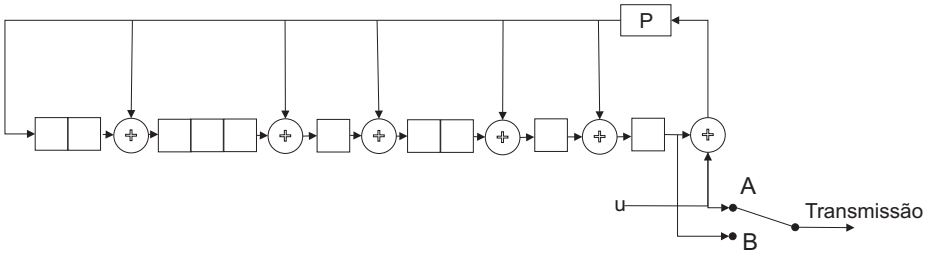


Figura 2.3: *Circuito codificador para $g(x) = 1 + x^2 + x^5 + x^6 + x^8 + x^9 + x^{10}$ do código cíclico C(15,5).*

As Figuras 2.4 e 2.5 ilustram o carregamento e codificação do vetor $\mathbf{u} = [10100]$.

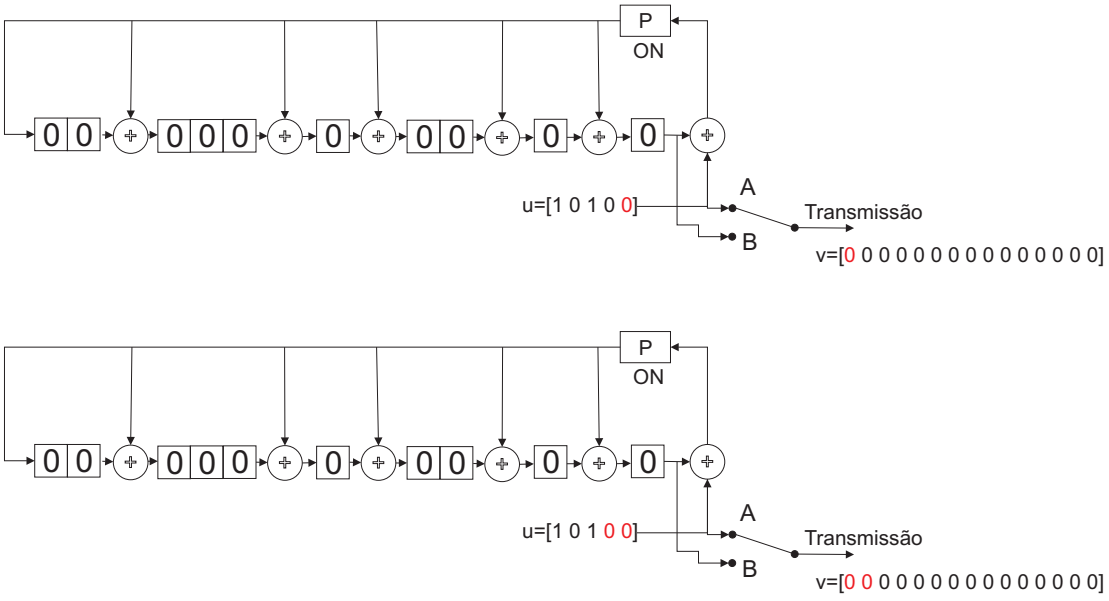


Figura 2.4: *Primeiro e segundo deslocamentos na codificação de $u(x) = 1 + x^2$.*

Figura 2.5: *Deslocamentos finais na codificação para* $u(x) = 1 + x^2$.

Após a comutação da chave de A para B o conteúdo de registrador é enviado, esvaziado e novamente preenchido com zeros para o envio de uma nova palavra código. Terminado o esvaziamento a palavra código $\mathbf{v} = [110111000010100]$ foi transmitida. Observa-se que é a mesma sequência binária do Exemplo 2.2.

2.2.2 SÍNDROME E DETECÇÃO DE ERRO

Conforme visto na Subseção 2.1.2, após a transmissão, erros podem ser inseridos e alterar o valor de alguma(s) posição(ões) da mensagem transmitida. Nos códigos cíclicos, toda palavra código é múltipla de $g(x)$, então esse é o teste considerado. Ao calcular a divisão do polinômio recebido $r(x)$ por $g(x)$, resulta em

$$r(x) \quad = \quad b(x)g(x) + s(x). \tag{2.17}$$

O resto da divisão, $s(x)$, é o fator determinante na indicação de erro no polinômio recebido, conhecido como polinômio síndrome.

Definição 2.6 – Polinômio Síndrome

Seja $r(x)$ o polinômio recebido de grau n-1 ou menor e $g(x)$ o polinômio gerador do código cíclico binário $C(n, k)$. O polinômio resto da divisão, de grau n-k-1 ou menor, representado em 2.17, $s(x)$, é denotado polinômio síndrome. □

Registradores de deslocamento realimentados baseados no $g(x)$ do código cíclico atuam na divisão e armazenamento da sequência binária de $s(x)$. A Figura 2.6 representa um circuito que realiza a divisão de $r(x)$ por $g(x)$ e armazena o vetor síndrome em suas células.

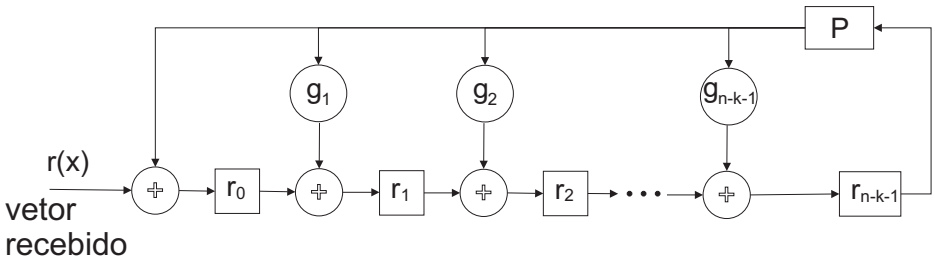


Figura 2.6: *Circuito genérico para a divisão de $r(x)$ por $g(x)$.*

Se o conteúdo do registrador representar o polinômio nulo, então $r(x)$ é um polinômio código e é considerado como o polinômio transmitido. Caso isso não ocorra, então houve adição de erro(s) e alteração da mensagem transmitida. Sendo necessária uma ação do circuito receptor.

A detecção pode ser realizada através da consulta a uma tabela previamente montada, na qual consta o valor da síndrome supondo erro em alguma posição determinada. Considerando que apenas um *bit* da palavra código foi alterado, então pode-se representar o erro por um monônio $e(x)$ de grau r, para $0 \leq r \leq n-1$.

A Tabela 2.1 foi montada para o código cíclico $C(15,5)$ gerado por $g(x) = 1 + x^2 + x^5 + x^6 + x^8 + x^9 + x^{10}$ e a Figura 2.6 ilustra o circuito que realiza o cálculo da síndrome para o mesmo $g(x)$.

Tabela 2.1: *Padrões de erros e suas síndromes correspondentes para o código cíclico C(15,5).*

Padrões de erros de peso um na forma polinomial	**Síndrome na forma polinomial**	**Síndrome na forma vetorial**
$e_{14}(x) = x^{14}$	$s(x) = x + x^4 + x^5 + x^7 + x^8 + x^9$	0100110111
$e_{13}(x) = x^{13}$	$s(x) = 1 + x^3 + x^4 + x^6 + x^7 + x^8$	1001101110
$e_{12}(x) = x^{12}$	$s(x) = x + x^2 + x^3 + x^4 + x^6 + x^8 + x^9$	0111101011
$e_{11}(x) = x^{11}$	$s(x) = 1 + x + x^2 + x^3 + x^5 + x^7 + x^8$	1111010110
$e_{10}(x) = x^{10}$	$s(x) = 1 + x^2 + x^5 + x^6 + x^8 + x^9$	1010011011
$e_9(x) = x^9$	$s(x) = x^9$	0000000001
$e_8(x) = x^8$	$s(x) = x^8$	0000000010
$e_7(x) = x^7$	$s(x) = x^7$	0000000100
$e_6(x) = x^6$	$s(x) = x^6$	0000001000
$e_5(x) = x^5$	$s(x) = x^5$	0000010000
$e_4(x) = x^4$	$s(x) = x^4$	0000100000
$e_3(x) = x^3$	$s(x) = x^3$	0001000000
$e_2(x) = x^2$	$s(x) = x^2$	0010000000
$e_1(x) = x$	$s(x) = x$	0100000000
$e_0(x) = x^0$	$s(x) = x^0$	1000000000

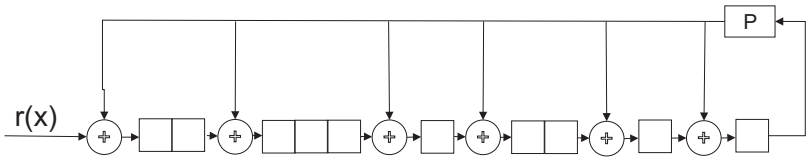


Figura 2.7: *Circuito para o cálculo de síndrome para* $g(x) = 1 + x^2 + x^5 + x^6 + x^8 + x^9 + x^{10}$.

No Exemplo 2.4 é apresentado um exemplo de detecção de erro por meio do cálculo da síndrome.

Exemplo 2.4

Seja o código cíclico binário $C(15,5)$ *gerado por* $g(x) = 1 + x^2 + x^5 + x^6 + x^8 + x^9 + x^{10}$. *Seja o polinômio código* $v(x) = 0$ *enviado, e o erro* $e(x) = x^{10}$ *afeta* $v(x)$. *Logo,* $r(x) = v(x) + e(x) = x^{10}$, *com representação vetorial* $r = [000000000010000]$.

As Figuras 2.8, 2.9 e 2.10 ilustram o carregamento e cálculo do vetor síndrome para $\mathbf{r} = [000000000010000]$.

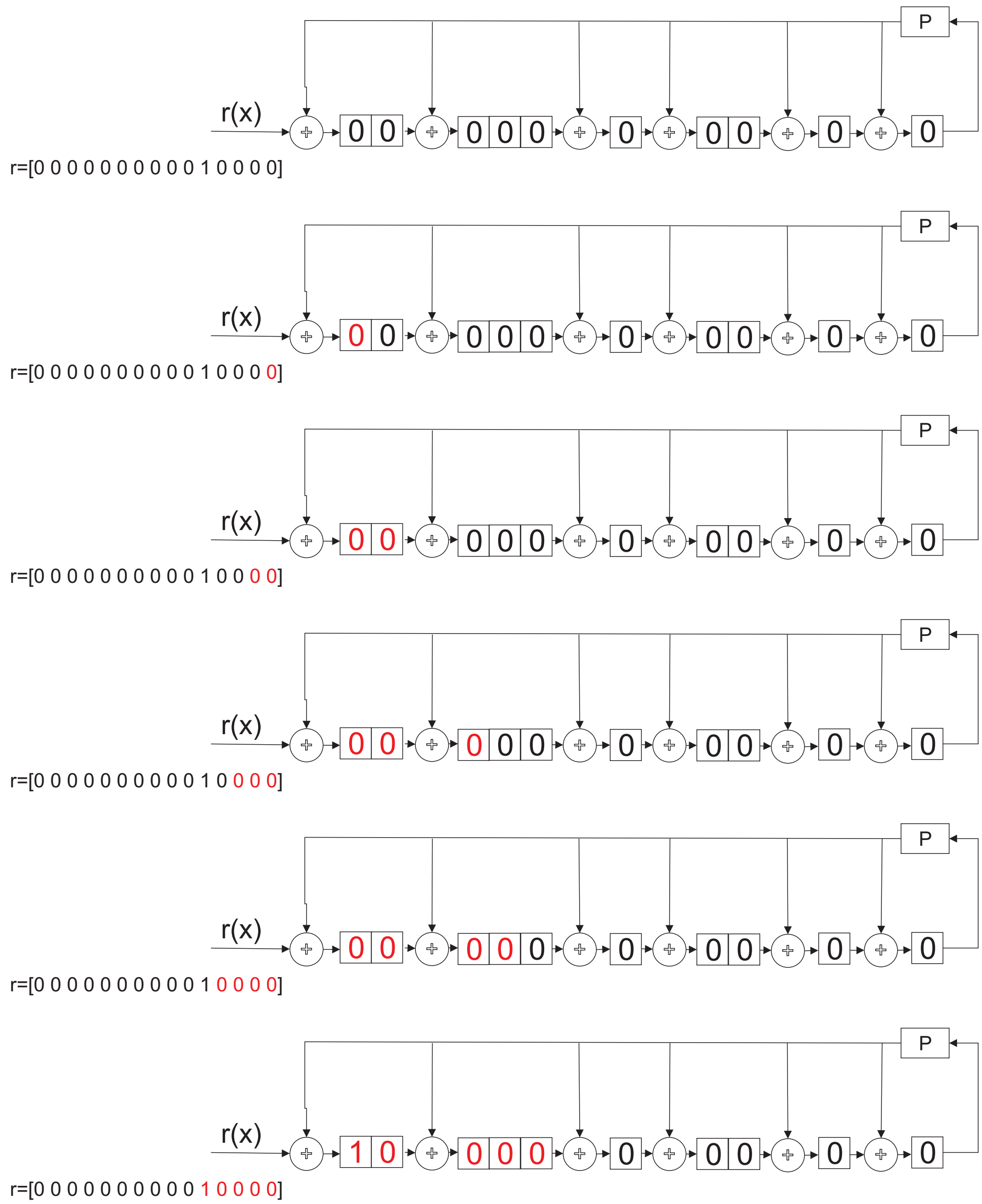


Figura 2.8: *Estado Inicial e deslocamentos 1 ao 5 no cálculo da Síndrome para* $r(x) = x^{10}$ *e* $g(x) = 1 + x^2 + x^5 + x^6 + x^8 + x^9 + x^{10}$ *de C(15,5).*

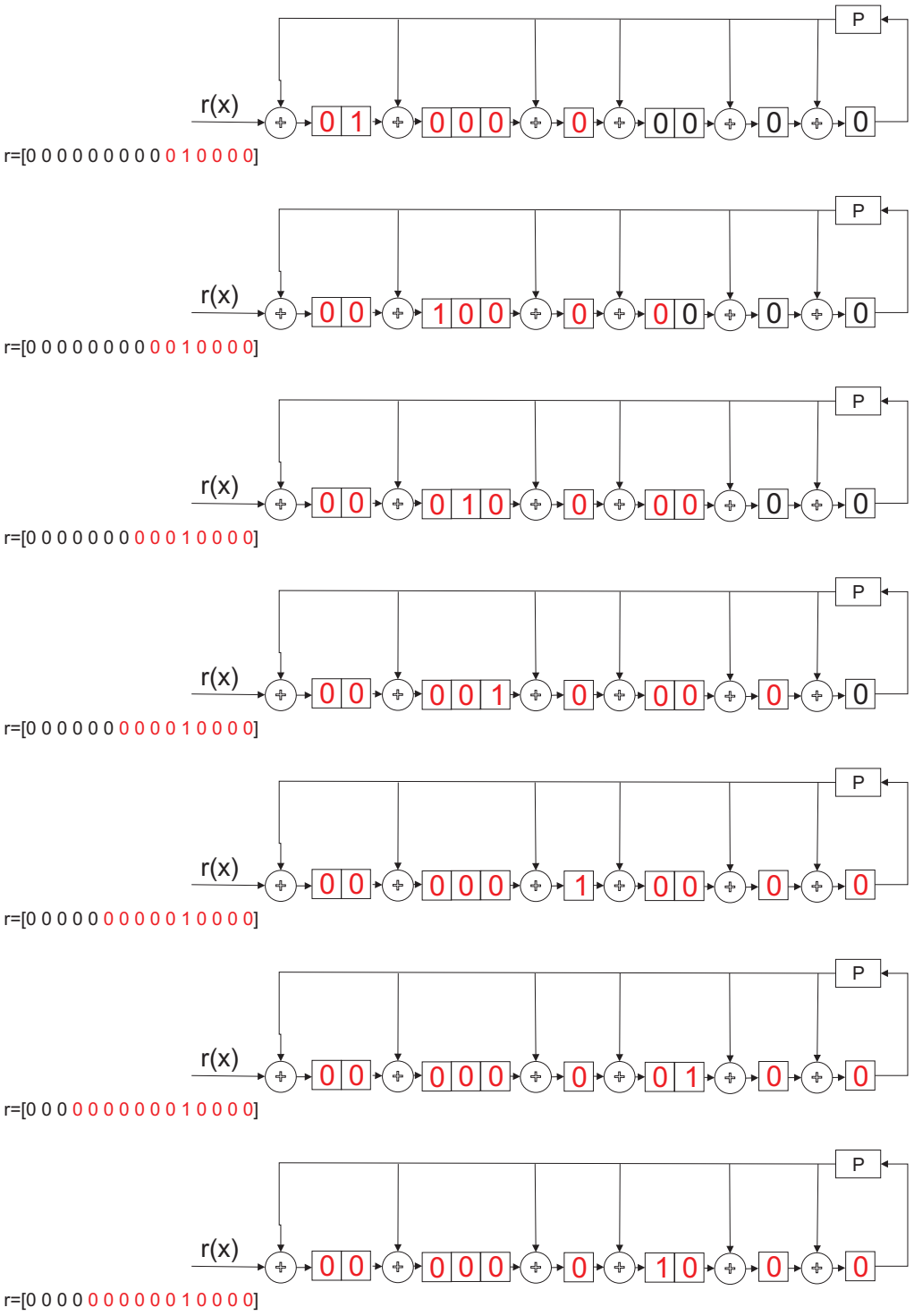


Figura 2.9: *Deslocamentos 6 ao 12 no cálculo da Síndrome para* $r(x) = x^{10}$ *e* $g(x) = 1 + x^2 + x^5 + x^6 + x^8 + x^9 + x^{10}$ *de C(15,5).*

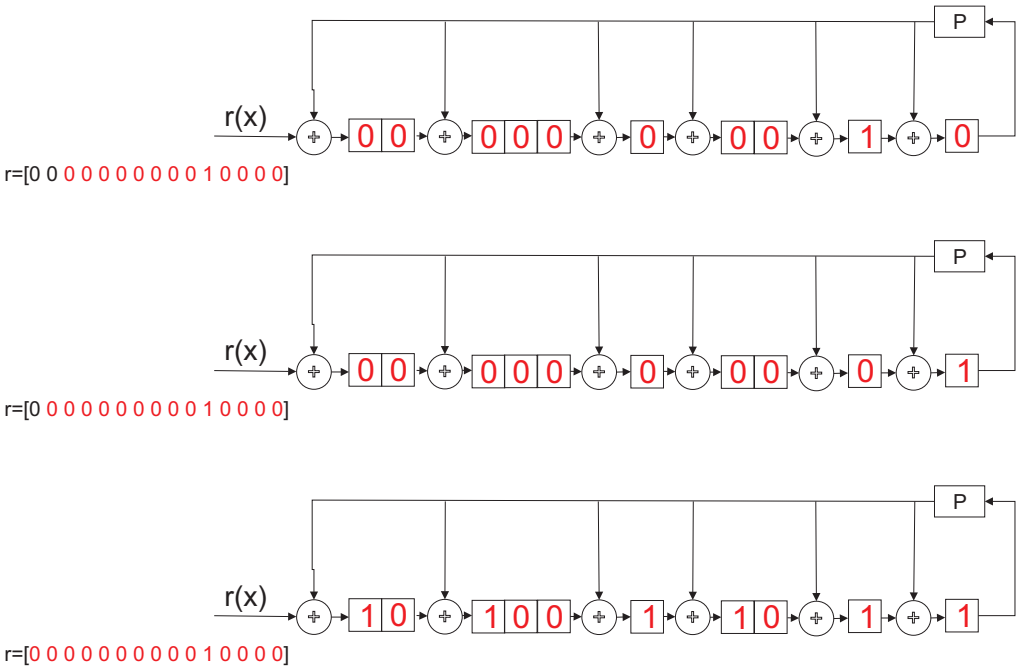


Figura 2.10: *Deslocamentos 13 ao 15 no cálculo da Síndrome para* $r(x) = x^{10}$ *e* $g(x) = 1 + x^2 + x^5 + x^6 + x^8 + x^9 + x^{10}$ *de C(15,5).*

Após o carregamento completo de ***r****, comparando o valor contido em seus registradores com a Tabela 2.1 percebe-se que essa sequência corresponde ao polinômio erro* $e(x) = x^{10}$*, indicando a localização do erro.* □

Com o cálculo da síndrome apenas é possível detectar o(s) erro(s) na palavra recebida. Esse circuito em conjunto com uma lógica combinacional formam a base do decodificador que atua na correção dos erros inseridos pelo canal. Códigos cíclicos são eficientes na correção de erros aleatórios e em surto por meio da construção de decodificadores adequados [13] [14]. A Figura 2.11 ilustra o esquema do decodificador de Meggitt [15] que é a base para os decodificadores implementados neste trabalho. Os blocos com a letra P representam chaves que controlam o fluxo de dados no circuito. No próximo capítulo, o foco é voltado para a correção de erros em surtos.

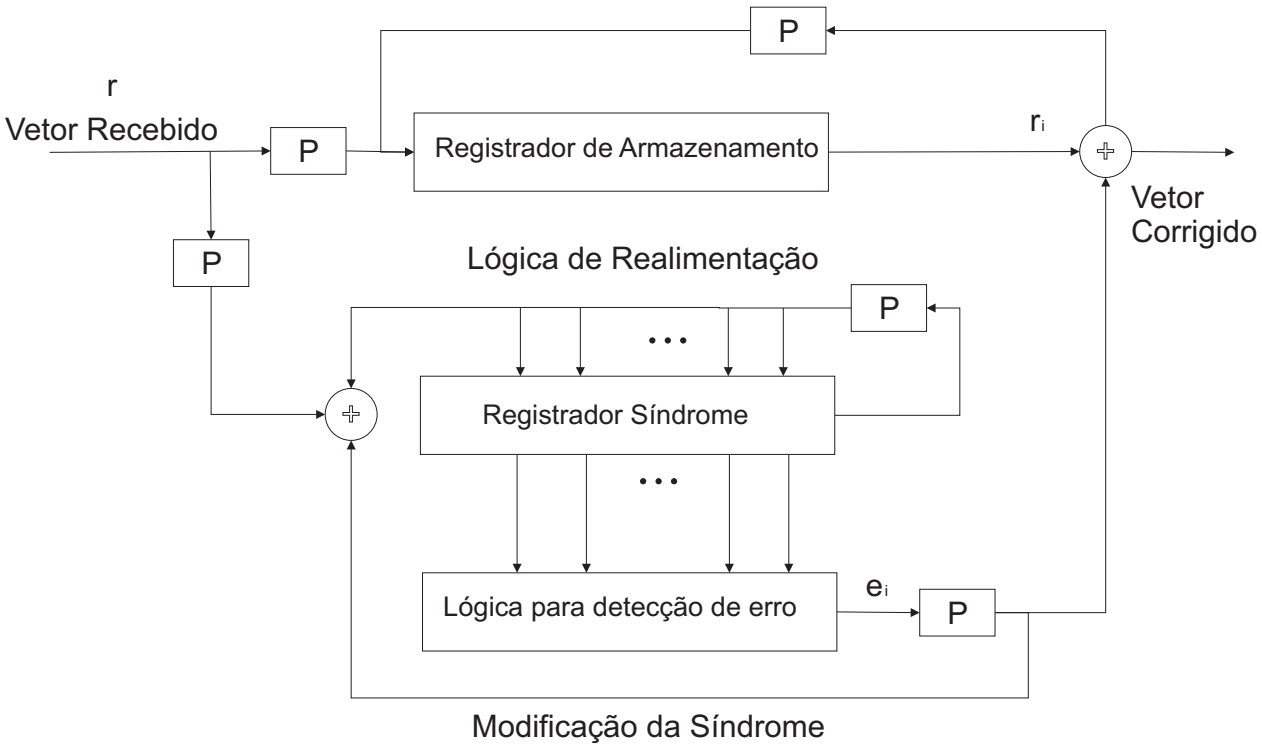


Figura 2.11: *Esquema básico de um decodificador de Meggitt.*

CAPÍTULO 3

CÓDIGOS CORRETORES DE ERROS EM SURTO

Os erros que atuam em sistemas de comunicação podem ser classificados como erros aleatórios e erros em surtos. No primeiro caso, cada dígito da sequência transmitida é afetado por ruído independentemente dos demais. Esse tipo de erro é comum em comunicações espaciais [6]. No entanto, alguns canais de comunicação como: linhas telefônicas ou sistemas de armazenamento magnético, podem inserir erros que afetem uma sequência de *bits* da palavra código transmitida, nesse caso tem-se um erro em surto. Para atuar na correção de erros em surtos, foram desenvolvidos os chamados *Códigos Corretores de Erros em Surto*. Códigos cíclicos foram utilizados ao longo de décadas para correção de erros em surto. Inicialmente estudados por Abramson [16] [17] para a correção de surtos isolados, seus estudos foram generalizados por Fire, originando os *Fire Codes*, utilizados na correção de surtos múltiplos [18]. À medida que o conhecimento na área aumentava, outros códigos cíclicos para erros em surto foram desenvolvidos e o seus desempenhos melhorados [19] [20] [21].

Nesta dissertação é feita uma abordagem geral sobre os códigos cíclicos. Ao longo do capítulo são abordadas as técnicas de correção para erros em surto por armadilha fixa e por armadilha adaptativa usando códigos cíclicos, exemplificando cada caso. Essas duas técnicas são utilizadas no auxílio à correção das manchas bidimensionais.

3.1 CONCEITOS BÁSICOS

Antes de determinar as condições para um código cíclico corrigir erros em surtos, há necessidade de definir o termo surto.

Definição 3.1 – Surto

Um surto de comprimento b é uma sequência binária com b bits consecutivos em que o primeiro e último não são nulos. □

O número de *bits* não nulos em determinado vetor de comprimento qualquer é denotado por p e conhecido como peso do vetor. No caso de surtos de comprimento b tem-se $2 \leq p \leq b$.

Exemplo 3.1

$e = [01101000000000]$; *Surto de comprimento* $b = 4$ *e* $p = 3$. □

Exemplo 3.2

$e = [11000000000100]$; *Surto de comprimento* $b = 5$ *e* $p = 3$. □

Uma primeira inspeção do surto do Exemplo 3.2 é possível considerar que ele possui comprimento $b = 12$. No entanto, ao se usar códigos cíclicos considera-se também os deslocamentos cíclicos do vetor. Ao realizar três deslocamentos para a direita do vetor $\mathbf{e}$ apresentado no Exemplo 3.2, o surto fica com comprimento $b = 5$. Esse menor valor é o adotado, pois admite-se que o surto atacou o fim e o começo do vetor ao mesmo tempo. Surtos desse tipo recebem o nome de surtos *end-around*.

Uma caracterítica essencial de um código desenvolvido para correção de erros em surtos é o comprimento máximo do surto que o código é capaz de corrigir.

Definição 3.2 – Código Corretor de Erros em Surtos

Um código linear é dito código corretor de erros em surtos de comprimento b, ou tem capacidade b de correção em erros em surtos se o código for capaz de corrigir todos os surtos de comprimento b ou menor, mas nem todos os surtos de comprimento $b + 1$. □

Dado o código linear $C(n, k)$, existe uma relação entre os parâmetros do código e a sua capacidade de correção de erros em surto. A busca por códigos que agreguem menor redundância à informação é sempre desejada, para tal a seguinte teorema deve ser obedecido. Teorema 20.1, [6].

Teorema 3.1 – Parâmetros do Código Linear com capacidade b de correção de erros em surto

O número de bits de reduntância n-k para um código linear $C(n,k)$ com capacidade de correção de erros em surto b deve ser maior ou igual a $2b$, isto é,

$$n - k \quad \geq \quad 2b. \tag{3.1}$$

□

Demonstração: A prova para o Teorema 3.1 é composta de duas partes. Primeiramente é necessário provar que nenhum surto de comprimento $2b$ ou menor é uma palavra código e em seguida demonstrar que o número de bits de reduntância, n-k deve ser maior ou igual a l, em que l-1 é o comprimento máximo de um surto que seja palavra código.

Considere a existência de um vetor $\mathbf{v}$ de comprimento $2b$ ou menor, com excessão do caso degenerado em que o comprimento é igual a 1, que seja uma palavra código. Este vetor pode ser expresso como uma soma de dois outros vetores $\mathbf{x}$ e $\mathbf{z}$ de comprimento b ou menor. Os vetores $\mathbf{x}$ e $\mathbf{z}$ podem pertencer à mesma classe lateral no arranjo padrão, no entanto, se um desses vetores for usado como líder de uma classe lateral o outro será classificado como um erro intedectável. Sendo assim, o código não terá capacidade b de correção para erros em surtos, pois existe um surto de comprimento b ou menor que o código não é capaz de corrigir. Então, nenhum surto de comprimento $2b$ ou menor pode ser uma palavra código.

Sejam os 2^l vetores cujas componentes não-nulas estão confinadas nas l primeiras posições. Dois vetores desta classe não podem pertencer à mesma linha do arranjo padrão do código em questão. No entanto, a sua soma, que resulta num vetor de comprimento l ou menor, pode ser uma palavra código. Dessa forma, esses 2^l vetores podem formar as 2^{n-k} classes laterais do código $C(n,k)$. Logo, $n - k \geq l$.

As duas partes desta prova resultam na prova do Teorema 3.1, pois $l = 2b$ e $n - k \geq l$. ■

Do Teorema 3.1 é obtido um limitante superior para a capacidade de correção de surtos de determinado código linear $C(n,k)$ dado pela Inequação 3.2, chamado de limitante de Reiger [22]:

$$b \quad \leq \quad \left\lfloor \frac{n-k}{2} \right\rfloor. \tag{3.2}$$

Códigos que satisfazem o limitante de Reiger na igualdade são considerados códigos ótimos e com base nele foi desenvolvida uma taxa, representada na Fórmula 3.3, usada como uma medida para determinar a eficiência na correção de surtos de determinado código

$$z \quad = \quad \frac{2b}{n-k}. \tag{3.3}$$

A decodificação por armadilha consiste em aprisonar o erro em determinado número de estágios do registrador síndrome. Ela foi desenvolvida por Mitchell em 1962 [23] e pode ser aplicada para a correção tanto de erros em surto quanto erros aleatórios. Inicialmente, é feita uma abordagem para erros aleatórios sobre a técnica, em seguida, ela é aplicada para os erros em surto.

Seja o código cíclico binário linear $C(n,k)$. Um polinômio código $v(x)$, codificado na forma sistemática, é transmitido e afetado pelo polinômio erro $e(x)$, resultando na recepção o polinômio $r(x)$. Seja $s(x)$ a síndrome de $r(x)$ de grau n-k-1 ou menor. Se os erros estiverem confinados nas n-k posições de grau superior de $r(x)$, tem-se que $e(x) = e_k x^k + \ldots + e_{n-2}x^{n-2} + e_{n-1}x^{n-1}$. Após n-k deslocamentos cíclicos de $e(x)$ encontra-se $e^{n-k}(x)$ que de acordo com 2.7 é igual a $e^{n-k}(x) = e_k x^0 + e_{k+1}x^1 + \ldots + e_{n-2}x^{n-k-2} + e_{n-1}x^{n-k-1}$. Por sua vez $e^{n-k}(x) = s(x)^{n-k}$, em que $s(x)^{n-k}$ é a síndrome de $r^{n-k}(x)$. Realizando a multiplicação de x^k por $s(x)^{n-k}$, tem-se

$$x^k s(x)^{n-k} \quad = \quad e(x). \tag{3.4}$$

De 3.4 retira-se a informação para corrigir determinado padrão de erro $e(x)$ confinado nas n-k posições de grau superior de $r(x)$. Dando continuidade, deve-se primeiro calcular a síndrome do vetor recebido. Em seguida, realizar os n-k deslocamentos cíclicos necessários e, após a multiplicação por x^k finalmente adicionar $x^k s(x)^{n-k}$ a $r(x)$. Esse polinômio resultante é considerado o polinômio código transmitido. Caso o erro não esteja localizado nas $n-k$ posições de maior grau de $r(x)$ mas, localizado em n-k posições consecutivas de $r(x)$, inclusive erros *end-around*. É possível, após um certo número de deslocamentos, confinar os erros nas n-k posições de maior grau de $r(x)$ e assim poder efetuar a correção do erro. A seguir são apresentadas as técnicas de decodificação usadas na elaboração desta dissertação.

3.2 DECODIFICAÇÃO DE SURTOS ISOLADOS UTILIZANDO CÓDIGOS CÍCLICOS

3.2.1 DECODIFICAÇÃO POR ARMADILHA FIXA

Para o caso de armadilha fixa considera-se uma armadilha de tamanho igual a b, em que o código $C(n,k)$ é um código corretor de erros em surtos de comprimento b. A idéia do algoritmo é, após o

recebimento completo de $r(x)$ e cálculo do respectivo $s(x)$, realizar deslocamentos do conteúdo do registrador síndrome até aprisionar o surto na armadilha. Seja o código cíclico binário linear $C(n,k)$ com capacidade de correção de erros em surtos b. Um polinômio código $v(x)$, codificado na forma sistemática, é transmitido e afetado pelo polinômio erro $e(x)$, resultando na recepção o polinômio $r(x)$. Seja $s(x)$ a síndrome de $r(x)$ de grau n-k-1 ou menor. Considere o surto confinado nas b posições de grau superior da região de paridade de $r(x)$, isto é, $e(x) = e_{n-k-b}x^{n-k-b} + \ldots + e_{n-k-2}x^{n-k-2} + e_{n-k-1}x^{n-k-1}$. Para este caso, a sequência dos b coeficientes de maior grau de $s(x)$ representa os *bits* do padrão de erro $e(x)$ inserido e os demais coeficientes de $s(x)$ devem ser nulos. Logo, o surto foi aprisionado na armadilha do decodificador. Os erros também podem estar em b posições consecutivas de $r(x)$, sendo do tipo *end-around* ou não, então, existe um número i de deslocamentos que aprisiona o erro nas b posições de ordem superior de $s^i(x)$, dessa forma possibilitando a correção do surto.

A Figura 3.1 ilustra o esquema do decodificador de armadilha simples. Em seguida o algoritmo para decodificação é apresentado.

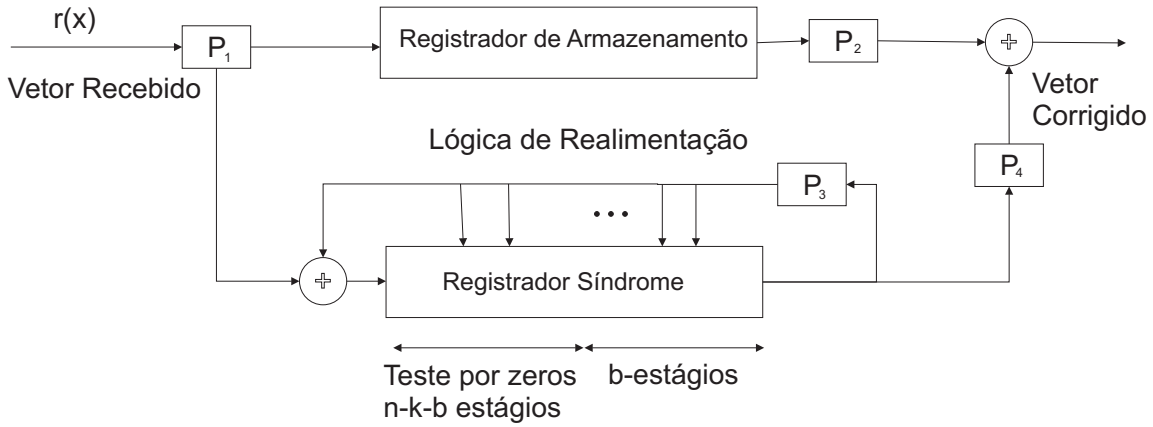


Figura 3.1: *Circuito decodificador genérico por armadilha fixa para um código cíclico C(n,k).*

1. Inicialmente todo o vetor **r** é recebido e armazenado no registrador de armazenamento. O mesmo vetor é usado no cálculo da síndrome que é armazenada no registrador síndrome com as chaves P1 e P3 ativas;

2. São realizados n-k-b deslocamentos do registrador síndrome com P3 ativa em busca de erros na região de paridade. A cada deslocamento é realizado o teste nos n-k-b estágios a esquerda do registrador síndrome. Se em algum momento a soma desses estágios for nula então o erro está localizado na região de paridade. Isto implica que os k dígitos de informação estão livres de erro e podem ser repassados com a ativação de P2. Caso não zere após esses n-k-b deslocamentos

o algoritmo segue para o Passo 3;

3. Nesta etapa a busca é por surtos *end-around* que ataquem ambas as regiões: de paridade e de informação. Se após n-k-b+i deslocamentos para $1 \leq i \leq b$, zerar os n-k-b dígitos à esquerda do registrador síndrome, então os dígitos contidos nos estágios b-i mais à direita do registrador síndrome corrigem os dígitos $x^0, x^1, \ldots, x^{b-i-2}, x^{b-i-1}$ na região de paridade de $r(x)$. E os demais i dígitos do registrador corrigem as posições $x^{n-i}, \ldots, x^{n-2}, x^{n-1}$ na região de informação de $r(x)$. Por meio de sincronização de relógio o registrador síndrome é deslocado com P3 desativada até o momento certo para que os *bits* corrijam o surto inserido. No momento de sincronismo exato, as chaves P2 e P4 são ativadas e a correção é efetuada. Caso o critério de zeramento não seja obedecido após esses n-k deslocamentos o algoritmo segue para o Passo 4;
4. Se após os n-k deslocamentos ainda não for registrada a sequência de zeros desejada então, o circuito realiza mais k deslocamentos para esvaziar os *bits* de informação do registro de armazenameno com P2 ativa. Ao mesmo tempo, o registrador síndrome é deslocado com P3 ativa, sempre observando os n-k-b estágios mais à esquerda do registrador síndrome. No momento em que esses estágios se anularem, P3 é desativada e os b dígitos mais à direita do registrador síndrome corrigem os próximos b dígitos que saírem do registro de armazenamento com P4 ativa.

Se após os n deslocamentos, os n-k-b estágios à esquerda do registrador síndrome não conter apenas zeros significa que um padrão de erro incorrigível foi detectado. A seguir há um exemplo da atuação do decodificador por armadilha fixa na correção de erros em surtos.

Exemplo 3.3

Seja $C(15, 5)$ o código cíclico binário corretor de erros em surtos de tamanho 5 gerado por $g(x) = 1 + x^2 + x^5 + x^6 + x^8 + x^9 + x^{10}$. A Figura 3.2 ilustra o circuito decodificador baseado na Figura 3.1, em destaque os $n - k - b = 15 - 5 - 5 = 5$ estágios que determinam o fim do algoritmo.

*Seja o polinômio $u(x) = 0$ codificado sistematicamente originando $v(x) = 0$. Este polinômio foi transmitido e o polinômio erro $e(x) = x^{10} + x^{11} + x^{12} + x^{14}$ adicionado a $v(x)$ resultando em $r(x) = x^{10} + x^{11} + x^{12} + x^{14}$. A representação vetorial de $r(x)$ $\mathbf{r}$ é carregado e a síndrome resultante é calculada conforme o Exemplo 2.4, sendo assim, a **etapa 1** do algoritmo é finalizada.*

As Figuras 3.3 e 3.4 ilustram cada deslocamento até a etapa dois do algoritmo.

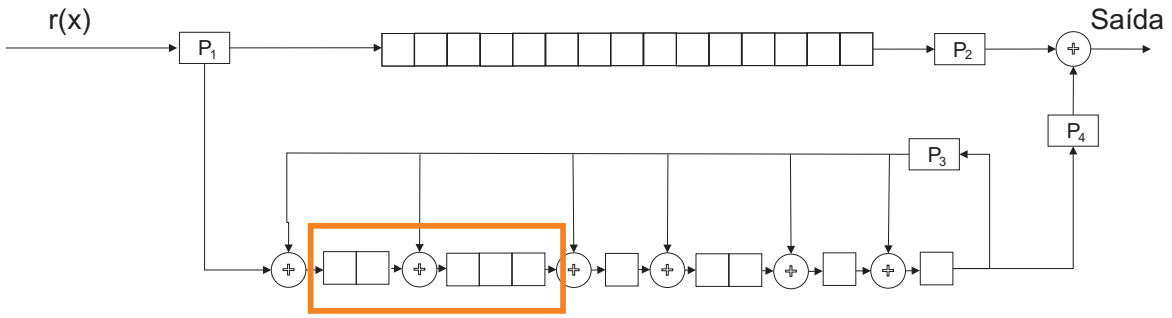


Figura 3.2: *Circuito decodificador para* $g(x) = 1 + x^2 + x^5 + x^6 + x^8 + x^9 + x^{10}$ *do código cíclico C(15,5), em destaque os cinco estágios que determinam o fim do algoritmo.*

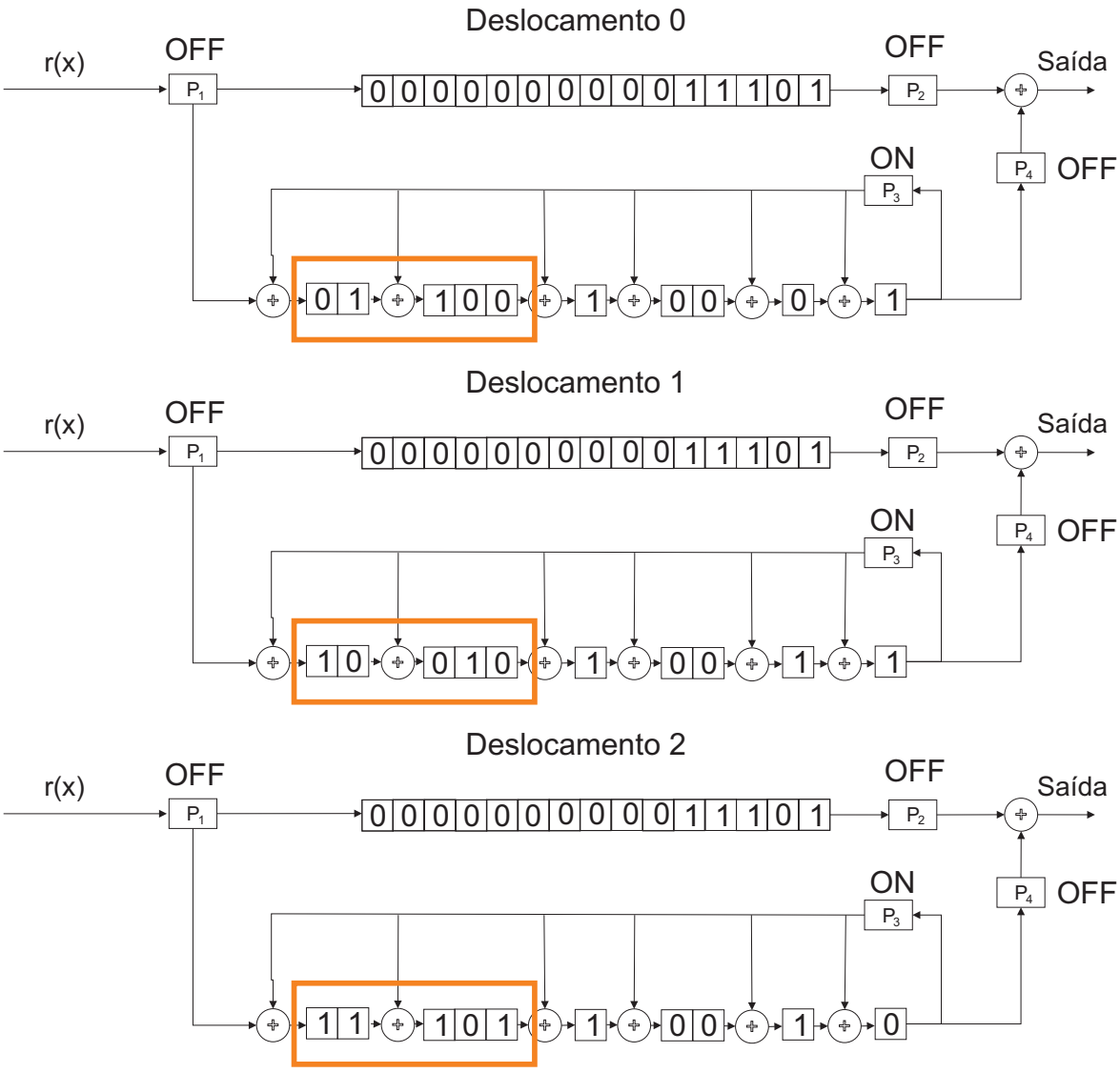


Figura 3.3: *Estado inicial ao deslocamento 2 do registrador síndrome na correção por armadilha simples até a etapa 2 do algoritmo.*

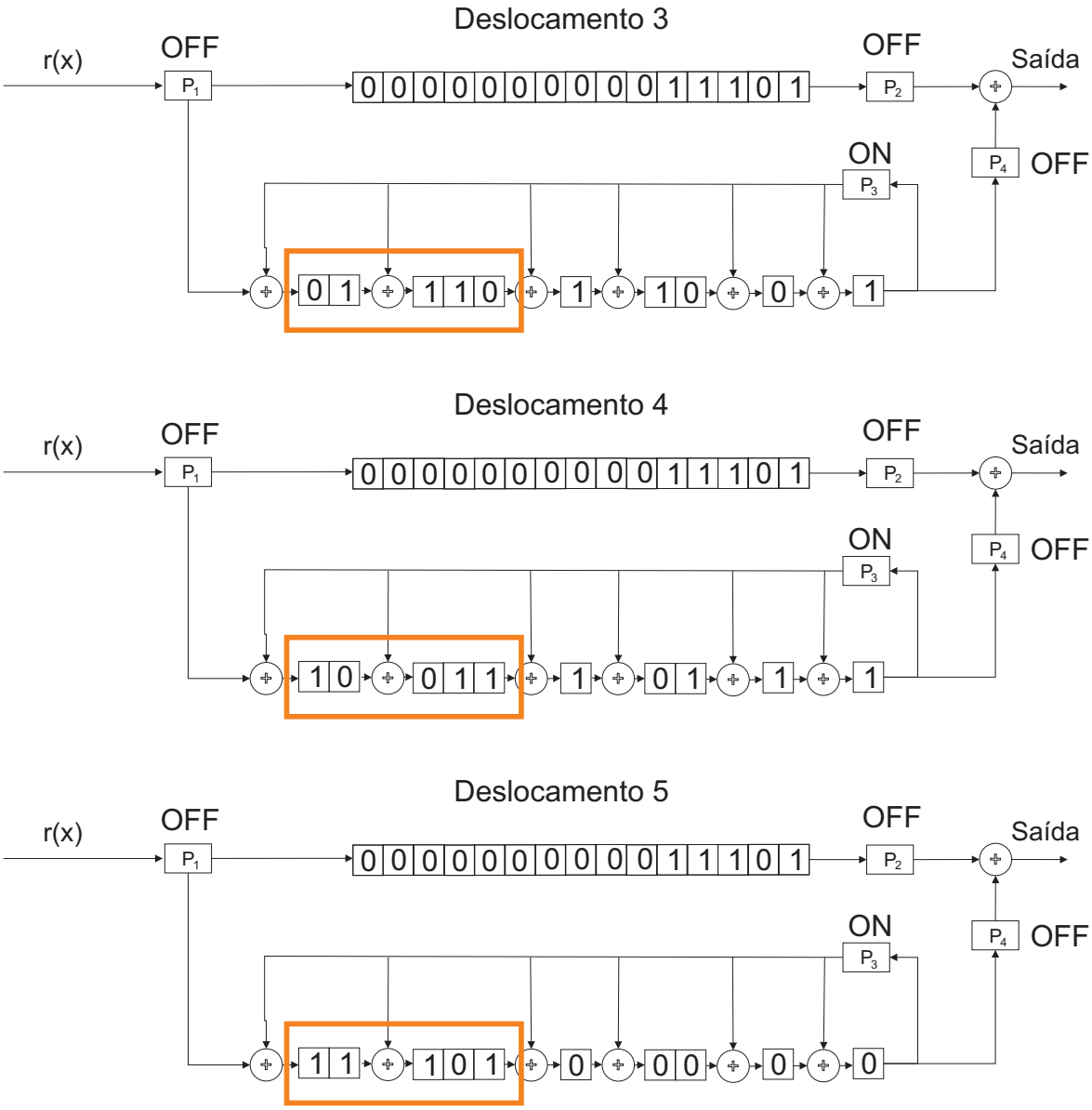


Figura 3.4: *Deslocamentos 3 ao 5 do registrador síndrome na correção por armadilha simples até a etapa 2 do algoritmo.*

Até o momento realizaram-se os n-k-l=5 deslocamentos e não há apenas zeros nos estágios em destaque da Figura 3.4. Então, o algoritmo segue para o Passo três. As Figuras 3.5 e 3.6 ilustram cada deslocamento até o fim da etapa três do algoritmo.

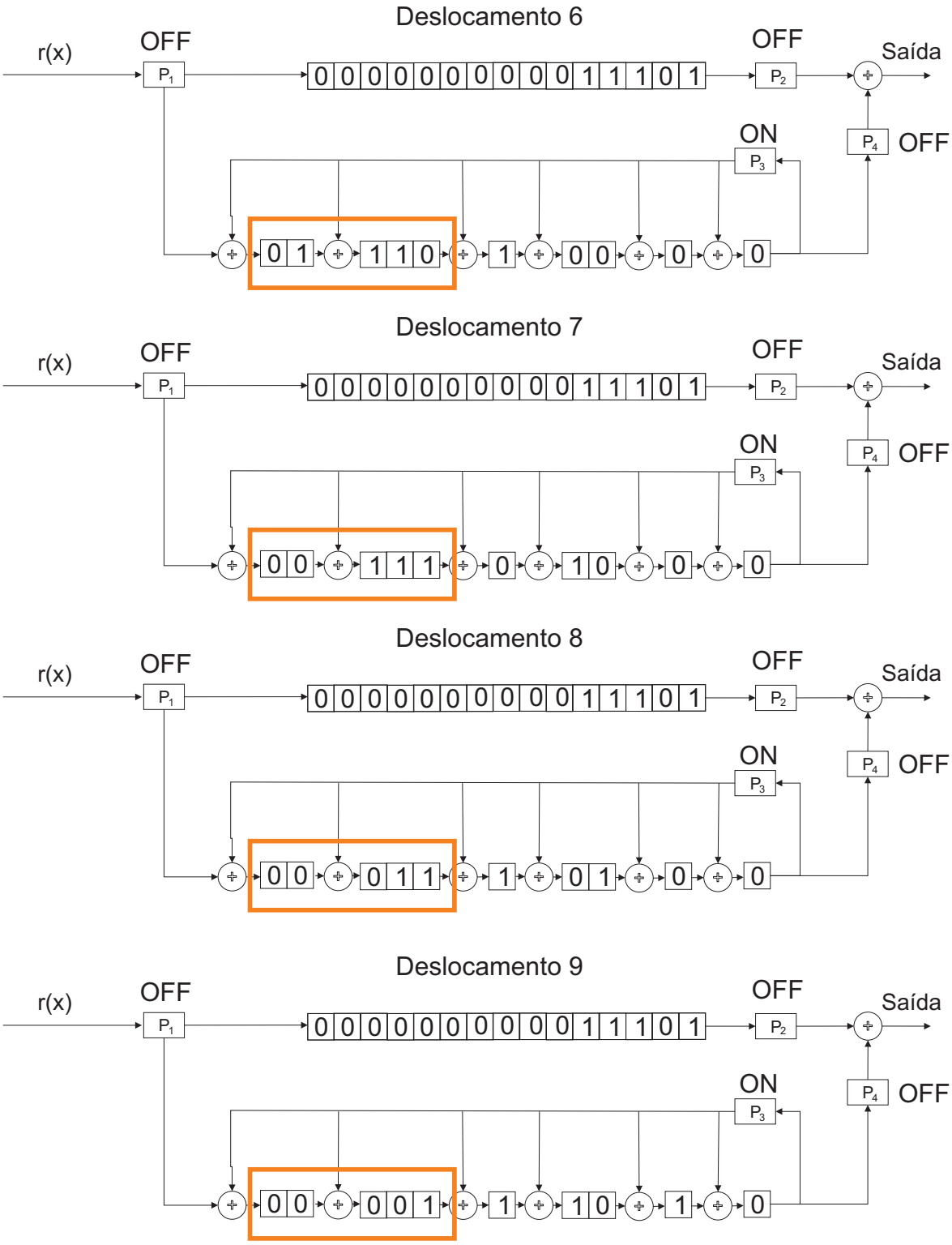


Figura 3.5: *Deslocamentos 6 ao 9 do registrador síndrome na correção por armadilha simples até a etapa 3 do algoritmo.*

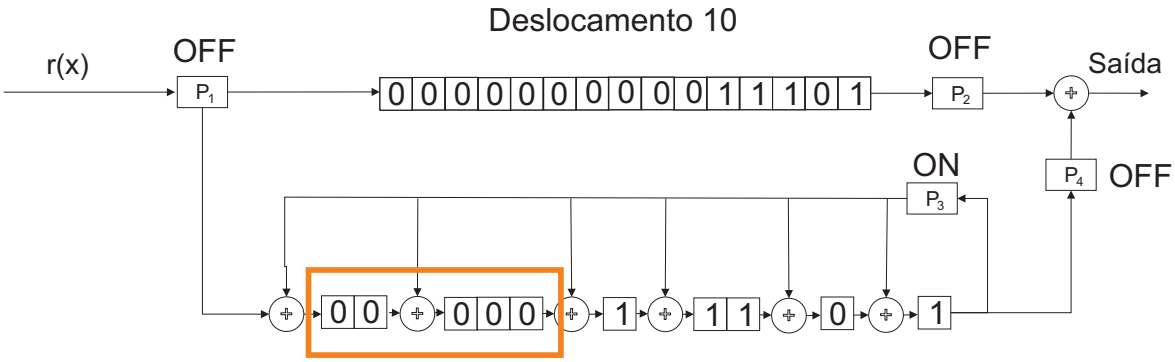


Figura 3.6: *Deslocamento 10 do registrador síndrome do registrador síndrome na correção por armadilha simples até a etapa 3 do algoritmo.*

Neste momento a condição de parada do algoritmo é satisfeita indicando que o erro está aprisionado na armadilha. Em seguida P3 é desligada e P2 e P4 ativadas para a correção do vetor recebido. As Figuras 3.7 e 3.8 ilustram a correção do vetor recebido ***r****.*

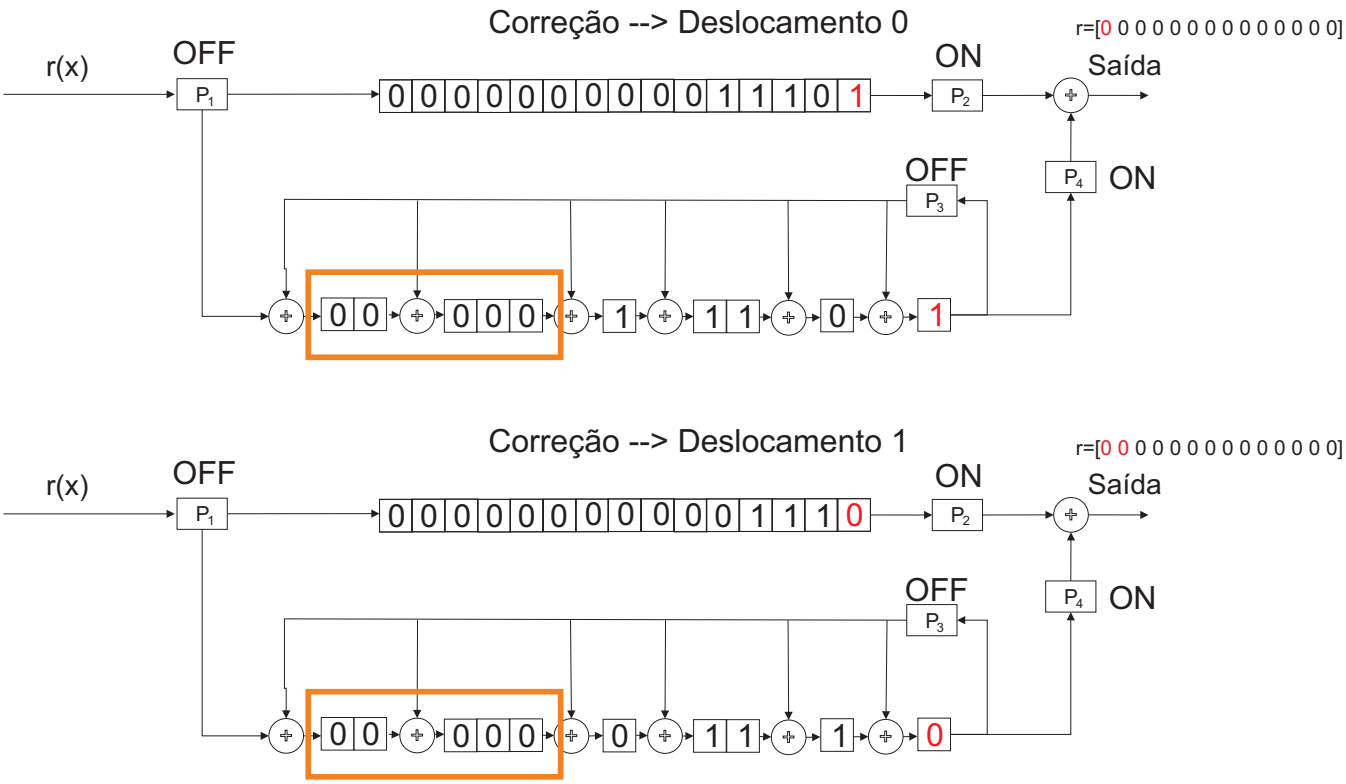


Figura 3.7: *Correção do vetor recebido - estado inicial e deslocamento 1.*

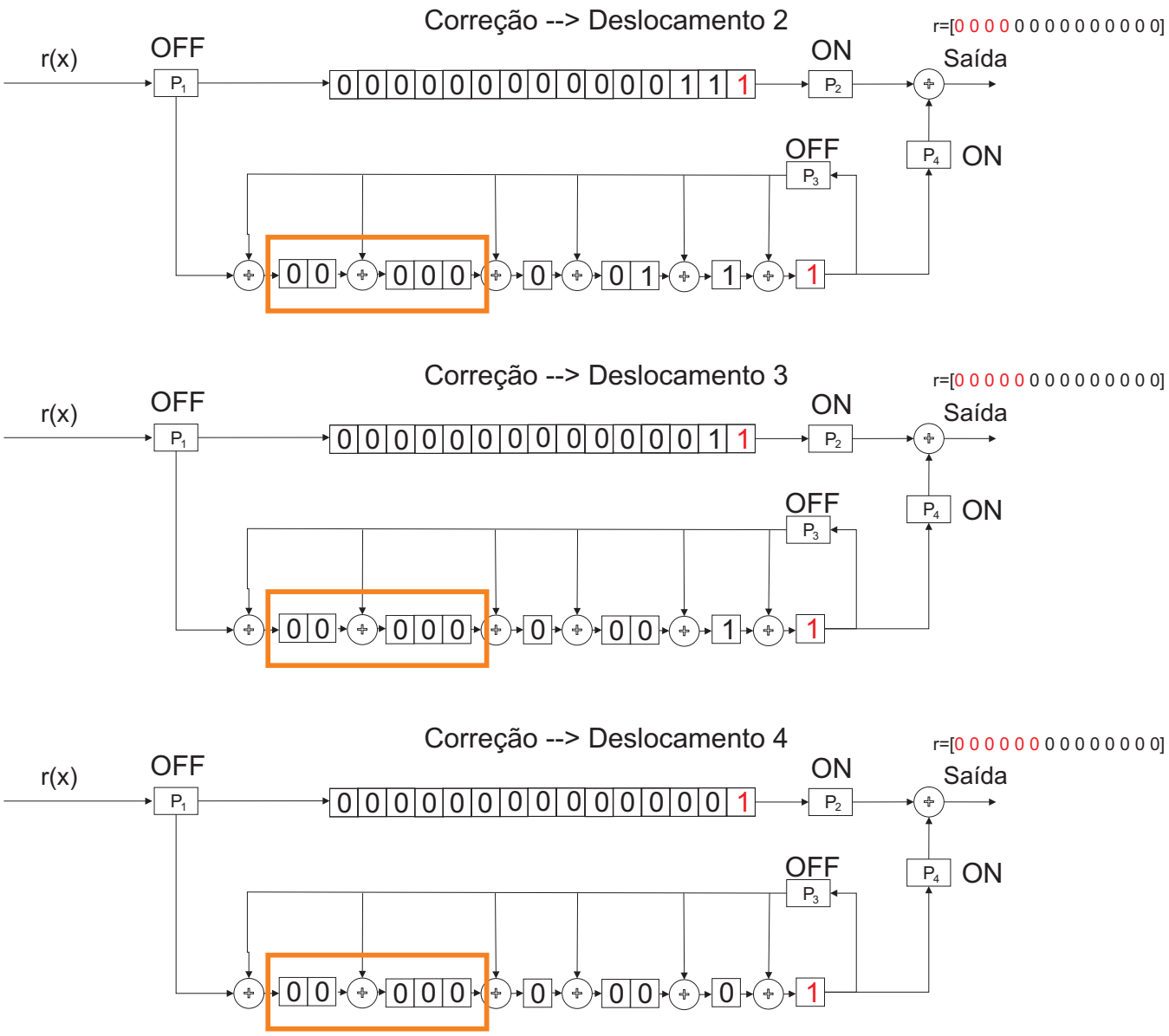


Figura 3.8: *Correção do vetor recebido - deslocamentos 2 a 4.*

Após o completo esvaziamento do registro de armazenamento temos que o surto foi corrigido do vetor recebido ***r****.* □

3.2.2 DECODIFICAÇÃO POR ARMADILHA ADAPTATIVA

Na decodificação por armadilha adaptativa, como o próprio nome induz, não há um tamanho determinado para a armadilha que aprisona o surto. Esta técnica se baseia no fato de que se determinado surto atacar a palavra código é mais provável que esse surto seja o de menor comprimento. Com este decodificador surtos com comprimento até n-k podem ser corrigidos. O circuito para a

decodificação por armadilha é basicamente o mesmo do apresentado na Figura 3.1 com a diferença de que não existe o teste por zeros nos n-k-b estágios mais à esquerda do registrador síndrome. A armadilha vai se adaptando à medida que o algoritmo segue. A seguir apresenta-se o algoritmo para esta decodificação proposto por Gallager [7].

1. Inicialmente, todos os coeficientes do polinômio $r(x)$ são recebidos e armazenados no registrador de armazenamento. O mesmo vetor é usado no cálculo da síndrome que é armazenada no registrador síndrome com P1 e P3 ativas;
2. Em seguida, são realizados n deslocamentos no registrador síndrome com P3 ativa. Em cada deslocamento é armazenado o tamanho da sequência de zeros a contar da extremidade direita do registrador síndrome, denotado por j. Também são armazenados o tamanho da armadilha para cada deslocamento dado por A = n-k-j, bem como o número do deslocamento realizado e a sequência presente no registrador síndrome;
3. No fim dos n deslocamentos há um histórico com os valores de A e cada sequência associada. O surto de menor comprimento está confinado nos A' estágios mais à esquerda do registrador síndrome. Em que A' corresponde ao menor valor de A do histórico armazenado.
4. Com o conhecimento do surto e do valor do deslocamento realizado o sincronismo é feito e realizado o deslocamento do vetor recebido até o momento em que o surto corrija os *bits* afetados no vetor recebido com P2 e P4 ativadas.

A seguir há um exemplo da atuação do decodificador por armadilha adaptativa na correção de erros em surtos.

Exemplo 3.4

Seja $C(15, 5)$ o código cíclico binário corretor de erros em surtos de tamanho 5 gerado por $g(x) = 1 + x^2 + x^5 + x^6 + x^8 + x^9 + x^{10}$. O circuito decodificador para este $g(x)$ é igual ao apresentado na Figura 3.2 com a diferença de que não há preocupação com os estágios iniciais do registro síndrome.

Considerando os mesmos polinômios informação, código e recebido do Exemplo 3.3, podemos dar início ao algoritmo para correção de armadilha adaptativa.

Segundo o algoritmo, é necessário realizar $n = 15$ deslocamentos e analisar a cada deslocamento o tamanho da janela e o surto que ela armazena. A fim de não tornar repetitivo, já se considera os dez deslocamentos realizados no Exemplo 3.3. As Figuras 3.9 e 3.10 ilustram os cinco deslocamentos restantes.

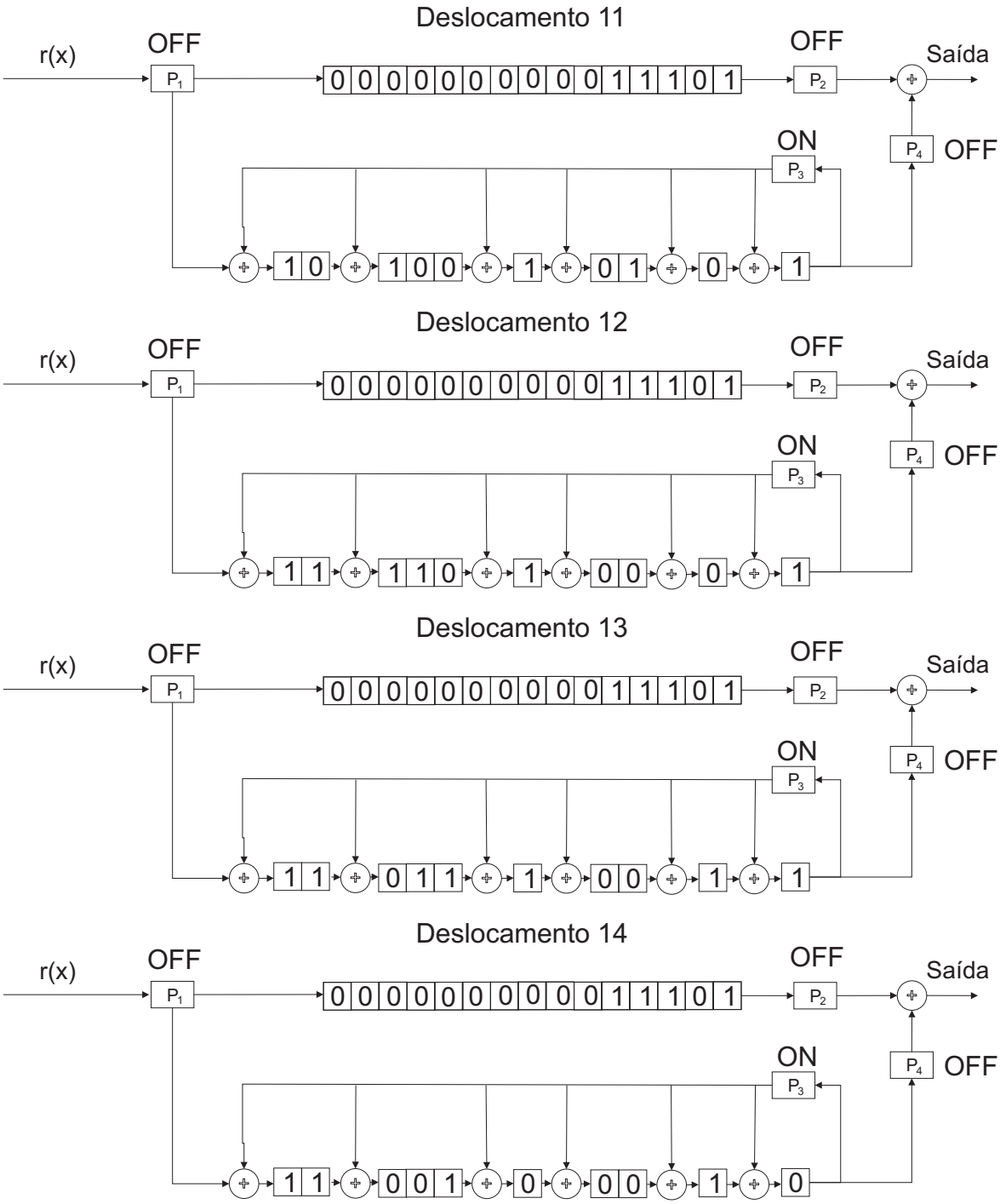


Figura 3.9: *Deslocamentos finais do registrador síndrome para a decodificação por armadilha adaptativa.*

Após os quinze deslocamentos a Tabela 3.1 é construída. Nela apresenta-se o tamanho de cada armadilha e o surto aprisionado por ela em cada deslocamento.

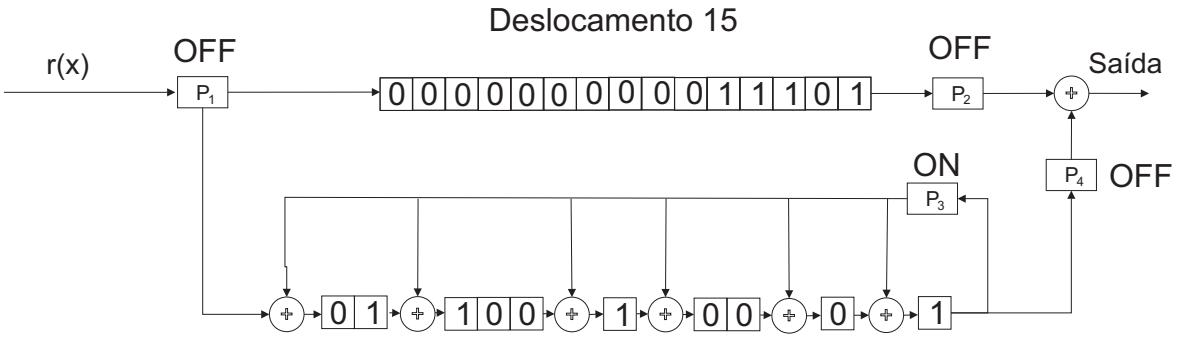


Figura 3.10: *Deslocamento final do registrador síndrome para a decodificação por armadilha adaptativa.*

Tabela 3.1: *Valor do tamanho da armadilha e o surto aprisionado para cada deslocamento.*

Deslocamento	**Tamanho da armadilha** (A)	**Surto aprisionado**
1	10	1 0 0 1 0 1 0 0 1 1
2	9	1 1 1 0 1 1 0 0 1 0
3	10	0 1 1 1 0 1 1 0 0 1
4	10	1 0 0 1 1 1 0 1 1 1
5	**5**	**1 1 1 0 1 0 0 0 0 0**
6	6	0 1 1 1 0 1 0 0 0 0
7	7	0 0 1 1 1 0 1 0 0 0
8	8	0 0 0 1 1 1 0 1 0 0
9	9	0 0 0 0 1 1 1 0 1 0
10	10	0 0 0 0 0 1 1 1 0 1
11	10	1 0 1 0 0 1 0 1 0 1
12	10	1 1 1 1 0 1 0 0 0 1
13	10	1 1 0 1 1 1 0 0 1 1
14	9	1 1 0 0 1 0 0 0 1 0
15	10	0 1 1 0 0 1 0 0 0 1

A linha em negrito, corresponde ao quinto deslocamento e consiste no valor de A', logo esta sequência é o surto considerado que corresponde à sequência dos coeficientes do polinômio erro inserido. Com o sincronismo dos deslocamentos dos registradores a correção é efetuada da mesma forma que nas Figuras 3.7 e 3.8.

Após exemplificar os algoritmos, pode-se agora desenvolver a teoria para a correção de manchas bidimensionais.

CAPÍTULO 4

SIMULAÇÃO DE CORREÇÃO DE MANCHAS DE ERROS EM ARRANJOS BIDIMENSIONAIS

MANCHAS de erros atuam em sistemas que utilizam matrizes para armazenar os *bits* de informação, como o sistema de armazenamento e transmissão de imagem, os sistemas de fitas magnéticas, os *chips*, entre outros [24]. A correção dessas manchas de erros foi investigada por vários autores, que consideram determinada forma [25] e [26], ou comprimento dos vetores que compõem as manchas [27] [28] que atacam a matriz de informação. Entre os trabalhos desenvolvidos nesta área pode-se referenciar Almeida e Palazzo [29], que introduziram o uso de reticulados na correção de manchas, e Rocha [24], que derivou as condições para o desenvolvimento do simples entrelaçamento (apenas em uma dimensão), em que a correção é efetuada por meio do uso de códigos corretores de erros aleatórios sobre uma dimensão do arranjo.

Esta dissertação é uma aplicação do que foi desenvolvido em [5], em que é aplicado entrelaçamento e código corrretor de erros em surto apenas em uma dimensão do arranjo. Outras técnicas que usam códigos corretores em duas dimensões [30] [31] podem ser empregadas para a correção de manchas em arranjos bidimensionais. Os blocos que compõem o sistema desenvolvido implementados com o uso do programa Matlab®, foram introduzidos no Capítulo 1. Neste capítulo são apresentados os resultados e o desenvolvimento do trabalho.

4.1 SISTEMA DESENVOLVIDO

Para avaliar o uso do sistema considerou-se imagens extraídas de [32] como fonte de informação. Imagens com resoluções de 128×128 *pixels* na escala de cinza foram convertidas para matrizes de dimensões 128×128, cujos elementos são números decimais. Cada elemento da matriz foi convertido em sua representação binária com 8 *bits*, resultando em matrizes binárias de dimensões 128×1.024. Por sua vez, esta matriz binária é dividida em matrizes menores. Essa divisão é necessária para que o processo de codificação seja realizado. De modo a tornar as dimensões das matrizes divisíveis pelos parâmetros do código cíclico binário linear $C(n,k)$, foi feito um preenchimento com zeros nas últimas linhas e ou colunas da matriz original. Obtendo um número inteiro de blocos a serem enviados pelo canal. O procedimento a seguir descreve o envio de um bloco obtido da matriz da imagem binária.

Um bloco de informação é representado por uma matriz $\mathbf{U_{n\times k}}$. As linhas da matriz $\mathbf{U_{n\times k}}$ são codificadas por um codificador sistemático de determinado código cíclico $C(n,k)$, como o da Figura 2.2, resultando na matriz $\mathbf{V_{n\times n}}$. Em seguida essa matriz é entrelaçada conforme descrito na Seção 4.2 e origina a matriz $\mathbf{V^*_{n\times n}}$. A nova matriz $\mathbf{V^*_{n\times n}}$ é considerada a matriz transmitida e a ela é adicionada uma mancha de erro. O tamanho e forma da moldura da mancha podem ser escolhidos aleatoriamente pelo sistema ou determinados pelo usuário. Dentre os tipos avaliados encontram-se moldura na forma quadrada, retangular ou cruz. A criação das manchas é detalhada na Seção 4.3. Esta mancha é somada módulo 2 aleatoriarmente à $\mathbf{V^*_{n\times n}}$, por meio da escolha de um elemento da matriz que demarca o início da região a ser afetada pela mancha. A Seção 4.3 ilustra exemplos dessa soma.

A matriz resultante da adição da mancha é considerada como a matriz recebida $\mathbf{R^*_{n\times n}}$ pelo sistema. Essa matriz $\mathbf{R^*_{n\times n}}$ é desentrelaçada resultando na matriz $\mathbf{R_{n\times n}}$. Após isso, cada linha de $\mathbf{R_{n\times n}}$ é decodificada pelo decodificador armadilha para erros em surtos do código cíclico $C(n,k)$,como o da Figura 3.1, resultando na matriz $\mathbf{\hat{U}_{n\times k}}$.

Os blocos são enviados em sequência e uma vez finalizado o envio de todos os blocos da matriz informação da imagem, tem-se, após a remoção dos zeros adicionados, uma matriz binária de dimensões 128×1024. Feito o devido agrupamento de 8 *bits* dos seus elementos constrói-se uma matriz com dimensões 128×128 composta de elementos decimais. Após isso a imagem pode ser reconstruída, finalizando seu envio. Na Figura 4.1 é apresentado o esquema de envio de um bloco de informação com a identificação das matrizes ao longo de cada etapa. No fim do envio de todos os blocos da imagem tem-se que um total de manchas inseridas na matriz binária da imagem.

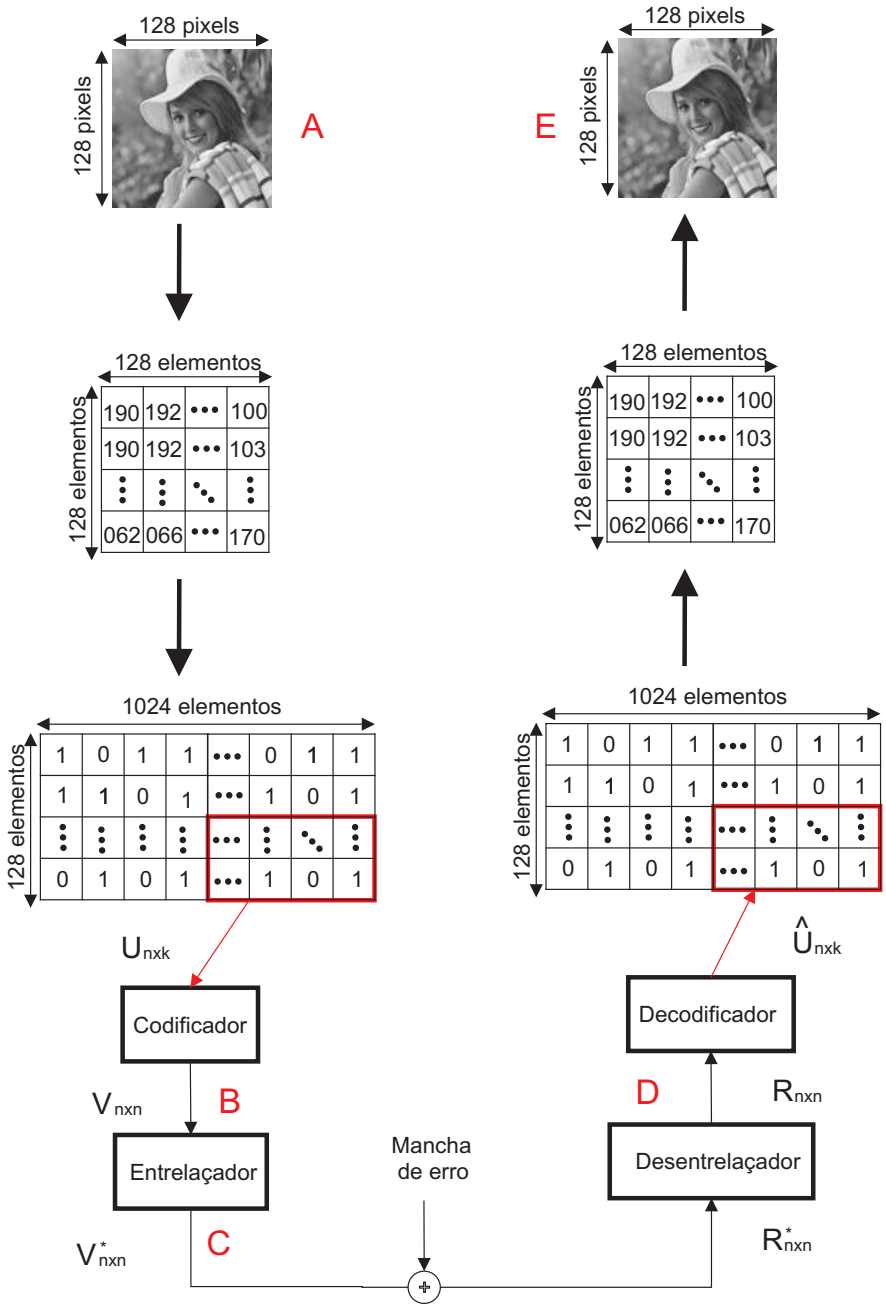


Figura 4.1: *Esquema de transmissão de um bloco da matriz de informação da imagem, supondo total eliminação dos erros adicionados à imagem.*

Na descrição anterior, cada mancha se aloca em um bloco específico a ser enviado. Foi feito um outro estudo em que um número de manchas específico é adicionado a matriz binária da imagem, codificada e entrelaçada. Da mesma forma que o processo anterior ocorre a divisão por blocos para a codificação e entrelaçamento, no entanto, é realizada a adição da quantidade específica de manchas em toda a matriz, podendo ocorrer sobreposição de manchas inclusive. Essa matriz recebida é

então novamente dividida em blocos para ser desentrelaçada e decodificada, dando continuidade a construção da imagem recebida assim como anteriormente.

4.2 ENTRELAÇADOR

Cada elemento da matriz $\mathbf{V_{n\times n}}$ é representado por índices (j,i) para $0 \leq i \leq n-1$ e $0 \leq j \leq n-1$. Considera-se os índices i para as linhas e j para as colunas de $\mathbf{V}$ respectivamente. Essa matriz é entrelaçada, deslocando os elementos da posição (j,i) para $(j, j-i)\texttt{mod}\ n$. Essa transformação linear, dada por 4.1, é efetuada pela matriz apresentada em 4.2.

$$(j,i)\mathbf{T} = (j, j-i)\texttt{mod}\ n = (j', i'). \tag{4.1}$$

$$\mathbf{T} = \begin{bmatrix} 1 & 1 \\ 0 & -1 \end{bmatrix}. \tag{4.2}$$

O Exemplo 4.1 mostra o entrelaçamento de uma matriz $\mathbf{V_{n\times n}}$, resultando na matriz $\mathbf{V^*_{n\times n}}$.

Exemplo 4.1

Entrelaçamento da matriz $\mathbf{V_{7\times 7}}$ pela transformação linear da Equação 4.1.

Matriz Original

$$\mathbf{V} = \begin{bmatrix} \clubsuit_0 & \clubsuit_1 & \clubsuit_2 & \clubsuit_3 & \clubsuit_4 & \clubsuit_5 & \clubsuit_6 \\ \diamondsuit_0 & \diamondsuit_1 & \diamondsuit_2 & \diamondsuit_3 & \diamondsuit_4 & \diamondsuit_5 & \diamondsuit_6 \\ \spadesuit_0 & \spadesuit_1 & \spadesuit_2 & \spadesuit_3 & \spadesuit_4 & \spadesuit_5 & \spadesuit_6 \\ \square_0 & \square_1 & \square_2 & \square_3 & \square_4 & \square_5 & \square_6 \\ \heartsuit_0 & \heartsuit_1 & \heartsuit_2 & \heartsuit_3 & \heartsuit_4 & \heartsuit_5 & \heartsuit_6 \\ \triangle_0 & \triangle_1 & \triangle_2 & \triangle_3 & \triangle_4 & \triangle_5 & \triangle_6 \\ \maltese_0 & \maltese_1 & \maltese_2 & \maltese_3 & \maltese_4 & \maltese_5 & \maltese_6 \end{bmatrix}. \tag{4.3}$$

Matriz entrelaçada

$$\mathbf{V}^* = \begin{bmatrix} \clubsuit_0 & \diamondsuit_1 & \spadesuit_2 & \square_3 & \heartsuit_4 & \triangle_5 & \maltese_6 \\ \maltese_0 & \clubsuit_1 & \diamondsuit_2 & \spadesuit_3 & \square_4 & \heartsuit_5 & \triangle_6 \\ \triangle_0 & \maltese_1 & \clubsuit_2 & \diamondsuit_3 & \spadesuit_4 & \square_5 & \heartsuit_6 \\ \heartsuit_0 & \triangle_1 & \maltese_2 & \clubsuit_3 & \diamondsuit_4 & \spadesuit_5 & \square_6 \\ \square_0 & \heartsuit_1 & \triangle_2 & \maltese_3 & \clubsuit_4 & \diamondsuit_5 & \spadesuit_6 \\ \spadesuit_0 & \square_1 & \heartsuit_2 & \triangle_3 & \maltese_4 & \clubsuit_5 & \diamondsuit_6 \\ \diamondsuit_0 & \spadesuit_1 & \square_2 & \heartsuit_3 & \triangle_4 & \maltese_5 & \clubsuit_6 \end{bmatrix} \tag{4.4}$$

□

O elemento $(1,1)$ da Matriz apresentada em 4.3 representado por $\diamondsuit_1$, tem sua posição alterada de acordo com a Equação 4.1 para $(1, 1-1)\texttt{mod}\ 7 = (1,0)\texttt{mod}\ 7 = (j', i')$ na nova matriz entrelaçada.

O mesmo ocorre para o elemento $(6,4)$ da Matriz 4.3 representado por $\heartsuit_6$, que tem sua posição alterada de acordo com a Equação 4.1 para $(6, 6-4)\texttt{mod}\ 7 = (6,2)\texttt{mod}\ 7 = (j', i')$ na nova matriz entrelaçada. A modificação das posições dos elementos é validada na matriz apresentada em 4.4. Percebe-se que a coordenada corresponde à coluna não é alterada para nenhum elemento, como sugere a transformação da Equação 4.1.

4.3 GERAÇÃO DAS MANCHAS DE ERRO

Nesta dissertação, foram consideradas manchas de erros que afetam a matriz informação codificada e entrelaçada, $\mathbf{V}^*_{\mathbf{n}\times\mathbf{n}}$, e que possuem a moldura na forma de quadrado, retângulo ou cruz. Cada mancha de erro é gerada a partir de uma distribuição aleatória de $1s$ e $0s$. A partir deste momento o termo mancha se refere à mancha de erro.

A mancha com moldura quadrada possui dimensão, a, para $2 \leq a \leq n$, e pode possuir peso, p, dado por $0 \leq p \leq a^2$. O Exemplo 4.2 ilustra uma mancha quadrada com $a = 8$ e $p = 32$.

Exemplo 4.2

Mancha quadrada com $a = 8$ *e* $p = 32$

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \end{bmatrix}. \tag{4.5}$$

□

A mancha com moldura retangular possui dimensões, a e b, para $2 \leq a \leq n$ e $2 \leq b \leq n$, excetuando os casos em que $a = b$. Em que a representa o número de linhas e b o número de colunas da mancha. De maneira análoga o peso, p, é dado por $0 \leq p \leq ab$. O Exemplo 4.3 ilustra uma mancha retangular com $a = 7$, $b = 10$ e $p = 28$.

Exemplo 4.3

Mancha retangular com $a = 7$, $b = 10$ *e* $p = 28$.

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 \end{bmatrix}. \tag{4.6}$$

□

A mancha com moldura em cruz possui dimensões, a e b, para $3 \leq a \leq n$ e $3 \leq b \leq n$. Em que a representa o número de linhas e b o número de colunas da mancha. Esse formato de cruz se diferencia dos demais modelos, pois para este caso os *bits* das extremidades da mancha nessa moldura são zerados. Para este caso o peso, p é dado por $0 \leq p \leq (ab - 4)$. O Exemplo 4.4 ilustra uma mancha cruz com $a = 6$, $b = 3$ e $p = 8$.

Exemplo 4.4

Mancha em cruz com $a = 6$, $b = 3$ e $p = 6$. Em negrito os elementos nulos das extremidades para formar um padrão em cruz.

$$\begin{bmatrix} \mathbf{0} & 1 & \mathbf{0} \\ 1 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \\ \mathbf{0} & 0 & \mathbf{0} \end{bmatrix}. \tag{4.7}$$

□

A fim de verificar a distribuição de geração das manchas, foram construídos histogramas para cada tipo de mancha. Os histogramas apresentados nas Figuras 4.2 e 4.3 ilustram a quantidade de manchas com moldura quadrada geradas em função do peso da mancha, p.

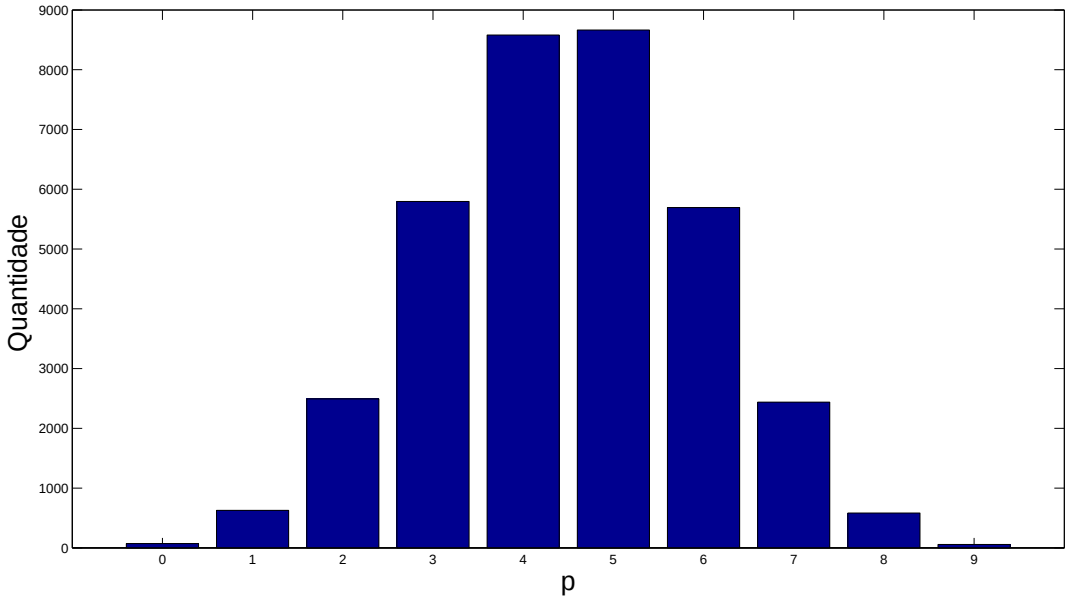


Figura 4.2: *Histograma com a distribuição para manchas quadradas com dimensão $a = 3$.*

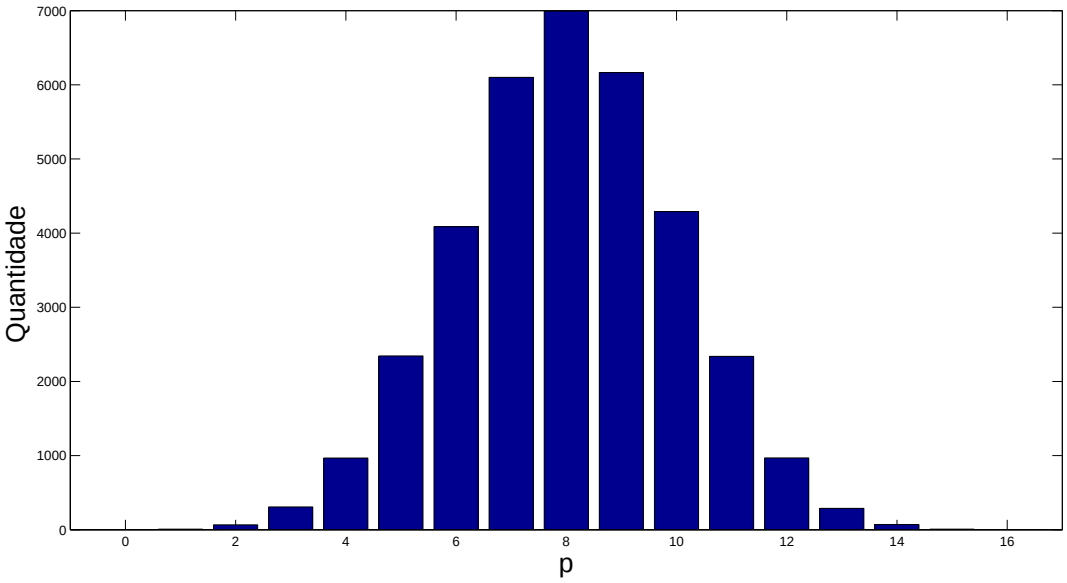


Figura 4.3: *Histograma com a distribuição para manchas quadradas com dimensão* $a = 4$.

Os histogramas apresentados nas Figuras 4.4 e 4.5 ilustram a quantidade de manchas com moldura retangular geradas em função do peso da mancha, p.

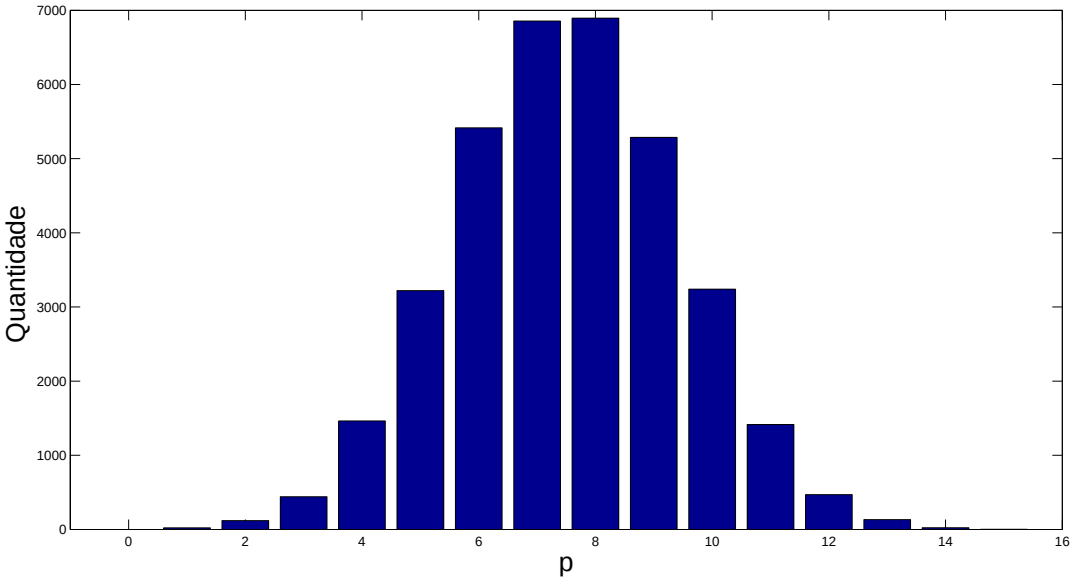


Figura 4.4: *Histograma com a distribuição para manchas retangulares com dimensões* $a = 3$ *e* $b = 5$.

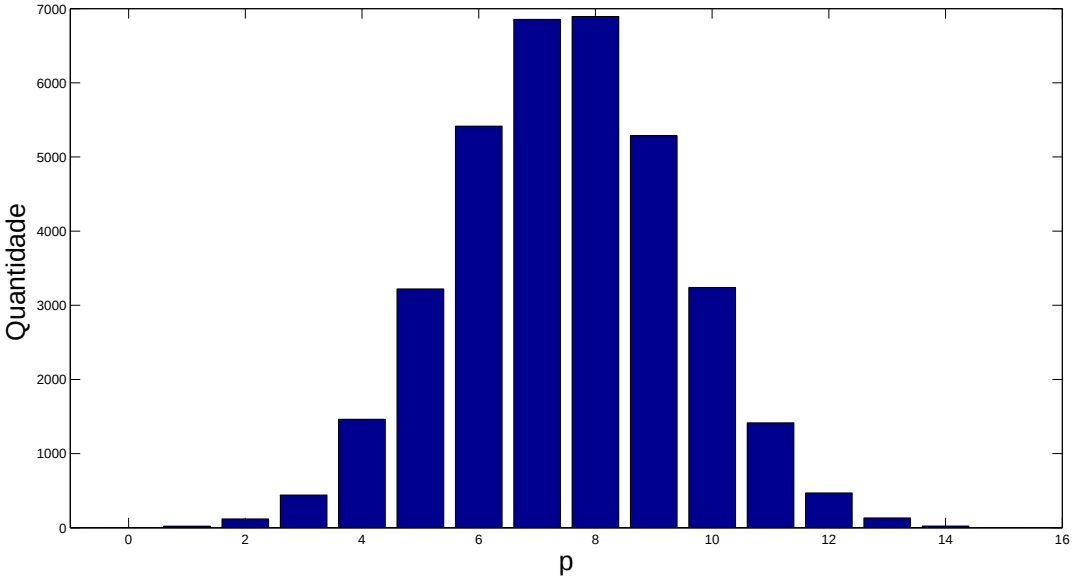


Figura 4.5: *Histograma com a distribuição para manchas retangulares com dimensões* $a = 5$ *e* $b = 3$.

Os histogramas apresentados nas Figuras 4.6 e 4.7 ilustram a quantidade de manchas com moldura em cruz geradas em função do peso da mancha, p.

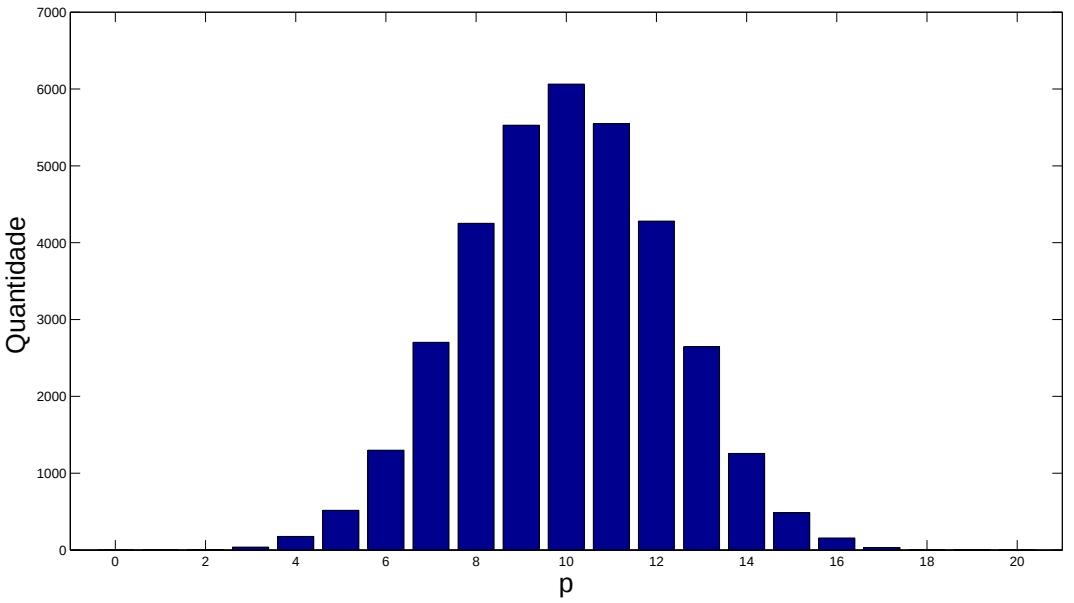


Figura 4.6: *Histograma com a distribuição para manchas em cruz com dimensões* $a = 4$ *e* $b = 6$.

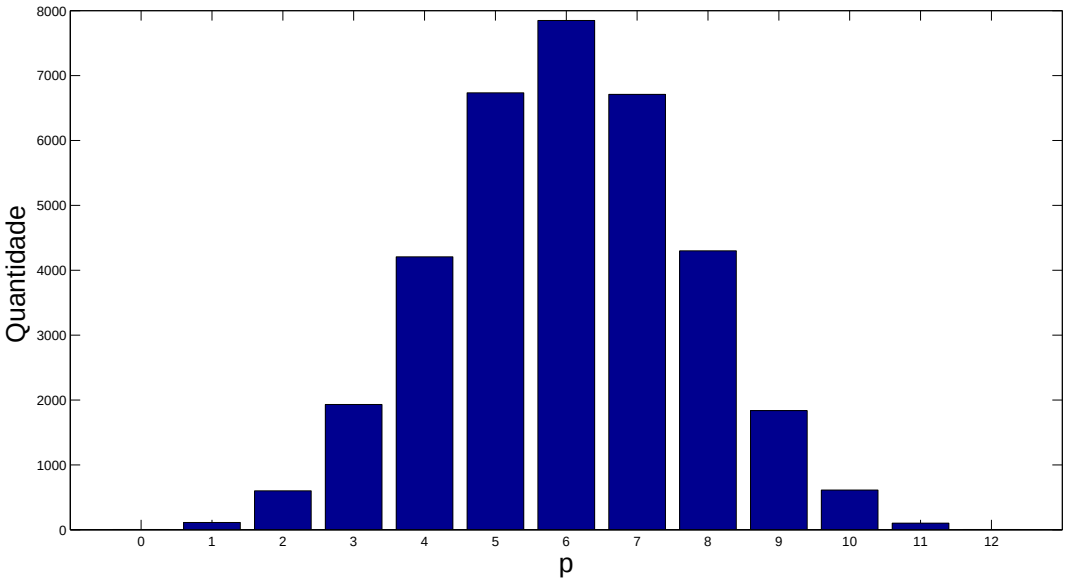


Figura 4.7: *Histograma com a distribuição para manchas cruz com dimensões* $a = 4$ *e* $b = 4$.

Para as Figuras 4.2 a 4.7, foram geradas 35.000 manchas para a otenção de cada histograma. Observa-se em todos as imagens um comportamento aproximado da distribuição de probabilidade binomial na geração das manchas em função de p. É sabido que o número de manchas para determinado p e P_{max} é dado por $C_p^{P_{max}}$, em que P_{max} é o peso máximo de determinada mancha, e C_m^n é obtido pela Fórmula apresentada em 4.8.

$$C_m^n = \frac{n!}{m!(n-m)!}. \tag{4.8}$$

A adição da mancha na matriz binária é módulo 2. No trabalho desenvolvido, é escolhido um elemento como referência para a região a ser afetada pela mancha. Esse elemento inicial da matriz $\mathbf{V^*_{n\times n}}$ é escolhido aleatoriamente e representado por $\mathbf{V^*_{ij}}$, em que i representa a linha e j a coluna do elemento contido na matriz. No Exemplo 4.5, ilustra-se o ataque da mancha na matriz $\mathbf{V^*_{n\times n}}$.

Exemplo 4.5

Seja a matriz codificada e entrelaçada $\mathbf{V}^*_{\mathbf{7\times 7}}$ *dada por*

$$\mathbf{V}^*_{\mathbf{7\times 7}} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}. \tag{4.9}$$

Ela é atacada pela mancha quadrada $\mathbf{M_{5\times 5}}$ *apresentada em 4.10. Esta matriz gerada por uma distribuição aleatória de* $1s$ *e* $0s$ *representa a adição de cinco surtos de comprimento cinco à matriz* $\mathbf{V}^*_{\mathbf{7\times 7}}$.

$$\mathbf{M_{5\times 5}} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}. \tag{4.10}$$

O ponto de ataque escolhido como início da região a ser afetada foi o elemento $\mathbf{V}^*_{\mathbf{32}}$. *Então, a submatriz composta pelos elementos em negrito na Matriz 4.11 terá seus valores alterados pelos bits de* $\mathbf{M_{5\times 5}}$.

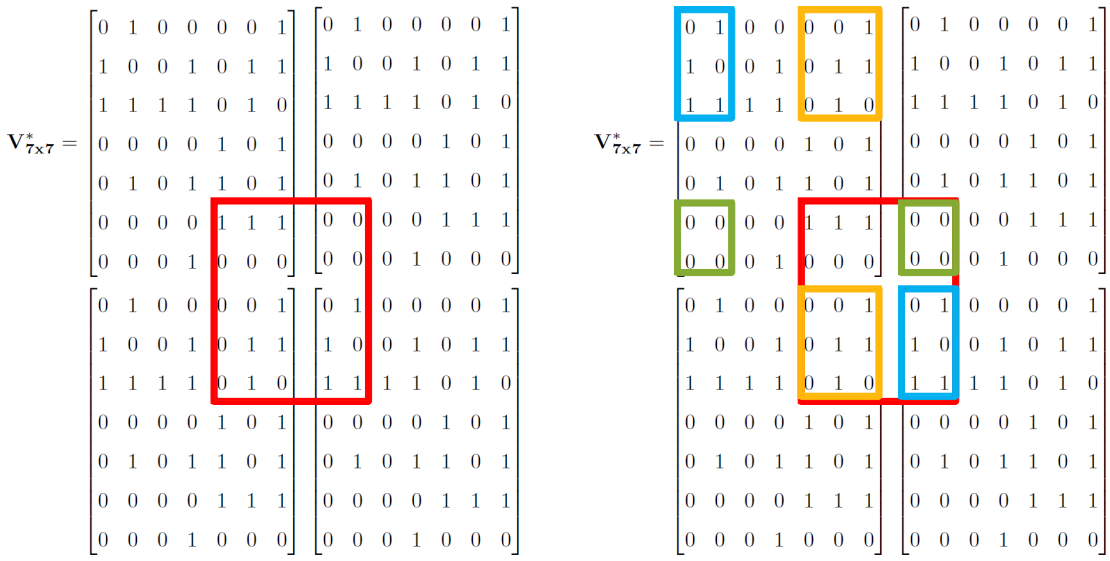
$$\mathbf{V}^*_{\mathbf{7\times 7}} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{1} & 0 \\ 0 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & 1 \\ 0 & \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{0} & 1 \\ 0 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{1} & 1 \\ 0 & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} & 0 \end{bmatrix}. \tag{4.11}$$

□

No Exemplo 4.5 o padrão de erro ficou contido na matriz de informação preservando sua moldura. No entanto, existem casos em que isto não ocorre. O Exemplo 4.6 ilustra esta ocorrência.

Exemplo 4.6

Sejam as mesmas matrizes entrelaçada e mancha a ser adicionada apresentadas em 4.9 e 4.10 respectivamente. Sendo que o ponto de ataque escolhido como início da região a ser afetada para este caso foi o elemento $\mathbf{V^*_{65}}$. *Como utiliza-se códigos cíclicos, então pode-se imaginar o arranjo apresentado na Figura 4.8 (a) em que temos cópias da matriz* $\mathbf{V^*_{7\times7}}$. *Neste arranjo tem-se, em destaque na moldura vermelha, a região afetada pela mancha de acordo com o elemento inicial escolhido. A região pode ser divida conforme a Figura 4.8 (b), dessa forma é possivel determinar os bits a serem afetados pela mancha na matriz* $\mathbf{V^*_{7\times7}}$ *através das cores de cada subregião.*



(a) Região afetada pela mancha. (b) Divisão da região afetada.

Figura 4.8: *Adição da mancha de erro para o elemento inicial* $\mathbf{V^*_{65}}$.

Dessa a forma submatriz composta pelos elementos em negrito apresentada em 4.12 terão seus valores alterados pelos bits de $\mathbf{M_{5\times5}}$

$$\mathbf{V^*_{7\times7}} = \begin{bmatrix} \mathbf{0} & \mathbf{1} & 0 & 0 & \mathbf{0} & \mathbf{0} & \mathbf{1} \\ \mathbf{1} & \mathbf{0} & 0 & 1 & \mathbf{0} & \mathbf{1} & \mathbf{1} \\ \mathbf{1} & \mathbf{1} & 1 & 1 & \mathbf{0} & \mathbf{1} & \mathbf{0} \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ \mathbf{0} & \mathbf{0} & 0 & 0 & \mathbf{1} & \mathbf{1} & \mathbf{1} \\ \mathbf{0} & \mathbf{0} & 0 & 1 & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}. \tag{4.12}$$

□

O Exemplo 4.6 ilustra um caso em que o erro não preserva sua moldura ao ser adicionado à matriz, no entanto, a mesma quantidade de surtos e adicionada, bem como a mesma quantidade de *bits* é afetada em ambos os exemplos. A adição da matriz $\mathbf{M_{5\times5}}$ representa a adição de cinco surtos de comprimento cinco a matriz $\mathbf{V^{*}_{7\times7}}$ apresentada em 4.9.

4.4 DESENTRELAÇADOR

O desentrelaçador realiza a operação inversa realizada pelo entrelaçador. Para tal é necessário encontrar a matriz $\mathbf{T^{-1}}$, matriz inversa da matriz de transformação $\mathbf{T}$ utilizada na Equação 4.1. Seja uma matriz binária quadrada $\mathbf{C_{c\times c}}$, a sua matriz inversa $\mathbf{C^{-1}_{c\times c}}$ é aquela matriz tal que $\mathbf{CC^{-1}} = \mathbf{I}$, em que $\mathbf{I}$ é a matriz identidade de ordem c. Então, ao resolver a Equação 4.13 encontra-se $\mathbf{T^{-1}} = \mathbf{T}$ [24].

$$\mathbf{TT^{-1}} = \mathbf{I}_2. \tag{4.13}$$

Com base no resultado da Equação 4.13, é necessário aplicar a mesma transformação realizada pela matriz $\mathbf{T}$, e o trabalho do desentrelaçador é desfeito. Dessa maneira, a matriz $\mathbf{R^{*}_{n\times n}}$ é convertida na matriz $\mathbf{R_{n\times n}}$ para ser decodificada e depois recuperada a informação.

A seguir são apresentadas em 4.14 e 4.15 as matrizes desentrelaçadas $\mathbf{R_{7\times7}}$ para os Exemplos 4.5 e 4.6 respectivamente.

$$\mathbf{R^{*}_{7\times7}} = \begin{bmatrix} 0 & 0 & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} & 0 \\ 0 & 1 & 0 & \mathbf{0} & \mathbf{0} & \mathbf{1} & 1 \\ 0 & \mathbf{1} & 0 & 1 & \mathbf{1} & \mathbf{1} & 1 \\ 0 & \mathbf{1} & \mathbf{1} & 0 & 0 & \mathbf{0} & 1 \\ 0 & \mathbf{0} & \mathbf{1} & \mathbf{0} & 0 & 1 & 0 \\ 1 & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & 0 & 1 \\ 1 & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{1} & 1 \end{bmatrix} \tag{4.14}$$

$$\mathbf{R}^{*}_{\mathbf{7\times 7}} = \begin{bmatrix} \mathbf{1} & \mathbf{1} & 1 & 0 & 1 & \mathbf{1} & \mathbf{0} \\ \mathbf{1} & \mathbf{0} & 0 & 1 & 1 & 0 & \mathbf{1} \\ \mathbf{1} & \mathbf{1} & 0 & 1 & \mathbf{1} & 0 & 1 \\ 0 & \mathbf{1} & 0 & 0 & \mathbf{1} & \mathbf{0} & 1 \\ 0 & 1 & 0 & 1 & \mathbf{1} & \mathbf{0} & \mathbf{1} \\ \mathbf{0} & 0 & 0 & 0 & \mathbf{1} & \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & 0 & 1 & \mathbf{0} & \mathbf{1} & \mathbf{0} \end{bmatrix}. \tag{4.15}$$

Os *bits* em negrito foram afetados pelos bits da matriz $\mathbf{M_{5\times 5}}$ apresentada em 4.10. É possível perceber que o comprimento dos surtos que atacaram as matrizes $\mathbf{R}^{*}_{\mathbf{7\times 7}}$ para as Matrizes 4.14 e 4.15 permaneceram ou tiveram seu comprimento reduzido em relação aos surtos das matrizes $\mathbf{V}^{*}_{\mathbf{7\times 7}}$ para as Matrizes 4.11 e 4.12, demonstrando a vantagem no uso do entrelaçador. A seguir é apresentado o desempenho dos decodificadores para as duas técnicas de decodificação estudadas.

4.5 DESEMPENHO DO DECODIFICADOR PARA AS TÉCNICAS DE ARMADILHA SIMPLES E A DE ARMADILHA ADAPTATIVA

Para avaliar o desempenho dos decodificadores utilizados, o seguinte procedimento foi realizado. Considere que o polinômio $u(x) = 0$ é codificado sistematicamente pelo codificador do código cíclico $C(15, 5)$ gerado por $g(x) = 1 + x^2 + x^5 + x^6 + x^8 + x^9 + x^{10}$ e com capacidade $b = 5$ de correção de erros em surto [6]. Este código é ótimo, pois satisfaz com igualdade o limitante de Reiger apresentado na Inequação 3.2. A codificação resulta no polinômio código $v(x) = 0$. Este polinômio tem as posições de menor grau afetadas por um polinômio erro, $e(x)$, de grau r para $1 \leq r \leq 6$. Para a Tabela 4.1 há conhecimento do surto a ser inserido, enquanto nas demais é estabelecido apenas o seu comprimento. O polinômio $r(x)$ resultante é usado como entrada dos decodificadores por armadilha fixa e por armadilha adaptativa, que dão como resultado dois polinômios estimados $\hat{u}_1(x)$ e $\hat{u}_2(x)$ respectivamente. Comparando $u(x)$ com $\hat{u}_1(x)$ e $\hat{u}_2(x)$ verifica-se o sucesso ou falha na correção do surto inserido para cada decodificador. Em seguida, o polinômio erro $e(x)$ é deslocado ciclicamente uma vez para a direita e todo o procedimento se repete até que se tenha efetuado $n = 15$ deslocamentos em $e(x)$, pois dessa forma ele terá percorrido todo o polinômio $v(x)$. Seguindo este raciocínio, como era de se esperar, todos os surtos de comprimento $b = 5$ ou menor foram corrigidos por ambas técnicas.

A Tabela 4.1 apresenta os surtos gerados e falhas percentuais para o caso de armadilha adapta-

tiva e comprimento de surto $b = 6$. Para o decodificador de armadilha fixa nenhum dos surtos foi decodificado, pois não podem ser aprisionados pela armadilha de tamanho cinco.

Tabela 4.1: *Desempenho do decodificador armadilha para surtos de comprimento $b = 6$, Em destaque os surtos não corrigidos.*

Surto Gerado		**Erros Armadilha Variável (%)**
b = 6	100001	0,00%
	100011	0,00%
	100101	0,00%
	100111	0,00%
	101001	**100,00%**
	101011	0,00%
	101101	0,00%
	101111	0,00%
	110001	0,00%
	110011	0,00%
	110101	0,00%
	110111	**100,00%**
	111001	0,00%
	111011	0,00%
	111101	**100,00%**
	111111	0,00%

Percebe-se a diferença na capacidade de correção dos surtos quando se usa o decodificador por armadilha adaptativa em relação ao de armadilha fixa. A correção para a maioria dos surtos utilizando o decodificador de armadilha adaptativa é obtida para o comprimento $b = 6$. Analisando a Tabela 4.1 nota-se que para os dezesseis possíveis surtos de comprimento $b = 6$ gerados, apenas três surtos não foram decodificados.

A não correção destes surtos é explicada pela teoria apresentada na Seção 3.1. No caso do surto da quinta linha da Tabela 4.1, 101001, de comprimento 6 é possível adicioná-lo com o surto $E = 00000010111$ de comprimento 5 e resultar no surto $F = 10100110111$. O surto F de comprimento 11 é uma palavra código, então, existe um surto de comprimento menor que $2b = 12$ que é uma palavra código, logo o código não é capaz de corrigir todos os erros em surto de comprimento 6. Para o surto da décima segunda linha da Tabela 4.1, 110111, a explicação é mesma sendo que considera-se os vetores $D = 1010000000$, $E = 00000010111$ e $F = 10100110111$, em que $F = D + E$.

Idem para o surto da décima quarta linha da Tabela 4.1, 111101, agora com $D = 10100000111$, $E = 00000110000$ e $F = 10100110111$.

As Tabelas 4.2, 4.3, 4.4 e 4.5 dão continuidade à análise do desempenho dos decodificadores. O procedimento realizado na obtenção dos dados é similar ao apresentado no início dessa seção. Neste caso define-se apenas o tamanho do surto, não há a construção de cada surto independentemente. No entanto, cada surto gerado ainda percorre todo o polinômio código.

Apresentam-se os valores médios e o desvio padrão do desempenho do decodificador armadilha variável para os códigos cíclicos binários e lineares $C(15, 5)$ (Tabelas 4.2 e 4.3) e $C(21, 7)$ (Tabelas 4.4 e 4.5). Para o decodificador de armadilha fixa nenhum dos surtos com comprimento maior que cinco para o código $C(15, 5)$ e comprimento sete para o código $C(21, 7)$ foram decodificados, pois não podem ser aprisionados pela armadilha do respectivo decodificador.

Tabela 4.2: *Resultados da Média (Med) e Desvio Padrão (DevPad) de dez amostras obtidas pelo decodificador de armadilha variável (AV), para o código C(15,5), gerado por,* $g(x) = 1 + x + x^2 + x^4 + x^5 + x^8 + x^{10}$*, com* $b = 5$*, para diferentes comprimentos de surtos (1005 surtos gerados).*

Tamanho do Surto	**Surtos não corrigidos AV 1005 surtos gerados**	
	Med	DevPad
b = 2	0,00%	0,00 E+00
b = 3	0,00%	0,00 E+00
b = 4	0,00%	0,00 E+00
b = 5	0,00%	0,00 E+00
b = 6	16,27%	3,48 E-02
b = 7	38,51%	6,38 E-02
b = 8	74,43%	4,67 E-02
b = 9	96,83%	1,64 E-02
b = 10	98,81%	1,37 E-02

Tabela 4.3: *Resultados da Média (Med) e Desvio Padrão (DevPad) de dez amostras obtidas pelo decodificador de armadilha variável (AV), para o código C(15,5), gerado por,* $g(x) = 1 + x + x^2 + x^4 + x^5 + x^8 + x^{10}$*, com* $b = 5$*, para diferentes comprimentos de surtos (10050 surtos gerados).*

Tamanho do Surto	**Surtos não corrigidos AV 10050 surtos gerados**	
	Med	DevPad
b = 2	0,00%	0,00 E+00
b = 3	0,00%	0,00 E+00
b = 4	0,00%	0,00 E+00
b = 5	0,00%	0,00 E+00
b = 6	18,84%	1,47 E-02
b = 7	37,41%	9,94 E-03
b = 8	73,45%	1,73 E-02
b = 9	96,54%	5,81 E-03
b = 10	98,49%	5,42 E-03

Tabela 4.4: *Resultados da Média (Med) e Desvio Padrão (DevPad) de dez amostras obtidas pelo decodificador de armadilha variável (AV), para o código C(21,7), gerado por,* $g(x) = 1 + x^3 + x^4 + x^5 + x^7 + x^8 + x^9 + x^{13} + x^{14}$*, com* $b = 7$*, para diferentes comprimentos de surtos (1008 surtos gerados).*

Tamanho do Surto	**Surtos não corrigidos AV 1008 surtos gerados**	
	Med	DevPad
b = 2	0,00%	0,00 E+00
b = 3	0,00%	0,00 E+00
b = 4	0,00%	0,00 E+00
b = 5	0,00%	0,00 E+00
b = 6	0,00%	0,00 E+00
b = 7	0,00%	0,00 E+00
b = 8	5,63%	3,03 E-02
b = 9	12,78%	4,08 E-02
b = 10	30,71%	7,18 E-02
b = 11	53,73%	7,03 E-02
b = 12	86,84%	4,31 E-02
b = 13	99,58%	6,94 E-03
b = 14	99,79%	6,94 E-03

Tabela 4.5: *Resultados da Média (Med) e Desvio Padrão (DevPad) de dez amostras obtidas pelo decodificador de armadilha variável (AV), para o código C(21,7), gerado por,* $g(x) = 1 + x^3 + x^4 + x^5 + x^7 + x^8 + x^9 + x^{13} + x^{14}$*, com* $b = 7$*, para diferentes comprimentos de surtos (10080 surtos gerados).*

Tamanho do Surto	**Surtos não corrigidos AV 10080 surtos gerados**	
	Med	DevPad
b = 2	0,00%	0,00 E+00
b = 3	0,00%	0,00 E+00
b = 4	0,00%	0,00 E+00
b = 5	0,00%	0,00 E+00
b = 6	0,00%	0,00 E+00
b = 7	0,00%	0,00 E+00
b = 8	6,33%	1,45 E-02
b = 9	12,82%	2,19 E-02
b = 10	27,92%	1,51 E-02
b = 11	55,19%	2,89 E-02
b = 12	87,65%	9,34 E-03
b = 13	99,37%	4,57 E-03
b = 14	99,88%	1,47 E-03

Analisando as Tabelas 4.2, 4.3, 4.4 e 4.5 percebe-se que à medida que o comprimento do surto aumenta, a percentagem de erro também acompanha este aumento. Dessa forma, quanto maior o comprimento do surto mais surtos podem ser decompostos em dois outros menores que somados resultam em uma palavra código e dessa forma torna a decodificação impossível para aquele determinado comprimento. Na próxima seção são abordados exemplos com imagens.

4.6 EXEMPLOS

Nos exemplos que se seguem considera-se uma imagem como fonte de informação. A imagem considerada foi a do arquivo *lenna128.bmp*, extraído de [32]. Para todos os exemplos, foi usado o código cíclico binário linear $C(21, 7)$ gerado por $g(x) = 1+x^3+x^4+x^5+x^7+x^8+x^9+x^{13}+x^{14}$ e com capacidade de correção de surtos de comprimento até $b = 7$. Na seção 4.6.1 é seguido o esquema apresentado na Figura 4.1. Nesse esquema, a matriz original é dividida em blocos. Cada bloco é codificado, entrelaçado e afetado por uma única mancha.

Na seção 4.6.2 também ocorre a divisão de blocos da matriz binária original para codificação e entrelaçamento. Em seguida, essa matriz é reconstruída e a adição de determinado número de manchas ocorre em toda a matriz, podendo ou não haver superposição de manchas.

4.6.1 Adição de mancha em cada Bloco

Seguindo o esquema apresentado na Figura 4.1, a Figura 4.9 (a) representa a imagem original. A localização da imagem no esquema da Figura 4.1 é representada pela letra A.

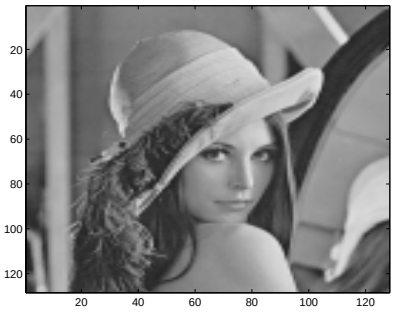
Essa imagem quando convertida em uma matriz de *bits* resulta em uma matriz com dimensões 128×1024. Após o ajuste para o código $C(21, 7)$, tem-se uma dimensão de 147×1029, o que resulta num total de 1029 blocos à serem enviados em sequência pelo sistema de transmissão. Cada bloco, representado por $\mathbf{U_{21\times 7}}$ é codificado sistematicamente resultando em $\mathbf{V_{21\times 21}}$. A Figura 4.9 (b) representa a imagem codificada. A localização da imagem no esquema da Figura 4.1 é representada pela letra B.

Em seguida, a matriz $\mathbf{V_{21\times 21}}$ é entrelaçada pelo entrelaçador descrito na Seção 4.2, gerando a matriz $\mathbf{V^*_{21\times 21}}$. A Figura 4.9 (c) representa a imagem entrelaçada. A localização da imagem no esquema da Figura 4.1 é representada pela letra C.

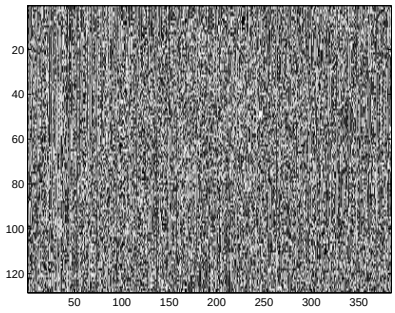
Logo após é adicionada uma mancha de erro ao bloco, conforme apresentado na Seção 4.3. Essa matriz resulta na matriz $\mathbf{R^*_{21\times 21}}$ que é desentrelaçada, resultando na matriz $\mathbf{R_{21\times 21}}$. Devido a codificação sistemática, é possivel separar os bits de informação de cada bloco e ao fim compor uma imagem. A localização destas imagens no esquema da Figura 4.1 é representada pela letra D.

A matriz $\mathbf{R^*_{21\times 21}}$ é decodificada gerando a matriz estimada $\mathbf{\hat{U}_{21\times 7}}$. Após a recepção e conversão para decimal de todos os blocos, é possível reconstruir a imagem. A localização dessa imagem no esquema da Figura 4.1 é representada pela letra E.

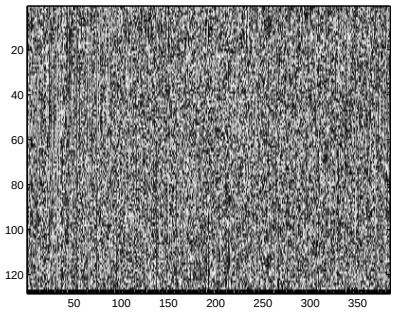
A Figura 4.9 ilustra a imagem original, a imagem codificada e a imagem entrelaçada que são usadas nos exemplos que seguem. Vale resaltar que as imagens foram construídas removendo o ajuste pela adição de zeros adicionada à matriz original.



(a) Imagem Original



(b) Imagem Codificada



(c) Imagem Entrelaçada

Figura 4.9: *Imagem original (a), imagem codificada (b) e imagem entrelaçada (c).*

No Exemplo 4.7 cada bloco codificado e entrelaçado foi afetado por uma mancha quadrada com $a = 7$ e peso p. O valor de p e o local em que a mancha ataca a imagem são escolhidos aleatorialmente. Uma vez que apenas o comprimento do vetor é fator de correção para o decodificador.

Exemplo 4.7

As Figuras 4.10 e 4.11 ilustram a sequência de imagens ao longo do esquema da Figura 4.1, para o caso da adição de mancha quadrada com $a = 7$ em cada bloco transmitido.

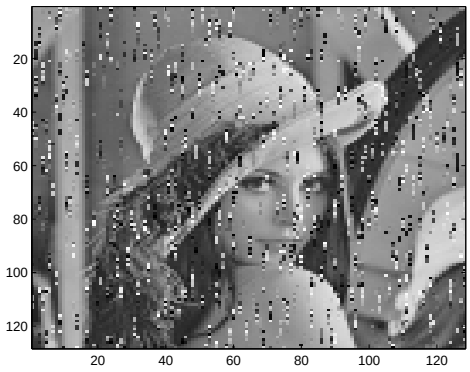
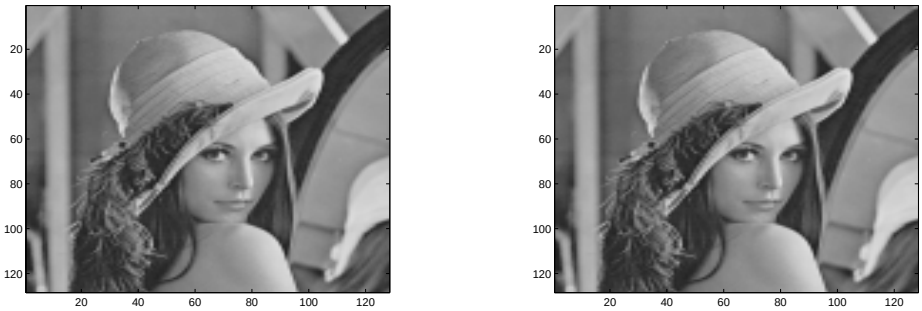


Figura 4.10: *Exemplo de imagem afetada por mancha quadrada com dimensão $a = 7$ em cada bloco. Imagem reconstruída com os bits de informação da imagem não decodificada*

.



(a) Imagem decodificada por armadilha adptativa

(b) Imagem decodificada por armadilha fixa

Figura 4.11: *Exemplo de imagem afetada por mancha quadrada com dimensão $a = 7$ em cada bloco. Imagem reconstruída com os bits de informação obtidos pelos decodificadores de armadilha adaptativa (a) e o de armadilha fixa (b).*

A fim de validar a eficiência da decodificação por cada técnica contou-se o número de pixels diferentes entre a imagem original e as imagens decodificadas por cada decodificador. A imagem utilizada como exemplo possui dimensão 128×128, logo um total de $128^2 = 16384$ *pixels*.

As imagens da Figura 4.11 obtida pelo decodificador de armadilha adaptativa (a) e pelo decodificador de armadilha fixa (b) apresentam 0 *pixels* diferentes em relação à imagem original. Enquanto a imagem da Figura 4.10 (b) que não sofreu decodificação possui um total de 4171 *pixels* diferentes em relação à imagem original, ou seja, 25,46 % dos *pixels* totais.

O procedimento do Exemplo 4.7 foi repetido mais nove vezes gerando os dados da Tabela 4.6.

Tabela 4.6: *Percentual do número de pixels diferentes em relação à imagem original para o decodificador por armadilha adaptativa, o decodificador por armadilha fixa e antes da decodificação. Realizada a adição da mancha com moldura quadrada e dimensão* $a = 7$.

Amostra	**Percentual Armadilha Adaptativa**	**Percentual Armadilha Fixa**	**Percentual Antes da decodificação**
1	0,00%	0,00%	25,46%
2	0,00%	0,00%	27,12%
3	0,00%	0,00%	27,52%
4	0,00%	0,00%	25,91%
5	0,00%	0,00%	26,54%
6	0,00%	0,00%	26,53%
7	0,00%	0,00%	27,67%
8	0,00%	0,00%	26,55%
9	0,00%	0,00%	26,31%
10	0,00%	0,00%	26,27%
Média	0,00%	0,00%	26,59%
Desvio Padrão	0,00 E-00	0,00 E-00	6,83 E-03

Pela análise da Tabela 4.6 percebe-se que todos os erros inseridos foram corrigidos para ambos os decodificadores, isto porque a mancha quadrada com dimensão $a = 7$ gera surtos de comprimento sete ou menor. Como o código é ótimo então, todos esses surtos são corrigidos.

No Exemplo 4.8 cada bloco codificado e entrelaçado foi afetado por uma mancha retangular com dimensões $a = 8$ e $b = 9$ e peso p. O valor de p e o local em que a mancha ataca a imagem são escolhidos aleatorialmente.

Exemplo 4.8

As Figuras 4.12 e 4.13 ilustram a sequência de imagens ao longo do esquema da Figura 4.1, para o caso de mancha retangular com $a = 8$ e $b = 9$.

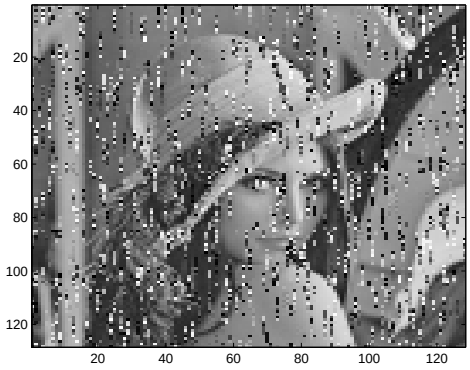
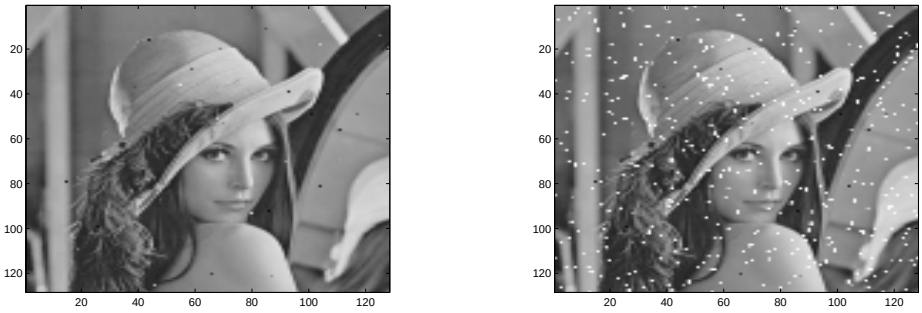


Figura 4.12: *Exemplo de imagem afetada por mancha retangular com $a = 8$ e $b = 9$ em cada bloco. Imagem reconstruída com os bits de informação da imagem não decodificada*

.



(a) Imagem decodificada por armadilha adptativa (b) Imagem decodificada por armadilha fixa

Figura 4.13: *Exemplo de imagem afetada por mancha retangular com dimensões $a = 8$ e $b = 9$ em cada bloco. Imagem reconstruída com os bits de informação obtidos pelos decodificadores de armadilha adaptativa (a) e o de armadilha fixa (b).*

Ao realizar a contagem do *pixels* diferentes, a imagem da Figura 4.13 obtida pelo decodificador de armadilha adaptativa (a), possui 49 *pixels* diferentes, ou seja, 0,30 % dos *pixels* totais. Enquanto a imagem obtida pelo decodificador de armadilha fixa (b) apresenta um total de 756 pixels diferentes

Tabela 4.7: *Percentual do número de pixels diferentes em relação à imagem original para o decodificador por armadilha adaptativa, o decodificador por armadilha fixa e antes da decodificação. Realizada a adição da mancha com moldura retangular de dimensões $a = 8$ e $b = 9$.*

Amostra	Percentual Armadilha Adaptativa	Percentual Armadilha Fixa	Percentual Antes da decodificação
1	0,30%	4,61%	36,19%
2	0,24%	4,59%	35,99%
3	0,18%	4,47%	37,92%
4	0,30%	4,38%	37,19%
5	0,31%	4,30%	36,27%
6	0,35%	4,41%	36,43%
7	0,26%	4,28%	36,60%
8	0,27%	4,65%	37,28%
9	0,19%	4,74%	35,89%
10	0,26%	4,61%	36,74%
Média	0,27%	4,50%	36,65%
Desvio Padrão	5,29 E-04	1,58 E-03	6,44 E-03

em relação à imagem original, ou seja, 4,61 % dos pixels totais. Já a imagem da Figura 4.12 (b), na qual não sofreu decodificação possui um total de 5930 *pixels* diferentes em relação à imagem original, ou seja, 36,19 % dos *pixels* totais.

O procedimento do Exemplo 4.8 foi repetido mais nove vezes gerando os dados da Tabela 4.7.

Pela análise da Tabela 4.7 percebe-se a melhora na correção para o decodificador por armadilha adaptativa em relação ao de armadilha fixa.

No Exemplo 4.9 cada bloco codificadao e entrelaçado foi afetado por uma mancha em cruz com dimensões $a = 10$ e $b = 10$ e peso p. O valor de p e o local em que a mancha ataca a imagem são escolhidos aleatorialmente.

Exemplo 4.9

As Figuras 4.14 e 4.15 ilustram a sequência de imagens ao longo do esquema da Figura 4.1, para o caso de mancha em cruz com $a = 10$ e $b = 10$.

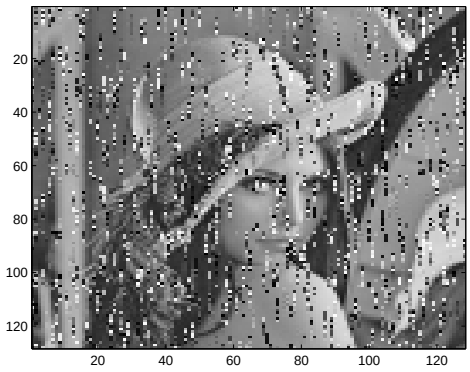
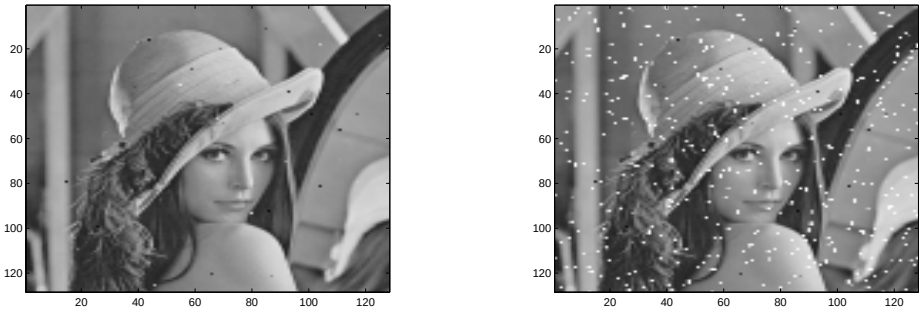


Figura 4.14: *Exemplo de imagem afetada por mancha em cruz com $a = 10$ e $b = 10$ em cada bloco. Imagem reconstruída com os bits de informação da imagem não decodificada*

.



(a) Imagem decodificada por armadilha adptativa (b) Imagem decodificada por armadilha fixa

Figura 4.15: *Exemplo de imagem afetada por mancha em cruz com dimensões $a = 10$ e $b = 10$ em cada bloco. Imagem reconstruída com os bits de informação obtidos pelos decodificadores de armadilha adaptativa (a) e o de armadilha fixa (b)*

Realizando a contagem de *pixels* diferentes verificou-se que a imagem da Figura 4.15, obtida pelo decodificador de armadilha adaptativa (a), possui 132 *pixels* diferentes, ou seja, 0,80 % dos pixels totais. Enquanto a imagem obtida pelo decodificador de armadilha fixa (b) apresenta um total de 2467 pixels diferentes em relação à imagem original ou seja, 15,05 % dos *pixels* totais. Já a imagem da Figura 4.12 (b) que não sofreu decodificação possui um total de 7300 *pixels* diferentes em relação à imagem original, ou seja, 44,55 % dos pixels totais.

Tabela 4.8: *Percentual do número de pixels diferentes em relação à imagem original para o decodificador por armadilha adaptativa, o decodificador por armadilha fixa e antes da decodificação. Realizada a adição da mancha com moldura em cruz e dimensões* $a = 10$ *e* $b = 10$.

Amostra	**Percentual Armadilha Adaptativa**	**Percentual Armadilha Fixa**	**Percentual Antes da decodificação**
1	0,80%	15,05%	44,55%
2	1,22%	15,80%	46,26%
3	1,09%	15,26%	45,45%
4	1,01%	15,58%	42,49%
5	1,17%	15,27%	45,01%
6	1,18%	15,34%	45,00%
7	1,03%	15,44%	45,56%
8	1,11%	15,30%	45,19%
9	1,16%	15,32%	45,18%
10	0,95%	14,56%	43,37%
Média	1,07%	15,29%	44,81%
Desvio Padrão	1,28 E-03	3,27 E-03	1,10 E-02

O procedimento do Exemplo 4.9 foi repetido mais nove vezes gerando os dados da Tabela 4.8.

Pela análise da Tabela 4.8 percebe-se a melhora na correção para o decodificador por armadilha adaptativa em relação ao de armadilha fixa.

No Exemplo 4.10 não há controle sobre o tipo de mancha, valor de p e local de ataque da mancha na imagem.

Exemplo 4.10

As Figuras 4.16 e 4.17 ilustram a sequência de imagens ao longo do esquema da Figura 4.1.

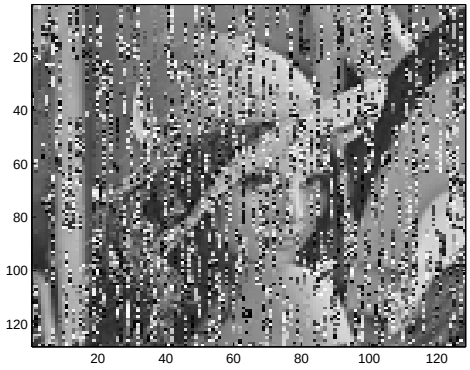
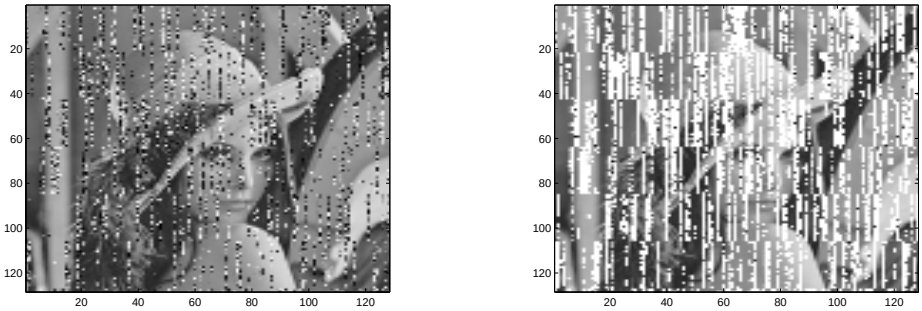


Figura 4.16: *Exemplo de imagem afetada por mancha de moldura aleatória em cada bloco. Imagem reconstruída com os bits de informação da imagem não decodificada*

.



(a) Imagem decodificada por armadilha adptativa (b) Imagem decodificada por armadilha fixa

Figura 4.17: *Exemplo de imagems afetada por mancha de moldura aleatória em cada bloco.Imagem reconstruída com os bits de informação obtidos pelos decodificadores de armadilha adaptativa (a) e o de armadilha fixa (b).*

Seguindo o raciocínio aplicado nos exemplos anteriores contou-se o número de *pixels* diferentes entre a imagem original e as imagens decodificadas por cada decodificador. A imagem da Figura 4.17(a) obtida pelo decodificador de armadilha adaptativa apresenta um total de 5161 *pixels* diferentes em relação à imagem original, ou seja, 31,50% de *pixels* diferentes. Já a imagem da Figura 4.15 (b) obtida pelo decodificador de armadilha fixa apresenta um total de 8203 *pixels* diferentes em relação

Tabela 4.9: *Percentual do número de pixels diferentes em relação à imagem original para o decodificador por armadilha adaptativa, o decodificador por armadilha fixa e antes da decodificação. Realizada a adição da mancha com moldura aleatória.*

Amostra	**Percentual Armadilha Adaptativa**	**Percentual Armadilha Fixa**	**Percentual Antes da decodificação**
1	31,50%	50,07%	55,95%
2	30,32%	49,07%	55,24%
3	30,57%	49,04%	57,36%
4	31,30%	49,38%	56,26%
5	30,00%	48,83%	55,52%
6	33,87%	52,42%	58,53%
7	33,59%	51,53%	56,40%
8	31,05%	50,23%	55,94%
9	35,70%	55,08%	59,33%
10	32,48%	51,39%	58,13%
Média	32,04%	50,70%	55,87%
Desvio Padrão	1,84 E-02	1,96 E-02	1,39 E-02

à imagem original, ou seja, 50,07 % de *pixels* diferentes. Enquanto a imagem da Figura 4.14 (b)que não sofreu decodificação possui um total de 9167 *pixels* diferentes em relação à imagem original, ou seja, 55,95 % dos *pixels* totais. O procedimento do Exemplo 4.10 foi repetido mais nove vezes gerando os dados da Tabela 4.9.

Pela análise da Tabela 4.9 percebe-se um menor percentual de pixels diferentes em relação à imagem original quando se utiliza o decodificador por armadilha adaptativa, comprovando assim sua maior eficiência.

Em todos os exemplos apresentados neste cápitulo é notada a melhora ou igual resultado quando se utiliza o decodificador com a técnica de armadilha adaptativa. Tal fato se deve, conforme mencionado no Capítulo 3, à sua armadilha não possuir tamanho fixo e ser capaz de corrigir surtos de comprimentos maiores em relação ao de armadilha fixa.

Para o Exemplo 4.8 tem-se $b = 9$, logo os surtos são reduzidos a comprimento nove ou menor. O decodificador de armadilha fixa se limita na correção dos surtos de comprimento cinco ou menor enquanto o de armadilha adaptativa comete erro apenas para 5,63 % dos surtos de comprimento $b = 8$ e 12,78 % dos surtos de comprimento $b = 9$ conforme apresentado na Tabela 4.4, logo um maior desempenho quando comparado ao de armadilha fixa.

4.6.2 ADIÇÃO DE QUANTIDADE FIXA DE MANCHAS

Para este caso considera-se a mesma imagem da Figura 4.9 (a) como imagem a ser transmitida. Bem como na Seção 4.6.1, a matriz decimal da imagem é convertida para uma matriz binária, ajustada aos parâmetros do código e dividida em blocos. Cada bloco é codificado e entrelaçado, como descrito nas Seções 4.1 e 4.2. Após, é feito um armazenamento desses blocos até que toda a matriz seja codificada e entrelaçada. Essa matriz possui dimensões 147×3087. A Figura 4.9 (c) representa a imagem até este passo.

A seguir é examinada a situação na qual um número fixo de manchas é adicionado à imagem transmitida, podendo cada mancha assumir qualquer posição na imagem, sendo possível inclusive haver superposição de manchas. Em seguida, a matriz é novamente dividida em blocos para ser desentrelaçada e decodificada pelas técnicas de decodificação aqui estudadas. Finalizado o ataque, pode-se reconstruir a imagem, pela conversão de seus elementos binários em decimais.

A seguir seguem exemplos deste tipo de soma. Mantendo um padrão com o esquema descrito na seção anterior, considera-se os seguintes exemplos: Exemplo 4.11 adição de manchas com moldura quadrada de dimensão $a = 7$, Exemplo 4.12 adição de manchas com moldura retangular de dimensões $a = 8$ e $b = 9$ e Exemplo 4.13 adição de manchas com moldura em cruz de dimensões $a = 10$ e $b = 10$. Para cada tipo de moldura considera-se dois experimentos: um com a adição de 1000 manchas e outro com a adição de 2000 manchas em qualquer local da matriz de dimensões 147×3087.

O Exemplo 4.11 segue os parâmetros do Exemplo 4.7, com a adição de 1000 manchas.

Exemplo 4.11

As Figuras 4.18 e 4.19 ilustram a sequência de imagens ao longo do esquema da Figura 4.1, para o caso da adição de 1000 manchas com moldura quadrada e dimensão $a = 7$.

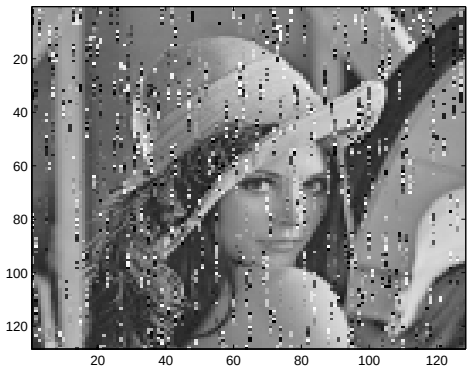
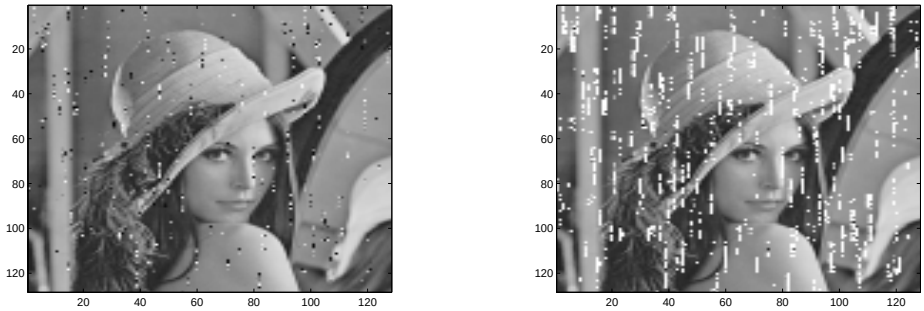


Figura 4.18: *Exemplo de imagem afetada por 1000 manchas com moldura quadrada e dimensão $a = 7$. Imagem reconstruída com os bits de informação da imagem não decodificada*

.



(a) Imagem decodificada por armadilha adptativa (b) Imagem decodificada por armadilha fixa

Figura 4.19: *Exemplo de imagem afetada por 1000 manchas com moldura quadrada e dimensão $a = 7$. Imagem reconstruída com os bits de informação obtidos pelos decodificadores de armadilha adaptativa (a) e o de armadilha fixa (b)*

Ao realizar a contagem do *pixels* diferentes, a imagem da Figura 4.19 obtida pelo decodificador de armadilha adaptativa (a), possui 845 *pixels* diferentes, ou seja, 5,16 % dos *pixels* totais. Enquanto a imagem obtida pelo decodificador de armadilha fixa (b) apresenta um total de 1989 pixels diferentes em relação à imagem original, ou seja, 12,14 % dos pixels totais. Já a imagem da Figura 4.18 (b),

Tabela 4.10: *Percentual do número de pixels diferentes em relação à imagem original para o decodificador por armadilha adaptativa, o decodificador por armadilha fixa e antes da decodificação. Realizada a adição de 1000 manchas com moldura quadrada e de dimensão* $a = 7$.

Amostra	**Percentual Armadilha Adaptativa**	**Percentual Armadilha Fixa**	**Percentual Antes da decodificação**
1	5,16%	12,14%	25,35%
2	5,24%	12,04%	25,52%
3	4,27%	10,33%	24,50%
4	5,16%	11,59%	26,33%
5	4,68%	11,93%	24,93%
6	5,05%	12,62%	25,63%
7	4,86%	12,40%	25,02%
8	5,36%	12,79%	24,74%
9	4,05%	10,72%	23,79%
10	4,98%	12,31%	26,39%
Média	4,88%	11,89%	25,20%
Desvio Padrão	8,00 E-01	4,30 E-01	8,00 E-01

na qual não sofreu decodificação possui um total de 4154 *pixels* diferentes em relação à imagem original, ou seja, 25,35 % dos *pixels* totais.

O procedimento do Exemplo 4.11 foi repetido mais nove vezes gerando os dados da Tabela 4.10.

Tabela 4.11: *Percentual do número de pixels diferentes em relação à imagem original para o decodificador por armadilha adaptativa, o decodificador por armadilha fixa e antes da decodificação. Realizada a adição de 2000 manchas com moldura quadrada e de dimensão* $a = 7$.

Amostra	**Percentual Armadilha Adaptativa**	**Percentual Armadilha Fixa**	**Percentual Antes da decodificação**
1	16,86%	34,84%	44,81%
2	15,81%	34,63%	43,33%
3	16,71%	34,34%	43,19%
4	16,67%	33,82%	42,88%
5	15,78%	32,17%	45,79%
6	15,77%	32,57%	42,21%
7	16,74%	34,86%	41,91%
8	16,00%	34,29%	42,78%
9	16,60%	34,18%	44,54%
10	15,29%	32,90%	43,82%
Média	16,22%	33,86%	43,53%
Desvio Padrão	5,50 E-01	9,70 E-01	1,22 E-00

Seguindo o mesmo raciocínio, adicionou-se 2000 manchas com os mesmos parâmetros do Exemplo 4.11. Após 10 repetições obteve-se os dados da Tabela 4.11.

O Exemplo 4.12 segue os mesmos parâmetros do Exemplo 4.8, com a adição de 1000 manchas.

Exemplo 4.12

As Figuras 4.20 e 4.21 ilustram a sequência de imagens ao longo do esquema da Figura 4.1, para o caso da adição de 1000 manchas com moldura retangular e dimensões $a = 8$ *e* $b = 9$.

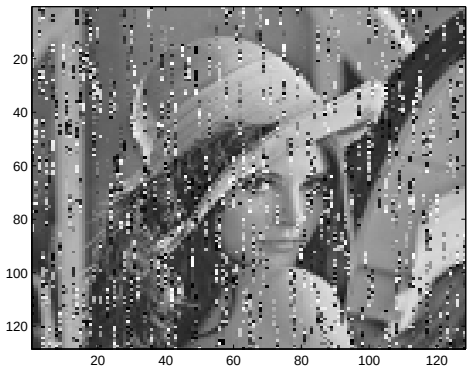
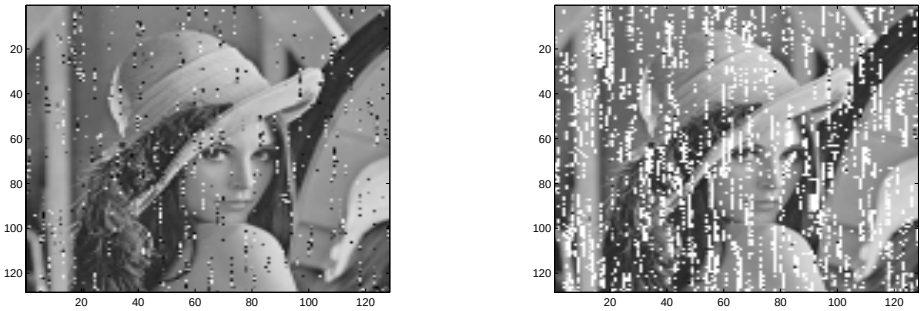


Figura 4.20: *Exemplo de imagem afetada por 1000 manchas de moldura retangular com dimensões* $a = 8$ *e* $b = 9$. *Imagem reconstruída com os bits de informação da imagem não decodificada*

.



(a) Imagem decodificada por armadilha adptativa (b) Imagem decodificada por armadilha fixa

Figura 4.21: *Exemplo de imagem afetada por 1000 manchas de moldura retangular com dimensões* $a = 8$ *e* $b = 9$. *Imagem reconstruída com os bits de informação obtidos pelos decodificadores de armadilha adaptativa (a) e o de armadilha fixa (b)*

Ao realizar a contagem do *pixels* diferentes, a imagem da Figura 4.21 obtida pelo decodificador de armadilha adaptativa (a), possui 1743 *pixels* diferentes, ou seja, 10,64 % dos *pixels* totais. Enquanto a imagem obtida pelo decodificador de armadilha fixa (b) apresenta um total de 4799 pixels diferentes em relação à imagem original, ou seja, 29,29 % dos pixels totais. Já a imagem da Figura

Tabela 4.12: *Percentual do número de pixels diferentes em relação à imagem original para o decodificador por armadilha adaptativa, o decodificador por armadilha fixa e antes da decodificação. Realizada a adição de 1000 manchas com moldura retangular e dimensões* $a = 8$ *e* $b = 9$.

Amostra	**Percentual Armadilha Adaptativa**	**Percentual Armadilha Fixa**	**Percentual Antes da decodificação**
1	10,64%	29,29%	32,15%
2	10,25%	29,13%	32,93%
3	10,50%	29,16%	32,27%
4	10,24%	29,78%	32,56%
5	10,19%	29,73%	33,43%
6	10,03%	29,43%	32,76%
7	10,54%	29,79%	31,52%
8	10,10%	29,26%	33,36%
9	9,50%	29,57%	33,95%
10	10,28%	29,00%	32,51%
Média	10,23%	29,41%	32,75%
Desvio Padrão	3,20 E-01	2,90 E-01	7,10 E-01

4.20 (b), na qual não sofreu decodificação possui um total de 5268 *pixels* diferentes em relação à imagem original, ou seja, 32,15 % dos *pixels* totais.

O procedimento do Exemplo 4.12 foi repetido mais nove vezes gerando os dados da Tabela 4.12.

Tabela 4.13: *Percentual do número de pixels diferentes em relação à imagem original para o decodificador por armadilha adaptativa, o decodificador por armadilha fixa e antes da decodificação. Realizada a adição de 2000 manchas com moldura retangular e dimensões* $a = 8$ *e* $b = 9$.

Amostra	**Percentual Armadilha Adaptativa**	**Percentual Armadilha Fixa**	**Percentual Antes da decodificação**
1	28,71%	56,37%	54,74%
2	28,00%	55,68%	54,10%
3	28,61%	56,38%	53,94%
4	28,19%	57,12%	54,22%
5	28,80%	55,39%	53,95%
6	28,83%	55,89%	53,68%
7	28,48%	55,76%	55,09%
8	28,22%	55,22%	55,24%
9	28,17%	54,81%	53,69%
10	28,34%	55,39%	55,08%
Média	28,43%	55,80%	54,37%
Desvio Padrão	2,90 E-01	6,70 E-01	6,10 E-00

Seguindo o mesmo raciocínio, adicionou-se 2000 manchas com os mesmos parâmetros do Exemplo 4.11. Após 10 repetições obteve-se os dados da Tabela 4.13.

O Exemplo 4.13 segue os mesmos parâmetros do Exemplo 4.9, com a adição de 1000 manchas.

Exemplo 4.13

As Figuras 4.22 e 4.23 ilustram a sequência de imagens ao longo do esquema da Figura 4.1, para o caso da adição de 1000 manchas com moldura em cruz e dimensões $a = 10$ *e* $b = 10$.

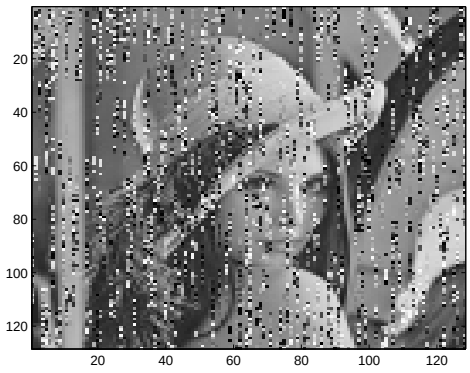
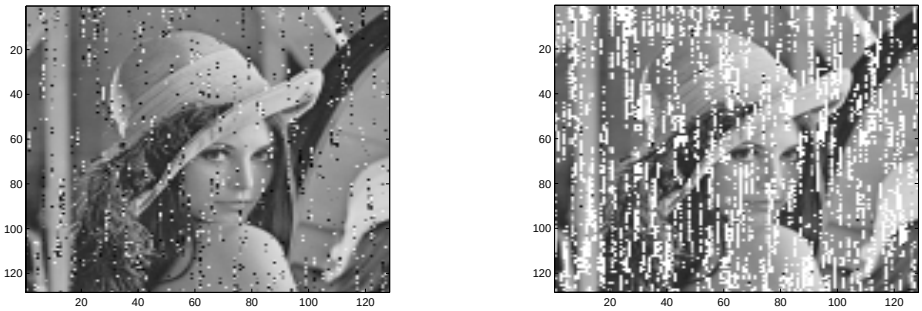


Figura 4.22: *Exemplo de imagem afetada por 1000 manchas de moldura em cruz com dimensões* $a = 10$ *e* $b = 10$. *Imagem reconstruída com os bits de informação da imagem não decodificada*

.



(a) Imagem decodificada por armadilha adptativa (b) Imagem decodificada por armadilha fixa

Figura 4.23: *Exemplo de imagem afetada por 1000 manchas de moldura em cruz com dimensões* $a = 10$ *e* $b = 10$. *Imagem reconstruída com os bits de informação obtidos pelos decodificadores de armadilha adaptativa (a) e o de armadilha fixa (b)*

Ao realizar a contagem do *pixels* diferentes, a imagem da Figura 4.23 obtida pelo decodificador de armadilha adaptativa (a), possui 2319 *pixels* diferentes, ou seja, 14,15 % dos *pixels* totais. Enquanto a imagem obtida pelo decodificador de armadilha fixa (b) apresenta um total de 6108 pixels diferentes em relação à imagem original, ou seja, 37,28 % dos pixels totais. Já a imagem da Figura

Tabela 4.14: *Percentual do número de pixels diferentes em relação à imagem original para o decodificador por armadilha adaptativa, o decodificador por armadilha fixa e antes da decodificação. Realizada a adição de 1000 manchas com moldura em cruz e dimensões* $a = 10$ *e* $b = 10$.

Amostra	**Percentual Armadilha Adaptativa**	**Percentual Armadilha Fixa**	**Percentual Antes da decodificação**
1	14,15%	37,28%	40,02%
2	15,80%	39,00%	38,89%
3	15,70%	39,22%	38,41%
4	16,27%	40,37%	40,84%
5	15,99%	38,02%	38,29%
6	16,54%	38,78%	41,05%
7	15,06%	38,24%	39,02%
8	15,95%	39,67%	39,95%
9	16,00%	39,25%	38,90%
10	15,35%	38,04%	38,63%
Média	15,68%	38,79%	39,41%
Desvio Padrão	6,80 E-01	9,10 E-01	9,90 E-01

4.22 (b), na qual não sofreu decodificação possui um total de 6557 *pixels* diferentes em relação à imagem original, ou seja, 40,02 % dos *pixels* totais.

O procedimento do Exemplo 4.13 foi repetido mais nove vezes gerando os dados da Tabela 4.14.

Tabela 4.15: *Percentual do número de pixels diferentes em relação à imagem original para o decodificador por armadilha adaptativa, o decodificador por armadilha fixa e antes da decodificação. Realizada a adição de 2000 manchas com moldura em cruz e dimensões* $a = 10$ *e* $b = 10$.

Amostra	**Percentual Armadilha Adaptativa**	**Percentual Armadilha Fixa**	**Percentual Antes da decodificação**
1	39,22%	66,66%	61,32%
2	40,20%	67,98%	63,52%
3	39,58%	67,44%	61,06%
4	39,69%	67,36%	62,79%
5	40,03%	68,40%	62,62%
6	39,42%	68,27%	62,43%
7	37,67%	65,78%	61,82%
8	40,86%	68,42%	64,28%
9	38,98%	66,48%	62,45%
10	38,80%	67,02%	61,41%
Média	39,24%	67,38%	62,37%
Desvio Padrão	8,70 E-01	9,00 E-01	1,01 E-00

Seguindo o mesmo raciocínio, adicionou-se 2000 manchas com os mesmos parâmetros do Exemplo 4.12. Após 10 repetições obteve-se os dados da Tabela 4.15.

Nesta seção, obteve-se resultados em que cada mancha é inserida em qualquer local da matriz. O tamanho da mancha determina a quantidade máxima de manchas que podem ser inseridas no arranjo matricial. A matriz a ser adicionado as manchas possui uma dimensão de 147×3087. Para uma melhor visualização do efeito da adição de 1000 e 2000 manchas a matriz da imagem, a Tabela 4.16 relaciona a dimensão da mancha com a quantidade máxima que pode ser inserida na matriz. Os valores percentuais apresentados para 1000 e 2000 manchas dão a idéia da quantidade de erros que é inserido na matriz.

Tabela 4.16: *Quantidade máxima de cada tipo de mancha que pode ser adicionada a matriz total da imagem e valores percentuais que representam a adição de 1000 e 2000 manchas.*

Tipo da moldura e dimensão(ões)	Quantidade Máxima	Percentual 1000 manchas	Percentual 2000 manchas
quadrada a= 7	9241	10,81%	21,60%
retangular a= 8 e b=9	6174	16,20%	32,40%
cruz a=10 e b=10	4312	23,19%	46,38%

Pela análise das Tabelas 4.9 a 4.12 percebe-se a melhora na correção para o decodificador por armadilha adaptativa em relação ao de armadilha fixa. Tal fato se deve pelo método adaptativo ser capaz de corrigir surtos com comprimentos maiores em relação ao não-adaptativo. No entanto, para as Tabelas 4.13 a 4.15 a decodificação por armadilha fixa removeu uma quantidade não representativa dos erros ou, em piores casos, adicionou mais erros a matriz. Isto ocorre devido ao comprimento dos surtos inseridos ultrapassarem a capacidade de correção do código. Quando isso ocorre, o código na tentativa de corrigir os erros inseridos acaba por inserir mais erros e danificar ainda mais a palavra-código. O próximo capítulo apresenta a conclusão do trabalho desenvolvido.

CAPÍTULO 5

CONCLUSÃO, COMENTÁRIOS E SUGESTÕES

NESTE capítulo, é apresentado de forma simplificada o trabalho desenvolvido nesta dissertação, são feitos comentários e, finalmente, propostas algumas sugestões de trabalhos futuros.

5.1 CONCLUSÃO E COMENTÁRIOS FINAIS

Os principais objetivos desta dissertação foram utilizar entrelaçamento e codificação em uma dimensão e analisar o desempenho das técnicas de decodificação por armadilha fixa e adaptativa na simulação de correção de manchas de erros em arranjos bidimensionais. Estas manchas de erros afetam sistemas que utilizam matrizes como forma de armazenamento de informação. Um sistema de comunicação digital foi simulado utilizando códigos cíclicos lineares binários específicos em apenas uma dimensão do arranjo. Nesse sistema a matriz de informação é dividida em blocos de matrizes menores. Obteve-se resultados quando uma única mancha ataca um bloco por vez e quando um determinado número de manchas ataca a matriz total. Cada bloco e a matriz total são afetados pela adição módulo 2 de uma mancha de erro que possui formatos de moldura quadrada, retangular e em cruz. Devido à capacidade de correção de erros em surtos dos códigos cíclicos, e utilizando o entrelaçamento proposto em [5], tais manchas de erros são distribuídas ao longo do arranjo, se tornando surtos em uma dimensão com comprimento igual ou menor ao original e dessa forma, podendo ser corrigidas com a técnica de decodificação por armadilha. Analisou-se a decodificação para o decodi-

ficador de armadilha fixa e o de armadilha adaptativa. A fim de observar o desempenho das técnicas de decodificação foram realizadas simulações nas quais se observou um melhor desempenho do sistema quando é utilizado o decodificador com armadilha adaptativa em relação ao de armadilha fixa para ambos os casos. Excetuando os casos em que devido ao comprimento do surto ser maior que a capacidade do código, o processo de decodificação inseriu mais erros a matriz, como apresentado em 4.14 e 4.15 por exemplo.

Iniciou-se com o modelo de sistema de comunicação digital simplificado descrevendo a função de cada bloco. O entendimento desse sistema é de fundamental importância para compreensão do trabalho desenvolvido.

Em seguida, a teoria sobre códigos de blocos lineares foi apresentada, juntamente com sua representação por matrizes e a sua capacidade de detecção de erros por meio do cálculo da síndrome. Dando continuidade, os códigos cíclicos foram introduzidos e com o uso de registradores de deslocamento, exemplos de codificação e detecção de erro foram apresentados. Logo após, os conceitos dos códigos corretores de erros em surtos e os algoritmos de decodificação por armadilha foram apresentados. Exemplos ilustraram a decodificação do mesmo surto para as duas técnicas utilizadas, observando o sucesso na correção de ambas.

Após a apresentação dos algoritmos de decodificação, foi explicado o passo a passo o sistema utilizado. Com o uso de uma imagem ilustrou-se a divisão dos blocos de informação e o caminho percorrido por ele. A teoria foi apresentada e exemplificados o entrelaçamento, a geração das manchas de erro e o desentrelaçamento de um bloco da imagem. Para a geração de manchas foram apresentados os limites das dimensões de cada moldura e histogramas que demonstram um comportamento aproximado da distribuição de probabilidade binomial para a distribuição de manchas em função do peso p.

Na Seção 4.4 é ilustrado o espalhamento de uma mancha de erro posteriormente adicionada aleatorialmente a alguma região da matriz. Percebe-se que os surtos tem seu comprimento reduzido ou mantido, conforme os Exemplos 4.5 e 4.6. O desempenho dos decodificadores de armadilha fixa e armadilha adaptativa são apresentadas por meio de tabelas, em que inicialmente se tem o controle do surto gerado, Tabela 4.1. Dando continuidade, é deixado livre o peso do surto, apenas especificando o seu comprimento, como nas Tabelas 4.2, 4.3, 4.4 e 4.5. Conforme esperado, percebeu-se a melhora na correção ao utilizar a técnica de decodificação por armadilha adaptativa em relação à armadilha fixa.

Em seguida após foram mostrados exemplos com cada tipo de moldura, como vistos nos Exem-

plos 4.7, 4.8 e 4.9. O Exemplo 4.10 simula mais fielmente a realidade uma vez que não há controle sobre a moldura da mancha de erro que atua em cada bloco. Para todos os exemplos ao se comparar as técnicas de decodificação percebeu-se um nível igual ou melhor quando se usa o decodificador de armadilha adaptativa proposto por Gallager [7], por meio da contagem dos *pixels* diferentes em relação à imagem original.

Finalizando,os Exemplos 4.10, 4.11 e 4.12 demonstram o efeito do ataque com quantidade estabelecida de 1000 e 2000 manchas para cada tipo de moldura. Para a maioria dos exemplos ao se comparar as técnicas de decodificação percebeu-se um nível igual ou melhor quando se usa o decodificador de armadilha adaptativa proposto por Gallager [7], por meio da contagem dos *pixels* diferentes em relação à imagem original. As tabelas da Seção 4.6.2 ilustram esse resultado.

5.2 CONTRIBUIÇÕES FUTURAS

A seguir, é apresentada uma sugestão para trabalhos futuros.

- Comparar o desempenho ao utilizar outros códigos corretores de erros de surto como, por exemplo, *Fire Codes* [18] ou *Burton Codes* [33] em vez dos códigos corretores de erros em surto deste trabalho. Esses códigos podem ser utilizados com um maior comprimento da palavra código, dessa forma um menor número de blocos da matriz de informação é gerado no caso de imagens e um ganho em processamento espera-se ser observado.

REFERÊNCIAS

[1] C. PIMENTEL, **Comunicação Digital**. Brasport, Rio de Janeiro, 2007.

[2] C. E. SHANNON, A mathematical theory of communication, *Bell System Technical Journal*, v. 27, n. 3 and 4, p. 379–423 and 623–656, julho e outubro 1948.

[3] ——, Communication theory of secrecy systems, *Bell System Technical Journal*, v. 28, n. 4, p. 656–715, Outubro 1949.

[4] R. W. HAMMING, Error detecting and error correcting codes, *Bell System Technical Journal*, v. 29, n. 2, p. 147–160, Abril 1950.

[5] V. C. DA ROCHA JR., W. P. S. GUIMARAES, & P. FARRELL, Two-dimensional interleaving with burst error-correcting codes, *IET Electronics Letters*, v. 38 (18), p. 1042–1043, Agosto 2002.

[6] S. LIN & D. J. COSTELLO, **Error Control Coding**, 2ª ed. Prentice Hall, 2004.

[7] R. G. GALLAGER, **Information Theory and Reliable Communication**, 1ª ed. John Wiley and Songs, Inc, 1968.

[8] R. E. BLAHUT, **Algebraic Codes for Data Transmission**. Cambridge University Press, 2003.

[9] S. RYAN, WILLIAN E. E LIN, **Channel Codes Classical and Modern**. Cambridge, 2009.

[10] T. M. COVER & J. A. THOMAS, **Elements of Information Theory**, 2ª ed. John Wiley and Sons, 2006.

[11] E. PRANGE, **Cyclic Error-Correcting Codes in Two Symbols**, 1ª ed. AFCRC-TN, 1957.

[12] S. B. WICKER, **Error Control Systems for Digital Communication and Storage**. Prentice Hall, 1995.

[13] E. R. BERLEKAMP, **Algebraic Coding Theory**. McGraw-Hill, 1968.

[14] W. W. PETERSON & E. J. W. JR., **Error correction codes**, 2ª ed. Cambridge, 1972.

[15] J. E. MEGGITT, Error correcting codes and their implementation, *IRE Trans. Inf. Theory*, v. IT-7, p. 232–244, outubro 1961.

[16] N. ABRAMSON, A class of systematic codes for non-independent errors, *IRE Trans. INf. Theory*, v. IT-4(4), p. 150–157, Dezembro 1959.

[17] N. ABRAMSON & B. ELSPAS, Double-error-correcting codes and decoders for non independent binary errors, *UNESCO Inf. Process. Conf. Paris*, 1959.

[18] P. FIRE, A class of multiple-error-correcting binary codes for non independent binary errors, *Sylvania Report No RSL-E-2; Sylvania Electronic Defense Laboratory*, Março 1959.

[19] J. J. STONE, Multiple burst error correction, *Inf Control*, v. 4, p. 324–331, Dezembro 1961.

[20] J. E. MEGGITT, Error correcting codes for correcting burst of errors, *IBM J. Res. Dev.*, v. 4, p. 329–334, Julho 1960.

[21] R. T. CHIEN, Burst-correction codes with high-speed decoding, *IEEE Trans. Inf. Theory*, v. IT-1, p. 109–113, janeiro 1969.

[22] S. H. REIGER, Codes for the correction of "clustered"errors, *IRE Trans. Inf. Theory*, v. IT-6, p. 16–21, Março 1960.

[23] M. E. MITCHELL, Error-trap decoding of cyclic codes, *G. E. report*, n. 62MCD3, Dezembro 1962.

[24] V. C. DA ROCHA JR., Two-dimensional interleaving, *in , Darnell, M. and Honary: 'Communications and Coding'*, p. 82–88, 1998.

[25] P. FARRELL, An introduction to array error control codes, *in LONGO, G., MARCHI, M., SGARRO, A.: Geometries, codes and cryptography*, p. 101–128, 1990.

[26] A. KAUFFMAN, M. D. MORAES, R. LIMA, & R. C. DE SOUZA, A technique for correcting clusters of errors, *Int. Telecommunications Symp., Acapulco, México*, p. 538–540, 1996.

[27] M. BLAUM, J. BRUCK, & A. VARDY, Interleaving schemes for multidimensional clusters errors, *IEEE Trans. Inf. Theory*, v. 44 (2), p. 730–743, Março 1998.

[28] T. ETZION. & A. VARDY, Two-dimensional interleaving schemes with repetitions: constructions and bounds, *IEEE Trans. Inf. Theory*, v. 48 (2), p. 428–457, 2002.

[29] C. D. ALMEIDA, C. SUEN, & R. PALAZZO, Efficient two-dimensional interleaving technique by use of the set partitioning concept, *Electron. Lett.*, v. 32 (6), p. 64–66, 1996.

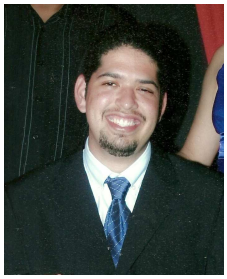
[30] M. SCHWARTZ & T. ETZION, Two-dimensional cluster-correcting codes, *IEEE Trans. Inform. Theory*, v. 51 (6), p. 2121 – 2132, 2005.

[31] T. ETZION & E. YAAKOBI, Error-correction of multidimensional bursts, *IEEE Trans. Inform. Theory*, v. 55 (3), p. 961 – 976, 2009.

[32] U. VITERBI, Signal and image processing institute, 2011, [Online; acessado 25-Dezembro-2011]. [Online]. Disponível: http://sipi.usc.edu/database/database.php?volume=misc&image=12#top

[33] H. O. BURTON, "a class of asymptotically optimal burst correcting block codes", *ICCC, Boulder, Color*, Junho 1969.

SOBRE O AUTOR



O autor nasceu em Recife, Pernambuco, no dia 25 de Julho de 1986. Formou-se em Engenharia Elétrica, modalidade Eletrônica, pela Universidade Federal de Pernambuco em 2009. Seus interesses de pesquisa incluem Teoria da Informação, Códigos Corretores de Erro, Sistemas de Comunicação Digital e Processamento Digital de Sinais.

Endereço: Rua Artur Wanderley, 205
50740-310, Várzea
Recife - PE
Brasil

e-mail: paulmrt.25@gmail.com

Esta dissertação foi diagramada usando LaTeX 2_ε[1] pelo autor.

[1] LaTeX 2_ε é uma extensão do LaTeX. LaTeX é uma coleção de macros criadas por Leslie Lamport para o sistema TeX, que foi desenvolvido por Donald E. Knuth. TeX é uma marca registrada da Sociedade Americana de Matemática ($\mathcal{AMS}$). O estilo usado na formatação desta dissertação foi escrito por Dinesh Das, Universidade do Texas. Modificado por Renato José de Sobral Cintra (2001) e por Andrei Leite Wanderley (2005), ambos da Universidade Federal de Pernambuco. Sua úlltima modificação ocorreu em 2010 realizada por José Sampaio de Lemos Neto, também da Universidade Federal de Pernambuco.