UNIVERSIDADE ESTADUAL DE FEIRA DE SANTANA

DEPARTAMENTO DE TECNOLOGIA

TEC498 PROJETO DE CIRCUITOS DIGITAIS

COMUNICAÇÃO SEM ERROS

Lucas Alves da Encarnação Oliveira, Velder Araújo Soares

Tutor: João Bosco Gertrudes

1 INTRODUÇÃO

A tecnologia da informação visa produzir, manipular e acessar de forma eficiente grandes quantidades de informação utilizando para tais finalidades, sistemas digitais. Entretando, estes sistemas estão sujeitos a falhas devido a ínumeras causas, principalmente interferências eletromagnéticas, resultando em processamentos errôneos.

A empresa Teiú Tecnologia Industrial S.A que desenvolve sistemas de controle de processo, solicitou a integração entre um conjunto de dispositivos e uma central de gerenciamento, focando em preservar a integridade da informação, já que este sistema de controle lida com processos críticos.

Este artigo descreve a implementação de um circuito de detecção e correção de erro, utilizando um dispositivo programável. Este circuito recebe quatro bits como entrada, verifica a integridade e corrige até um bit incorreto.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 CÓDIGO DE HAMMING

Essa técnica de detecção e correção foi desenvolvida por Richard Hamming, em meados de 1950, e é caracterizado por não só localizar o erro, mas também corrigir até 1 bit. Este método de detecção e correção de erro usa um determinado número de bits, localizados em pontos estratégicos da mensagem codificada. O número de bits de paridade, depende do número de bits da informação. O código de *Hamming* usa a relação entre bits de redundância e os bits de dados, assim pode ser aplicado a qualquer número de bits de dados (HAMMING, 1950), a tabela 1 mostra esta relação.

Tabela 1 – Relação entre bits de paridade e bits de dados

Bits de paridade	**Total de bits**	**Bits de dados**	**Nome**	**Proporção**
2	3	1	*Hamming(3, 1)*	$1/3 = 0.333$
3	7	4	*Hamming(7, 4)*	$4/7 = 0.571$
4	15	11	*Hamming(15, 11)*	$11/15 = 0.733$
5	31	26	*Hamming(31, 26)*	$26/31 = 0.839$

Fonte: (HAMMING. . . , 2017)

2.1.1 Bit de redundância

Um bit de redundância é a diferença entre o número de bits da sequência de dados reais e os bits transmitidos. Esses bits de redundância são utilizados num sistema de comunicação para detectar e corrigir o erro se houver.

No código *Hamming*, os bits de redundância são colocados em determinadas posições calculadas para corrigir o erro. A distância entre os bits de redundância é chamada de Distância *Hamming*.

2.1.2 Número de bits de paridade

O número de bits de paridade a serem adicionados a uma sequência de dados depende do número de bits de informação da cadeia de dados a ser transmitida. O número de bits de paridade será calculado usando os bits de dados. Esta relação é dada abaixo.

$$2^p >= d + p + 1 \tag{2.1}$$

onde d é o número de bits de dados e p representa o número de bits de paridade.

2.2 PORTA LÓGICA OU-EXCLUSIVO

A porta lógica OU-Exclusivo é uma simplificação da expressão $A \oplus B = A\overline{B} + \overline{A}B$, retornando nível lógico 1, se o número de '1' na entrada é par, e consequentemente retornando nível lógico 0 caso ocorra o contrário, como descrito na tabela 2. Devido a essa propriedade, (WAKERLY, 2010) afirma que : "Exclusive-OR and Exclusive-NOR gates may be viewed as 1-bit comparator"

[1]

[1] OU-Exclusivo e não OU-Exclusivo podem ser vistos como comparadores de 1 bit

Tabela 2 – OU-Exclusivo

A	B	x=A⊕B
0	0	0
0	1	1
1	0	1
1	1	0

Fonte: (TOCCI; WIDMER; MOSS, 2003)

2.2.1 Como inversora

A porta Ou-exclusivo, Considerando $B = 1$, possui o comportamento de inverter sua outra entrada

$$Y = A\overline{1} + \overline{A}1$$
$$Y = A * 0 + \overline{A}$$
$$Y = 0 + \overline{A}$$
$$Y = \overline{A}$$

2.2.2 Como passe-livre

$$Y = A\overline{0} + \overline{A} * 0$$
$$Y = A + 0$$
$$Y = A$$

2.2.3 Teoremas de DeMorgan

O matemático De Morgan contribuiu com dois teoremas para simplificações de expressões booleanas, (TOCCI; WIDMER; MOSS, 2003) define estes teoremas como:

> Os teoremas de DeMorgan são muito úteis na simplificação de expressões nas quais um produto ou uma soma de váriaveis aparacem negador [..] O teorema aponta que, quando a soma lógica (OR) de duas váriaveis é invertida, equivale a inverter cada variável individualmente e, em seguida, fazer a operação AND entre elas.

Seu teorema consiste em : $\overline{AB} \leftrightarrow \overline{A} + \overline{B}$, e pode ser visto através da tabela 3

Tabela 3 – De Morgan

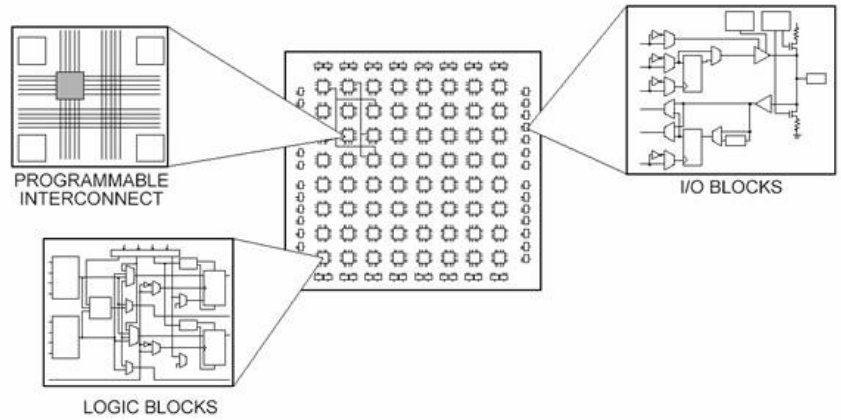
A	B	$\overline{AB}$	$\overline{A}+\overline{B}$
0	0	1	1
0	1	1	1
1	0	1	1
1	1	0	0

Fonte: Próprio autor

2.3 FUNDAMENTOS DA TECNOLOGIA FPGA

São chips de sílicio reprogramáveis, desenvolvidos em 1985 por Ross Freeman. Estes circuitos integrados combinam a flexibilidade de sistemas baseados em micro-processadores, com o desempenho de um circuito de aplicação específica (FUNDAMENTOS. . . , 2013). *Field Programmable Gate Array* consiste em blocos lógicos que são utilizadas para implementação de funções lógicas, como descrito na figura 1.

Figura 1 – Diferentes partes de uma FPGA



Fonte: (FUNDAMENTOS. . . , 2013)

2.4 MAPA DE KARNAUGH

Mapa de Karnaugh (Mapa-K) é um método de simplificação de expressões booleanas desenvolvido por Maurice Karnaugh em meados de 1953, designado para a síntese de circuitos combinacionais (KARNAUGH, 1953), este método também pode ser implementando utilizando recursos computacionais, visto que como acréscimo de muitas variáveis, o método perde a intuitividade. (WAKERLY, 2010) define esta técnica de minimização como:

> A Karnaught map is a graphical representation of a logic function's truth table. [..] The rows and columns of a Karnaugh map are labeled so that the input combination for any cell is easily determined from the row and column heading for that cell. The small number inside each cell is the correspomding minterm

number in the truth table [..] To represent a logic function on a Karnaugh map, we simply copy 1s and 0s from the truth table or equivalent to the corresponding cell of the map.

[2]

[2] O mapa de Karnaugh é uma representação gráfica da tabela verdade de uma função [..] As linhas e colunas do mapa de Karnaugh são nomeados de modo que a combinação de entradas de qualquer célula seja facilmente determinado pelo cabeçalho da linha e coluna daquela célula. O pequeno número dentro de cada célula é o *minterm* correspondente na tabela verdade [..] Para representar uma função lógica no mapa de Karnaugh, nós simplismente copiamos os 1s and 0s da tabela verdade ou equivalente para a célula correspondente no mapa

3 METODOLOGIA

Para realizar este trabalho, empregou-se uma metodologia dividida em 7 etapas, as quais podem ser vistas de forma detalhada na Tabela 4.

Tabela 4 – Metodologia empregada no desenvolvimento do circuito

Etapas de Metodologia	Descrição da Etapa
Escolha de uma técnica de detecção e correção do erro	Nesta etapa, foi definido o método para detectar o erro e corrigir.
Prototipação do circuito	Nesta etapa foi feito o diagrama de blocos do circuito.
Módulo codificador	Nesta etapa, foi construído o módulo do codificador, o qual é formado por circuitos geradores de paridade, resultante das expressões algébricas booleanas extraídas de uma tabela verdade.
Módulo decodificador	Nesta etapa, foi construído o módulo do decodificador, que é constituído de 3 circuitos: circuito verificador de paridade, decodificador 3x8 e um multiplexador.
Módulo indutor de erro	Nesta etapa foi construído o módulo que introduz um erro no circuito.
Plano de Testes	Nesta etapa, foi definido o plano de testes para cada módulo do circuito.
Aréa utilizada da FPGA	Nesta etapa, foi definido a área de utilização da FPGA para este trabalho.

A técnica de detecção e correção de erro escolhida para ser implementada neste trabalho, foi o *Hamming code* (Código de Hamming). Nas sessões tutoriais foi definido que o número de bits de entrada seria de 4 bits e que a paridade utilizada seria ímpar. Através da Equação 2.1 foi constatado que 3 bits de paridade seriam necessários para transferir uma mensagem de 4 bits com a correção de um único bit.

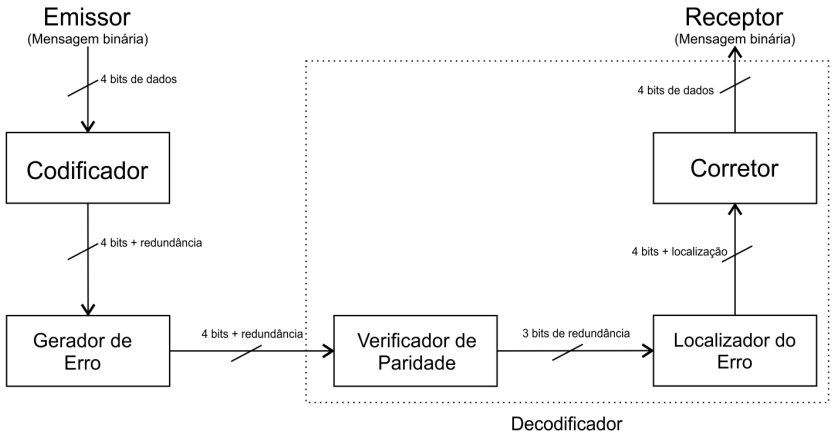
3.0.1 Modelagem do circuito

A Figura 2 mostra o diagrama de blocos do circuito, construído a partir das discussões nas sessões tutoriais.

3.0.2 Módulo codificador

Para o desenvolvimento do circuito codificador, utilizou-se 4 bits de entrada, os quais resultam em 3 bits de paridade. Para obter tais resultados, foi implementado um gerador de paridade no qual verifica as entradas A, B, C e tem P como o bit de paridade de saída. O número total de 1s deve ser ímpar para o gerar o bit de paridade ímpar.

Figura 2 – Diagrama de blocos do circuito



Na Tabela 5, o 1 é colocado no bit de paridade quando o número total de 1s é ímpar.

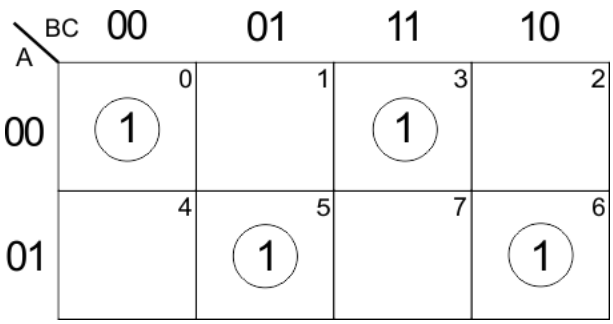
Tabela 5 – Tabela verdade do gerador de paridade

A	**B**	**C**	**P**
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

Fonte: Próprio autor

Aplicou-se o método de simplificação Mapa K para simplificar a expresão da Tabela 5, que pode ser visto na Figura 3.

Figura 3 – Simplificação pelo método de Karnaugh



A partir da Figura 3 a simplificação do bit de paridade, pode ser descrita como:

$$P = \overline{A}.\overline{B}.\overline{C} + \overline{A}.B.C + A.\overline{B}.C + A.B.\overline{C}$$

$$P = C(A\overline{B} + \overline{A}B) + \overline{C}(\overline{AB} + AB)$$

$$P = C(A \oplus B) + \overline{C}(\overline{A \oplus B})$$

$$P = A \oplus B \odot C$$

Logo, a relação para cada bit de paridade, de acordo com o código de *Hamming* é descrita da seguinte forma:

$$P_1 = D1 \oplus D2 \odot D4$$
$$P_2 = D1 \oplus D3 \odot D4$$
$$P_3 = D2 \oplus D3 \odot D4$$

3.1 MÓDULO DECODIFICADOR

O decodificador é um circuito lógico combinacional que converte entrada codificada em saídas decodificadas, desde que ambas sejam diferentes uma da outra. Este decodificador foi desenvolvido a partir de um circuito verificador de paridade ímpar, o qual recebe os bits de dados e de paridade e retorna uma sequência de bits. De acordo com a sequência de bits retornada, um circuito decodifcador 3x8 decodifica a mensagem para um circuito combinacional que inverte o bit errado.

3.1.1 Verificador de paridade ímpar

É um circuito lógico que verifica possíveis erros na transmissão. Este circuito pode ser um verificador de paridade par ou verificador de paridade ímpar dependendo do tipo de paridade gerada no final de transmissão. Neste trabalho, foi considerada a paridade ímpar.

Uma mensagem de três bits, juntamente com um bit de paridade ímpar é transmitido em algum ponto da transmissão. O verificador de paridade recebe esses 4 bits e verifica se há algum erro nos dados.

Se o número total de 1s nos dados é ímpar, então não aponta nenhum erro, entretanto, se o número total de 1s é ímpar, o erro é indicado.

A Tabela 6 mostra a tabela verdade do verificador de paridade ímpar, onde $C_p = 1$ se a mensagem recebida de 4 bits consiste em um número par de 1s e $C_p = 0$ se a mensagem não contém erro.

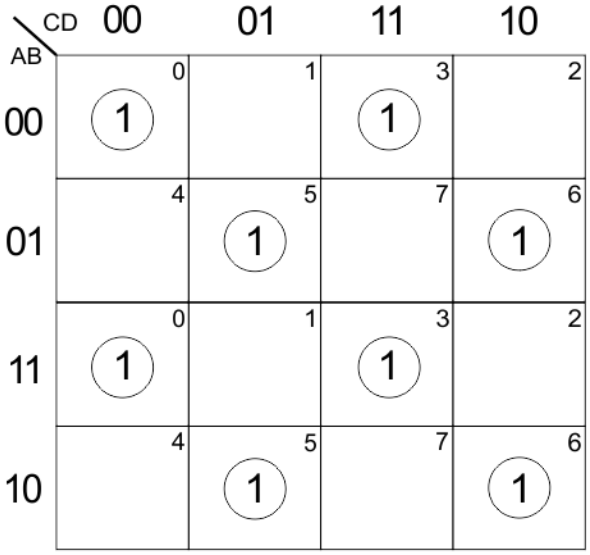
Tabela 6 – Tabela verdade do circuito para checar paridade ímpar

A	**B**	**C**	**P**	C_p
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

Fonte: Próprio autor

A Tabela 6 pode ser simplificada através do Mapa K, como mostra a Figura 4.

Figura 4 – Simplificação do circuito verificador de paridade ímpar pelo mapa de Karnaugh



$C_p = \overline{A}.\overline{B}.\overline{C}.\overline{D} + \overline{A}.\overline{B}.C.D + \overline{A}.B.\overline{C}.D + \overline{A}.B.C.\overline{D} + A.\overline{B}.\overline{C}.D + A.\overline{B}.C.\overline{D} + A.B.\overline{C}.\overline{D} + A.B.C.D$

$C_p = (C\overline{D} + \overline{C}D)(A\overline{B} + \overline{A}B) + (\overline{C\overline{D}} + CD)(\overline{A\overline{B}} + AB)$

$C_p = (C \oplus D)(A \oplus B) + (\overline{C \oplus D})(\overline{A \oplus B})$

$C_p = (A \odot B) \odot (C \odot D)$

A partir da simplificação encontrada acima, podemos definir para cada bit de verificação, as seguintes expressões:

$C_1 = D1 \odot D2 \odot D4 \odot P1$
$C_2 = D1 \odot D3 \odot D4 \odot P2$
$C_3 = D2 \odot D3 \odot D4 \odot P3$

3.1.2 Decodificador 3x8

Em um decodificador 3x8, três entradas são decodificadas em oito saídas. Tem três entradas como A, B e C e oito saídas de X0 a X7. Com base nas combinações das três entradas, apenas uma das oito saídas é ativada.

A Tabela 7 mostra a tabela verdade de um decodificador 3x8. A saída decodificada depende das combinações de entradas em A, B e C.

Tabela 7 – Tabela verdade do decodificador 3x8

Entradas			Saídas							
A	**B**	**C**	**X7**	**X6**	**X5**	**X4**	**X3**	**X2**	**X1**	**X0**
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

Fonte: Próprio autor

A partir da tabela verdade acima, os mintermos representa a equação de cada saída e são dadas como:

$X0 = \overline{A}.\overline{B}.\overline{C}$
$X1 = \overline{A}.\overline{B}.C$
$X2 = \overline{A}.B.\overline{C}$

$X3 = \overline{A}.B.C$
$X4 = A.\overline{B}.\overline{C}$
$X5 = A.\overline{B}.C$
$X6 = A.B.\overline{C}$
$X7 = A.B.C$

O circuito decodificador 3x8 foi implementado com três portas NOT e 8 portas AND.

3.1.3 Multiplexador

Com a informação da posição do bit errado, desenvolvou-se um pequeno circuito multiplexador, capaz de corrigir qualquer um dos quatro bits de dados que sofrerem erros. Para tal, utilizou-se a tabela verdade 8.

Tabela 8 – Tabela verdade multiplexador

A	**B**	**C**	**x**
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

Fonte: Próprio autor

Através da notação de soma de produtos, chegou-se as seguintes expressões booleanas:

$\overline{A}BC$ $A\overline{B}C$ $AB\overline{C}$ ABC

Deste modo, considerando D_1, D_2, D_3, D_4, os bits de dados, aplica-se: $D_1 \oplus \overline{A}BC$, $D_2 \oplus A\overline{B}C$, $D_3 \oplus AB\overline{C}$, $D_4 \oplus ABC$

Corrige-se o erro, até um bit, sendo D_1 o bit de dados mais significado e assim por diante.

3.2 MÓDULO INDUTOR DE ERRO

Para induzir ao erro, foi proposto um multiplexador, que recebe como entrada um habilitador e dois digítos binários, sendo assim capaz de identificar a posição do bit, para a síntese deste circuito combinacional foi utilizada a tabela 9.

Tabela 9 – Tabela verdade indutor de erro

Entradas			Saídas			
$Enable$	E_0	E_1	Bit_0	Bit_1	Bit_2	Bit_3
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1

Fonte: Próprio autor

3.3 PLANO DE TESTES

Para a realização dos testes de cada componente que compõe os módulos do Codificador e Decodificador, foi utilizada uma abordagem que pode ser vista na Tabela 10.

Tabela 10 – Abordagem utilizada para realização dos testes

Item	Descrição
Componente	Identificação do Componente.
Objeto de teste	Verificar o funcionamento das saídas.
Pré-condição	Conhecer as entradas e saídas previamente.
Resultado	Resultado esperado.

Vale ressaltar, que todos os testes a nível de *software* foram funcionais, realizados a partir do *Wave Form* da ferramenta Quartus II.

3.4 UTILIZAÇÃO DA FPGA

Para a implementação deste circuito, foi utilizado um kit de desenvolvimento *FPGA* da família *ACEX1 EP1K100QC208-3*. Um *dip-switch* foi utilizado para a entrada de dados e 8 LED's para saída.

Para a entradas de dados, as quatro primeiras chaves do *dip-switch* foram utilizadas, as duas chaves seguintes informam a posição em que o erro será inserido e logo após, uma chave que ativa ou desativa o módulo de introdução de erro. Os LED's verdes 1-4, indicam a mensagem corrigida, os vermelhos 9-12 indicam onde foi inserido o erro.

A Tabela 11 mostra a pinagem utilizada na FPGA.

Tabela 11 – Tabela pinagem utilizada neste protótipo

Função no Protótipo	**Nome do Sinal**	**Pino da FPGA**	**Descrição**
Entrada de Dados	DIP[0]	99	Chave tipo DIP 1
Entrada de Dados	DIP[1]	101	Chave tipo DIP 2
Entrada de Dados	DIP[2]	103	Chave tipo DIP 3
Entrada de Dados	DIP[3]	111	Chave tipo DIP 4
Posição do erro	DIP[4]	113	Chave tipo DIP 5
Posição do erro	DIP[5]	115	Chave tipo DIP 6
Habilitador de erro	DIP[6]	119	Chave tipo DIP 7
Exibir mensagem corrigida	LED[0]	127	LED 1
Exibir mensagem corrigida	LED[1]	131	LED 2
Exibir mensagem corrigida	LED[2]	133	LED 3
Exibir mensagem corrigida	LED[3]	135	LED 4
Indicador de erro	LED[8]	149	LED 9
Indicador de erro	LED[9]	157	LED 10
Indicador de erro	LED[10]	159	LED 11
Indicador de erro	LED[11]	161	LED 12

Fonte: Próprio autor

Em relação aos recursos da *FPGA*, foram utilizados 8/4.992 elementos lógicos, 15/147 *pins* e 0 *bits de memória*

4 RESULTADOS E DISCUSSÕES

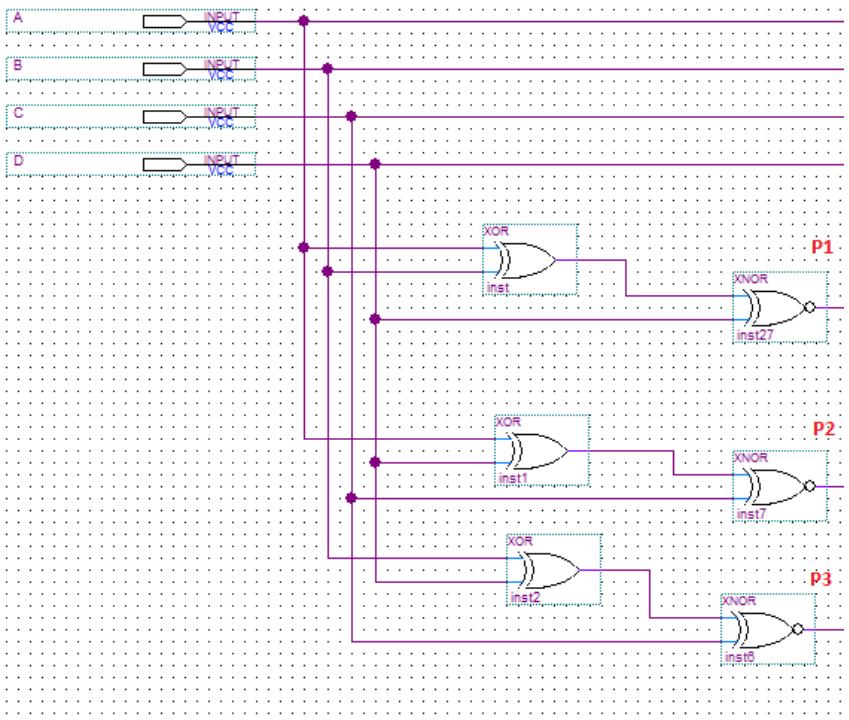
Nas primeiras sessões tutoriais foi entendido que usaríamos um dip-switch de 4 entradas e sua saída seriam os LED's assim como propõe o problema. As seções 4.1 e 3.1 descrevem a implemententação e o funcionamento do codificador e decodificador respectivamente.

4.1 DESENVOLVIMENTO DO CODIFICADOR

O codificador consiste em 4 *inputs* que combinados com os geradores de paridade resultam em 7 bits, 4 bits de dados e 3 de bits de paridade. As portas lógicas utilizadas nos geradores de paridades são resultantes da simplificação realizada na seção 3.0.2. A Figura 5 mostra a implementação do codificador no Quartus II.

Sendo P_1, P_2 e P_3, a saída referente aos bits de paridade.

Figura 5 – Módulo do Codificador implementado



Fonte: Próprio autor

4.2 DESENVOLVIMENTO DO DECODIFICADOR

O módulo completo do decodificador é baseado em 3 circuitos combinacionais responsáveis por encontrar e corrigir o bit errado. Esses circuitos são descritos nas etapas seguintes, baseados nas etapas do módulo do decodificador.

4.2.1 Detecção do erro

A Figura 6 mostra o circuito verificador de paridade ímpar, que é responsável pela verificação de integridade da mensagem. Como visto em 3.1.1, este circuito recebe 7 bits, 4 de dados e 3 de paridade que são verificados e resultam em uma sequência de bits. Logo, temos C_1, C_2 e C_3. Se o resultado for 000, não houve um erro, do contrários, o resultado será a posição em binário, sendo a posição 0 o bit mais significativo.

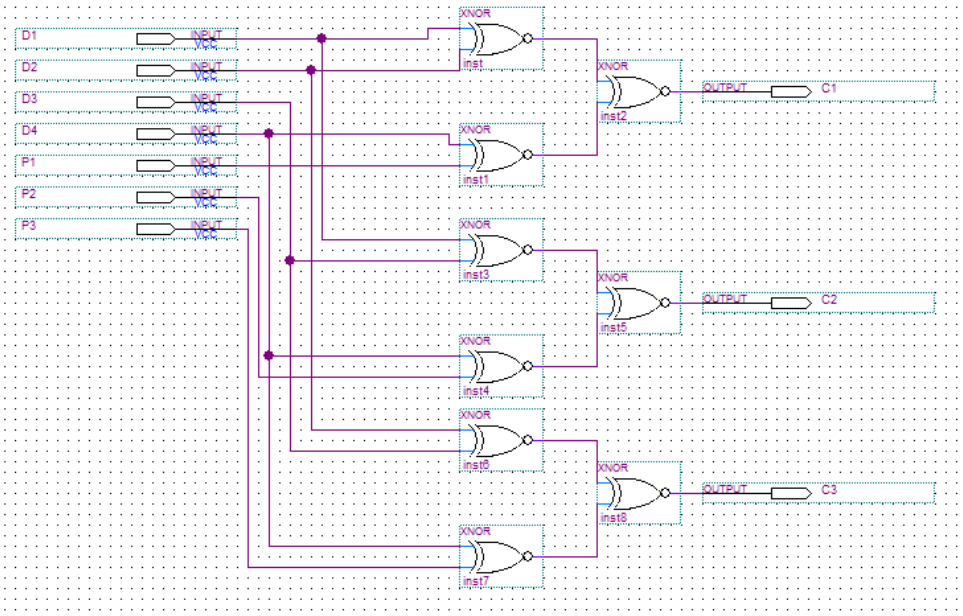
4.2.2 Decodificador

A Figura 7 mostra a implementação do circuito decodificador.

A implementação do circuito 3x8 consiste em 3 portas NOT e 8 portas AND. Este circuito, apesar de indicar a posição do erro, desconsidera os bits de paridade, pois os mesmos não são exibidos nos LED's.

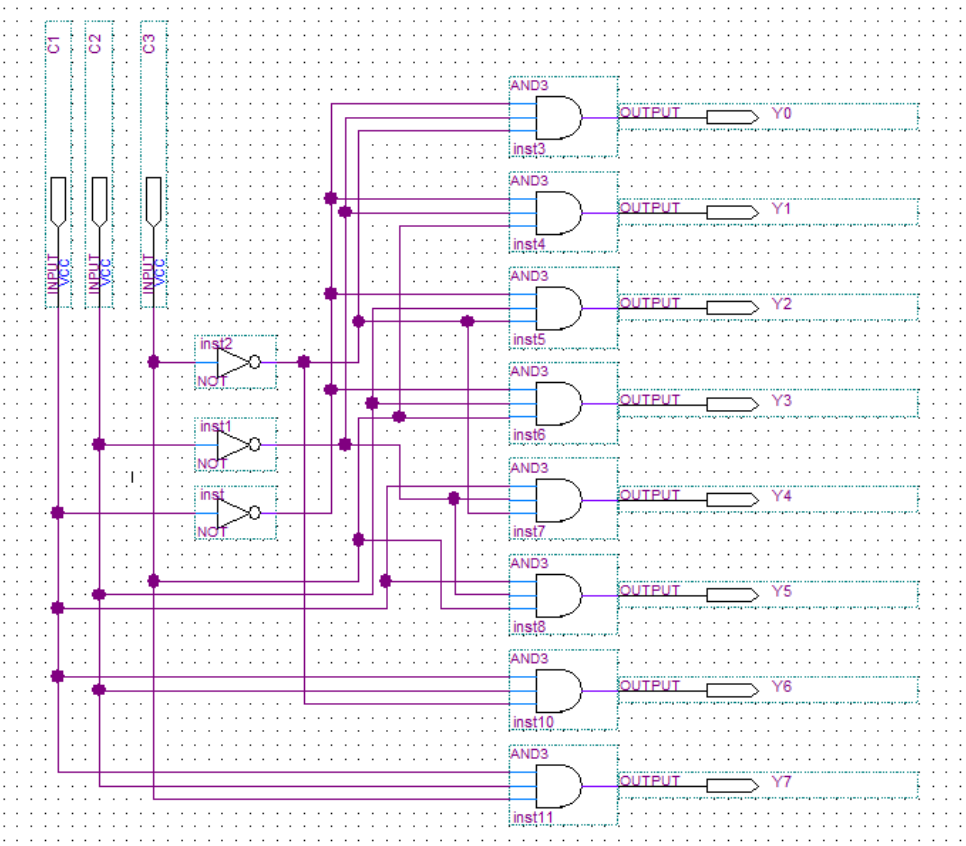
A partir dos 3 bits de entrada, este circuito tem saída 1 em uma das saídas, caso os bits de entrada sejam $\neq$ de 000, acusando assim a posição do erro.

Figura 6 – Circuito verificador de paridade ímpar



Fonte: Próprio autor

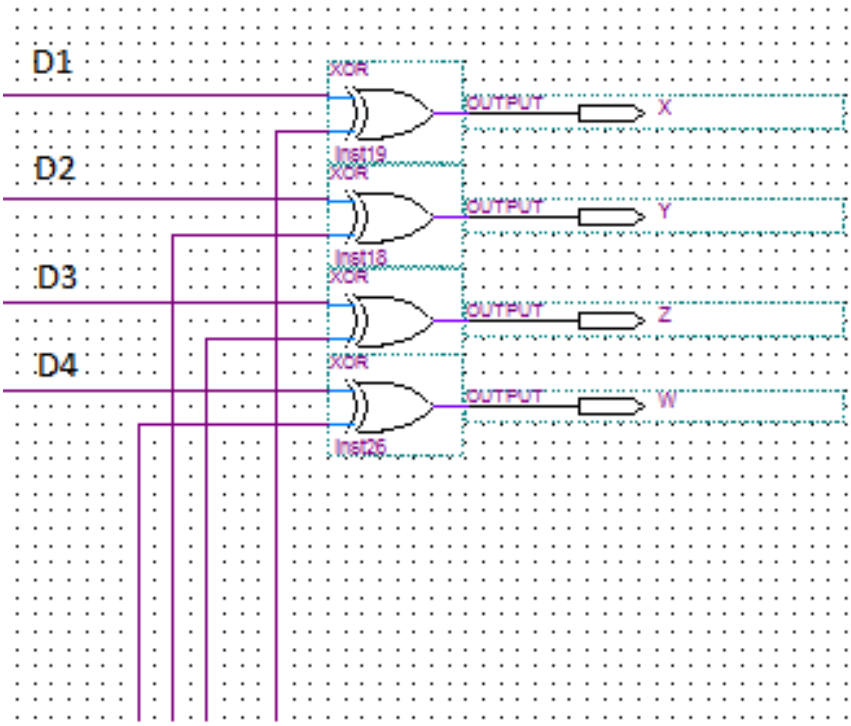
Figura 7 – Decodificador 3x8



Fonte: Próprio autor

4.2.3 Correção do Erro

Figura 8 – Circuito de correção do bit



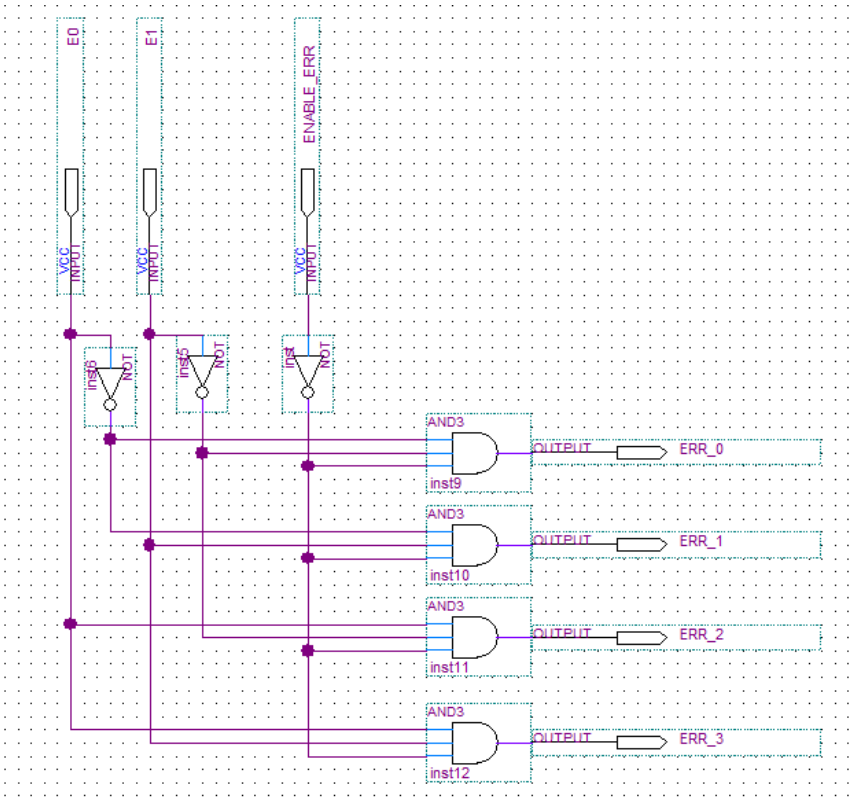
Fonte: Próprio autor

A Figura 8 exibe a implementação do circuito de correção do bit. Para corrigir o erro, esse circuito combinacional foi implementado utilizando 4 portas XOR. Já sabendo qual o bit errado, ele simplesmente inverte o bit.

4.2.4 Indução ao erro

O circuito é induzido ao erro ao habilitar um multiplexador através de um *non-debounced push button* e passar uma posição válida dentro do faixa de bits de dados representado em binário através de *dip-switch*, para que seja realizada uma inversão neste bit, como mostra a figura 9.

Figura 9 – Circuito de indução ao erro



Fonte: Próprio autor

4.2.5 Circuito Final

A Figura 10 mostra o circuito final implementado para solução do problema.

4.3 SIMULAÇÕES E TESTES NO QUARTUS II

Para simulações em nível de software, foi utilizado o simulator *Vector WaveForm* do Quartus II.

Tabela 12 – Saída do Codificador

Módulo	Entradas				Saídas		
Codificador	**D1**	**D2**	**D3**	**D4**	**P1**	**P2**	**P3**
Mensagem 1	0	0	0	0	1	1	1
Mensagem 2	0	1	1	0	0	0	1
Mensagem 3	1	1	0	0	1	0	0
Mensagem 4	1	1	1	1	0	0	0

Fonte: Próprio autor

A Tabela 12 mostra o resultado da saída do codificador a partir de uma mensagem de 4 bits de dados, onde as colunas Dn e Pn representam os bits de dados e os paridade respectivamente.

Tabela 13 – Saída do Verificador de Paridade

Módulo	Entradas							Saídas		
Verif. de Paridade	**P1**	**P2**	**D1**	**P3**	**D2**	**D3**	**D4**	**C1**	**C2**	**C3**
Mensagem 1	1	1	0	1	0	0	0	0	0	0
Mensagem 2	0	0	1	1	1	1	0	1	1	0
Mensagem 3	1	0	1	0	0	0	0	1	0	1
Mensagem 4	0	0	1	0	1	1	0	1	1	1

Fonte: Próprio autor

No circuito decodificador, temos como entrada as mensagens em binário e os bits de verificação de paridade ímpar como saída. Se ocorre um erro, como ocorre nas mensagens 2, 3 e 4, os bits de verificação de modo inverso mostra a posição do erro, como pode ser visto na Tabela 13.

5 CONCLUSÃO

Para o projeto de circuitos digitais descrito acima, foram utilizadas técnicas básicas de desenvolvimento de circuitos lógicos combinacionais, como a confecção das tabelas verdade, a simplificação de equações booleanas e a projeção do circuito completo no Quartus.

Os resultados estavam dentro do esperado, já que este projeto soluciona o problema,

tendo também um sinal indicativo quando algum erro acontece, logo em seguida corrigindo-o. Tendo a possibilidade de pequenas melhorias, como: indicar a posição exata de onde aconteceu o erro, determinar em qual bit será inserido o erro, entre outros.

Ao finalizar este projeto, pôde-se verificar a importância de técnicas de síntese e minimização de circuitos combinacionais, o real funcionamento de um dispositivo *Field Programmable Gate Array* e os passos a serem seguidos no desenvolvimento de um projeto de circuito digital,

REFERÊNCIAS

FUNDAMENTOS da Tecnologia FPGA. 2013. Disponível em: <http://www.ni.com/white-paper/6983/pt/>. Acesso em: 21 abril. 2017. Citado na página 4.

HAMMING code. 2017. Disponível em: <https://en.wikipedia.org/wiki/Hamming_code>. Acesso em: 22 abril. 2017. Citado na página 2.

HAMMING, R. W. Error detecting and error correcting codes. *Bell Labs Technical Journal*, Wiley Online Library, v. 29, n. 2, p. 147–160, 1950. Citado na página 1.

KARNAUGH, M. The map method for synthesis of combinational logic circuits. *Transactions of the American Institute of Electrical Engineers, Part I: Communication and Electronics*, IEEE, v. 72, n. 5, p. 593–599, 1953. Citado na página 4.

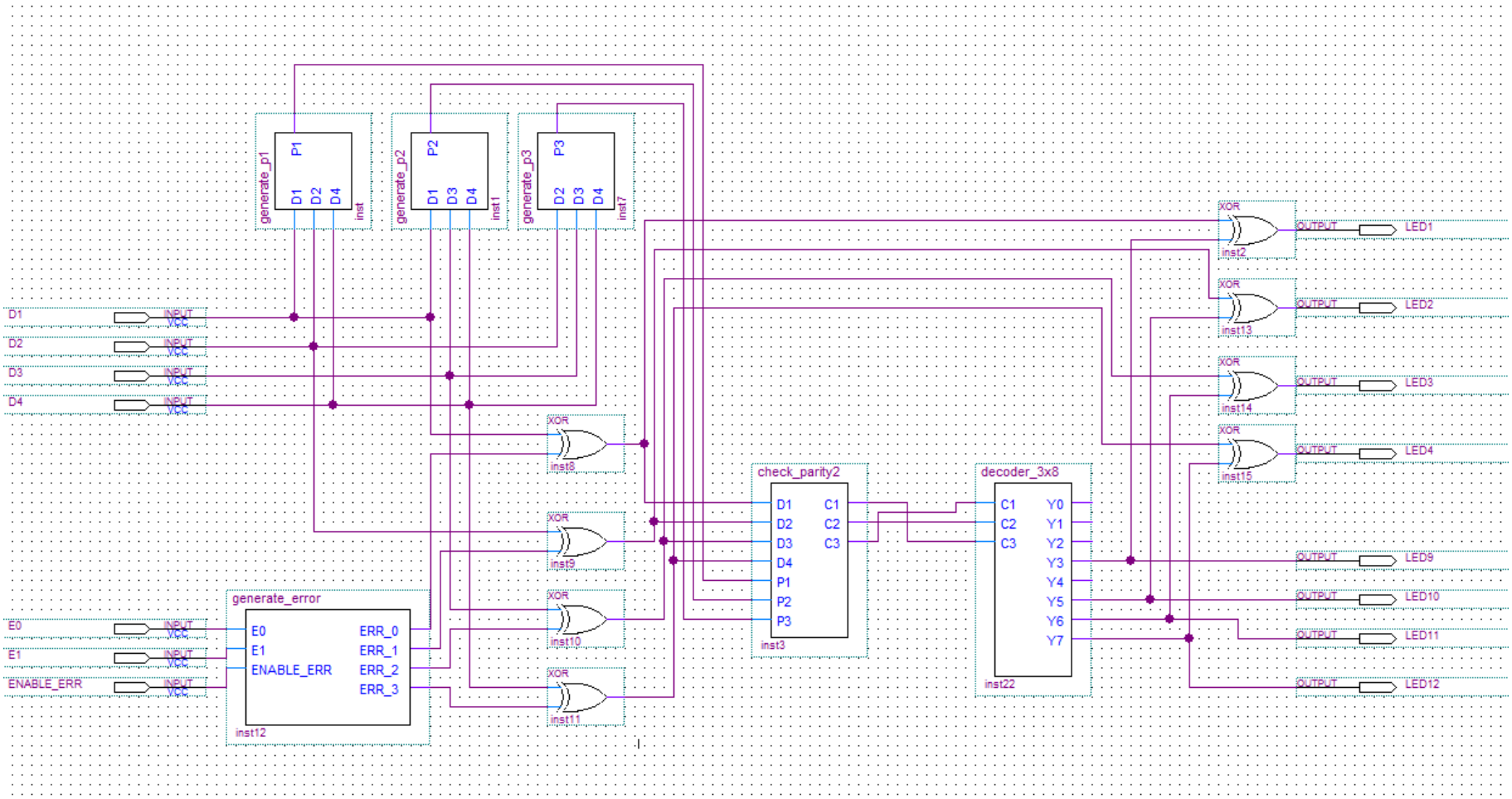
TOCCI, R. J.; WIDMER, N. S.; MOSS, G. L. *Sistemas digitais: princípios e aplicações*. [S.l.]: Prentice Hall, 2003. v. 8. Citado na página 3.

WAKERLY, J. F. *Digital Design: principles & practices*. [S.l.]: Prentice Hall, 2010. Citado 2 vezes nas páginas 2 e 4.

Anexos

A CIRCUITO FINAL

Figura 10 – Circuito Final



Fonte: Próprio autor