Experimento 5
Circuitos Combinacionais: Codificadores e Decodificadores

Grupo D1
Alexandre Augusto, 15/0056940
Anderson Vieira, 19/0102322
Gabriel de Castro, 21/1055432

[1]Dep. Ciência da Computação – Universidade de Brasília (UnB)
CIC0231 - Laboratório de Circuitos Lógicos
12 de março de 2023

150056940@aluno.unb.br, 190102322@aluno.unb.br, 211055432@aluno.unb.br

Abstract. *This report will present the implementation of combinational circuits that encode and decode from an X code to binary, the project will be carried out in the Quartus-II software version 13.0 SP1.*

Resumo. *Este relatório apresentará a implementação de circuitos combinacionais que codificam e decodificam de um código X para binário, o projeto será realizado no software Quartus-II versão 13.0 SP1.*

1. Introdução

O sistema binário possui várias vantagens devido à sua simplicidade nos circuitos, porém a relação entre os humanos e o sistema da tecnologia são meio distantes, principalmente pelo fato de que os humanos foram acostumados apenas com o sistema decimal, por isso é conveniente a criação de um codificador e decodificador, que transformam de um código para binário, e vice-versa analogamente. Tais circuitos foram implementado no Software Quartus-II versão 13.0 SP1 [Altera 2013], e os detalhes do mesmo podem ser observados em 2.

Codificadores são circuitos que transformam informações em formato de bits em outro conjunto de informações sem perda de informações, de tal forma que seu tamanho se reduza. Fazendo uso de tais circuitos, é possível obter n saídas para 2^n entradas.

1.1. Objetivos

Através de técnicas como a minimização de funções, conversões de bases e etc, os alunos deverão elaborar e implementar circuitos de codificação e decodificação no Quartus - II.

1.2. Materiais

Neste experimento foram utilizados os seguintes materiais e equipamentos:

- Software Quartus-II versão 13.0 SP1;
- Kit de desenvolvimento em FPGA DE2 ou DE2-70 Intel.

2. Procedimentos e Resultados

Na seção 2.1 definimos as funções que correspondem a tabela verdade do circuito do codificador. Na seção 2.2, projetamos o circuito obtido no software *Quartus-II*. Nas seções 2.3 e 2.4 fizemos o mesmo das seções 2.1 e 2.2, respectivamente. Na seção 2.5, juntamos tanto os projetos de codificador e decodificador para observar no kit FPGA as conversões realizadas.

2.1. Obtendo funções booleanas para o codificador

Tabela 1. Tabela verdade para o circuito do Codificador

Entradas										Saídas			
A_0	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9	**A**	**B**	**C**	**D**
1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	0	0	0	1	1
0	0	0	1	0	0	0	0	0	0	0	0	1	0
0	0	0	0	1	0	0	0	0	0	0	1	1	0
0	0	0	0	0	1	0	0	0	0	0	1	1	1
0	0	0	0	0	0	1	0	0	0	0	1	0	1
0	0	0	0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	0	0	0	1	0	1	1	0	0
0	0	0	0	0	0	0	0	0	1	1	1	0	1

No codificador, teremos o valor 0 definido como 000000001 (somente $A_(0)$ ativo), valor 1 como 000000010 (somente $A_(1)$ ativo) e assim por diante até 9 como 100000000 (somente $A_(9)$ ativo). Para obter as funções, é necessário averiguar saída a saída. Desta forma, obtemos:

$$F(A) = A_8 + A_9$$

$$F(B) = A_4 + A_5 + A_6 + A_7 + A_8 + A_9$$

$$F(C) = A_2 + A_3 + A_4 + A_5$$

$$F(D) = A_1 + A_2 + A_5 + A_6 + A_9$$

2.2. Projetando circuito do codificador no *Quartus-II*

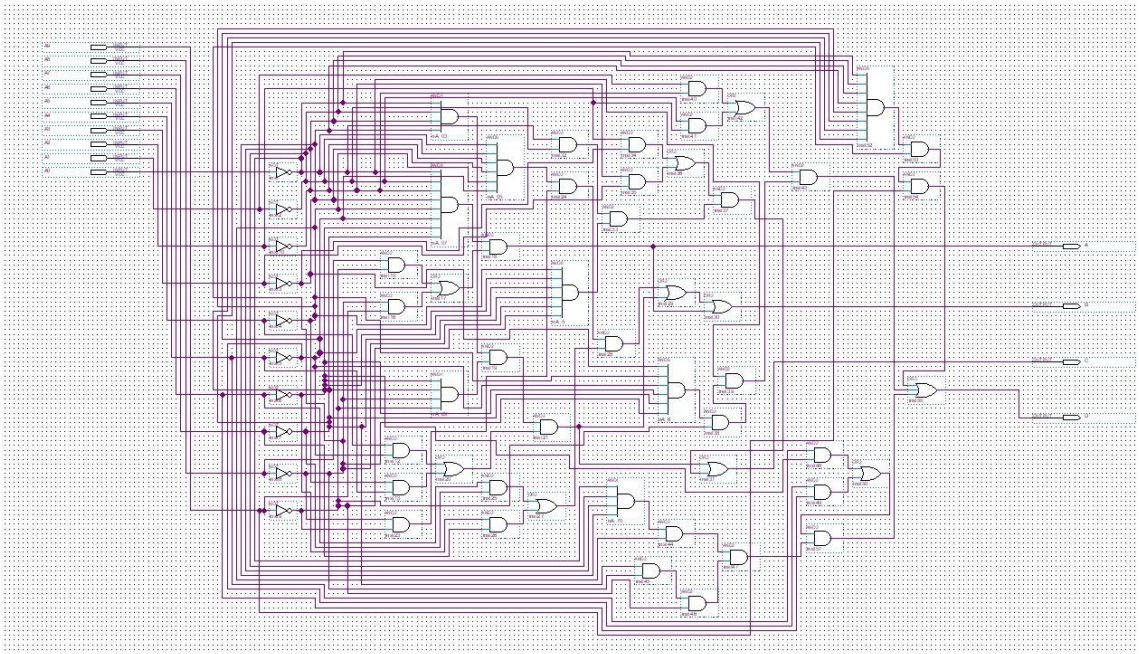

Figura 1. Implementação do codificador no Quartus-II

Este projeto de circuito não se encontra minimizado.

2.3. Obtendo funções booleanas para o decodificador

Tabela 2. Tabela verdade para o circuito do Decodificador

Entradas				Saídas									
A	**B**	**C**	**D**	A_0	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9
0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	0	1	0	1	0	0	0	0	0	0	0	0
0	0	1	1	0	0	1	0	0	0	0	0	0	0
0	0	1	0	0	0	0	1	0	0	0	0	0	0
0	1	1	0	0	0	0	0	1	0	0	0	0	0
0	1	1	1	0	0	0	0	0	1	0	0	0	0
0	1	0	1	0	0	0	0	0	0	1	0	0	0
0	1	0	0	0	0	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	0	0	0	1	0
1	1	0	1	0	0	0	0	0	0	0	0	0	1

No decodificador, teremos o comportamento oposto ao do codificador, isto é, o valor 000000001 (somente A_0 ativo) definido como 0, valor 000000010 (somente A_1 ativo) como 1 e assim por diante até 100000000 (somente A_9 ativo) como 9. Para obter as funções, é necessário averiguar saída a saída. Desta forma, obtemos:

$$F(A_0) = \overline{A} \cdot \overline{B} \cdot \overline{C} \cdot \overline{D}$$

$$F(A_1) = \overline{A} \cdot \overline{B} \cdot \overline{C} \cdot D$$
$$F(A_2) = \overline{A} \cdot \overline{B} \cdot C \cdot D$$
$$F(A_3) = \overline{A} \cdot \overline{B} \cdot C \cdot \overline{D}$$
$$F(A_4) = \overline{A} \cdot B \cdot C \cdot \overline{D}$$
$$F(A_5) = \overline{A} \cdot B \cdot C \cdot D$$
$$F(A_6) = \overline{A} \cdot B \cdot \overline{C} \cdot D$$
$$F(A_7) = \overline{A} \cdot B \cdot \overline{C} \cdot \overline{D}$$
$$F(A_8) = A \cdot B \cdot \overline{C} \cdot \overline{D}$$
$$F(A_9) = A \cdot B \cdot \overline{C} \cdot D$$

2.4. Projetando circuito do decodificador no *Quartus-II*

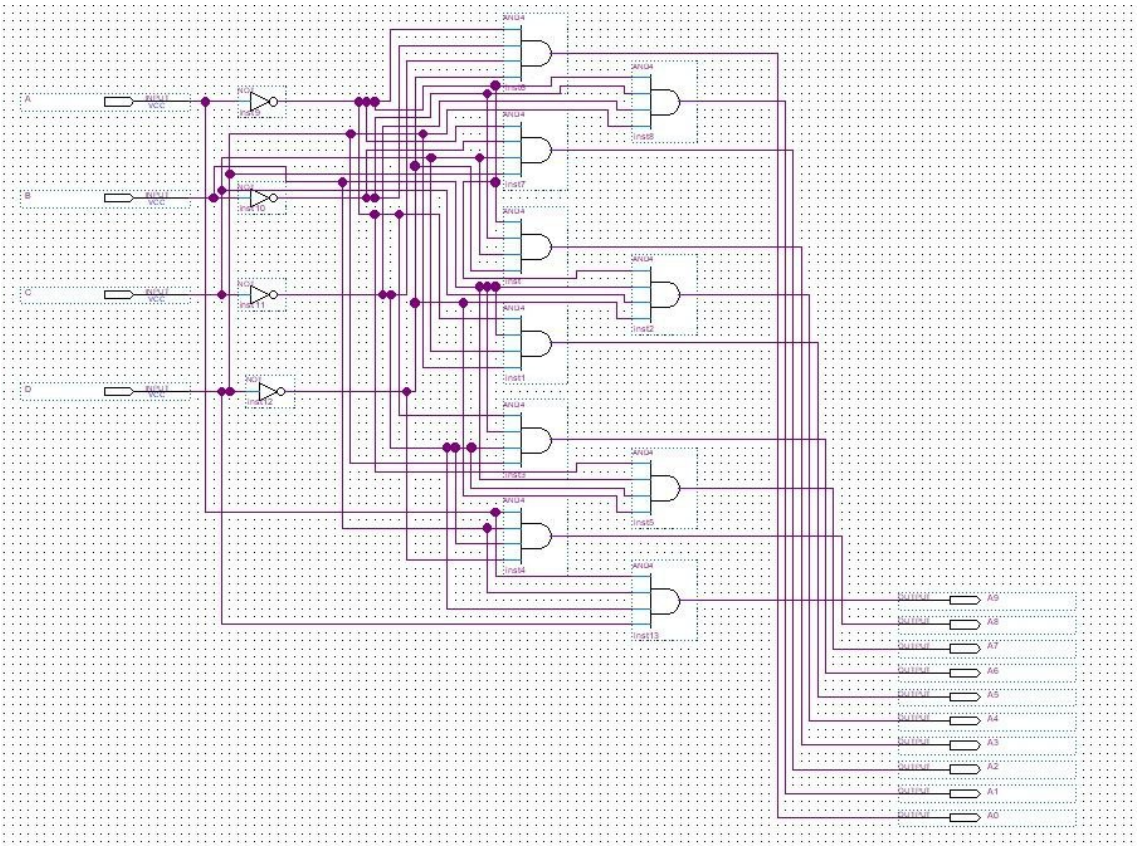

Figura 2. Implementação do decodificador no Quartus-II

Este projeto de circuito não se encontra minimizado.

2.5. Sintetização dos circuitos no kit FPGA DE2

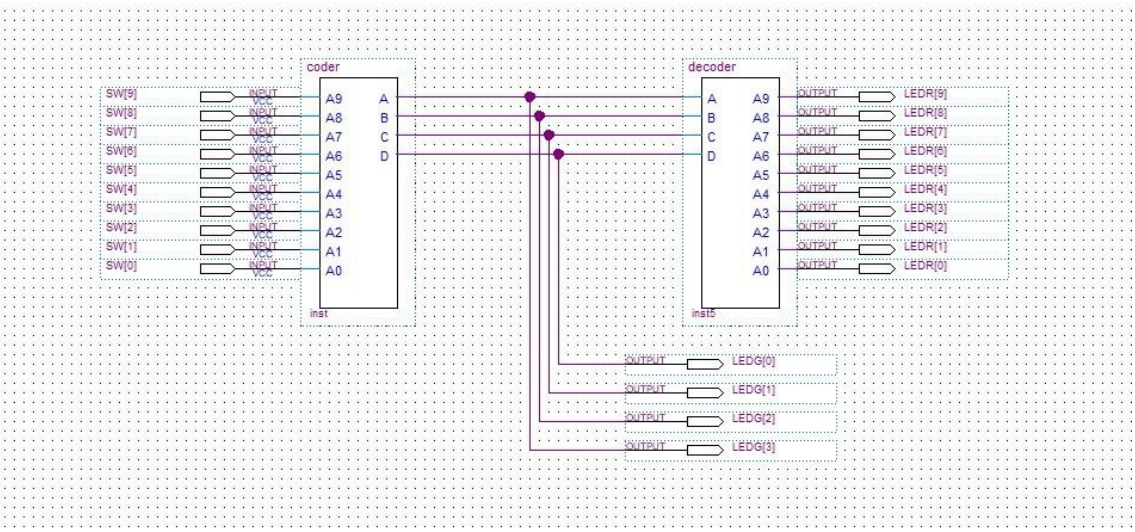

Figura 3. Implementação do codificador e decodificador no Quartus-II

Para visualizar o vídeo gravado, basta seguir o seguinte link do vídeo: 2.5

3. Conclusões

Códigos binários possuem muitas vantagens e desvantagens, e algumas dessas desvantagens podem ser resolvidas com os codificadores e decodificadores. Com o objetivo de implementa-las, utilizamos o software Quartus-II para montar o circuito, e para montar o circuito primeiro fizemos a tabela verdade e criamos a função, assim como no 2.1 e 2.3. Com as funções obtidas fizemos a implementação no Quartus-II como podemos ver nos 2.2 e 2.4. Para finalizar fizemos a sintetização dos dois circuitos 2.5. Com esses passos, podemos afirmar que o circuito foi implementado de forma correta.

Referências

[Altera 2013] Altera (2013). Quartus ii handbook. https://www.intel.com/content/dam/support/us/en/programmable/support-resources/bulk-container/pdfs/literature/hb/qts/archives/quartusii-handbook-archive-130.pdf. [Online; accessed 29-July-2022].

Auto-Avaliação

1. C
2. C
3. A
4. B
5. C
6. Opção E, porta $\overline{A}$ defeituosa, com saída no 0
7. Opção B, porta $\overline{D}$ defeituosa em 1
8. Opção F, porta $\overline{A}$ defeituosa em 1