Experimento 4
Circuitos Combinacionais: Comparador de palavras

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. *A word comparator compares two words of N bits, if it is equal, the output will be 1, otherwise it will be 0. This report will present a simplification of a truth table, implementation of a combinational circuit which compares words of N bits, besides the difficulties and decisions made by the students during the implementation of this experiment.*

Resumo. *Um comparador de palavras compara duas palavras de n bits, caso as mesmas sejam iguais, a saída será 1, e se forem diferentes a saída será 0. Este relatório apresentará a simplificação de uma tabela verdade, implementação de um circuito combinacional que compara palavras de n bits e escolhas, além de dificuldades dos alunos durante a implementação do experimento.*

1. Introdução

Os circuitos serão construídos através do software *Quartus-II*. Um comparador de duas palavras do mesmo comprimento terá a saída 1 quando as palavras forem iguais, caso sejam diferentes a saída será 0. Já um comparador de 3 saídas terá a saída igual a 1 quando $A > B$ ou $A = B$ ou $A < B$.

1.1. Objetivos

Este experimento tem como objetivo a projeção de um comparador de palavras binárias. Através do programa *Quartus-II*, os alunos deverão aprender a utilizar o aplicativo, para assim simplificarem uma função booleana adequada para a implementação do circuito combinacional.

1.2. Materiais

Neste experimento foram utilizados os seguintes materiais e equipamentos:

- Software Quartus-II versão 13.0 SP1

2. Procedimentos e Resultados

No experimento 2.1 projetaremos um comparador de palavras de 3 bits utilizando apenas portas NAND de duas entradas, e no 2.2 faremos um comparador de duas palavras de 2 bits com 3 saídas, utilizando dois comparadores de duas palavras de 1 bit de 3 saídas, tal que $Y_1 = 1$ se $A > B$, $Y_2 = 1$ se $A = B$ e $Y_3 = 1$ se $A < B$.

2.1. Comparador de palavras de 3 bits

O circuito deve comparar duas palavras de 3 bits usando apenas portas NAND de duas entradas. Antes de construir o circuito, utilizaremos a lógica da porta XNOR. Na tabela verdade da XNOR com entradas A_i, B_i e saída Z_i, onde $0 \leq i \leq 2$ temos:

A_i	B_i	Z_i
0	0	1
0	1	0
1	0	0
1	1	1

Podemos observar que se as entradas A e B forem diferentes a saída será 0. Caso forem iguais, a saída será 1. Isso é exatamente o que estamos procurando para o circuito. Assim, para implementar o comparador, precisamos implementar a porta XNOR de duas entradas com portas NAND de duas entradas. a conversão pode ser vista na figura 1.

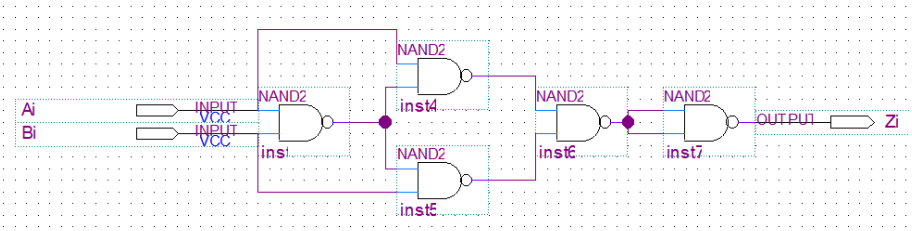

Figura 1. Diagrama para a porta XNOR implementada com NAND de duas entradas.

Agora, apresentaremos algumas simulações no *Quartus-II* com a porta XNOR. Primeiro, montamos o diagrama (figura 1) na interface do software. Para auxiliar a montagem da porta XNOR com NAND, consultamos [Wikipedia 2022]. Após compilar o projeto, configuramos uma simulação com as seguintes opções:

- A_i e B_i como *inputs*;
- Intervalo de 10 nanossegundos entre as transições de valores para A_i e B_i;
- Tempo final de contagem para análise da simulação funcional em forma de onda: 40 nanossegundos.

Ao executar a simulação, obtemos o seguinte resultado em forma de onda:

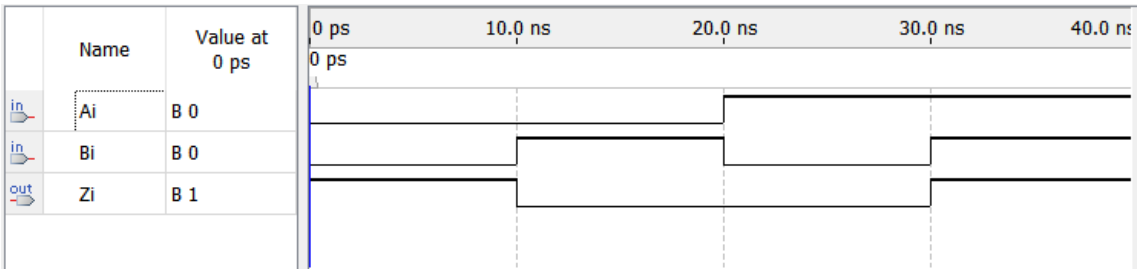

Figura 2. Tabela verdade em forma de onda da porta XNOR.

Pela figura 2, podemos observar que se A_i for igual a B_i, a saída Z_i será 1, caso contrário será 0. Isso corresponde a tabela verdade da porta XNOR. Em seguida, nós

projetamos o circuito para o comparador de duas palavras de 3 bits, porém nós utilizamos o recurso de blocos do *Quartus-II*:

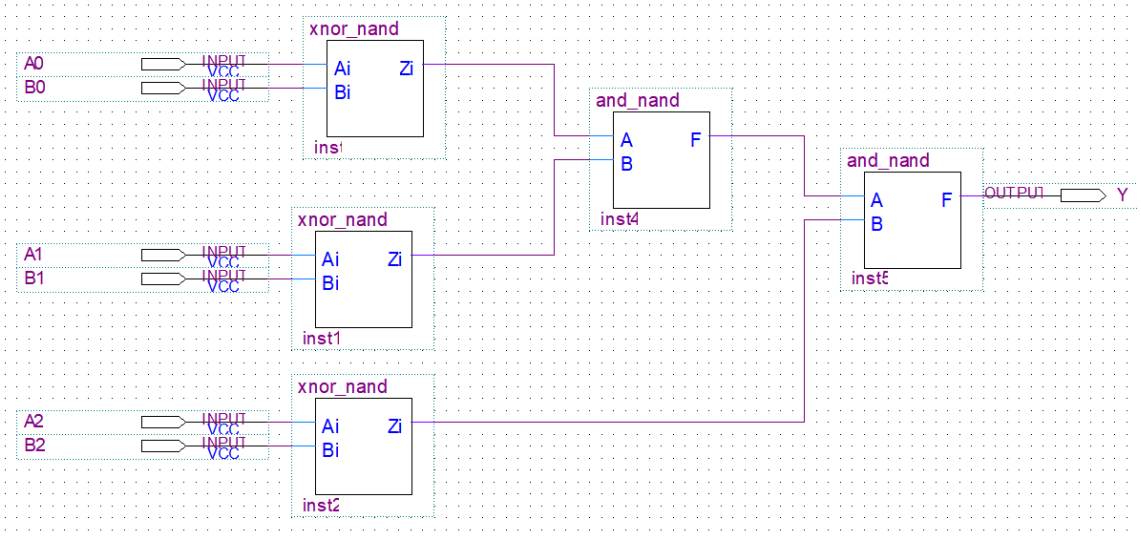

Figura 3. Comparador de palavras de 3 bits com portas NAND de duas entradas.

A simulação é feita com as seguintes configurações:

- As entradas $A_{(0,1,2)}$ e $B_{(0,1,2)}$ são agrupadas em A e B, respectivamente;
- A e B têm valor inicial 0, porém A tem incremento 3 a cada 5 segundos e B tem incremento 1 a cada 3 segundos;
- A saída corresponde à variável Y e o tempo final é de 42 nanossegundos.

É importante destacar que na figura 3, os blocos com os nomes *xnor_nand* e *and_nand* correspondem às portas lógicas XNOR e AND implementadas com portas NAND de 2 entradas. Ao realizar a simulação em forma de onda, obtemos o resultado da figura 4. Observe que quando os valores de A e B coincidem, a saída em Y é 1, enquanto no caso contrário a saída é 0.

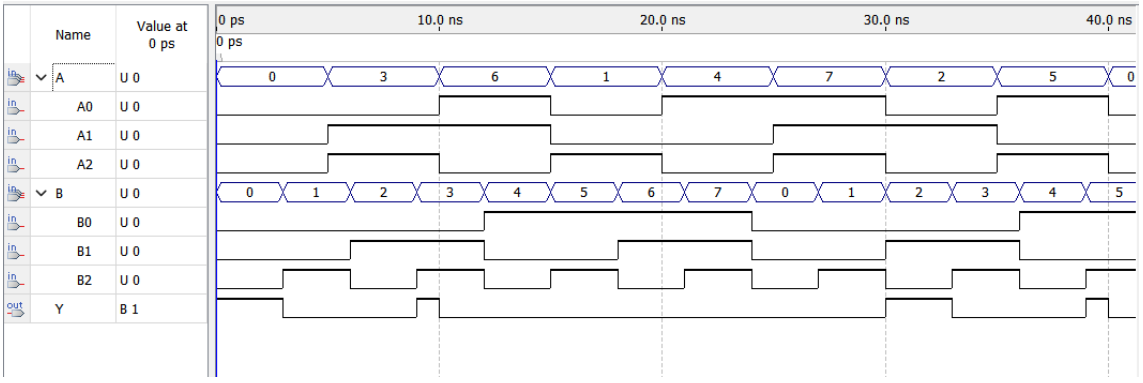

Figura 4. Resultado da simulação do comparador de palavras de 3 bits.

2.2. Comparador de duas palavras de 2 bits com 3 saídas

Essa parte do experimento consiste em projetar um comparador que receba duas palavras de 2 bits - A e B - e retorne as seguintes saídas:

- 1, se $A > B$. Se não, 0;
- 1, se $A = B$. Se não, 0;
- 1, se $A < B$. Se não, 0;

Para começar o projeto, primeiro vamos montar a tabela verdade para cada situação. Seja Y_1 a saída que corresponde a $A > B$, Y_2 corresponde a $A = B$ e Y_3 corresponde a $A < B$ e as variáveis de entrada A e B, temos:

A	B	Y_1
0	0	0
0	1	0
1	0	1
1	1	0

A	B	Y_2
0	0	1
0	1	0
1	0	0
1	1	1

A	B	Y_3
0	0	0
0	1	1
1	0	0
1	1	0

Com as tabelas-verdade em mãos, podemos minimizar as funções Y_1, Y_2 e Y_3 utilizando mapas de Karnaugh. Entretanto, tanto Y_1 como Y_3 são bastante simples, sendo elas:

$$Y_1 = A \cdot \overline{B}$$
$$Y_3 = \overline{A} \cdot B$$

A função para Y_2 vimos na seção 2.1. Assim, $Y_2 = A \quad XNOR \quad B$. Note que as tabelas verdade são para apenas 1 bit de cada palavra. Temos que elaborar um comparador de 1 bit tendo como base as expressões obtidas até então. A figura 5 mostra o diagrama para essa primeira etapa:

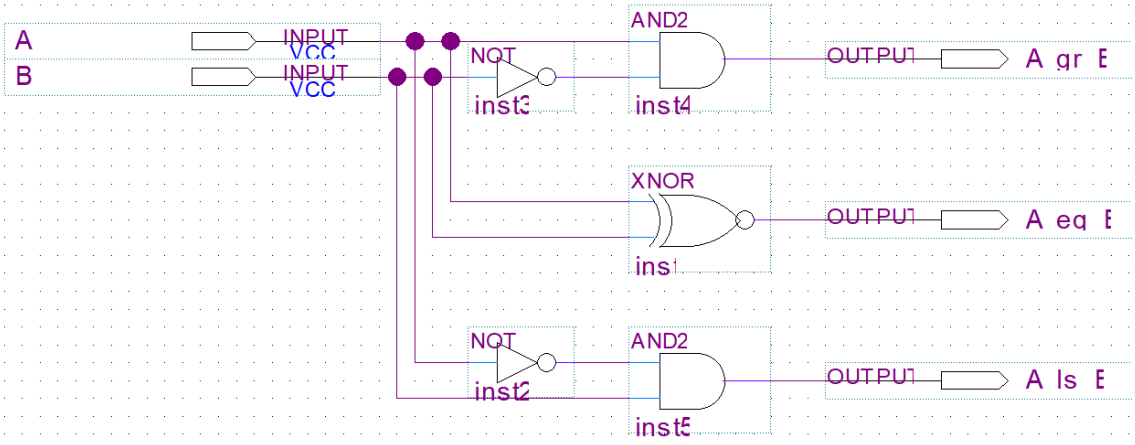

Figura 5. Diagrama para o circuito do comparador de 1 bit.

Ao realizar a simulação no *Quartus-II*, adotamos um intervalo de 20 nanossegundos para as transições. A figura 6 mostra os resultados da simulação. Para conferir se o comparador está produzindo a saída esperada, basta observar as saídas em A_gr_B ($A > B$), A_eq_B ($A = B$) e A_ls_B ($A < B$).

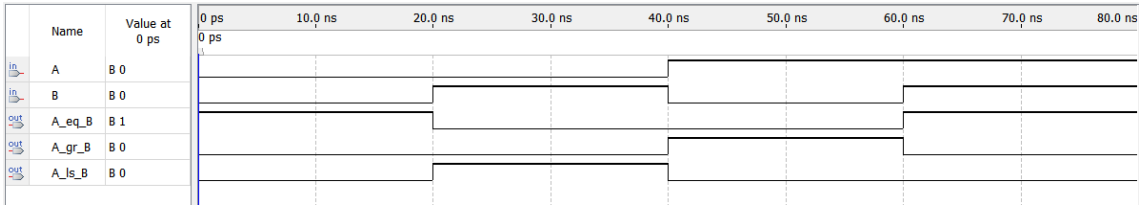

Figura 6. Simulação funcional do comparador de 1 bit.

Com o comparador de 1 bit, podemos então projetar o comparador de palavras de 2 bits. A figura 7 mostra o diagrama do circuito que desejamos simular no *Quartus-II*:

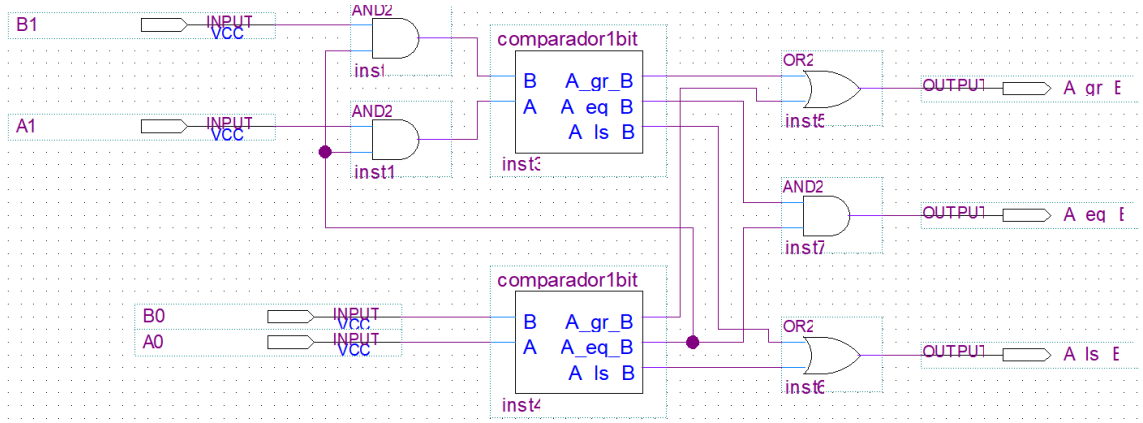

Figura 7. Diagrama para o comparador de palavras de 2 bits.

Para simular o circuito, adotamos um intervalo de 10 nanossegundos para as transições e um tempo final de 160 nanossegundos. Na figura 8 é possível observar que a cada 40 nanossegundos, ocorre a comparação entre o valor de A com os valores de B, cujo resultado é expresso por meio das saídas em A_gr_B ($A > B$), A_eq_B ($A = B$) e A_ls_B ($A < B$).

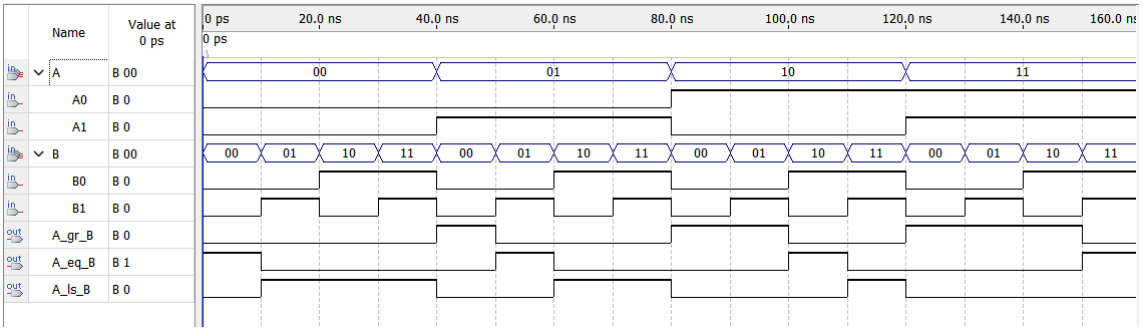

Figura 8. Simulação funcional do comparador de palavras de 2 bits.

3. Conclusões

Concluímos que ao utilizar o software *Quartus-II* para projetar circuitos combinacionais, podemos visualizar circuitos complexos sem a necessidade de implementá-los em

protoboard. Com o projeto do comparador de palavras, obtivemos resultados em forma de onda, o que nos forneceu a ideia de *clock* em um circuito.

Referências

[Wikipedia 2022] Wikipedia (2022). Nand logic — wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/NAND_logic. [Online; accessed 21-July-2022].

Auto-Avaliação

1. c
2. a
3. d
4. a
5. a