Universidade de Brasília
Departamento de Engenharia Elétrica

Aluna: Mariana Soares Oliveira
Matrícula: 231013663
Turma 01
16/11/2024

Relatório Experimento 3

1. Introdução

O intuito do seguinte experimento é desenvolver um multiplexador 8x1 e um decodificador 4x16 , utilizando a linguagem de descrição de hardware VHDL, e simular o seu comportamento por meio de *testbenchs* realizados no *software* ModelSim.

2. Teoria

O multiplexador 8x1 é um circuito combinacional que seleciona uma entre oito entradas possíveis e a direciona para a saída, de acordo com a combinação de três linhas de seleção. Essas linhas determinam qual entrada será conectada à saída, funcionando como um "canal seletor".

O seu funcionamento pode ser ilustrado pela Tabela Verdade 1.

Entradas (S)	Saídas (Y)
000	D0
001	D1
010	D2
011	D3
100	D4
101	D5
110	D6
111	D7

Tabela verdade 1

Já o decodificador 4x16 é um circuito combinacional que converte um código binário de 4 bits em uma das 16 saídas possíveis. Para cada combinação única nas entradas, apenas uma saída será ativada (nível lógico alto), enquanto as demais permanecem desativadas (nível lógico baixo).

O seu funcionamento pode ser ilustrado pela Tabela Verdade 2.

Entradas (S)	Saídas (Y)
0000	0000 0000 0000 0001
0001	0000 0000 0000 0010
0010	0000 0000 0000 0100
0011	0000 0000 0000 1000
0100	0000 0000 0001 0000
0101	0000 0000 0010 0000
0110	0000 0000 0100 0000
0111	0000 0000 1000 0000
1000	0000 0001 0000 0000
1001	0000 0010 0000 0000
1010	0000 0100 0000 0000
1011	0000 1000 0000 0000
1100	0001 0000 0000 0000
1101	0010 0000 0000 0000
1110	0100 0000 0000 0000
1111	1000 0000 0000 0000

Tabela Verdade 2

3. Códigos

Neste experimento utilizamos a linguagem de descrição de hardware VHDL por meio do software Modelsim para desenvolver um multiplexador 8x1 e um decodificador 4x16 conforme as figuras 1 e 2. Posteriormente, foi desenvolvido um código auxiliar chamado *testbench* para cada circuito, descrito nas figuras 3 e 4.

C:/Users/maria/Desktop/231013663_Projeto3/Mux8x1/mux8x1.vhd - Default

```
Ln#
 1  ---- biblioteca
 2
 3  library IEEE;
 4  use IEEE.STD_LOGIC_1164.all;
 5
 6  ---- entidade
 7
 8  entity ent_mux8x1 is port (
 9
10  D: in STD_LOGIC_vector (7 downto 0);
11  S: in STD_LOGIC_vector (2 downto 0);
12  Y: out STD_LOGIC);
13
14  end ent_mux8x1;
15
16  ---- arquitetura
17
18  architecture arch_mux8x1 of ent_mux8x1 is
19
20  begin
21
22  Y <= D(0) when S = "000" else
23       D(1) when S = "001" else
24       D(2) when S = "010" else
25       D(3) when S = "011" else
26       D(4) when S = "100" else
27       D(5) when S = "101" else
28       D(6) when S = "110" else
29       D(7);
30
31  end arch_mux8x1;
32
```

Figura 1. Codificação do multiplexador 8x1

C:/Users/maria/Desktop/231013663_Projeto3/Deco4x16/deco4x16.vhd - Default

```
Ln#
 1  ---- biblioteca
 2  library IEEE;
 3  use IEEE.STD_LOGIC_1164.all;
 4
 5  ---- entidade
 6  entity ent_deco4x16 is
 7      port (
 8          A: in STD_LOGIC_VECTOR (3 downto 0);
 9          Y: out STD_LOGIC_VECTOR (15 downto 0)
10      );
11  end ent_deco4x16;
12
13  ---- arquitetura
14  architecture arch_deco4x16 of ent_deco4x16 is
15  begin
16      with A select
17      Y <= "0000000000000001" when "0000",
18           "0000000000000010" when "0001",
19           "0000000000000100" when "0010",
20           "0000000000001000" when "0011",
21           "0000000000010000" when "0100",
22           "0000000000100000" when "0101",
23           "0000000001000000" when "0110",
24           "0000000010000000" when "0111",
25           "0000000100000000" when "1000",
26           "0000001000000000" when "1001",
27           "0000010000000000" when "1010",
28           "0000100000000000" when "1011",
29           "0001000000000000" when "1100",
30           "0010000000000000" when "1101",
31           "0100000000000000" when "1110",
32           "1000000000000000" when "1111",
33           "0000000000000000" when others; ---- valor padrão para outros casos
34  end arch_deco4x16;
35
```

Figura 2. Codificação do decodificador 4x16

C:/Users/maria/Desktop/231013663_Projeto3/Mux8x1/tb_mux8x1.vhd - Default

```
Ln#
 1  library IEEE;
 2  use IEEE.STD_LOGIC_1164.all;
 3  ---- entidade
 4  entity testbench1 is
 5  end;
 6  ---- arquitetura
 7  architecture tb_mux8x1 of testbench1 is
 8      component ent_mux8x1 is
 9          port (
10              D: in STD_LOGIC_vector (7 downto 0);
11              S: in STD_LOGIC_vector (2 downto 0);
12              Y: out STD_LOGIC
13          );
14      end component;
15      -- sinais para teste
16      signal D : STD_LOGIC_VECTOR (7 downto 0) := "00000000";
17      signal S : STD_LOGIC_VECTOR (2 downto 0) := "000";
18      signal Y : STD_LOGIC;
19
20  begin
21      ---- instância do multiplexador 8x1
22      mux8x1 : ent_mux8x1
23          port map (
24              D => D,
25              S => S,
26              Y => Y
27          );
28      ---- geração de estímulos
29      D(0) <= not D(0) after 1 ns;
30      D(1) <= not D(1) after 2 ns;
31      D(2) <= not D(2) after 4 ns;
32      D(3) <= not D(3) after 8 ns;
33      D(4) <= not D(4) after 16 ns;
34      D(5) <= not D(5) after 32 ns;
35      D(6) <= not D(6) after 64 ns;
36      D(7) <= not D(7) after 128 ns;
37      S(0) <= not S(0) after 256 ns;
38      S(1) <= not S(1) after 512 ns;
39      S(2) <= not S(2) after 1024 ns;
40  end tb_mux8x1;
```

Figura 3. *Testbench* do multiplexador 8x1

C:/Users/maria/Desktop/231013663_Projeto3/Deco4x16/tb_deco4x16.vhd - Default

```
Ln#
 1  ---- biblioteca
 2  library IEEE;
 3  use IEEE.STD_LOGIC_1164.all;
 4  -- entidade
 5  entity testbench2 is
 6  end;
 7  -- arquitetura
 8  architecture tb_deco4x16 of testbench2 is
 9      component ent_deco4x16 is
10          port (
11              A: in STD_LOGIC_VECTOR (3 downto 0);
12              Y: out STD_LOGIC_VECTOR (15 downto 0)
13          );
14      end component;
15      -- sinais para teste
16      signal A : STD_LOGIC_VECTOR (3 downto 0) := "0000";
17      signal Y : STD_LOGIC_VECTOR (15 downto 0);
18  begin
19      -- instância do decodificador 4x16
20      deco4x16 : ent_deco4x16
21          port map (
22              A => A,
23              Y => Y
24          );
25      -- geração de estímulos
26      A(0) <= not A(0) after 3 ns;
27      A(1) <= not A(1) after 6 ns;
28      A(2) <= not A(2) after 12 ns;
29      A(3) <= not A(3) after 24 ns;
30  end tb_deco4x16;
31
32
```

Figura 4. *Testbench* do decodificador 4x16

4. Compilação

Os códigos gerados anteriormente foram submetidos a uma compilação com o intuito de garantir seu funcionamento, como mostrado nas figuras 5 e 6 ambos os códigos não apresentam erros de sintaxe.

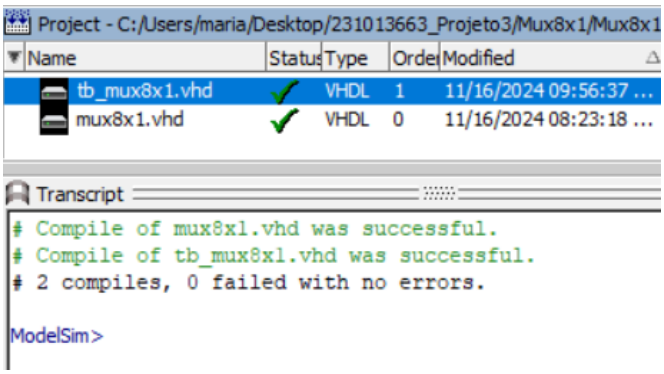

Figura 5. Compilação do multiplexador 8x1

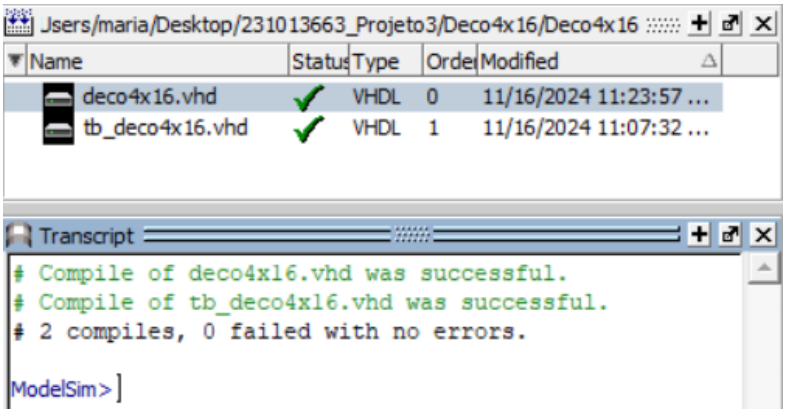

Figura 6. Compilação do decodificador 4x16

5. Simulação

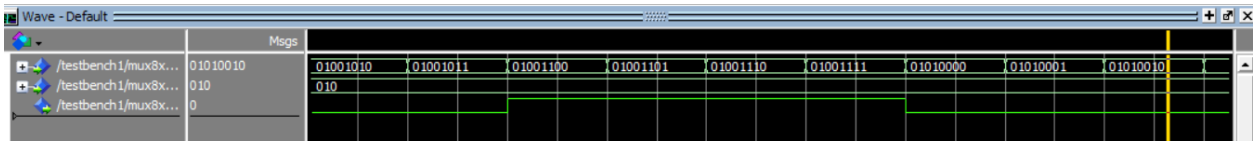

Figura 7. Simulação de onda do banco de testes do multiplexador 8x1

Figura 8. Simulação de onda do banco de testes do decodificador 4x16

6. Análise

Neste experimento, foram analisadas duas estruturas: o multiplexador 8x1 e o decodificador 4x16 . Com base nas Tabelas Verdade 1 e 2 e na simulação de de onda apresentada nas Figuras 7 e 8, é possível compreender o funcionamento esperado de ambos os circuitos. Dessa forma, pode-se afirmar que os códigos desenvolvidos correspondem ao comportamento esperado, uma vez que os valores obtidos coincidem com aqueles descritos nas tabelas verdade

7. Conclusão

No experimento, foi possível descrever o comportamento das estruturas propostas e entender suas características. As simulações geraram os dados esperados, que foram comparados com as tabelas verdade dos circuitos (Tabelas 1 e 2). Não houveram erros ou divergências observados durante a realização do experimento.