

Relatório 3

Nome: Gabriel Cruz Vaz Santos

Matrícula: 200049038

Turma: C

• Questão 1

Introdução

Foi solicitado a implementação de uma entidade com 2 vetores de entrada (S com 3 bits e D com 8 bits) e uma saída Y (um bit) que implementem um multiplexador 8 para 1.

Teoria

As entradas do vetor S são as responsáveis por definir a saída (Y), que irá nos levar a algum dos vetores de D como resposta.

Código

Os códigos foram programados no ambiente de desenvolvimento integrado Visual Studio Code, na linguagem VHDL e compilados pelo software do ModelSim. A figura 1 é referente a implementação da entidade e arquitetura do multiplexador, enquanto as demais imagens são do teste.

```
Relatorio3 > multiplexador_8x1.vhd
1  library ieee;
2  use ieee.std_logic_1164.all;
3
4  entity multiplexador_8x1 is
5      port(
6          S: in std_logic_vector ( 2 downto 0 );
7          D: in std_logic_vector ( 7 downto 0 );
8          Y: out std_logic
9      );
10 end multiplexador_8x1;
11
12 architecture multiplexador_8x1_arch of multiplexador_8x1 is
13     begin
14         Y <= d(0) when s = "000" else
15             d(1) when s = "001" else
16             d(2) when s = "010" else
17             d(3) when s = "011" else
18             d(4) when s = "100" else
19             d(5) when s = "101" else
20             d(6) when s = "110" else
21             d(7) when s = "111";
22     end multiplexador_8x1_arch;
```

Relatorio3 > multiplexador_8x1_tb.vhd

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3
4  entity multiplexador_8x1_tb is end;
5
6  architecture multiplexador_8x1_arch of multiplexador_8x1_tb is
7      component multiplexador_8x1 is
8          port(
9              S: in std_logic_vector ( 2 downto 0 );
10             D: in std_logic_vector ( 7 downto 0 );
11             Y: out std_logic
12          );
13      end component;
14
15      signal S_1: std_logic_vector (2 downto 0);
16      signal D_1: std_logic_vector (7 downto 0);
17
18      constant time_0: time := 2 ns;
19      constant time_1: time := 4 ns;
20      constant time_2: time := 8 ns;
21      constant time_3: time := 16 ns;
22      constant time_4: time := 32 ns;
23      constant time_5: time := 64 ns;
24      constant time_6: time := 128 ns;
25      constant time_7: time := 256 ns;
26      constant time_8: time := 512 ns;
27      constant time_9: time := 1024 ns;
28      constant time_10: time := 2048 ns;
29
30      begin
31          multiplexador_8x1_1: multiplexador_8x1 port map (D => D_1, S => S_1, Y => open );
32
33          clock0: process
34              begin
35                  D_1(0) <= '0', '1' after time_0/2, '0' after time_0;
36                  wait for time_0;
37              end process clock0;
```

```
38
39 clock1: process
40 begin
41     D_1(1) <= '0', '1' after time_1/2, '0' after time_1;
42     wait for time_1;
43 end process clock1;
44
45 clock2: process
46 begin
47     D_1(2) <= '0', '1' after time_2/2, '0' after time_2;
48     wait for time_2;
49 end process clock2;
50
51 clock3: process
52 begin
53     D_1(3) <= '0', '1' after time_3/2, '0' after time_3;
54     wait for time_3;
55 end process clock3;
56
57 clock4: process
58 begin
59     D_1(4) <= '0', '1' after time_4/2, '0' after time_4;
60     wait for time_4;
61 end process clock4;
62
63 clock5: process
64 begin
65     D_1(5) <= '0', '1' after time_5/2, '0' after time_5;
66     wait for time_5;
67 end process clock5;
68
69 clock6: process
70 begin
71     D_1(6) <= '0', '1' after time_6/2, '0' after time_6;
72     wait for time_6;
73 end process clock6;
```

```

73     end process clock6;
74
75     clock7: process
76     begin
77         D_1(7) <= '0', '1' after time_7/2, '0' after time_7;
78         wait for time_7;
79     end process clock7;
80
81     clock8: process
82     begin
83         S_1(0) <= '0', '1' after time_8/2, '0' after time_8;
84         wait for time_8;
85     end process clock8;
86
87     clock9: process
88     begin
89         S_1(1) <= '0', '1' after time_9/2, '0' after time_9;
90         wait for time_9;
91     end process clock9;
92
93     clock10: process
94     begin
95         S_1(2) <= '0', '1' after time_10/2, '0' after time_10;
96         wait for time_10;
97     end process clock10;
98
99 end multiplexador_8x1_arch;

```

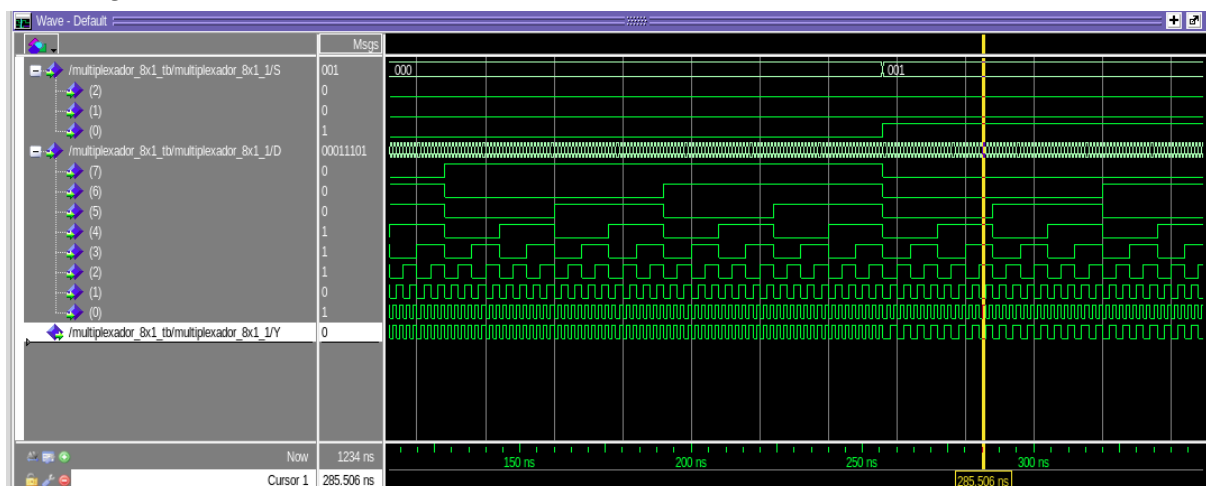
Compilação

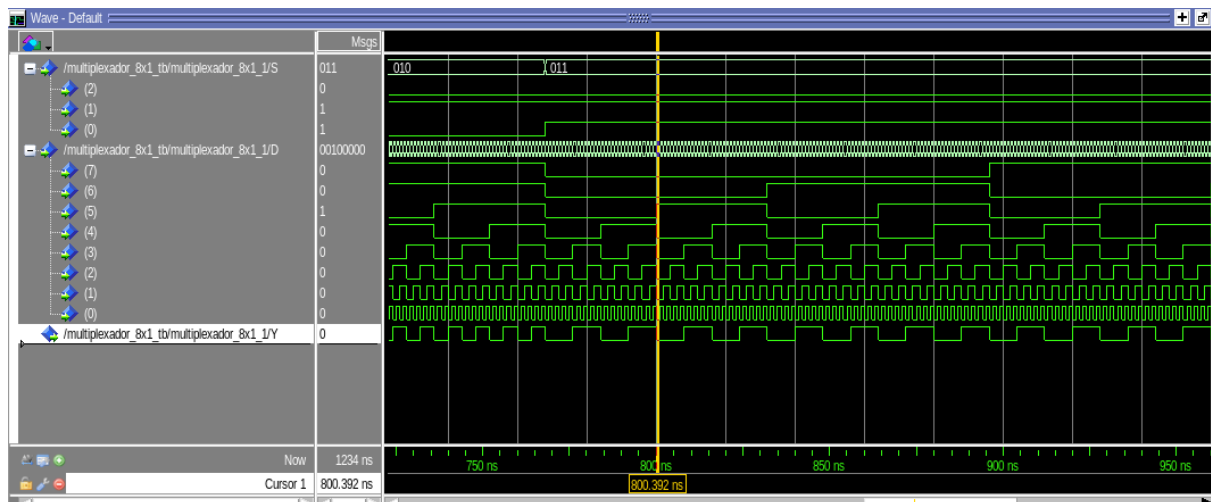
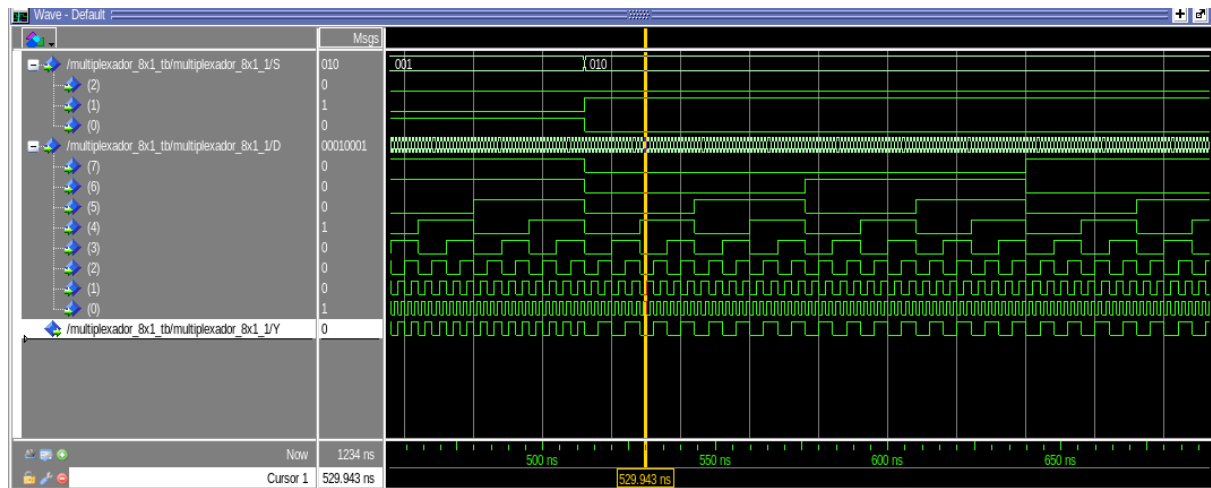
Os códigos acima foram compilados para garantir seu funcionamento, através do compilador do Modelsim. Não apresentam erros de sintaxe;

 multiplexador_8x1_t... ✓ VHDL 1 02/25/2022 11:48:52 ...
 multiplexador_8x1.v... ✓ VHDL 0 02/25/2022 11:20:04 ...

Compile of multiplexador_8x1.vhd was successful.
Compile of multiplexador_8x1_tb.vhd was successful.

Simulação





Análise

Os valores dos vetores de entrada D e S mudam conforme os clocks definidos. Nesse caso, os vetores S são aqueles que definem qual componente do vetor D será a resposta da saída Y.

Notamos que durante até o instante 256 ns, a saída é o vetor d(0), cujo valor varia entre 0 e 1 ao decorrer do tempo (o clock usado para esse vetor foi de 2 ns).

O mesmo acontece conforme os outros vetores vão variando de sinal.

Conclusão

Nesse experimento conseguimos com êxito descrever o multiplexador 8X1. As simulações se comportaram da maneira esperada e não foram encontrados erros de sintaxe no código.

• Questão 2

Introdução

Foi solicitado a implementação de uma entidade com uma entrada A de 4 bits e uma saída Y com 16 bits que implementem um decodificador 4x16.

Teoria

As entradas do vetor A são as responsáveis por definir qual dos componentes do vetor Y será diferente de zero.

Código

Os códigos foram programados no ambiente de desenvolvimento integrado Visual Studio Code, na linguagem VHDL e compilados pelo software do ModelSim. As figuras 1, 2 e 3 é referente a implementação da entidade e arquitetura do multiplexador, enquanto as demais imagens são do teste.

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3
4  entity decodificador_4x16 is
5      port (
6          A: in std_logic_vector (3 downto 0);
7          Y: out std_logic_vector (15 downto 0)
8      );
9  end decodificador_4x16;
10
11  architecture decodificador_4x16_arch of decodificador_4x16 is
12      begin
13          with A select
14              Y(0) <= '1' when "0000",
15                  '0' when others;
16
17          with A select
18              Y(1) <= '1' when "0001",
19                  '0' when others;
20
21          with A select
22              Y(2) <= '1' when "0010",
23                  '0' when others;
24
25          with A select
26              Y(3) <= '1' when "0011",
27                  '0' when others;
28
29          with A select
30              Y(4) <= '1' when "0100",
31                  '0' when others;
32
33          with A select
34              Y(5) <= '1' when "0101",
35                  '0' when others;
36
37          with A select
38              Y(6) <= '1' when "0110".
```

```
38         Y(6) <= '1' when "0110",
39         |         '0' when others;
40
41     with A select
42         Y(7) <= '1' when "0111",
43         |         '0' when others;
44
45     with A select
46         Y(8) <= '1' when "1000",
47         |         '0' when others;
48
49     with A select
50         Y(9) <= '1' when "1001",
51         |         '0' when others;
52
53     with A select
54         Y(10) <= '1' when "1010",
55         |         '0' when others;
56
57     with A select
58         Y(11) <= '1' when "1011",
59         |         '0' when others;
60
61     with A select
62         Y(12) <= '1' when "1100",
63         |         '0' when others;
64
65     with A select
66         Y(13) <= '1' when "1101",
67         |         '0' when others;
68
69     with A select
70         Y(14) <= '1' when "1110",
71         |         '0' when others;
72
73     with A select
74         Y(15) <= '1' when "1111",
```

```

74         Y(15) <= '1' when "1111",
75         '0' when others;
76
77     end decodificador_4x16_arch;

```

```

1  library ieee;
2  use ieee.std_logic_1164.all;
3
4  entity decodificador_4x16_tb is end;
5
6  architecture decodificador_4x16_arch of decodificador_4x16_tb is
7      component decodificador_4x16 is
8          port(
9              A: in std_logic_vector ( 3 downto 0 );
10             Y: out std_logic_vector ( 15 downto 0 )
11          );
12      end component;
13
14      signal A_1: std_logic_vector (3 downto 0);
15
16      constant time_0: time := 2 ns;
17      constant time_1: time := 4 ns;
18      constant time_2: time := 8 ns;
19      constant time_3: time := 16 ns;
20
21      begin
22          decodificador_4x16_1: decodificador_4x16 port map (A => A_1, Y => open);
23
24          clock0: process
25          begin
26              A_1(0) <= '0', '1' after time_0/2, '0' after time_0;
27              wait for time_0;
28          end process clock0;
29
30          clock1: process
31          begin
32              A_1(1) <= '0', '1' after time_1/2, '0' after time_1;
33              wait for time_1;
34          end process clock1;

```



```

35
36          clock2: process
37          begin
38              A_1(2) <= '0', '1' after time_2/2, '0' after time_2;
39              wait for time_2;
40          end process clock2;
41
42          clock3: process
43          begin
44              A_1(3) <= '0', '1' after time_3/2, '0' after time_3;
45              wait for time_3;
46          end process clock3;
47
48      end decodificador_4x16_arch;
49

```

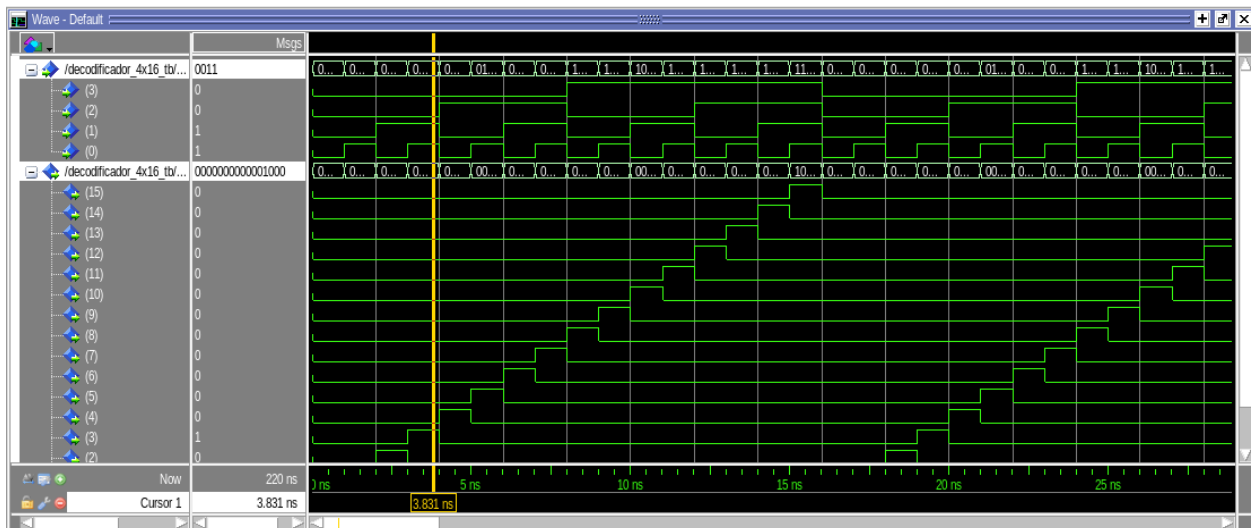
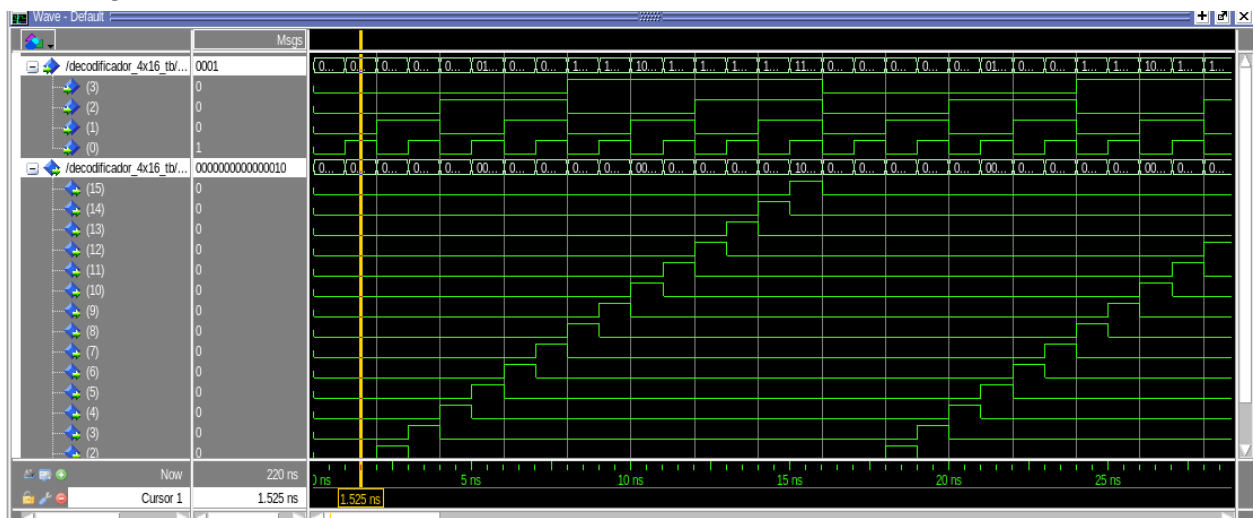

Compilação

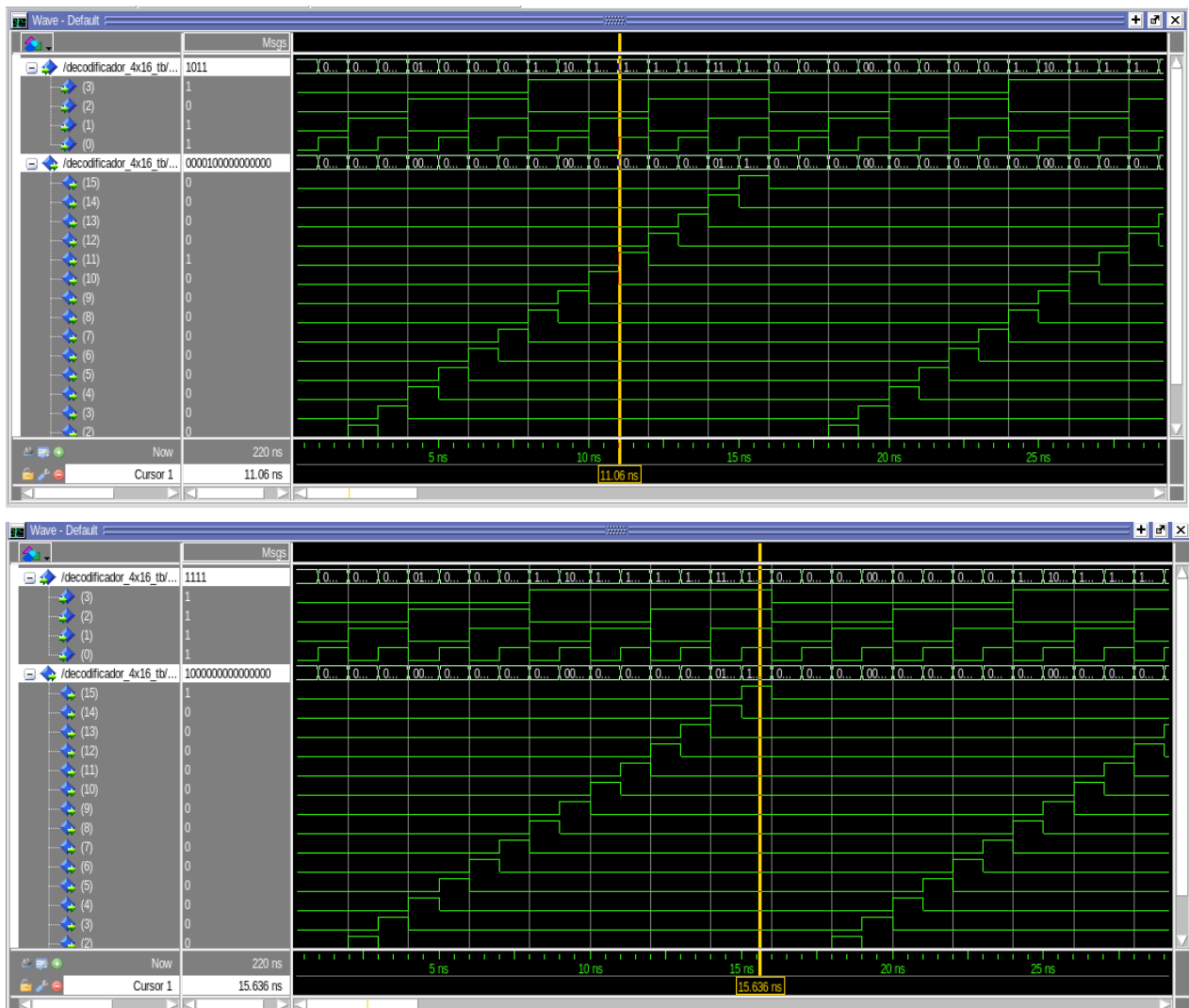
Os códigos acima foram compilados para garantir seu funcionamento, através do compilador do Modelsim. Não apresentam erros de sintaxe;

	decodificador_4x16...	✓	VHDL	3	02/25/2022 03:24:12 ...
	decodificador_4x16....	✓	VHDL	2	02/25/2022 12:29:11 ...

```
# Compile of decodificador_4x16.vhd was successful.  
# Compile of decodificador_4X16_tb.vhd was successful.
```

Simulação





Análise

Os valores da entrada “A” mudam conforme o clock definido, o que reflete em qual saída do vetor Y será diferente de zero.

Podemos observar na imagem 1 da simulação, o valor do vetor A era “0001”, o que fazia com que a saída Y fosse “0000 0000 0000 0010”.

Enquanto que no tempo 3.8 ns, no qual o vetor A correspondia a “0011”, a saída Y também foi modificada para “0000 0000 0000 1000”

A cada 16 ns o ciclo se repete completamente, com os valores da entrada indo de “0000” até “1111”.

Conclusão

Nesse experimento conseguimos com êxito descrever o decodificador 4x16. As simulações se comportaram da maneira esperada e não foram encontrados erros de sintaxe no código.