Relatório 4

Nome: Gabriel Cruz Vaz Santos
Matrícula: 200049038
Turma: C

- ## Questão 1

Introdução

Foi solicitado a implementação de uma função booleana com 3 bits de entrada (A,B e C) e 2 bits de saída (X e Y), utilizando dois multiplicadores 4x1 e uma porta inversora.

Teoria

Um multiplexador é uma entidade que com determinada quantidade de entradas, tem um ou mais seletores e possui somente uma saída. Nesse caso, para cada uma das saídas da função booleana será utilizado um multiplexador 4x1, com a porta inversora funcionando como entradas para cada uma dos multiplexadores.

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 e 2 é referente a implementação da entidade e arquitetura da função booleana e do multiplexador, enquanto as demais imagens são do teste da função booleana.

Relatorio4 > multiplexador.vhd

```vhdl
library ieee;
use ieee.std_logic_1164.all;

entity multiplexador is
    port(
        D: in std_logic_vector (3 downto 0);
        S: in std_logic_vector (1 downto 0);
        Y: out std_logic
    );
end multiplexador;

architecture multiplexador_arch of multiplexador is
    begin
        y <= d(0) when s = "00" else
             d(1) when s = "01" else
             d(2) when s = "10" else
             d(3);
end multiplexador_arch;
```

```vhdl
library ieee;
use ieee.std_logic_1164.all;

entity funcao_booleana is
    port(
        A: in std_logic;
        B: in std_logic;
        C: in std_logic;
        X: out std_logic;
        Y: out std_logic
    );
end funcao_booleana;

architecture funcao_booleana_arch of funcao_booleana is
    component multiplexador is
        port(
            D: in std_logic_vector (3 downto 0);
            S: in std_logic_vector (1 downto 0);
            Y: out std_logic
        );
    end component;

    signal notC: std_logic;

    begin
        notC <= not C;

        saida_1: multiplexador port map ( S(0) => B, S(1) => A, D(0) => '0', D(1) => C, D(2) => notC, D(3) => '1', y => X );
        saida_2: multiplexador port map ( S(0) => B, S(1) => A, D(0) => '1', D(1) => notC, D(2) => '0', D(3) => C, y => Y );

end funcao_booleana_arch;
```

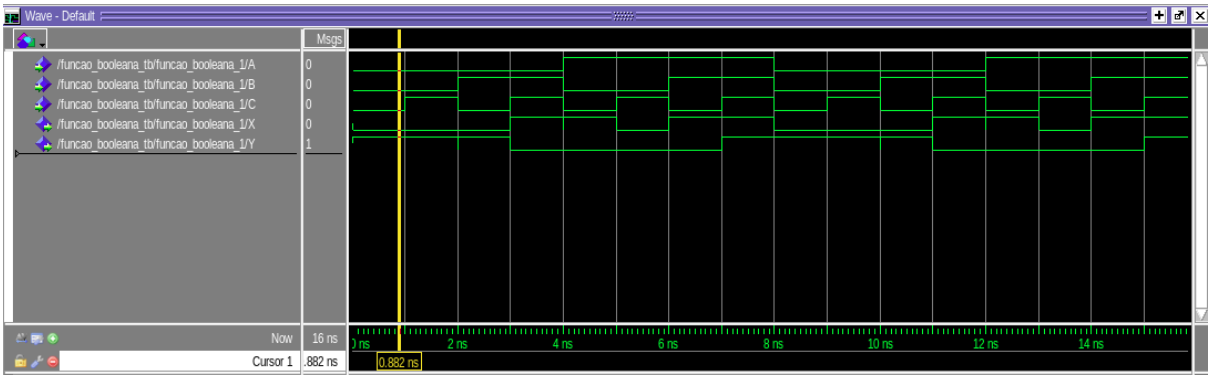
```vhdl
library ieee;
use ieee.std_logic_1164.all;

entity funcao_booleana_tb is end;

architecture funcao_booleana_arch of funcao_booleana_tb is
    component funcao_booleana is
        port (
            A: in std_logic;
            B: in std_logic;
            C: in std_logic;
            X: out std_logic;
            Y: out std_logic
        );
    end component;

    --quanto mais significativo maior o tempo
    signal entrance: std_logic_vector (2 downto 0) := "000";

    constant time_0: time := 2 ns;
    constant time_1: time := 4 ns;
    constant time_2: time := 8 ns;

    begin
        funcao_booleana_1: funcao_booleana port map ( A => entrance(2), B => entrance(1), C => entrance(0), X => open, Y => open );

        entrance(2) <= not entrance(2) after time_2/2;
        entrance(1) <= not entrance(1) after time_1/2;
        entrance(0) <= not entrance(0) after time_0/2;

end funcao_booleana_arch;
```

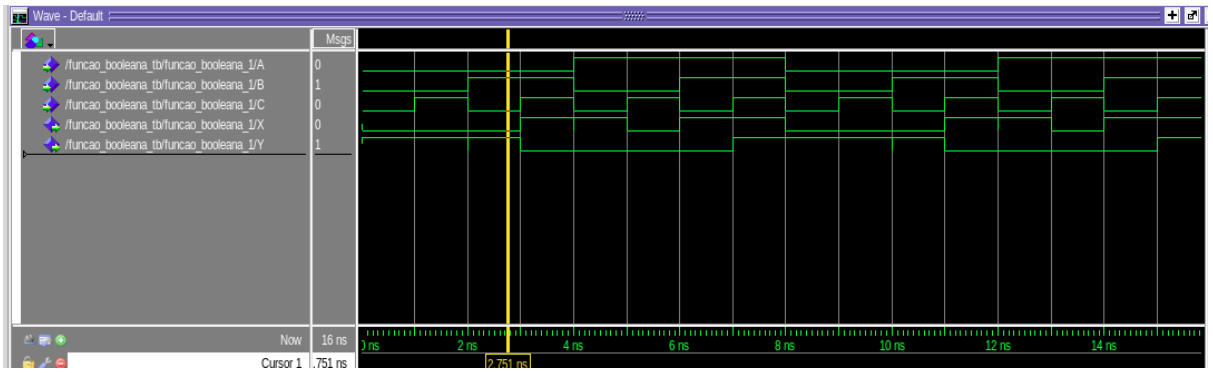
Compilação

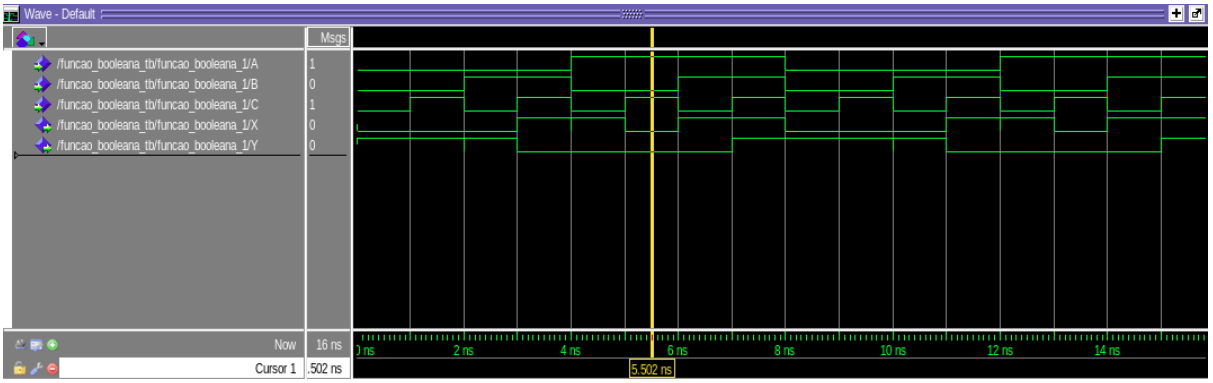
Os códigos acima foram compilados para garantir seu funcionamento, através do compilador do Modelsim. Não apresentam erros de sintaxe;

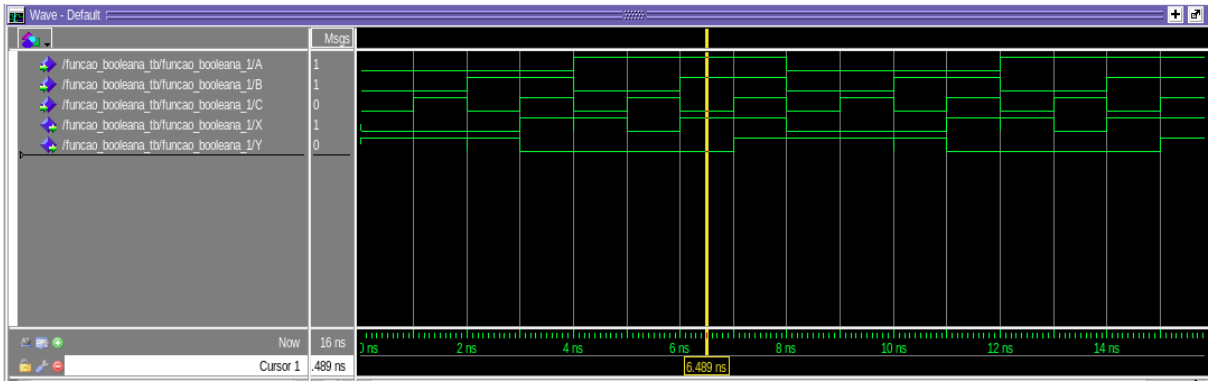
```
# Compile of funcao_booleana.vhd was successful.
# Compile of funcao_booleana_tb.vhd was successful.
# Compile of multiplexador.vhd was successful.
# 3 compiles, 0 failed with no errors.
```

Simulação









Análise

As entradas A e B são escolhidas como seletores, pois tanto na expressão da saída X e da saída Y, eles aparecem em todos os mintermos.

Podemos observar que quando A e B valem zero, a saída X sempre será zero, independente do valor de C. Enquanto isso, a saída Y sempre será um. Ao

analisarmos um pouco o código, podemos ver que quando o vetor S do multiplexador = “00”, a saída do multiplexador será D(0).

Quando a entrada A = 0 e a entrada B = 1, a saída X será o valor da variável C e a saída Y será NOT C. Ao analisarmos um pouco o código, podemos ver que quando o vetor S do multiplexador = “01”, a saída do multiplexador será D(1).

Quando a entrada A = 1 e a entrada B = 0, a saída X será NOT C e a saída Y será “0”.Ao analisarmos um pouco o código, podemos ver que quando o vetor S do multiplexador = “10”, a saída do multiplexador será D(2).

Quando a entrada A = 1 e a entrada B = 1, a saída X será “1” e a saída Y será C.Ao analisarmos um pouco o código, podemos ver que quando o vetor S do multiplexador = “11”, a saída do multiplexador será D(3).

Conclusão

Nesse experimento conseguimos com êxito descrever a função booleana pedida usando dois multiplexadores. 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 função booleana com 7 bits de entrada (A, B, C, D, E, F G) e uma saída (S), utilizando somente um decodificador 4x16, um multiplexador 8x1 e três portas OR.

Teoria

Usaremos E, F e G como nossas variáveis de seleção para a expressão fornecida no enunciado e com base nela e em seus mintermos, montamos a tabela verdade. As variáveis A, B, C e D foram as entradas do nosso decodificador 4x16. Com isso em mente, cada um dos possíveis valores do vetor de entrada do decodificador (0000, 0001, 0010, …, 1111) seria relacionado a um elemento de sua saída. Após isso, teremos o multiplexador, que será responsável por enviar a saída do sistema.

Código

Os códigos foram programados no ambiente de desenvolvimento integrado Visual Studio Code, na linguagem VHDL e compilados pelo software do ModelSim. As primeiras são referentes a implementação da entidade e arquitetura da função booleana2 e do multiplexador 8x1 e do decodificador 4x16, enquanto as demais imagens são do teste da função booleana2.

Relatori04 > decodificador_4x16.vhd

```
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",
```

Relatorio4 > decodificador_4x16.vhd

```
35                          '0' when others;
36
37          with A select
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
```

```
72
73          with A select
74              Y(15) <= '1' when "1111",
75                       '0' when others;
76
77      end decodificador_4x16_arch;
```

Relatorio4 > multiplexador_8x1.vhd

```
library ieee;
use ieee.std_logic_1164.all;

entity multiplexador_8x1 is
    port(
        S: in std_logic_vector ( 2 downto 0 );
        D: in std_logic_vector ( 7 downto 0 );
        Y: out std_logic
    );
end multiplexador_8x1;

architecture multiplexador_8x1_arch of multiplexador_8x1 is
    begin
        Y <= d(0) when s = "000" else
             d(1) when s = "001" else
             d(2) when s = "010" else
             d(3) when s = "011" else
             d(4) when s = "100" else
             d(5) when s = "101" else
             d(6) when s = "110" else
             d(7) when s = "111";
end multiplexador_8x1_arch;
```

Relatorio4 > funcao_booleana2.vhd

```
library ieee;
use ieee.std_logic_1164.all;

entity funcao_booleana2 is
    --Dica usar EFG como seletores para o multiplexador 8x1
    port(
        A: in std_logic;
        B: in std_logic;
        C: in std_logic;
        D: in std_logic;
        E: in std_logic;
        F: in std_logic;
        G: in std_logic;
        S: out std_logic
    );

end funcao_booleana2;

architecture funcao_booleana2_arch of funcao_booleana2 is
    component decodificador_4x16 is
        port (
            A: in std_logic_vector (3 downto 0); -- A, B, C, D
            Y: out std_logic_vector (15 downto 0) -- 0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111
        );
    end component;

    component multiplexador_8x1 is
        port(
            S: in std_logic_vector ( 2 downto 0 ); --S(0), S(1), S(2)
            D: in std_logic_vector ( 7 downto 0 ); -- 000 001 010 011 100 101 110 111
            Y: out std_logic
        );
    end component;

    signal out_decodificador: std_logic_vector (15 downto 0);
    signal in_multiplexador: std_logic_vector (7 downto 0);
```

```
    begin

        decodificador: decodificador_4x16 port map ( A(3) => A, A(2) => B, A(1) => C, A(0) => D, Y => out_decodificador );

        in_multiplexador(7) <= '1';
        in_multiplexador(6) <= out_decodificador(10) or out_decodificador(11);
        in_multiplexador(5) <= '0';
        in_multiplexador(4) <= out_decodificador(15) or out_decodificador(9);
        in_multiplexador(3) <= '1';
        in_multiplexador(2) <= out_decodificador(7);
        in_multiplexador(1) <= out_decodificador(15) or out_decodificador(0);
        in_multiplexador(0) <= '0';

        multiplexador: multiplexador_8x1 port map ( S(2) => E, S(1) => F, S(0) => G,D => in_multiplexador, y => S );


end funcao_booleana2_arch;
```

```vhdl
library ieee;
use ieee.std_logic_1164.all;

entity funcao_booleana2_tb is end;

architecture funcao_booleana2_arch of funcao_booleana2_tb is
    component funcao_booleana2 is
        port(
            A: in std_logic;
            B: in std_logic;
            C: in std_logic;
            D: in std_logic;
            E: in std_logic;
            F: in std_logic;
            G: in std_logic;
            S: out std_logic
        );
    end component;

    signal entrance: std_logic_vector (6 downto 0) := "0000000";

    constant time_0: time := 2 ns;
    constant time_1: time := 4 ns;
    constant time_2: time := 8 ns;
    constant time_3: time := 16 ns;
    constant time_4: time := 32 ns;
    constant time_5: time := 64 ns;
    constant time_6: time := 128 ns;

    begin
        funcao_booleana2_1: funcao_booleana2 port map (A => entrance(6), B => entrance(5), C => entrance(4), D => entrance(3), E=> entrance(2), F => entrance(1

        entrance(6) <= not entrance(6) after time_6/2;
        entrance(5) <= not entrance(5) after time_5/2;
        entrance(4) <= not entrance(4) after time_4/2;
        entrance(3) <= not entrance(3) after time_3/2;
        entrance(2) <= not entrance(2) after time_2/2;
        entrance(1) <= not entrance(1) after time_1/2;
```

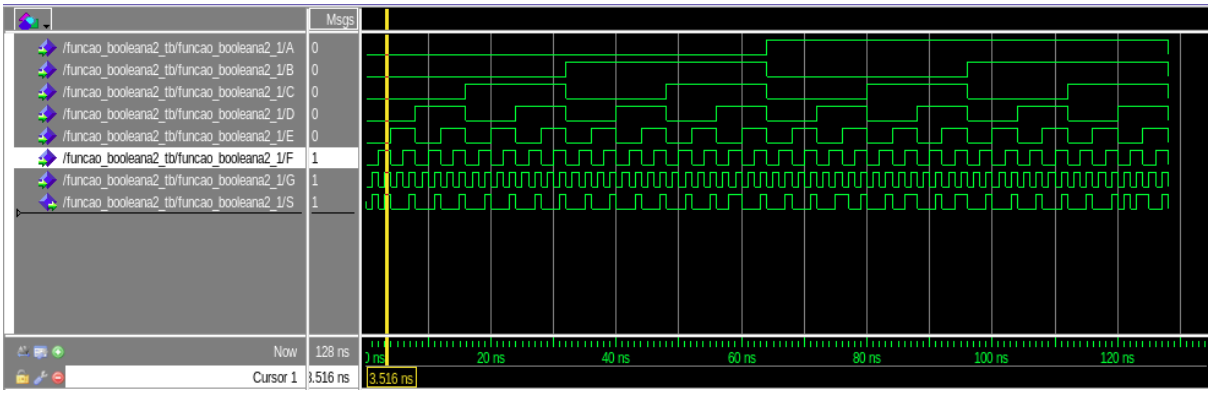
```vhdl
        entrance(1) <= not entrance(1) after time_1/2;
        entrance(0) <= not entrance(0) after time_0/2;

end funcao_booleana2_arch;
```

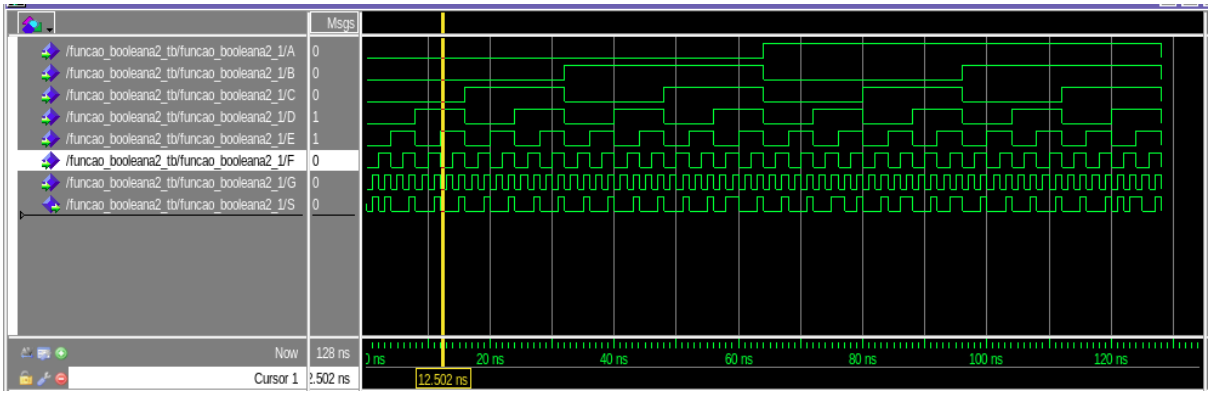
Compilação

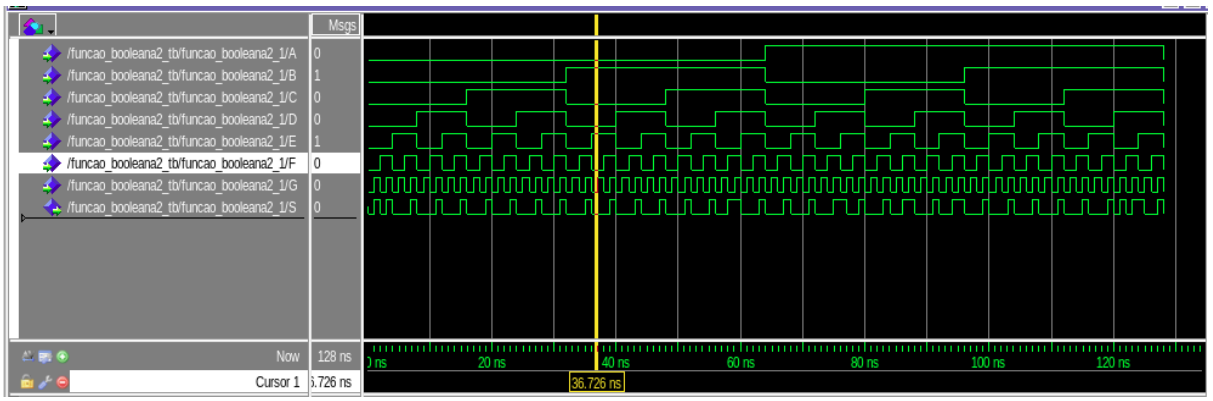
Os códigos acima foram compilados para garantir seu funcionamento, através do compilador do Modelsim. Não apresentam erros de sintaxe;

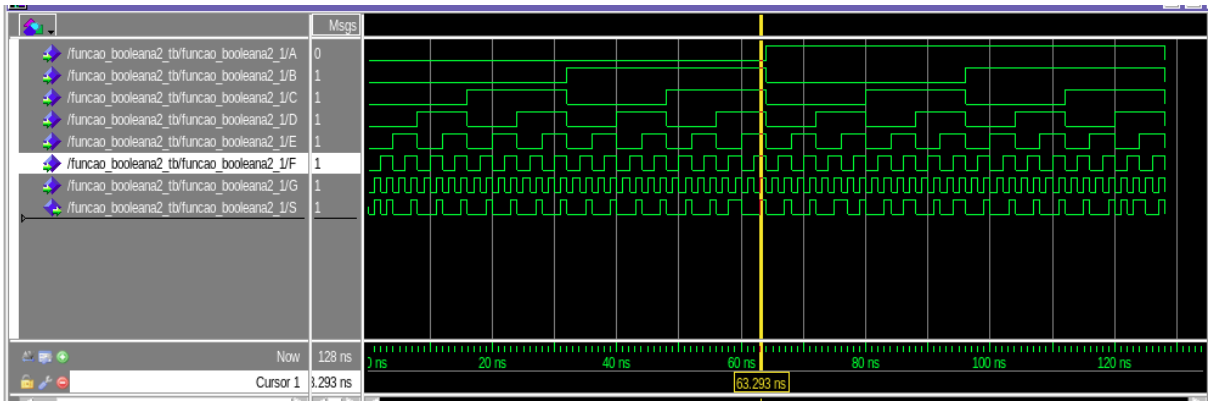
```
# Compile of funcao_booleana.vhd was successful.
# Compile of funcao_booleana_tb.vhd was successful.
# Compile of multiplexador.vhd was successful.
# Compile of funcao_booleana2.vhd was successful.
# Compile of decodificador_4x16.vhd was successful.
# Compile of multiplexador_8x1.vhd was successful.
# Compile of funcao_booleana_2_tb.vhd was successful.
# 7 compiles, 0 failed with no errors.
```

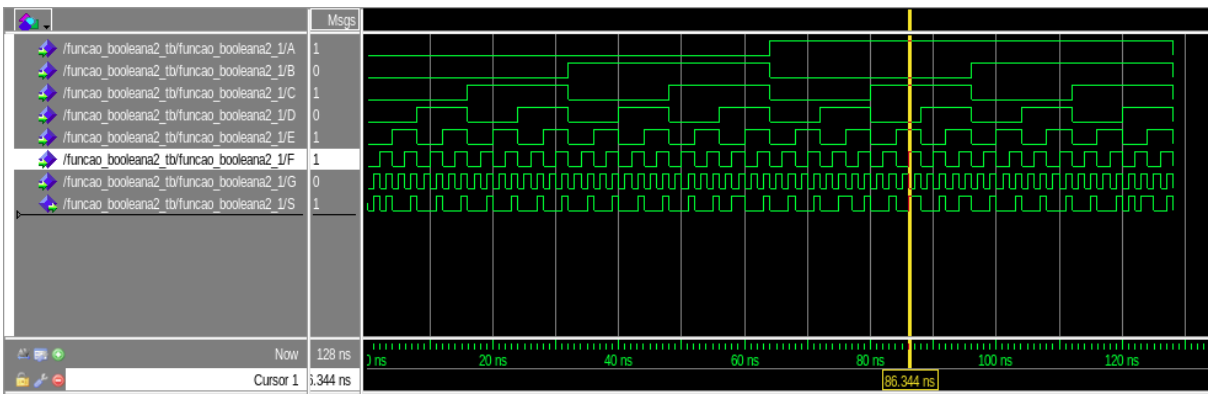
Simulação

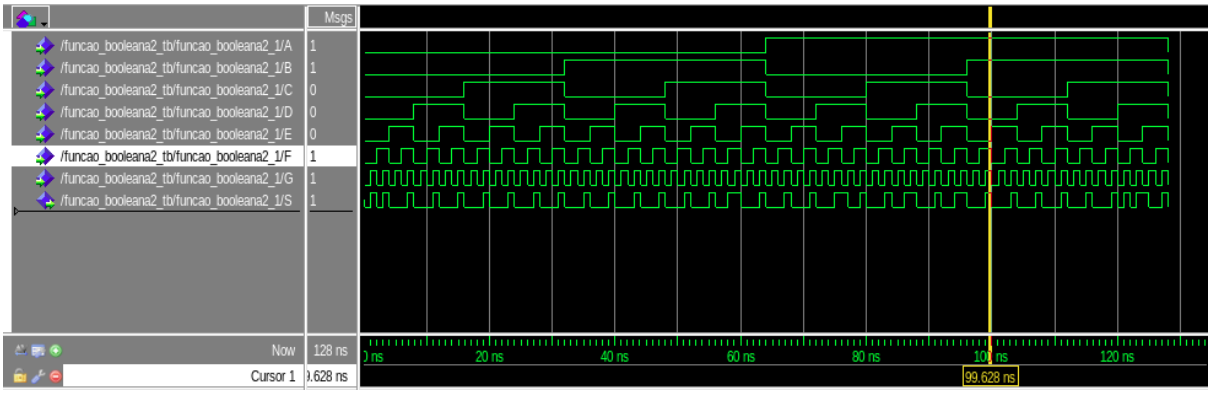



Msgs
/funcao_booleana2_tb/funcao_booleana2_1/A
/funcao_booleana2_tb/funcao_booleana2_1/B
/funcao_booleana2_tb/funcao_booleana2_1/C
/funcao_booleana2_tb/funcao_booleana2_1/D
/funcao_booleana2_tb/funcao_booleana2_1/E
/funcao_booleana2_tb/funcao_booleana2_1/F
/funcao_booleana2_tb/funcao_booleana2_1/G
/funcao_booleana2_tb/funcao_booleana2_1/S
0
0
0
1
1
0
0
0
Now
128 ns
Cursor 1
.502 ns
20 ns
40 ns
60 ns
80 ns
100 ns
120 ns
12.502 ns


Msgs
/funcao_booleana2_tb/funcao_booleana2_1/A
/funcao_booleana2_tb/funcao_booleana2_1/B
/funcao_booleana2_tb/funcao_booleana2_1/C
/funcao_booleana2_tb/funcao_booleana2_1/D
/funcao_booleana2_tb/funcao_booleana2_1/E
/funcao_booleana2_tb/funcao_booleana2_1/F
/funcao_booleana2_tb/funcao_booleana2_1/G
/funcao_booleana2_tb/funcao_booleana2_1/S
0
1
0
0
1
0
0
0
Now
128 ns
Cursor 1
.726 ns
20 ns
40 ns
60 ns
80 ns
100 ns
120 ns
36.726 ns


Msgs
/funcao_booleana2_tb/funcao_booleana2_1/A
/funcao_booleana2_tb/funcao_booleana2_1/B
/funcao_booleana2_tb/funcao_booleana2_1/C
/funcao_booleana2_tb/funcao_booleana2_1/D
/funcao_booleana2_tb/funcao_booleana2_1/E
/funcao_booleana2_tb/funcao_booleana2_1/F
/funcao_booleana2_tb/funcao_booleana2_1/G
/funcao_booleana2_tb/funcao_booleana2_1/S
1
1
1
1
1
1
1
1
Now
128 ns
Cursor 1
.293 ns
20 ns
40 ns
60 ns
80 ns
100 ns
120 ns
63.293 ns


Msgs
/funcao_booleana2_tb/funcao_booleana2_1/A
/funcao_booleana2_tb/funcao_booleana2_1/B
/funcao_booleana2_tb/funcao_booleana2_1/C
/funcao_booleana2_tb/funcao_booleana2_1/D
/funcao_booleana2_tb/funcao_booleana2_1/E
/funcao_booleana2_tb/funcao_booleana2_1/F
/funcao_booleana2_tb/funcao_booleana2_1/G
/funcao_booleana2_tb/funcao_booleana2_1/S
1
0
1
0
1
1
0
1
Now
128 ns
Cursor 1
.344 ns
20 ns
40 ns
60 ns
80 ns
100 ns
120 ns
86.344 ns



Análise

Sejam os seletores EFG = “000”, a saída do sistema do sistema também será 0, independente do valor das variáveis A, B, C e D;

Sejam os seletores EFG ”001”, a saída do sistema será 1 caso A= B = C = D

Sejam os seletores EFG “010”, a saída do sistema será 1 caso A = 0 e B = C = D = 1. Caso contrário, a saída S será 0.

Sejam os seletores EFG “011”, a saída S será “1”, independente dos valores A, B, C e D

Sejam os seletores EFG “100”, a saída será 1 caso A = B = C = D = 1 ou A = 1 e B = 0 e C = 0 e D = 1

Sejam os seletores EFG “101”, a saída será sempre ‘0’, independente dos valores das variáveis A, B, C e D

Sejam os seletores EFG “110”, a saída será ‘1’ somente se A = 1, B = 0, C = 1. Independente do valor de D.

Sejam os seletores EFG “111”, a saída será sempre ‘1’, independente dos valores das variáveis ABCD.

Conclusão

Nesse experimento conseguimos com êxito descrever a função booleana pedida usando um multiplexador 8x1, um decodificador 4X16 e 3 portas OR. As simulações se comportaram da maneira esperada e não foram encontrados erros de sintaxe no código.