

# Relatório 2

Nome: Gabriel Cruz Vaz Santos

Matrícula: 200049038

Turma: C

- Questão 1

## Introdução

Foi solicitado a criação de uma entidade com 3 entradas diferentes ( A, B, Cin) e duas saídas (S e Cout) que implementem um somador completo.

## Teoria

A saída S é o XOR (ou exclusivo) das entradas (A XOR B XOR Cin), enquanto a saída Cout é a sequência de AB em paralelo a sequência de ACin e BCin ((A AND B) OR (Cin AND B) OR (B AND Cin))

## Códigos

Os códigos foram programados no ambiente de desenvolvimento integrado Visual Studio Code, na linguagem VHDL e compilados pelo software do ModelSim. A figura 1 representa o código para implementação do somador completo e a figura 2 o teste

```
somador_completo.vhd
1  library ieee;
2  use ieee.std_logic_1164.all;
3
4  entity somador_completo is
5      port(
6          A: in std_logic;
7          B: in std_logic;
8          Cin: in std_logic;
9          S: out std_logic;
10         Cout: out std_logic
11     );
12 end somador_completo;
13
14 architecture somador_completo_arch of somador_completo is
15     begin
16         S <= A xor B xor Cin;
17         Cout <= ( A and B ) or ( A and Cin ) or ( B and Cin );
18     end somador_completo_arch;
```



```

1  library ieee;
2  use ieee.std_logic_1164.all;
3
4  entity somador_completo_tb is end;
5
6  architecture somador_completo_tb_arch of somador_completo_tb is
7      component somador_completo is
8          port(
9              A: in std_logic;
10             B: in std_logic;
11             cin: in std_logic;
12             S: out std_logic;
13             Cout: out std_logic
14          );
15      end component;
16
17      signal A_1: std_logic;
18      signal B_1: std_logic;
19      signal Cin_1: std_logic;
20
21      constant time_1: time := 4 ns;
22      constant time_2: time := 8 ns;
23      constant time_3: time := 16 ns;
24
25      begin
26          somador_comp_1: somador_completo port map ( A => A_1, B => B_1, Cin => Cin_1, S => open, Cout => open );
27
28          clock0: process
29              begin
30                  A_1 <= '0', '1' after time_1/2, '0' after time_1;
31                  wait for time_1;
32              end process clock0;
33
34          clock1: process
35              begin
36                  B_1 <= '0', '1' after time_2/2, '0' after time_2;
37                  wait for time_2;
38              end process clock1;
39
40          clock2: process
41              begin
42                  Cin_1 <= '0', '1' after time_3/2, '0' after time_3;
43                  wait for time_3;
44              end process clock2;
45
46      end somador_completo_tb_arch;

```

## Compilação

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

	somador_completo....	✓	VHDL	0	02/11/2022 04:54:58 ...
	somador_completo_...	✓	VHDL	1	02/11/2022 05:56:33 ...

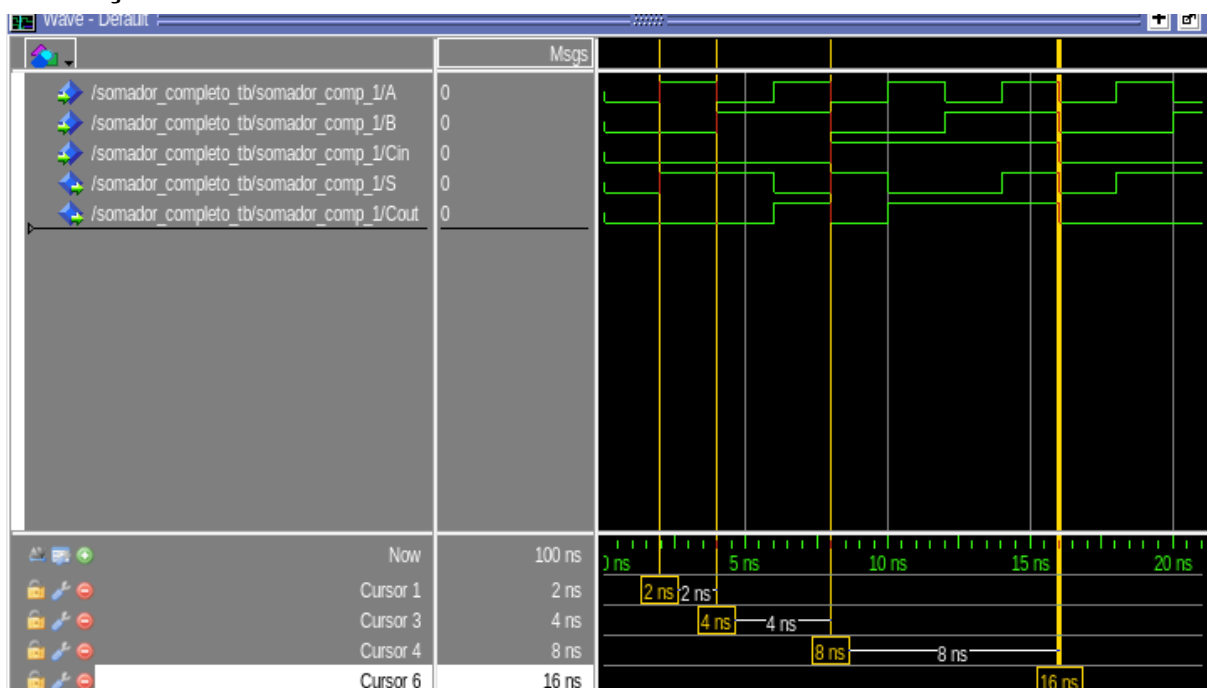
Library x Project x

Transcript

```
# Compile of somador_completo.vhd was successful.
# Compile of somador_completo_tb.vhd was successful.
# 2 compiles, 0 failed with no errors.
```

ModelSim>

## Simulação



## Análise

No instante 0 ns: tanto a saída S quanto Cout são 0, por todas as entradas serem zero;

No instante 4 ns: As entradas A e Cin são 0 enquanto B é 1. Com isso, a saída S vale 1 por se tratar de XORs e a saída Cout é 0, pois apenas uma das entradas é diferente de 0;

No instante 8 ns: As entradas A e B são 0, enquanto Cin é 1. Assim como em 4 ns, a saída S vale 1 e Cout é 0;

No instante 10 ns: As entradas A e Cin são 1, enquanto B é 0. Nesse caso, a saída S vale 0, por se tratar de XOR, enquanto a saída Cout, que possui OR e duas entradas não nulas, vale 1.

## Conclusão

Nesse experimento conseguimos com êxito descrever um somador completo igual ao proposto. As simulações se comportaram da maneira esperada e não foram encontrados erros de sintaxe no código.

## ● Questão 2

### Introdução

Foi solicitado uma entidade com 2 vetores de entrada: D com 4 bits e S com 2 bits para fazer um multiplicador 4x1.

### Teoria

O programa possui apenas uma saída: Y. Os bits do vetor D funcionarão como entradas, enquanto os bits do vetor S funcionarão como os seletores que determinarão quais entradas de dados serão observadas pela saída Y.

### 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 representa o código para implementação do somador completo e as figuras 2, 3 e 4 representam o teste.

```
1 library ieee;
2 use ieee.std_logic_1164.all;
3
4 entity multiplexador is
5     port(
6         D: in std_logic_vector (3 downto 0);
7         S: in std_logic_vector (1 downto 0);
8         Y: out std_logic
9     );
10 end multiplexador;
11
12 architecture multiplexador_arch of multiplexador is
13     begin
14         Y <= ( D(0) and not(S(1)) and not(S(0)) ) or ( D(1) and not(S(1)) and S(0) ) or ( D(2) and S(1) and not(S(0)) ) or ( D(3) and S(1) and S(0) );
15     end multiplexador_arch;
```

multiplexador\_tb.vhd

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3
4  entity multiplexador_tb is end;
5
6  architecture multiplexador_tb_arch of multiplexador_tb is
7      component multiplexador is
8          port(
9              D: in std_logic_vector (3 downto 0);
10             S: in std_logic_vector (1 downto 0);
11             Y: out std_logic
12          );
13      end component;
14
15      signal D_1: std_logic_vector (3 downto 0);
16      signal S_1: std_logic_vector (1 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
25      begin
26          multiplexador_1: multiplexador port map (D => D_1, S => S_1, Y => open );
27
28          clock0: process
29              begin
30                  D_1(0) <= '0', '1' after time_0/2, '0' after time_0;
31                  wait for time 0;
```

```

multiplexador_tb.vhd
32     end process clock0;
33
34     clock1: process
35     begin
36         D_1(1) <= '0', '1' after time_1/2, '0' after time_1;
37         wait for time_1;
38     end process clock1;
39
40     clock2: process
41     begin
42         D_1(2) <= '0', '1' after time_2/2, '0' after time_2;
43         wait for time_2;
44     end process clock2;
45
46     clock3: process
47     begin
48         D_1(3) <= '0', '1' after time_3/2, '0' after time_3;
49         wait for time_3;
50     end process clock3;
51
52     clock4: process
53     begin
54         S_1(0) <= '0', '1' after time_4/2, '0' after time_4;
55         wait for time_4;
56     end process clock4;
57
58     clock5: process
59     begin
60         S_1(1) <= '0', '1' after time_5/2, '0' after time_5;
61         wait for time_5;
62     end process clock5;
63
64
65 end multiplexador_tb_arch;

```

## Compilação

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

Project - /home/gabrielcruz/Documents/Unb/labSD/Relatorio2/Experimento2

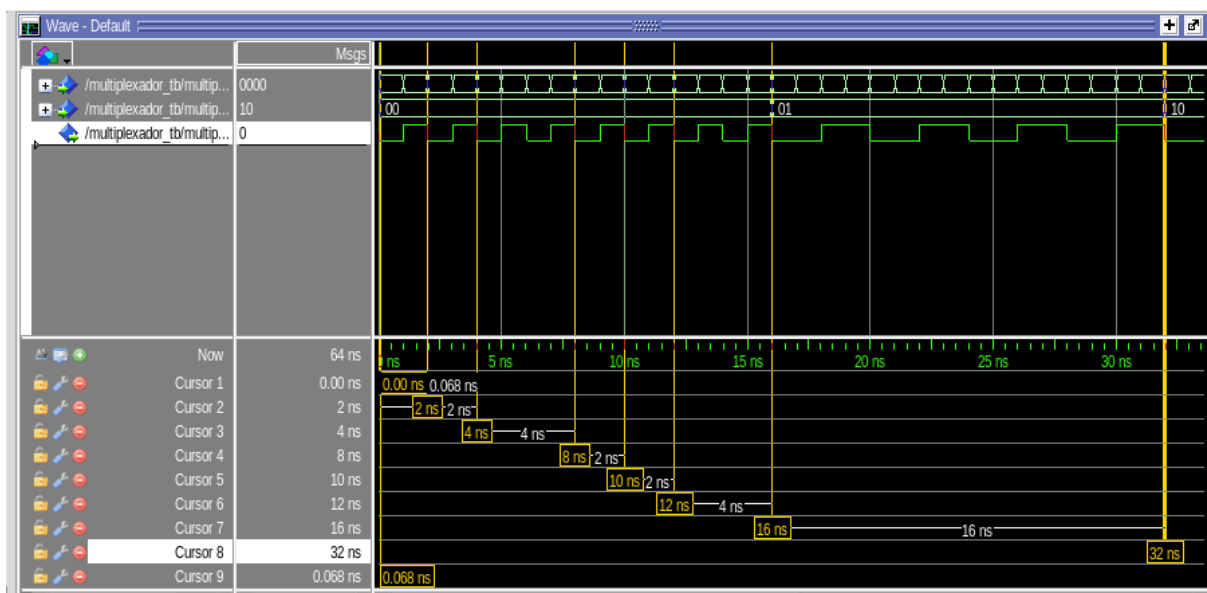
Name	Status	Type	Order	Modified
somador_completo_...	✓	VHDL	1	02/18/2022 09:17:30 ...
somador_completo....	✓	VHDL	0	02/11/2022 04:54:58 ...
multiplexador_tb.vh...	✓	VHDL	3	02/18/2022 09:57:25 ...
multiplexador.vhd	✓	VHDL	2	02/18/2022 08:52:33 ...

Library x Project x

Transcript

```
# Compile of multiplexador.vhd was successful.
# Compile of multiplexador_tb.vhd was successful.
# 4 compiles, 0 failed with no errors.
```

## Simulação



## **Análise**

Quando S0 e S1 valem zero, a saída só será 1 caso D(0) seja 1. Caso contrário, a saída será sempre zero.

Caso S0 seja igual a 1 e S1 igual a 0, a saída só será 1 caso D(1) seja 1. Caso contrário, a saída será sempre zero.

Caso S0 seja igual a 0 e S1 igual a 1, a saída só será 1 caso D(2) seja 1. Caso contrário, a saída será sempre zero.

E caso S0 e S1 sejam iguais a 1, a saída só será 1 caso D(3) seja igual a 1. Caso contrário, a saída será sempre zero.

## **Conclusão**

Nesse experimento conseguimos com êxito descrever um multiplexador 4 x 1 igual ao proposto. As simulações se comportaram da maneira esperada e não foram encontrados erros de sintaxe no código.