



Aluna: Mariana Soares Oliveira  
Matrícula: 231013663  
Turma 01  
28/10/2024

## Relatório Experimento 2

### 1. Introdução

O intuito do seguinte experimento é desenvolver um somador completo (*full adder*) e um multiplexador 4x1 (*MUX 4x1*), utilizando a linguagem de descrição de hardware VHDL, e simular o seu comportamento por meio de *testbenches* realizados no *software* ModelSim.

### 2. Teoria

#### 2.1. Somador Completo (Full Adder)

O somador completo (*full adder*) é um circuito lógico que realiza a soma de três bits: duas entradas principais  $A$  e  $B$  e um bit de carry-in ( $Cin$ ). Ele produz uma saída de soma ( $S$ ) e uma saída de carry-out ( $Cout$ ), representando o "vai-um" que será utilizado na próxima operação de soma.

As funções lógicas são:

$$S = A \oplus B \oplus Cin \quad (1)$$

e

$$Cout = (A \cdot B) + (A \cdot Cin) + (B \cdot Cin) \quad (2)$$

E a sua tabela verdade:

A	B	Cin	S	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



Tabela 1. Tabela verdade do Somador Completo

## 2.2. Multiplexador

Já o multiplexador 4x1 é um circuito que permite selecionar uma dentre quatro entradas de dados ( $D0$ ,  $D1$ ,  $D2$ ,  $D3$ ) para a saída  $Y$ , com base nas entradas de seleção  $S0$  e  $S1$ . Dependendo dos valores de  $S0$  e  $S1$ , uma das entradas  $D$  será direcionada para  $Y$ .

A função lógica é:

$$Y = D0 \cdot \overline{S1} \cdot \overline{S0} + D1 \cdot \overline{S1} \cdot S0 + D2 \cdot S1 \cdot \overline{S0} + D3 \cdot S1 \cdot S0 \quad (3)$$

E a tabela verdade:

S1	S0	Saída Y
0	0	D0
0	1	D1
1	0	D2
1	1	D3

Tabela 2. Tabela verdade reduzida do Multiplexador 4x1

## 3. Códigos

Neste experimento utilizamos a linguagem de descrição de hardware VHDL por meio do software Modelsim para desenvolver um somador completo e um multiplexador 4x1 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/Experimento2/Somador/somador.vhd - Default
Ln#
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3
4  -- definição da entidade
5  entity ent_somador is
6  Port (
7      A : in STD_LOGIC;
8      B : in STD_LOGIC;
9      Cin : in STD_LOGIC;
10     S : out STD_LOGIC;
11     Cout : out STD_LOGIC
12 );
13 end ent_somador;
14
15 -- definição da arquitetura
16 architecture arch_somador of ent_somador is
17 begin
18     -- definindo a soma e o carry-out segundo a fórmula
19     S <= A xor B xor Cin;
20     Cout <= (A and B) or (A and Cin) or (B and Cin);
21 end arch_somador;
```

Figura 1. Codificação do Somador Completo

```
C:/Users/maria/Desktop/Experimento2/Mux4x1/mux4x1.vhd - Default
Ln#
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3
4  -- entidade do multiplexador
5  entity ent_mux4x1 is
6  port (
7      S : in std_logic_vector(1 downto 0);
8      D : in std_logic_vector(3 downto 0);
9      Y : out std_logic
10 );
11 end ent_mux4x1;
12
13 -- arquitetura do multiplexador
14 architecture arch_mux4x1 of ent_mux4x1 is
15 begin
16     process(S, D)
17     begin
18         case S is
19             when "00" =>
20                 Y <= D(0); -- seleciona D0
21             when "01" =>
22                 Y <= D(1); -- seleciona D1
23             when "10" =>
24                 Y <= D(2); -- seleciona D2
25             when "11" =>
26                 Y <= D(3); -- seleciona D3
27             when others =>
28                 Y <= '0'; -- estado default (não usado no multiplexador)
29         end case;
30     end process;
31 end arch_mux4x1;
```

Figura 2. Codificação do Multiplexador 4x1



```
C:/Users/maria/Desktop/Experimento2/Somador/tb_somador.vhd - Default *
Ln#
1  entity testbench1 is
2  end testbench1;
3
4  library IEEE;
5  use IEEE.STD_LOGIC_1164.ALL;
6  use std.textio.all;
7
8  -- arquitetura da testbench
9  architecture tb_somador of testbench1 is
10
11     component ent_somador is
12     port(
13         A      : in std_logic;
14         B      : in std_logic;
15         Cin    : in std_logic;
16         S      : out std_logic;
17         Cout   : out std_logic;
18     );
19     end component;
20
21     signal A      : std_logic;
22     signal B      : std_logic;
23     signal Cin    : std_logic;
24     signal S      : std_logic;
25     signal Cout   : std_logic;
26
27     -- instância do componente ent_somador e mapeamento dos sinais
28     begin
29         somador1: ent_somador port map (A => A, B => B, Cin => Cin, S => S, Cout => Cout);
30
31         -- processo de estímulo para testar o somador
32         estimulo: process
33         begin
34             wait for 5 ns; A <= '0'; B <= '0'; Cin <= '0'; -- Teste 1: S=0, Cout=0
35             wait for 5 ns; Cin <= '1'; -- Teste 2: S=1, Cout=0
36             wait for 5 ns; B <= '1'; -- Teste 3: S=0, Cout=1
37             wait for 5 ns; Cin <= '0'; -- Teste 4: S=1, Cout=0
38             wait for 5 ns; A <= '1'; -- Teste 5: S=0, Cout=1
39             wait for 5 ns; Cin <= '1'; -- Teste 6: S=1, Cout=1
40             wait for 5 ns; B <= '0'; -- Teste 7: S=0, Cout=1
41             wait for 5 ns; Cin <= '0'; -- Teste 8: S=1, Cout=0
42             wait;
43         end process;
44     end tb_somador;
```

Figura 3. *Testbench* do Somador Completo



```
C:/Users/maria/Desktop/Experimento2/Mux4x1/tb_mux4x1.vhd - Default
Ln#
1  library ieee;
2  use ieee.std_logic_1164.ALL;
3  use std.textio.all;
4
5  -- entidade da testbench
6  entity testbenchmux is
7  end testbenchmux;
8
9  -- arquitetura da testbench
10 architecture tb_mux4x1 of testbenchmux is
11
12     component ent_mux4x1 is
13     port(
14         S : in std_logic_vector(1 downto 0);
15         D : in std_logic_vector(3 downto 0);
16         Y : out std_logic
17     );
18     end component;
19
20     signal D : std_logic_vector(3 downto 0);
21     signal S : std_logic_vector(1 downto 0);
22     signal Y : std_logic;
23
24 begin
25
26     mux4x1_l: ent_mux4x1 PORT MAP (D => D, S => S, Y => Y)
27
28     estimulo: process
29     begin
30         wait for 5 ns; D <= "1010"; S <= "00";
31         wait for 5 ns; S <= "01";
32         wait for 5 ns; S <= "10";
33         wait for 5 ns; S <= "11";
34         wait for 5 ns; D <= "0101"; S <= "00";
35         wait for 5 ns; S <= "01";
36         wait for 5 ns; S <= "10";
37         wait for 5 ns; S <= "11";
38         wait for 5 ns; D <= "0000"; S <= "00";
39         wait for 5 ns; S <= "01";
40         wait for 5 ns; S <= "10";
41         wait for 5 ns; S <= "11";
42         wait for 5 ns; D <= "1111"; S <= "00";
43         wait for 5 ns; S <= "01";
44         wait for 5 ns; S <= "10";
```

Figura 4a. Testbench do Multiplexador 4x1

```
        wait for 5 ns; S <= "11";
        wait; -- finaliza o processo
    end process;
end tb_mux4x1;
```

Figura 4b. Testbench do Multiplexador 4x1 (continuação)

## 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 apresentaram erros de sintaxe.



project - C:/Users/maria/Desktop/Experimento2/Somador/Somador

Name	Status	Type	Order	Modified
somador.vhd	✓	VHDL	0	10/30/2024 09:17:16 ...
tb_somador.vhd	✓	VHDL	1	10/30/2024 09:19:28 ...

Transcript

```
# Reading pref.tcl
# Loading project Somador
# Errors: 0, Warnings: 0
# Compile of somador.vhd was successful.
# Errors: 0, Warnings: 0
# Compile of tb_somador.vhd was successful.
# Compile of somador.vhd was successful.
# Compile of tb_somador.vhd was successful.
# 2 compiles, 0 failed with no errors.

ModelSim>
```

Figura 5. Compilação do Somador Completo

Project - C:/Users/maria/Desktop/Experimento2/Mux4x1/Mux4x1

Name	Status	Type	Order	Modified
mux4x1.vhd	✓	VHDL	0	10/30/2024 10:45:26 ...
tb_mux4x1.vhd	✓	VHDL	1	10/30/2024 11:11:34 ...

Transcript

```
# Reading pref.tcl
project open C:/Users/maria/Desktop/Experimento2/Mux4x1/Mux4x1
# Loading project Mux4x1
# Compile of mux4x1.vhd was successful.
# Compile of tb_mux4x1.vhd was successful.
# 2 compiles, 0 failed with no errors.

ModelSim>
```

Figura 6. Compilação do Multiplexador 4x1

## 5. Simulação

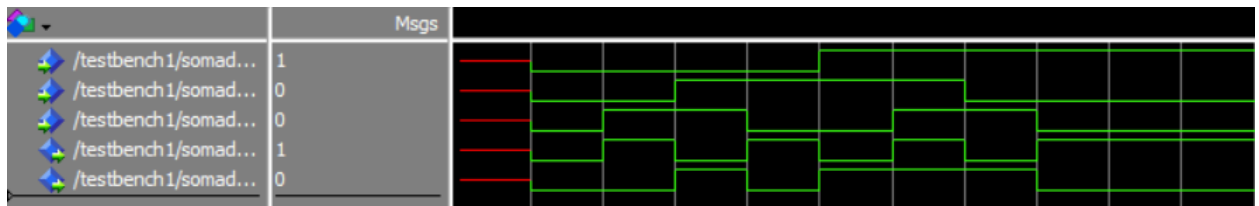


Figura 7. Simulação de onda do banco de testes do Somador Completo

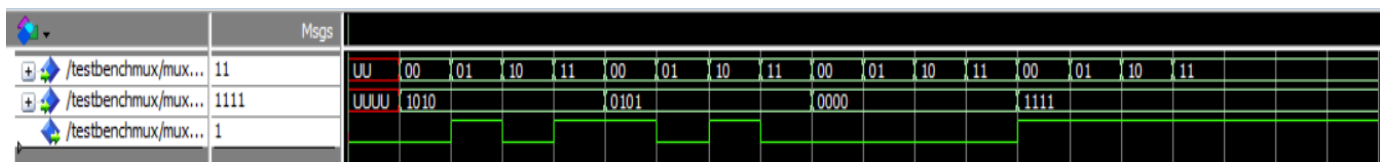


Figura 8. Simulação de onda do banco de testes do Multiplexador 4x1



## **6. Análise**

Neste experimento, foram analisadas duas estruturas: o somador completo e o multiplexador 4x1. Com base nas Tabelas Verdade 1 e 2, nas Equações 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.