**Universidade de Brasília**
Departamento de Engenharia Elétrica

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 *testbenchs* 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:

$$S = A \oplus B \oplus Cin \tag{1}$$

e

$$Cout = (A \cdot B) + (A \cdot Cin) + (B \cdot Cin) \tag{2}$$

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

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- definição da entidade
entity ent_somador is
    Port (
        A : in STD_LOGIC;
        B : in STD_LOGIC;
        Cin : in STD_LOGIC;
        S : out STD_LOGIC;
        Cout : out STD_LOGIC
    );
end ent_somador;

-- definição da arquitetura
architecture arch_somador of ent_somador is
begin
    -- definindo a soma e o carry-out segundo a fórmula
    S <= A xor B xor Cin;
    Cout <= (A and B) or (A and Cin) or (B and Cin);
end arch_somador;
```

Figura 1. Codificação do Somador Completo

C:/Users/maria/Desktop/Experimento2/Mux4x1/mux4x1.vhd - Default

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- entidade do multiplexador
entity ent_mux4x1 is
    port (
        S : in  std_logic_vector(1 downto 0);
        D : in  std_logic_vector(3 downto 0);
        Y : out std_logic
    );
end ent_mux4x1;

-- arquitetura do multiplexador
architecture arch_mux4x1 of ent_mux4x1 is
begin
    process(S, D)
    begin
        case S is
            when "00" =>
                Y <= D(0); -- seleciona D0
            when "01" =>
                Y <= D(1); -- seleciona D1
            when "10" =>
                Y <= D(2); -- seleciona D2
            when "11" =>
                Y <= D(3); -- seleciona D3
            when others =>
                Y <= '0'; -- estado default (não usado no multiplexador)
        end case;
    end process;
end arch_mux4x1;
```

Figura 2. Codificação do Multiplexador 4x1

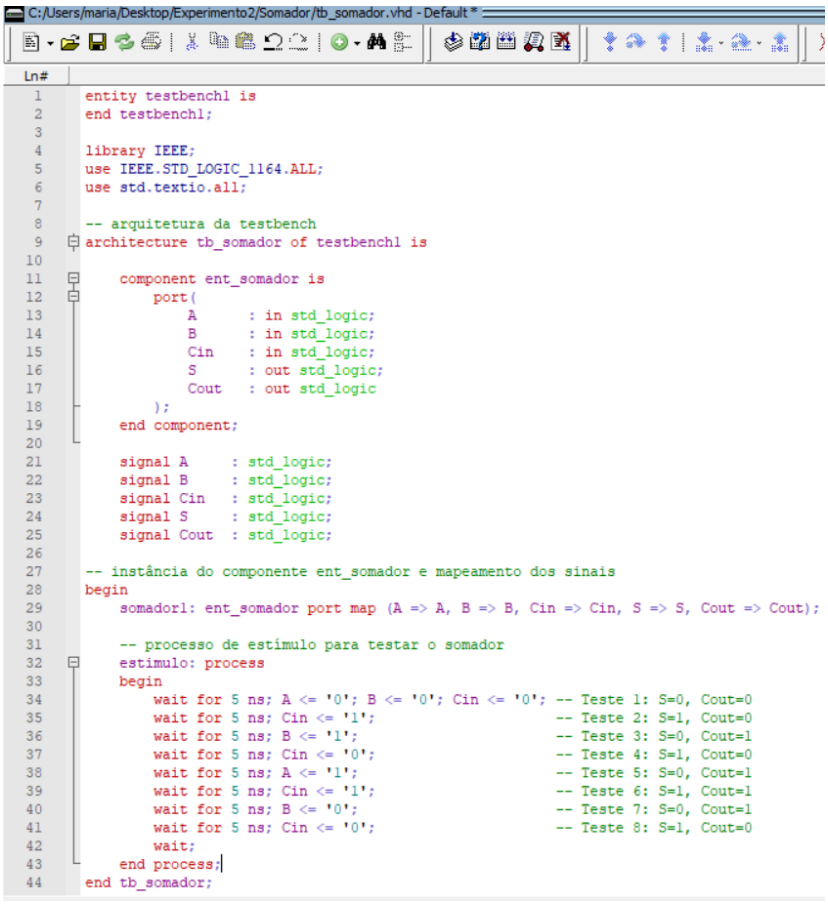
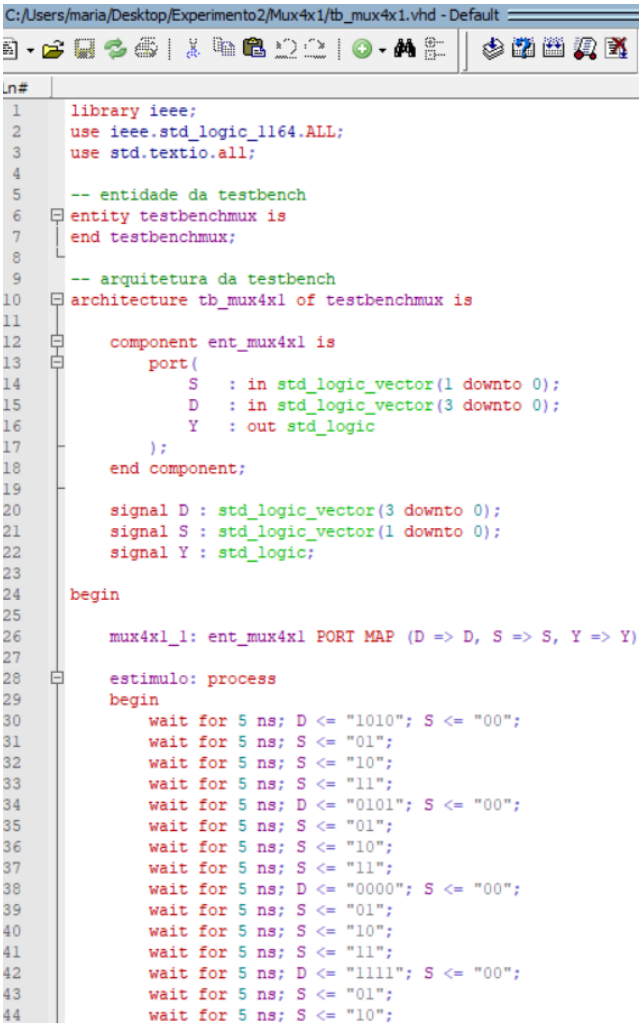



Figura 3. *Testbench* do Somador Completo




```
library ieee;
use ieee.std_logic_1164.ALL;
use std.textio.all;

-- entidade da testbench
entity testbenchmux is
end testbenchmux;

-- arquitetura da testbench
architecture tb_mux4x1 of testbenchmux is

    component ent_mux4x1 is
        port(
            S   : in std_logic_vector(1 downto 0);
            D   : in std_logic_vector(3 downto 0);
            Y   : out std_logic
        );
    end component;

    signal D : std_logic_vector(3 downto 0);
    signal S : std_logic_vector(1 downto 0);
    signal Y : std_logic;

begin

    mux4x1_1: ent_mux4x1 PORT MAP (D => D, S => S, Y => Y)

    estimulo: process
    begin
        wait for 5 ns; D <= "1010"; S <= "00";
        wait for 5 ns; S <= "01";
        wait for 5 ns; S <= "10";
        wait for 5 ns; S <= "11";
        wait for 5 ns; D <= "0101"; S <= "00";
        wait for 5 ns; S <= "01";
        wait for 5 ns; S <= "10";
        wait for 5 ns; S <= "11";
        wait for 5 ns; D <= "0000"; S <= "00";
        wait for 5 ns; S <= "01";
        wait for 5 ns; S <= "10";
        wait for 5 ns; S <= "11";
        wait for 5 ns; D <= "1111"; S <= "00";
        wait for 5 ns; S <= "01";
        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.

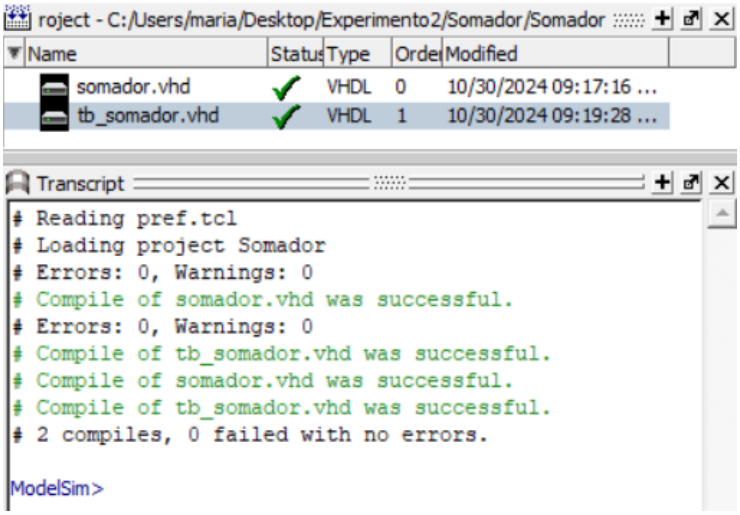



Figura 5. Compilação do Somador Completo

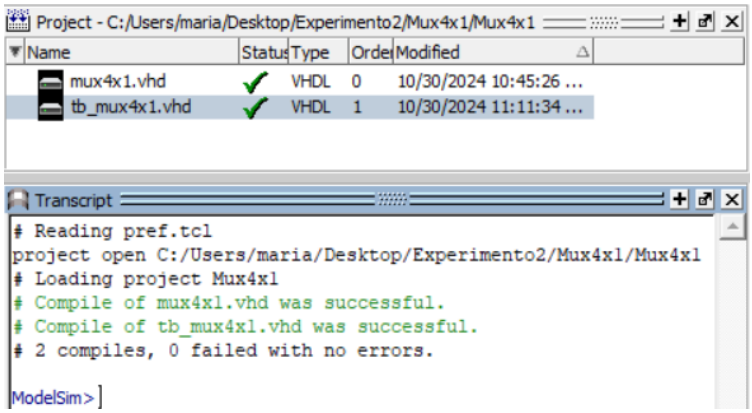



Figura 6. Compilação do Multiplexador 4x1

## 5. Simulação

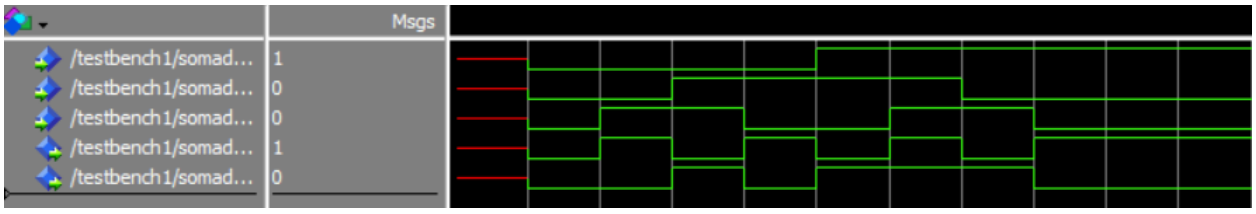



Figura 7. Simulação de onda do banco de testes do Somador Completo

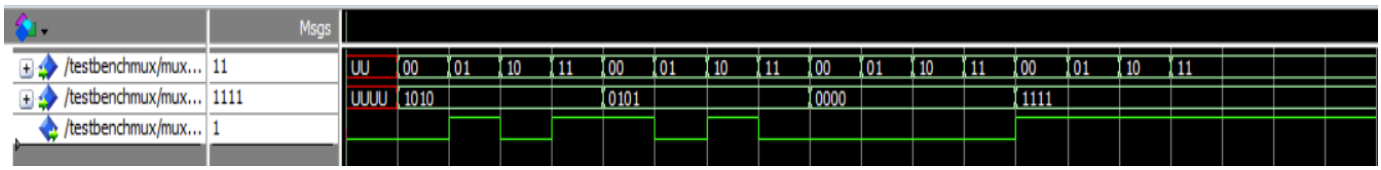



Figura 8. Simulação de onda do banco de testes do Multiplexador 4x1

## 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.