Universidade de Brasília
Faculdade de Tecnologia
Laboratório de Sistemas Digitais

Relatório 02

Carla de Araujo Clementino Ribeiro Mat:180030736

Professor:
Guilherme de Sousa Torres

1 Objetivos

Implementar um somadorcompleto e um multiplexador utilizando a linguagem de descrição de hardware VHDL.

2 Questões Propostas

1. Descrever em VHDL e implementar em FPGA uma entidade com três bits de entrada (A, B e C_{in}) e dois bits de saída (S e C_{out}) e sua arquitetura, que implemente um somador completo, descrito pelas seguintes funções lógicas. Associe cada um dos bits de entrada a diferentes chaves (SW0 a SW7) e os de saída a diferentes LEDs (LD0 a LD7).

$$S = A \oplus B \oplus C_{in} \tag{1}$$

$$C_{out} = AB + AC_{in} + BC_{in} \tag{2}$$

Considerando a questão proposta, primeiramente, foi implementado um componente para realizar as operações lógicas descritas pelas equações (1) (2):

```
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 end somador_completo;
12
13 architecture Behavioral of somador_completo is
14
15 begin
16
17 S <= A xnor B xnor Cin;
18
19 Cout <= (A and B) or (A and Cin) or (B and Cin);
20
21
22 end Behavioral;
23
```

Figura 1: Somador Completo. Fonte: Autor.

Dessa forma, podemos analisar que há 3 entradas (A, B e C_{in}) e 2 saídas(S e C_{out}), sendo as saídas os resultados das operações lógicas realizadas com as entradas.

Em seguida, criamos uma entidade para que fosse possível realizar simulações do nosso circuito por meio de sinais controlados:

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

entity master is
    Port (
        sw0 : in  STD_LOGIC;
        sw1 : in  STD_LOGIC;
        sw2 : in  STD_LOGIC;
        led : out STD_LOGIC;
        Cout: out STD_LOGIC
    );
end master;

architecture Behavioral of master is

    signal U0, U1, U2 : STD_LOGIC;
    signal S : STD_LOGIC;
    signal CarryOut : STD_LOGIC;

    component somador_completo
    Port (
        A : in STD_LOGIC;
        B : in STD_LOGIC;
        Cin : in STD_LOGIC;
        S : out STD_LOGIC;
        Cout : out STD_LOGIC
    );
    end component;
begin

    U0 <= sw0;
    U1 <= sw1;
    U2 <= sw2;

    somador : somador_completo
        port map (
            A => U0,
            B => U1,
            Cin => U2,
            S => S,
            Cout => CarryOut
        );

    led <= S;
    Cout <= CarryOut;
end Behavioral;
```

Figura 2: Entidade - Somador. Fonte: Autor.

Cada um dos bits de entrada são associados a uma chave (sw0, sw1 e sw2) e os de saída a leds (led e Cout).

Por fim, foi criado um aquivo para simular todas as possíveis entradas do circuito:

```
1   library IEEE;
2   use IEEE.STD_LOGIC_1164.ALL;
3
4   entity Test_banch is
5   end Test_banch;
6
7   architecture Behavioral of Test_banch is
8     signal U0, U1, U2 : STD_LOGIC;
9     signal S, Cout : STD_LOGIC;
10
11    component Master is
12      Port (
13        sw0 : in  STD_LOGIC;
14        sw1 : in  STD_LOGIC;
15        sw2 : in  STD_LOGIC;
16        led : out STD_LOGIC;
17        Cout : out STD_LOGIC
18      );
19    end component;
20
21  begin
22
23  UUT: Master port map (
24    sw0 => U0,
25    sw1 => U1,
26    sw2 => U2,
27    led => S,
28    Cout => Cout
29  );
30
```

```
30
31  process
32  begin
33    U0 <= '0'; U1 <= '0'; U2 <= '0';
34    wait for 10ns;
35    U0 <= '0'; U1 <= '0'; U2 <= '1';
36    wait for 10ns;
37    U0 <= '0'; U1 <= '1'; U2 <= '0';
38    wait for 10ns;
39    U0 <= '0'; U1 <= '1'; U2 <= '1';
40    wait for 10ns;
41    U0 <= '1'; U1 <= '0'; U2 <= '0';
42    wait for 10ns;
43    U0 <= '1'; U1 <= '0'; U2 <= '1';
44    wait for 10ns;
45    U0 <= '1'; U1 <= '1'; U2 <= '0';
46    wait for 10ns;
47    U0 <= '1'; U1 <= '1'; U2 <= '1';
48    wait for 10ns;
49
50   end process;
51
52  end Behavioral;
53
```

Figura 3: Casos de Teste - Somador. Fonte: Autor.

Os possíveis casos podem ser resumidos a seguinte tabela verdade:

A	B	C_{in}	S	C_{out}
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 de um Somador Completo. Fonte: Autor

Realizando a simulação e comparando com a tabela verdade acima vemos que os resultados batem:

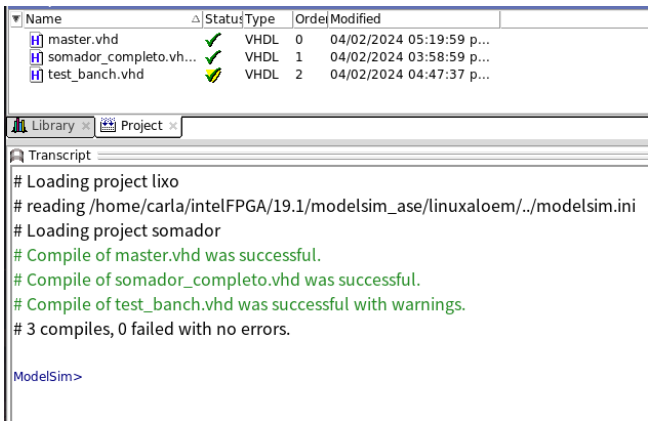

Figura 4: Compilação dos arquivos - Somador Completo. Fonte: Autor.

Figura 5: Simulação do Somador Completo. Fonte: Autor.

2. Descrever em VHDL e implementar em FPGA uma entidade com dois vetores de entrada (S com 2 bits e com 4 bits) e um bit de saída (D) e sua arquitetura, que implemente um multiplexador de 4 para 1, descrito pela função lógica abaixo. Associe cada um dos bits de entrada a diferentes chaves (SW0 a SW7) e a saída a um dos LEDs (LD0 a LD7).

$$Y = D_0 \cdot \overline{S_1} \cdot \overline{S_0} + D_1 \cdot \overline{S_1} \cdot S_0 + D_2 \cdot S_1 \cdot \overline{S_0} + D_3 \cdot S_1 \cdot S_0 \tag{3}$$

Assim como na questão anterior, começamos fazendo um componente para realizar a equação lógica acima (3):

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3
4  entity mux is
5    Port (
6         S : in STD_LOGIC_VECTOR (1 downto 0); -- vetor de 2 bits (entrada)
7         D : in STD_LOGIC_VECTOR (3 downto 0); -- vetor de 4 bits (entrada)
8         Y : out STD_LOGIC  -- (saida)
9    );
10 end mux;
11
12 architecture Behavioral of mux is
13
14 begin
15
16 -- Y = D0.S1'.S2' + D1.S1'.S0 + D2.S1.S0' + D3.S1.S2
17 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));
18
19 end Behavioral;
20
```

Figura 6: Multiplexador. Fonte: Autor.

Analisando o código podemos ver que temos 2 entradas, sendo uma um array de 4 bits e a outra um array de 2 bits, e uma saída de 1 bit.

Implementando uma entidade para manipular que possamos realizar simulações temos:

```
4  entity master is
5    Port (
6      sw1 : in  STD_LOGIC_VECTOR(3 downto 0);
7      sw0 : in  STD_LOGIC_VECTOR(1 downto 0);
8      led : out STD_LOGIC
9    );
10 end master;
11
12 architecture Behavioral of master is
13
14   signal U0 : STD_LOGIC_VECTOR(1 downto 0);
15   signal U1 : STD_LOGIC_VECTOR(3 downto 0);
16   signal Y : STD_LOGIC;
17
18   component mux is
19     Port (
20       S : in STD_LOGIC_VECTOR(1 downto 0);
21       D : in STD_LOGIC_VECTOR(3 downto 0);
22       Y : out STD_LOGIC
23     );
24   end component;
25 begin
26
27   U0 <= sw0;
28   U1 <= sw1;
29
30   somador : mux
31       port map (
32         S => U0,
33         D => U1,
34         Y => Y
35       );
36
37   led <= Y;
38
39 end Behavioral;
```

Figura 7: Entidade - Multiplexador. Fonte: Autor.

E por fim, para nosso arquivo de teste:

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

entity test_bench is
end test_bench;

architecture Behavioral of test_bench is
    signal U0 : STD_LOGIC_VECTOR(1 downto 0) := "00";
    signal U1 : STD_LOGIC_VECTOR(3 downto 0) := "0000";
    signal Y : STD_LOGIC;

    component master is
        Port (
            sw0 : in  STD_LOGIC_VECTOR(1 downto 0);
            sw1 : in  STD_LOGIC_VECTOR(3 downto 0);
            led : out STD_LOGIC
        );
    end component;

begin

UUT: master port map (
    sw0 => U0,
    sw1 => U1,
    led => Y
);

-- Para gerar todas as 16 possivilidades de numeros com 4 bits
U1(0) <= not U1(0) after 2ns;
U1(1) <= not U1(1) after 4ns;
U1(2) <= not U1(2) after 8ns;
U1(3) <= not U1(3) after 16ns;


process
begin

    -- Para gerar todas as possibilidades de numeros com 2 bits
    U0 <= "00";
    wait for 32ns;
    U0 <= "01";
    wait for 32ns;
    U0 <= "10";
    wait for 32ns;
    U0 <= "11";
    wait for 32ns;

    wait;
end process;

end Behavioral;
```

Figura 8: Casos de Teste - Multiplexador. Fonte: Autor.

Os testes podem ser resumidos por essa tabela verdade:

S_1	S_0	D_3	D_2	D_1	D_0	Y
0	0	0	0	0	0	0
0	0	0	0	0	1	0
0	0	0	0	1	0	0
0	0	0	0	1	1	0
...	...	...	...	...	...	...
1	1	1	1	0	1	1
1	1	1	1	1	0	1
1	1	1	1	1	1	1

Tabela 2: Tabela Verdade de um Multiplexador de 4 Bits (64 Possibilidades). Fonte: Autor

Realizando a simulação e comparando com a tabela obtemos os seguintes dados:

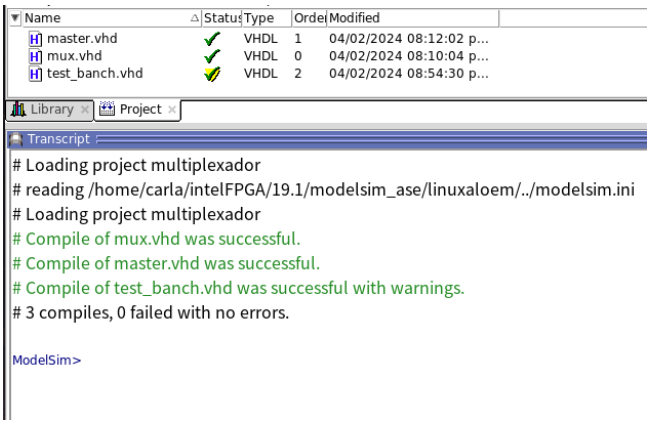

Figura 9: Compilação dos arquivos - Multiplexador. Fonte: Autor.

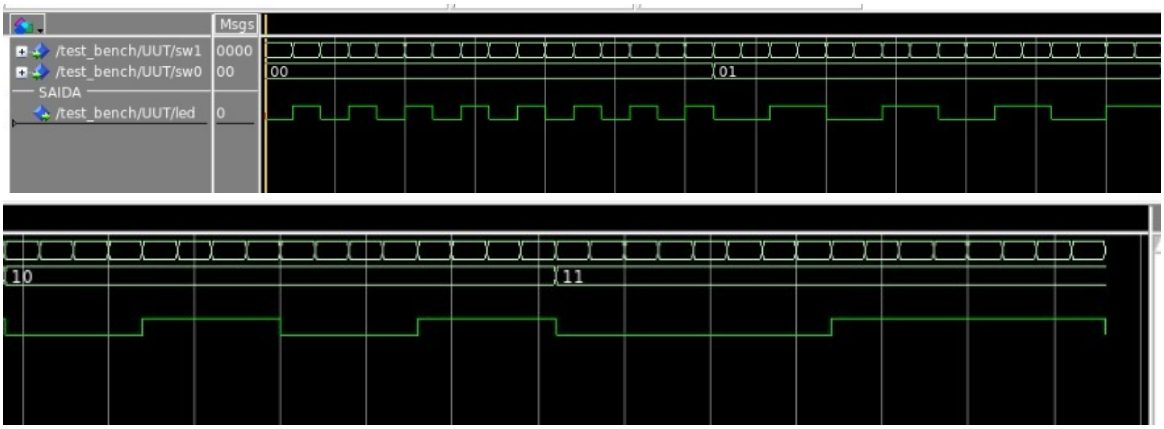

Figura 10: Simulação do Multiplexador. Fonte: Autor.