

Atividade 3: Display de 7 segmentos

Maria Eduarda Soares Romana Silva - 2408830

1 Introdução

O presente relatório descreve a terceira atividade da disciplina de Lógica Reconfigurável. Nela, foi explorado o funcionamento dos displays de 7 segmentos (SSD) em uma FPGA, por meio da linguagem VHDL. O objetivo principal foi desenvolver um código que recebesse dois números binários, de 0 a 9, por meio das chaves da placa, e os exibisse em dois displays distintos.

Além disso, a atividade reforçou o uso do tipo *std_logic_vector*, o mapeamento de portas no *Pin Planner* e a representação do circuito por meio do diagrama RTL.

2 Código VHDL comentado

```
1 -- Atividade 3 - Display de 7 segmentos
2 -- MARIA EDUARDA SOARES ROMANA SILVA
3
4 library ieee;
5 use ieee.std_logic_1164.all;
6
7 entity atv03 is
8     port (
9         switches : in std_logic_vector(7 downto 0); -- 8 chaves para
10        representar 2 numeros de 4 bits cada
11        SSD0 : out std_logic_vector(6 downto 0); --- display 1
12        SSD1 : out std_logic_vector(6 downto 0) -- display 2
13    );
14 end atv03;
15
16 architecture Behavioral of atv03 is
17 begin
18
19     -- Display 1: recebe os bits do primeiro numero atraves das chaves
20     -- 0 ate 3
21     with switches(3 downto 0) select
22         SSD0 <= "1000000" when "0000", -- 0
23             "1111001" when "0001", -- 1
24             "0100100" when "0010", -- 2
25             "0110000" when "0011", -- 3
26             "0011001" when "0100", -- 4
27             "0010001" when "0101", -- 5
28             "0000010" when "0110", -- 6
29             "1111000" when "0111", -- 7
30             "0000000" when "1000", -- 8
31             "0010000" when "1001", -- 9
32             "1111111" when others; -- Apaga
```

```
32      -- Display 2: recebe os bits do segundo numero atraves das chaves 4
33      -- ate 7
34      with switches(7 downto 4) select
35          SSD1 <= "1000000" when "0000", -- 0
36              "1111001" when "0001", -- 1
37              "0100100" when "0010", -- 2
38              "0110000" when "0011", -- 3
39              "0011001" when "0100", -- 4
40              "0010010" when "0101", -- 5
41              "0000010" when "0110", -- 6
42              "1111000" when "0111", -- 7
43              "0000000" when "1000", -- 8
44              "0010000" when "1001", -- 9
45              "1111111" when others; -- Apaga
46
end Behavioral;
```

Listing 1: Código VHDL - representação de 2 números nos displays de 7 segmentos

3 Foto da placa com os displays apresentando números

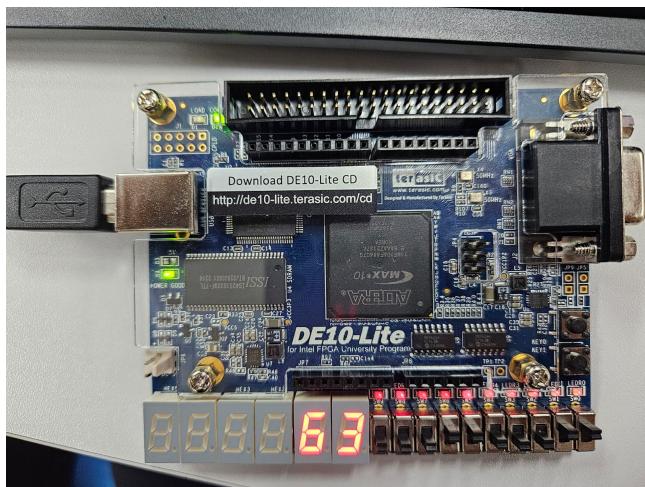


Figura 1: Placa mostrando o número 63

4 Pin Planner

O *Pin Planner* é uma ferramenta do software Quartus utilizada para associar os sinais lógicos do projeto aos pinos físicos da FPGA.

Nesta atividade 3 vetores de sinais foram definidos: `switches`, com 8 bits, responsável por receber os dois valores binários através das chaves da placa, e dois vetores de saída, `SSD0` e `SSD1`, com 7 bits, que correspondem os segmentos dos displays de 7 segmentos

De acordo com a documentação da placa, cada bit dos vetores definidos foram associados a um pino físico da FPGA.

Essa mapeamento se faz necessário para garantir o funcionamento do circuito na prática da forma desejada, pois sem o mesmo, os sinais lógicos não seriam representados corretamente no hardware.

<code>SSD0[6]</code>	Output	<code>PIN_C17</code>	7	<code>B7_NO</code>	<code>PIN_C17</code>	2.5 V		12mA (default)	2 (default)		
<code>SSD0[5]</code>	Output	<code>PIN_D17</code>	7	<code>B7_NO</code>	<code>PIN_D17</code>	2.5 V		12mA (default)	2 (default)		
<code>SSD0[4]</code>	Output	<code>PIN_E16</code>	7	<code>B7_NO</code>	<code>PIN_E16</code>	2.5 V		12mA (default)	2 (default)		
<code>SSD0[3]</code>	Output	<code>PIN_C16</code>	7	<code>B7_NO</code>	<code>PIN_C16</code>	2.5 V		12mA (default)	2 (default)		
<code>SSD0[2]</code>	Output	<code>PIN_C15</code>	7	<code>B7_NO</code>	<code>PIN_C15</code>	2.5 V		12mA (default)	2 (default)		
<code>SSD0[1]</code>	Output	<code>PIN_E15</code>	7	<code>B7_NO</code>	<code>PIN_E15</code>	2.5 V		12mA (default)	2 (default)		
<code>SSD0[0]</code>	Output	<code>PIN_C14</code>	7	<code>B7_NO</code>	<code>PIN_C14</code>	2.5 V		12mA (default)	2 (default)		
<code>SSD1[6]</code>	Output	<code>PIN_B17</code>	7	<code>B7_NO</code>	<code>PIN_B17</code>	2.5 V		12mA (default)	2 (default)		
<code>SSD1[5]</code>	Output	<code>PIN_A18</code>	7	<code>B7_NO</code>	<code>PIN_A18</code>	2.5 V		12mA (default)	2 (default)		
<code>SSD1[4]</code>	Output	<code>PIN_A17</code>	7	<code>B7_NO</code>	<code>PIN_A17</code>	2.5 V		12mA (default)	2 (default)		
<code>SSD1[3]</code>	Output	<code>PIN_B16</code>	7	<code>B7_NO</code>	<code>PIN_B16</code>	2.5 V		12mA (default)	2 (default)		
<code>SSD1[2]</code>	Output	<code>PIN_E18</code>	6	<code>B6_NO</code>	<code>PIN_E18</code>	2.5 V		12mA (default)	2 (default)		
<code>SSD1[1]</code>	Output	<code>PIN_D18</code>	6	<code>B6_NO</code>	<code>PIN_D18</code>	2.5 V		12mA (default)	2 (default)		
<code>SSD1[0]</code>	Output	<code>PIN_C18</code>	7	<code>B7_NO</code>	<code>PIN_C18</code>	2.5 V		12mA (default)	2 (default)		
<code>switches[7]</code>	Input	<code>PIN_A14</code>	7	<code>B7_NO</code>	<code>PIN_A14</code>	2.5 V		12mA (default)			
<code>switches[6]</code>	Input	<code>PIN_A13</code>	7	<code>B7_NO</code>	<code>PIN_A13</code>	2.5 V		12mA (default)			
<code>switches[5]</code>	Input	<code>PIN_B12</code>	7	<code>B7_NO</code>	<code>PIN_B12</code>	2.5 V		12mA (default)			
<code>switches[4]</code>	Input	<code>PIN_A12</code>	7	<code>B7_NO</code>	<code>PIN_A12</code>	2.5 V		12mA (default)			
<code>switches[3]</code>	Input	<code>PIN_C12</code>	7	<code>B7_NO</code>	<code>PIN_C12</code>	2.5 V		12mA (default)			
<code>switches[2]</code>	Input	<code>PIN_D12</code>	7	<code>B7_NO</code>	<code>PIN_D12</code>	2.5 V		12mA (default)			
<code>switches[1]</code>	Input	<code>PIN_C11</code>	7	<code>B7_NO</code>	<code>PIN_C11</code>	2.5 V		12mA (default)			
<code>switches[0]</code>	Input	<code>PIN_C10</code>	7	<code>B7_NO</code>	<code>PIN_C10</code>	2.5 V		12mA (default)			
<<new node>>											

Figura 2: Pin Planner

5 Diagrama RTL

O diagrama RTL é uma representação gráfica do circuito lógico desenvolvido a partir do código VHDL. Ele mostra como os sinais de entrada e saída estão conectados por meio de blocos lógicos internos.

Esse tipo de representação é útil para verificar se a lógica comportamental descrita em VHDL foi corretamente convertida em um circuito físico, permitindo validar o funcionamento do projeto ante de implementa-lo na FPGA.

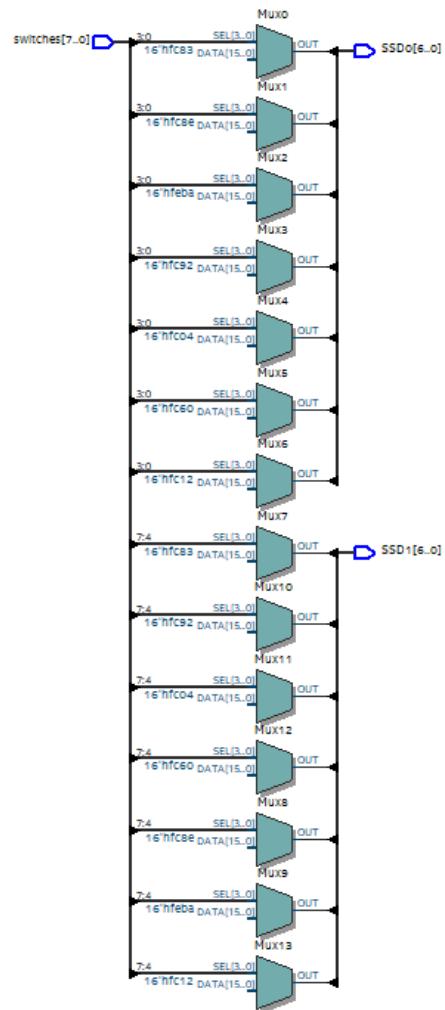


Figura 3: Diagrama RTL