



Prática N^o3 - SEL0606

Alunos:

Vitor Alexandre Garcia Vaz
Gabriel Dezejácomo Maruschi

Sumário

1	Introdução	3
1.1	<i>Objetivos</i>	3
1.2	<i>Decodificador Binário</i>	3
2	Materiais e métodos	4
2.1	<i>Obtenção das expressões booleanas</i>	4
2.2	<i>Código VHDL</i>	4
2.3	<i>Previsão sobre utilização do CI TTL 7400</i>	5
3	Conclusão	6
3.1	<i>Círculo RTL</i>	6
3.2	<i>Número de portas lógicas</i>	6
3.3	<i>Funcionamento do círculo</i>	7

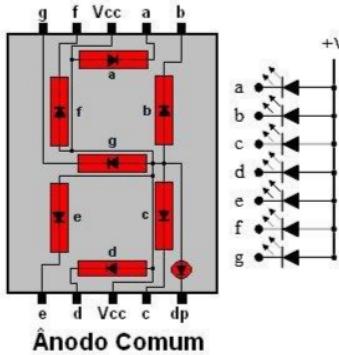
1 Introdução

1.1 Objetivos

Em tal prática, buscamos por meio do uso da FPGA (dispositivo reconfigurável), em conjunto com o uso da ferramenta computacional Quartus de elaboração de sistemas digitais, implementar efetivamente um decodificador binário para sete segmentos.

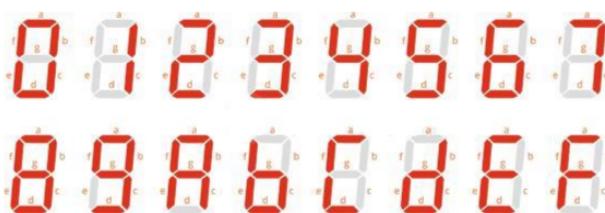
Nesse sentido, os resultados de cada implementação devem ser visualizados por meio do display de sete segmentos mais ponto decimal (fig.1), no qual cada segmento é um led configurado na forma de anodo comum (ativação no nível lógico 0), correspondendo à uma das saídas do decodificador, de forma que os segmentos acesos correspondam a representação hexadecimal (fig.2) do sinal de entrada de 4 bits.

Figure 1: Display de sete segmentos do tipo anodo comum



Fonte: [Site da internet](#)

Figure 2: Modos de acendimento display de 7 segmentos



Fonte: [Site da internet](#)

1.2 Decodificador Binário

O decodificador binário (fig3) implementado na prática tem como objetivo decodificar um sinal de entrada de 4 bits (x_3, x_2, x_1, x_0) em 7 sinais de 1 bit, os quais correspondem aos segmentos do display. Nesse sentido, vale ressaltar que cada bit do sinal de entrada é proveniente do sinal de uma das chaves do dispositivo reconfigurável: Sw_3, Sw_2, Sw_1, Sw_0 (do bit mais para o menos significativo).

Dessa forma, sendo os sete segmentos (a, b, c, d, e, f, g) pertencentes ao primeiro display da FPGA, tais termos terão expressões booleanas, de combinação dos sinais das chaves mecânicas, de modo que os leds acesos formem o símbolo (na forma hexadecimal) do código binário formado pelos 4 bits dos sinal de entrada do decodificador.

Outrossim, vale ressaltar que, para o primeiro display da FPGA, os sete segmentos mencionados correspondem, respectivamente, aos sinais $Hex0_0, Hex0_1, Hex0_2, Hex0_3, Hex0_4, Hex0_5, Hex0_6$ especificados na lista de pinos da FPGA disponibilizada na página da disciplina.

Por fim, para o caso desta prática, vale ressaltar um sétimo segmento (denominado h), o qual está associado ao led ponto decimal e, no display da FPGA, corresponde ao pino $Hex0_7$.

Figure 3: Decodificador binário de sete segmentos



Fonte: [Site da internet](#)

2 Materiais e métodos

2.1 Obtenção das expressões booleanas

A partir do mapeamento dos segmentos do display de led, foi possível relacionar em uma tabela verdade (com lógica *low-active*) as entradas das chaves e a resposta em cada segmento. As expressões obtidas para cada segmento estão explícitas abaixo:

- $a = \bar{x}_3\bar{x}_2\bar{x}_1x_0 + \bar{x}_3x_2\bar{x}_1\bar{x}_0 + x_3x_2\bar{x}_1x_0 + x_3\bar{x}_2x_1x_0$
- $b = \bar{x}_3x_2\bar{x}_1x_0 + x_3x_2\bar{x}_0 + x_2x_1\bar{x}_0 + x_3x_2x_0$
- $c = \bar{x}_3\bar{x}_2x_1\bar{x}_0 + x_3x_2\bar{x}_0 + x_3x_2x_1$
- $d = \bar{x}_3\bar{x}_2\bar{x}_1x_0 + \bar{x}_3x_2\bar{x}_1\bar{x}_0 + x_2x_1x_0 + x_3\bar{x}_2x_1\bar{x}_0$
- $e = \bar{x}_3x_0 + \bar{x}_2\bar{x}_1x_0 + \bar{x}_3x_2\bar{x}_1$
- $f = \bar{x}_3\bar{x}_2x_0 + \bar{x}_3\bar{x}_2x_1 + \bar{x}_3x_1x_0 + x_3x_2\bar{x}_1x_0$
- $g = \bar{x}_3\bar{x}_2\bar{x}_1 + \bar{x}_3x_2x_1x_0 + x_3x_2\bar{x}_1\bar{x}_0$
- $h = 1$

2.2 Código VHDL

Utilizando o mapeamento e barramento dos sinais descrito no manual do DE10-LITE, escrevemos o código em VHDL para o display de 7 segmentos, utilizando o display mais à direita (HEX0) e as quatro chaves mais à direita (SW(3) a SW(0)).

Figure 4: Print do código em VHDL

```

1  -- Copyright
2  -- Name: DE10_LITE_decoder
3  -- Date: 25/09/2024
4  -- Version: 1.2
5  -- Owners: Gabriel Dezejacomo Maruschi
6  -- Vitor Alexandre Garcia Vaz
7
8  entity DE10_LITE_decoder is
9    port(
10      SW : in bit_vector(9 downto 0);      --Chaves
11      HEX0 : out bit_vector(7 downto 0);   --Display
12    );
13  end DE10_LITE_decoder;
14
15  architecture netlist of DE10_LITE_decoder is
16
17  begin
18    -- Segmento 'a'
19    HEX0(0) <= (NOT SW(3) AND NOT SW(2) AND NOT SW(1) AND SW(0)) OR (NOT SW(3) AND SW(2) AND NOT SW(1) AND NOT SW(0)) OR (SW(3) AND SW(2) AND NOT SW(1) AND SW(0)) OR (SW(3) AND NOT SW(2) AND SW(1) AND SW(0));
20
21    -- Segmento 'b'
22    HEX0(1) <= (NOT SW(3) AND SW(2) AND NOT SW(1) AND SW(0)) OR (SW(3) AND SW(2) AND NOT SW(0)) OR (SW(2) AND SW(1) AND NOT SW(0)) OR (SW(3) AND SW(1) AND SW(0));
23
24    -- Segmento 'c'
25    HEX0(2) <= (NOT SW(3) AND NOT SW(2) AND SW(1) AND NOT SW(0)) OR (SW(3) AND SW(2) AND NOT SW(0)) OR (SW(3) AND SW(1) AND SW(0));
26
27    -- Segmento 'd'
28    HEX0(3) <= (NOT SW(3) AND NOT SW(2) AND NOT SW(1) AND SW(0)) OR (NOT SW(3) AND SW(2) AND NOT SW(1) AND NOT SW(0)) OR (SW(2) AND SW(1) AND SW(0)) OR (SW(3) AND NOT SW(2) AND SW(1) AND NOT SW(0));
29
30    -- Segmento 'e'
31    HEX0(4) <= (NOT SW(3) AND SW(0)) OR (NOT SW(2) AND NOT SW(1) AND SW(0)) OR (NOT SW(3) AND SW(2) AND NOT SW(1));
32
33    -- Segmento 'f'
34    HEX0(5) <= (NOT SW(3) AND NOT SW(2) AND SW(0)) OR (NOT SW(3) AND NOT SW(2) AND SW(1)) OR (NOT SW(3) AND SW(1) AND SW(0)) OR (SW(3) AND SW(2) AND NOT SW(1) AND SW(0));
35
36    -- Segmento 'g'
37    HEX0(6) <= (NOT SW(3) AND NOT SW(2) AND NOT SW(1)) OR (NOT SW(3) AND SW(2) AND SW(1) AND SW(0)) OR (SW(3) AND SW(2) AND NOT SW(1) AND NOT SW(0));
38
39    -- Ponto decimal
40    HEX0(7) <= '1';      --Apagado
41
42 end;

```

Fonte: os autores

Todo o código e a compilação foi feita no software Quartus, com a versão gratuita.

2.3 Previsão sobre utilização do CI TTL 7400

Caso, no lugar de um dispositivo reprogramável, utilizássemos apenas circuitos integrados TTL 7400, que possuem 4 NANDS, utilizaríamos aproximadamente 34 CI's. Para chegar neste valor, consideramos as seguintes igualdades:

- $a.b = \overline{(\bar{a} \cdot b)(\bar{a} \cdot \bar{b})}$
- $\bar{a} = (\bar{a} \cdot a)$
- $a + b = \overline{(\bar{a} \cdot a)(\bar{b} \cdot b)}$

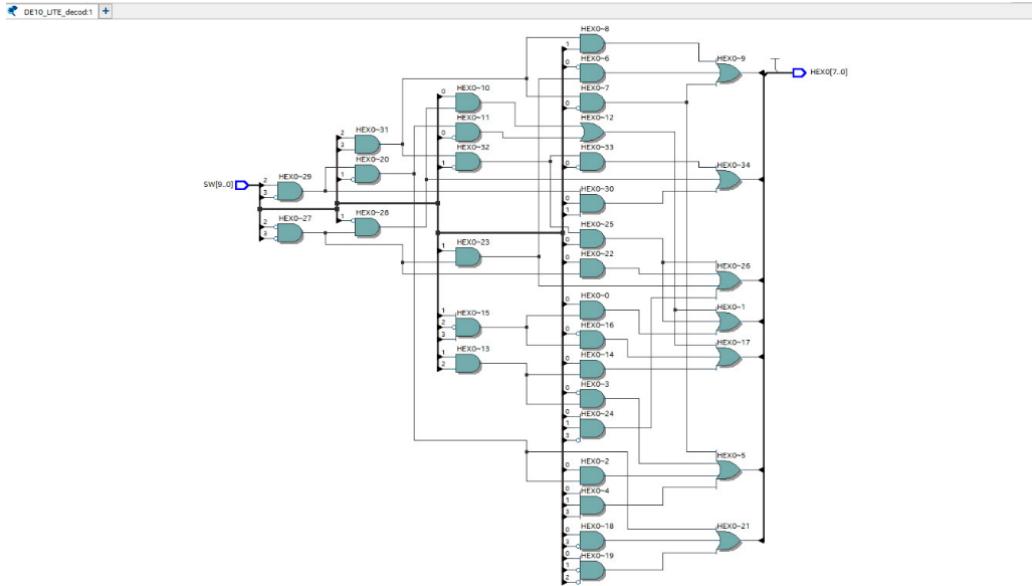
Sendo assim, um CI foi usando inteiramente para obtenção dos complementares dos sinais de entrada, e para cada AND e OR de sinais 2 a 2 foram utilizados 4 CI's. Ao total, descontado termos repetidos, foram realizados 134 NANDs, totalizando 34 circuitos integrados.

3 Conclusão

3.1 Circuito RTL

A partir do código em VHDL (fig4) feito pela dupla, e usando a ferramenta de síntese disponibilizada pelo quartus, chegamos ao circuito da fig5.

Figure 5: Print do circuito sintetizado

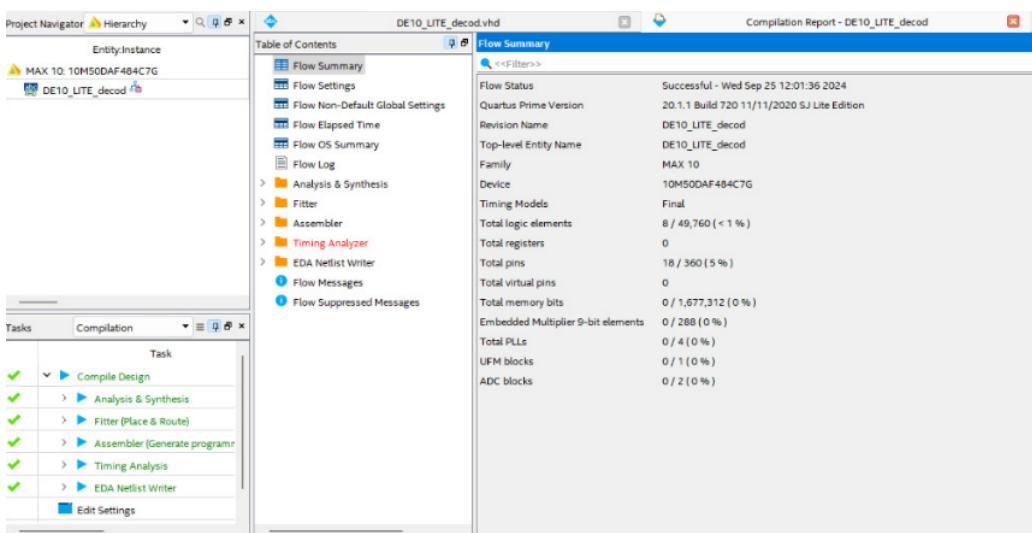


Fonte: os autores

3.2 Número de portas lógicas

Ademais, por meio do resumo do funcionamento e constituição do circuito descrito em VHDL no software (fig6), chegamos à conclusão de que o circuito sintetizado apresentou um uso de 8 células lógicas e 18 pinos do dispositivo reconfigurável (FPGA).

Figure 6: Print do resumo de funcionamento



Fonte: os autores

3.3 Funcionamento do circuito

Por fim, o circuito funcionou da forma esperada por os alunos, já que, ao ligarmos as chaves nas posições correspondentes aos valores binários (de 0000_2 a 1111_2) obtemos os símbolos correspondentes, em hexadecimal, no display da FPGA, conforme às imagens fig.7 e fig.8, que correspondem ao acendimento do display para as entradas $Sw = 4'0001$ e $Sw = 4'1111$, respectivamente.

Figure 7: Display formando número 1



Fonte: os autores

Figure 8: Display formando letra F



Fonte: os autores