



Prática N^o4 - SEL0606

Alunos:

Vitor Alexandre Garcia Vaz
Gabriel Dezejácomo Maruschi

Sumário

1	Introdução	3
1.1	<i>Objetivos</i>	3
1.2	<i>Multiplexador</i>	3
2	Materiais e métodos	3
2.1	<i>Obtenção das expressões booleanas</i>	3
2.2	<i>Implementação</i>	4
2.3	<i>Previsão sobre utilização do CI TTL 7400</i>	6
3	Conclusão	7
3.1	<i>Círculo RTL</i>	7
3.2	<i>Número de células lógicas</i>	8
3.3	<i>Funcionamento do circuito</i>	9

List of Figures

1	Representação do Multiplexador	3
2	Esquemático MUX 2-1	4
3	Esquemático MUX 3-1	4
4	Localização das chaves, botões e LEDS no DE10_lite	5
5	Código do MUX2-1 em VHDL	6
6	Montagem do MUX com o TTL7400	7
7	Círculo RTL sintetizado	8
8	Resumo de funcionamento	8
9	Acendimento de leds com botão pressionado	9
10	Acendimento dos leds sem o pressionamento do botão	10

1 Introdução

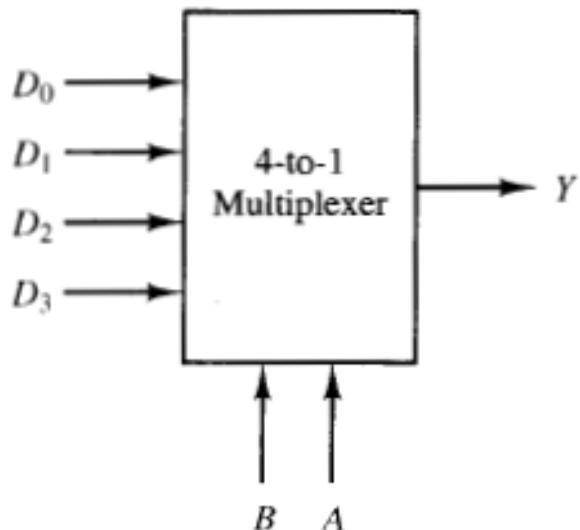
1.1 Objetivos

Nesta prática do Laboratório de Sistemas Digitais, implementamos um multiplexador (2 para 1), com entradas e saída de 4 bits na linguagem VHDL. Além disso, aplicamos o código na FPGA do kit DE10-Lite e testamos o funcionamento do circuito visualizando a resposta das entradas nos LEDS vermelhos, de acordo com as entradas nos chaves e no botão lateral. Também analisamos como seria a implementação do circuito utilizando o CI TTL7400 e comentamos a possibilidade de implementação do MUX3-1 de barramento de 4 bits.

1.2 Multiplexador

O Multiplexador é um circuito combinacional simples cuja função é selecionar, por meio de um sinal de controle, uma das entradas e rotear esta para a saída. O número máximo de entradas (i) é dependente do número de bits (n) pela fórmula $i = 2^n$.

Figure 1: Representação do Multiplexador



Fonte: Nelson, V. P., Nagle, H. T., Carroll, B. D., & Irwin, J. D. (1995). Digital Logic Circuit analysis and design

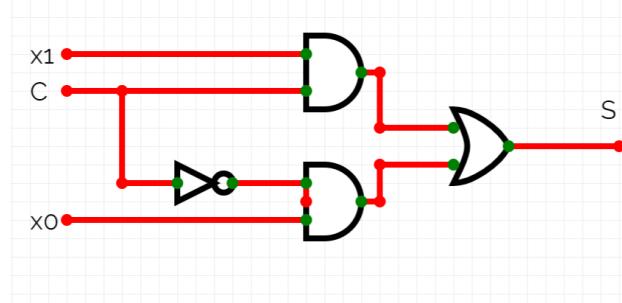
2 Materiais e métodos

2.1 Obtenção das expressões booleanas

Através das tabelas verdade do Multiplexador 2 para 1, obtivemos a expressão booleana genérica. Considere x_0 e x_1 entradas e 4 bits, C o sinal de controle de 1 bit e S a saída de 4 bits:

$$\bullet S = x_0 \bar{C} + x_1 C$$

Figure 2: Esquemático MUX 2-1

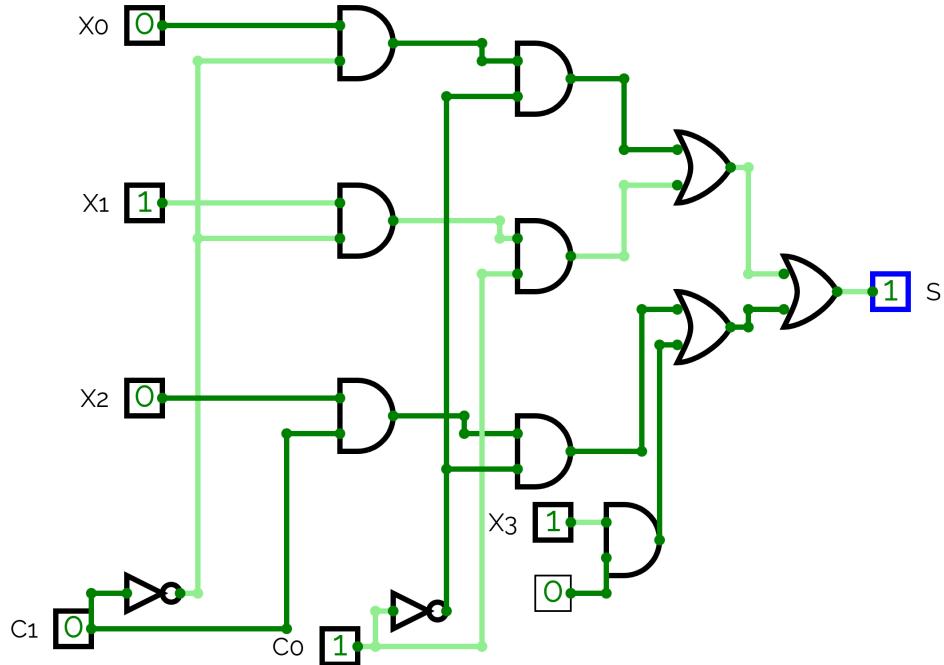


Fonte: Os autores

Ademais, por meio das tabelas verdade do Multiplexador 3 para 1, também chegamos a uma expressão booleana correspondente. Assim, considerando x_0 , x_1 e x_2 entradas de 4 bits, C_0 e C_1 sinais de controle (ambos de 1 bit) e saída S de 4 bits:

- $S = x_0 \bar{C}_1 \bar{C}_0 + x_1 \bar{C}_1 C_0 + x_2 C_1 C_0$

Figure 3: Esquemático MUX 3-1



Fonte: Os autores

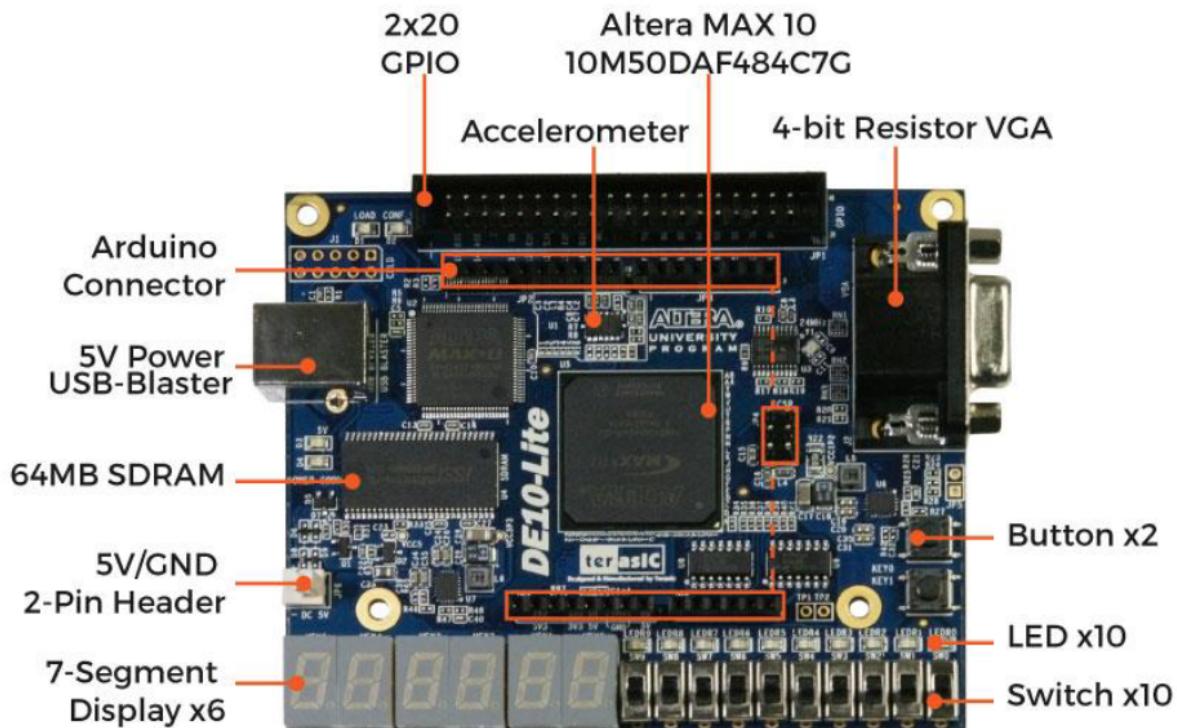
2.2 Implementação

Utilizando o mapeamento e barramento dos sinais descrito no manual do DE10-LITE, escrevemos o código em VHDL para o Multiplexador. As entradas SW(0) a SW(3) correspondem à entrada x_0 de 4 bits e as entradas SW(4) a SW(7) correspondem à entrada x_1 . São sinais provenientes das chaves. Além disso, o sinal de controle,

KEY(0) é proveniente do primeiro botão do lado inferior direito.

As saídas, por sua vez, foram transmitidas para LEDR(0) a LEDR(3), que espelham as chaves cujos sinais foram roteados de acordo com o sinal de controle lógico do MUX.

Figure 4: Localizacão das chaves, botões e LEDS no DE10_lite



Fonte: DE10_lite User Manual

Todo o código e a compilação foi feita no software Quartus, com a versão gratuita. A implementação valeu-se da estrutura lógica "WHEN-ELSE" da linguagem VHDL, na qual um sinal recebe outro de acordo com condicionais

Figure 5: Código do MUX2-1 em VHDL

```

1 --Copyright:
2 --Date: 02/10/24
3 --Version: 1.0
4 --Owners: Gabriel D. Maruschi
5 --          Vitor Alexandre Garcia Vaz
6
7 entity DE10_Lite_mux is
8   port (
9     SW    : in bit_vector(9 downto 0);
10    KEY   : in bit_vector(1 downto 0);
11    LEDR  : out bit_vector(9 downto 0)
12  );
13 end DE10_Lite_mux;
14
15 architecture concurrent of DE10_Lite_mux is
16
17   --Declaração de sinais internos
18
19 begin
20
21   LEDR(3 downto 0) <=
22     SW(3 downto 0) when KEY(0) = '0' else
23     SW(7 downto 4);
24
25
26
27 end concurrent;

```

Fonte: os autores

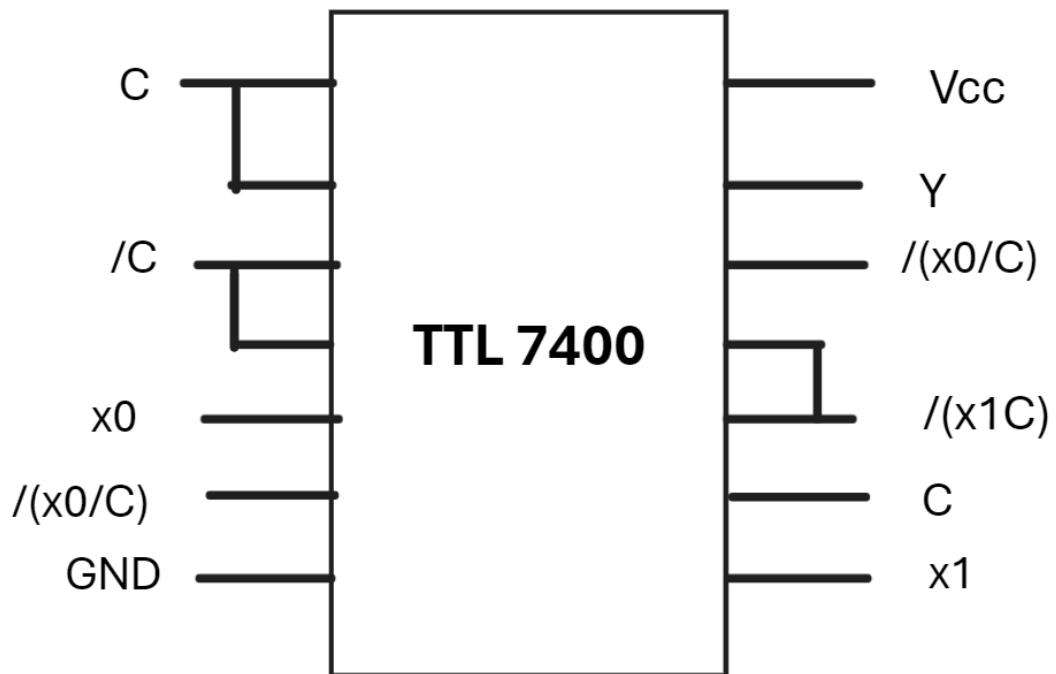
2.3 Previsão sobre utilização do CI TTL 7400

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

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

O montagem do MUX21 pode ser vizualizada abaixo:

Figure 6: Montagem do MUX com o TTL7400



Fonte: os autores

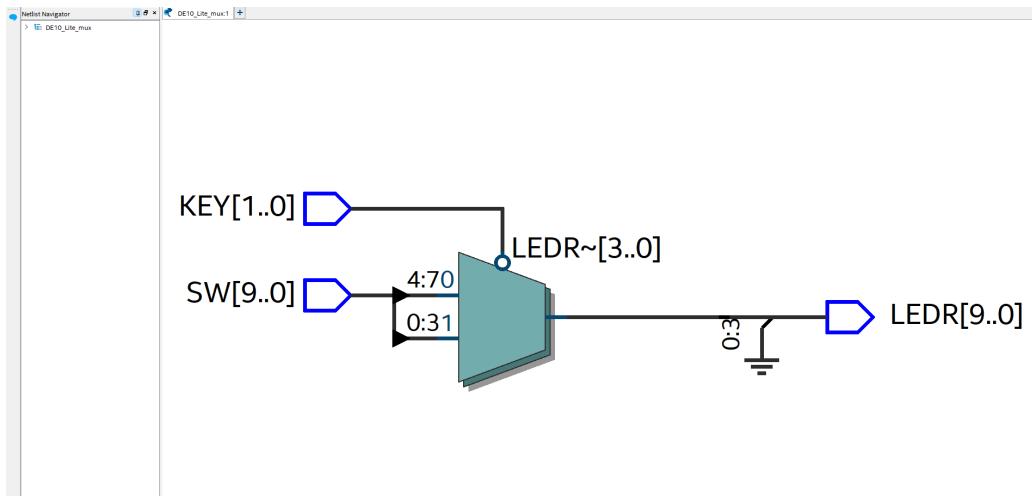
O equivalente para o MUX31, por outro lado, seria um pouco mais complexo, necessitando de no mínimo 7 CI's, precisando de 27 NANDS

3 Conclusão

3.1 Circuito RTL

A partir do código em VHDL (fig5) feito pela dupla, e usando a ferramenta de síntese disponibilizada pelo quartus, chegamos ao circuito da fig7.

Figure 7: Circuito RTL sintetizado



Fonte: os autores

3.2 Número de células lógicas

Ademais, por meio do resumo do funcionamento e constituição do circuito descrito em VHDL no software (fig8), chegamos à conclusão de que o circuito sintetizado apresentou um uso de 5 células lógicas e 22 pinos do dispositivo reconfigurável (FPGA).

Figure 8: Resumo de funcionamento

Table of Contents		Flow Summary	Compilation Report - DE10_Lite_mux
	Flow Summary		
	Flow Settings		
	Flow Non-Default Global Settings		
	Flow Elapsed Time		
	Flow OS Summary		
	Flow Log		
	Analysis & Synthesis		
	Filter		
	Flow Messages		
	Flow Suppressed Messages		
	Assembler		
	Timing Analyzer		
		<<Filter>>	
		Flow Status	Successful - Wed Oct 02 10:56:56 2024
		Quartus Prime Version	20.1.1 Build 720 11/11/2020 SJ Lite Edition
		Revision Name	DE10_Lite_mux
		Top-level Entity Name	DE10_Lite_mux
		Family	MAX 10
		Device	10M50DAF484C7G
		Timing Models	Final
		Total logic elements	5 / 49,760 (< 1 %)
		Total registers	0
		Total pins	22 / 360 (6 %)
		Total virtual pins	0
		Total memory bits	0 / 1,677,312 (0 %)
		Embedded Multiplier 9-bit elements	0 / 288 (0 %)
		Total PLLs	0 / 4 (0 %)
		UFM blocks	0 / 1 (0 %)
		ADC blocks	0 / 2 (0 %)

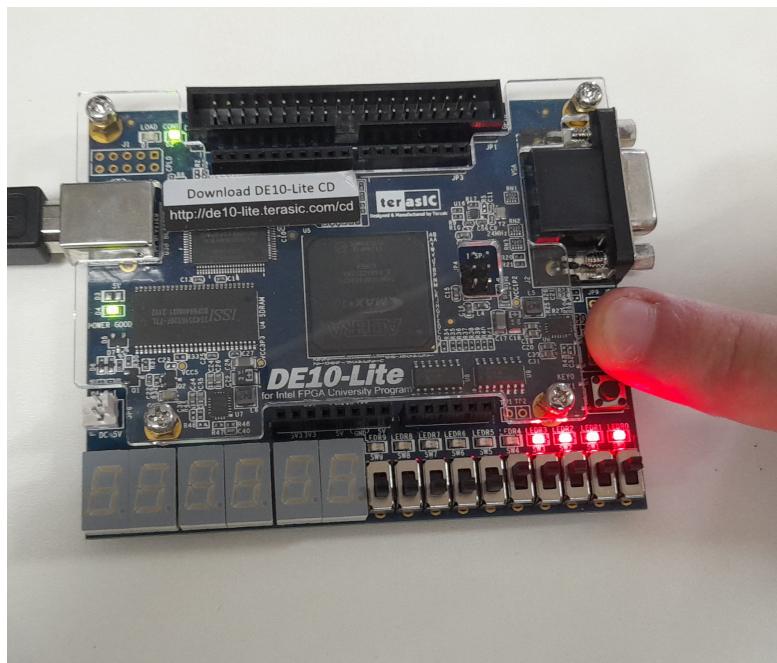
Fonte: os autores

3.3 Funcionamento do circuito

Nesse sentido, o circuito funcionou da forma esperada pelos alunos, já que, ao ligarmos as chaves $Sw(3 : 0)$ e $Sw(4 : 7)$ nas posições correspondentes aos valores binários de 1111_2 e 0001_2 , respectivamente, visualizamos o acendimento dos leds LEDR(3:0) pertencentes à FPGA, conforme às imagens fig.9 e fig.10, ou seja, houve um acendimento correspondente à entrada das chaves 0 a 3 para o botão pressionado (nível lógico 0) e correspondente à entrada das chaves 7 a 4 para o caso sem o pressionamento do botão (nível lógico 1).

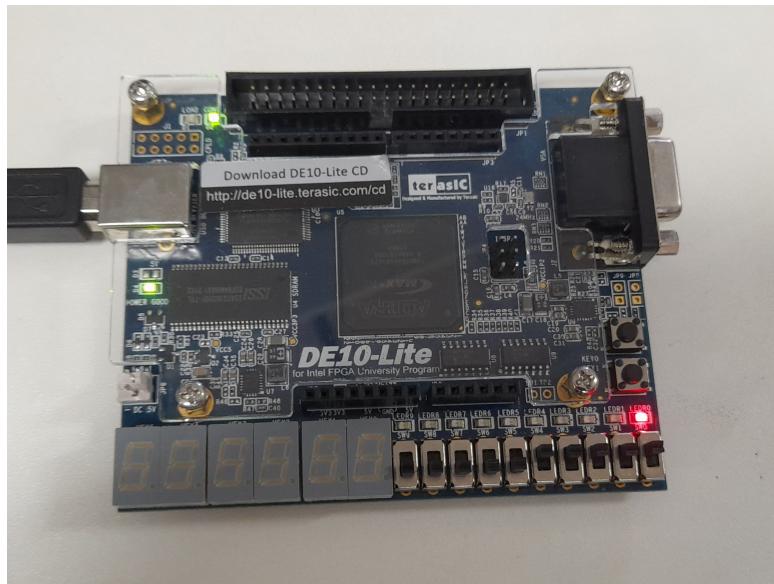
Por fim, tal comportamento mostrou-se de concordante com o código implementado para a elaboração do MUX2-1 envolvendo entradas e saída de 4 bits, fato o qual mostra a efetividade do uso de estruturas lógicas como "WHEN-ELSE" para a elaboração de códigos combinacionais que, como o multiplexador implementado, geram diferentes saídas de acordo com um sinal de seleção.

Figure 9: Acendimento de leds com botão pressionado



Fonte: os autores

Figure 10: Acendimento dos leds sem o pressionamento do botão



Fonte: os autores