Laboratório 1

Chaves, LEDs e Multiplexadores

O objetivo deste laboratório é aprender como conectar sinais simples de entrada e saída no chip FPGA e implementar um circuito que use estes dispositivos. Usaremos as chaves SW_{17-0} como entradas do circuito, e os LEDs como saída.

Parte I

A placa de desenvolvimento fornece 18 chaves, chamadas SW_{17-0}, que podem ser usadas como entradas para o circuito, e 18 LEDs vermelhos, chamados $LEDR_{17-0}$, que podem ser usados para mostrar valores de saída. A Figura 3 mostra uma entidade de VHDL simples que utiliza estas chaves e LEDs. Como existem 18 chaves e LEDs, é conveniente representá-los como vetores no código VHDL, como mostrado. Utilizamos uma única atribuição para todos as 18 saídas, o que é equivalente às atribuições individuais:

```
LEDR(17) <= SW(17);
LEDR(16) <= SW(16);
...
LEDR(0) <= SW(0);
```

A placa de desenvolvimento tem conexões físicas entre o chip FPGA e as portas e LEDs. Para usar os vetores SW_{17-0} e $LEDR_{17-0}$ é necessário incluir no seu projeto no Quartus II a atribuição correta de pinagem, conforme fornecido no manual da placa. Uma forma conveniente de fazer esta atribuição é através de arquivos de atribuição de pinagem (arquivo com extensão .qsf). Para tanto, é importante utilizar o menu Advanced Import Settings através do botão Advanced... na tela Import Assignments como mostra a Figura 1. Então, selecione Global assignments como mostrado na Figura 2 e pressione OK.

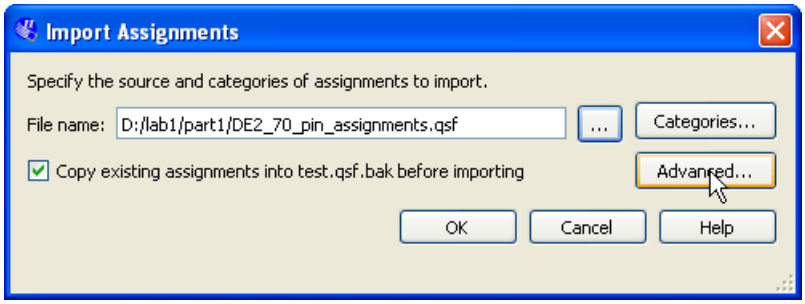


Figura 1. Janela de importação de atribuição.

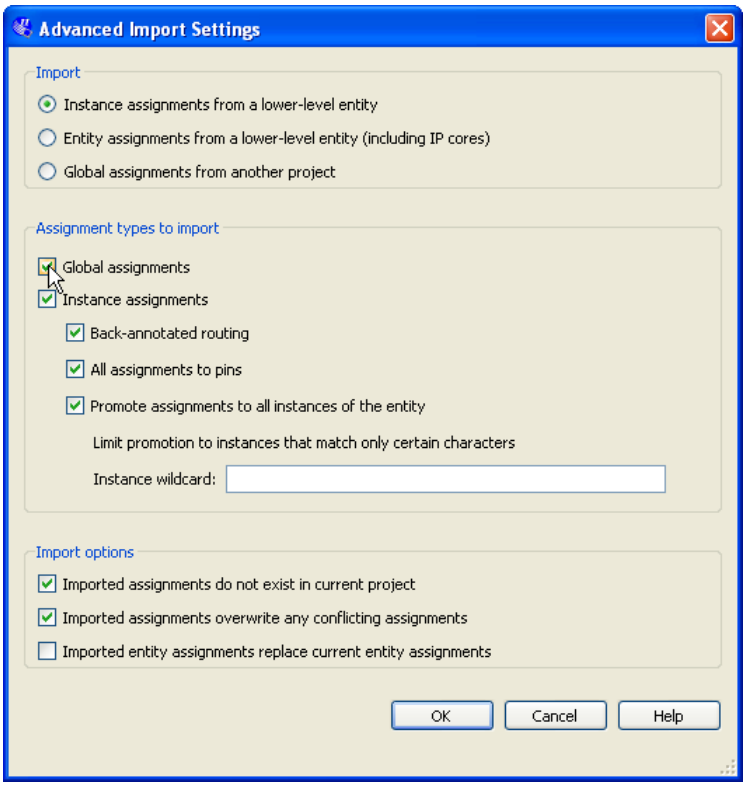


Figura 2. Janela de importação avançada de atribuição.

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

- - Módulo que conecta as chaves aos LEDs
ENTITY Exp1a IS
   PORT ( SW     : IN     STD_LOGIC_VECTOR(17 DOWNTO 0);
          LEDR : OUT    STD_LOGIC_VECTOR(17 DOWNTO 0)); - - red LEDs
END Exp1a;

ARCHITECTURE Behavior OF Exp1a IS
BEGIN
   LEDR <= SW;
END Behavior;
```

Figura 3. Código VHDL que usa as chaves e luzes da placa de desenvolvimento.

Execute os seguintes passos para implementar um circuito correspondente ao código da Figura 3.

1. Crie um novo projeto para o seu circuito (File > New Project Wizard) com o nome Exp1a. Selecione o dispositivo Cyclone IV GX EP4CGX150DF31C7.

2. Crie um novo arquivo do tipo VHDL (File/New/VHDL file), que deverá ser salvo com o nome Exp1a. Note que devem ter o mesmo nome: a pasta, o arquivo e a entidade principal do projeto. Edite o arquivo com o código da Figura 3.

3. Inclua no projeto a atribuição de pinagem como explicado.

4. Antes de compilar, acesse o menu (Assignments > Device), clique em Device and Pin Options, entre na opção Unused Pins, e selecione As input tri-state.

5. Compile o código (Processing > Start Compilation)

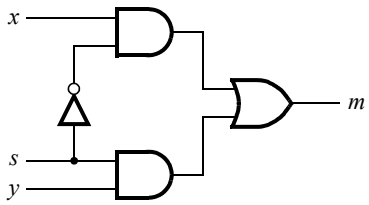
6. Para baixar o código para a placa de desenvolvimento, entre no menu (Tools > Programmer). Observe na tela do módulo Programmer que o campo Hardware Setup que deve mostrar *USB-Blaster*. Caso seja mostrado o valor No Hardware, clique em Hardware Setup e em seguida em USB-Blaster.

7. Verifique se em (File) aparece um arquivo com extensão sof. Caso contrário, clique em (Add File) e selecione o arquivo com essa extensão. Clique em Start e aguarde até completar.

8. Teste a funcionalidade do circuito alterando a posição das chaves e observando os LEDs.

Parte II

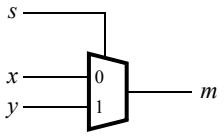
A Figura 4a mostra uma figura que implementa um *multiplexador* 2-para-1 com entrada de seleção s. Se $s = 0$ a saída do multiplexador m é igual à entrada x, e se $s = 1$ a saída é igual a y. A parte b da figura fornece a tabela verdade para este circuito, e a parte c mostra seu símbolo.



a) Circuit

s	m
0	x
1	y

b) Truth table



c) Symbol

Figura 4. Multiplexador 2-para-1.

O multiplexador pode ser descrito através da seguinte expressão em VHDL:

```
m <= (NOT (s) AND x) OR (s AND y);
```

Escreva uma entidade VHDL que inclua 8 atribuições como a fornecida para descrever o circuito da Figura 5a. Este circuito tem duas entradas de 8 bits, X e Y, e produz a saída de 8 bits. Se $s = 0$ então $M = X$, enquanto se $s = 1$ então $M = Y$. Nós denominamos este circuito como um multiplexador 2-para-1 de 8 bits. Seu símbolo é mostrado na Figura 5b, onde X, Y, e M são representados como barramentos de 8 bits. Efetue os passos abaixo.

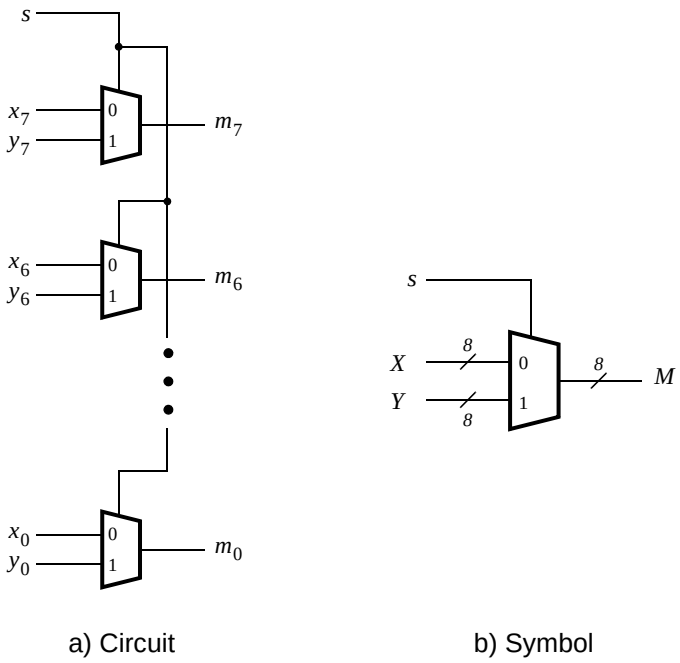


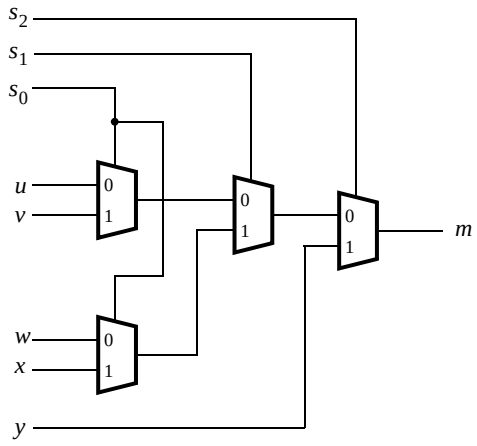
Figure 5. A eight-bit wide 2-to-1 multiplexer.

1. Crie um novo projeto Quartus II (Exp1b) para este circuito.
2. Implemente o circuito discutido. Use a chave SW_{17} como a entrada s, as chaves SW_{7-0} como entrada X e SW_{15-8} para a entrada Y. Conecte as chaves *SW* nos LEDs vermelhos *LEDR* e a saída M nos LEDs verdes $LEDG_{7-0}$.
3. Compile e teste a funcionalidade.
4. ***Apresente este projeto ao professor para validar a atividade em sala de aula.***

Parte III

Na Figura 4 mostramos um multiplexador 2-to-1 que seleciona entre duas entrdas *x* e *y*. Nesta parte, considere um circuito cuja saída *m* precisa ser selecionada entre cinco entradas u, v, w, x, e y. Parte a da Figura 6 mostra como podemos contruir este multiplexador 5-to-1 usando quatro multiplexadores 2-to-1. O circuito usa um sinal de entrada para seleção de 3 bits $s_2s_1s_0$ e implementa a tabela verdade mostrada na Figura 6b. O símbolo do circuito para este multiplexador é dado na parte c da figura.

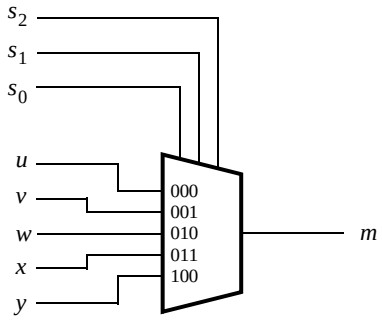
Relembre da Figura 5 que um multiplexador 2-para-1 de 8 bits pode ser construído usando oito instâncias de um multiplexador 2-para-1 de 1 bit. A Figura 7 aplica este conceito para definir um multiplexador 5-para-1 de 3 bits. Ele contém três instâncias do circuito na Figura 6a.



a) Circuit

s_2	s_1	s_0	m
0	0	0	u
0	0	1	v
0	1	0	w
0	1	1	x
1	0	0	y
1	0	1	y
1	1	0	y
1	1	1	y

b) Truth table



c) Symbol

Figura 6. Multiplexador 5-to-1.

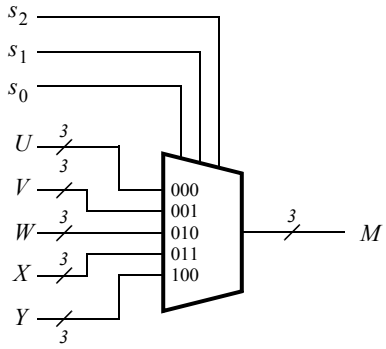


Figure 7. Multiplexador 5-to-1 de 3 bits.

Realize os seguintes passos para implementar o multiplexador 5-para-1 de 3 bits.

1. Crie um novo projeto Quartus II para este circuito.

2. Implemente o circuito discutido. Use as chaves SW_{17-15} como a entrada de seleção e as outras 15 chaves SW_{14-0} para gerar 5 entradas de 3 bits U até Y. Conecte as chaves *SW* nos LEDs vermelhos *LEDR* e a saída M nos LEDs verdes $LEDG_{2-0}$.

3. Compile e teste a funcionalidade.

4. Envie todos os arquivos de código desta parte para o Ensino Aberto.