Laboratório 3

Instruções:

Antes de iniciar o laboratório faça o download do arquivo 'lab03_material_v2018.1.zip' no moodle, ele contém todos as descrições de *entity* necessárias para implementar o circuito. A interface deve ser respeitada e isso será avaliado.

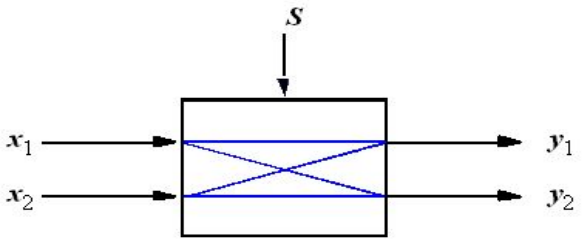
Exemplo do arquivo xbar_v1.vhd da questão 1 (a):

```
library ieee;
use ieee.std_logic_1164.all;

entity xbar_v1 is port(
    x1, x2, s: in std_logic;
    y1, y2: out std_logic
  );
end xbar_v1;

architecture rtl of xbar_v1 is begin
  -- code
end rtl;
```

Figura 1: Arquivo xbar_v1.vhd

Os arquivos devem ser enviados conforme a orientação de cada questão, nesse laboratório nenhuma entrega deve ser comprimida em qualquer formato, como *zip, tar, tar.gz*.

1. Seja o componente *xbar* que implementa um *crossbar switch* (a inversão só ocorre se o S estiver no no nível lógico alto, ou seja, se S = '1'). Projete os circuitos abaixo em VHDL e verifique o funcionamento de todos os



projetos com simulação.

a) Projete este circuito usando a construção WITH, SELECT e WHEN [**sem usar processo**].

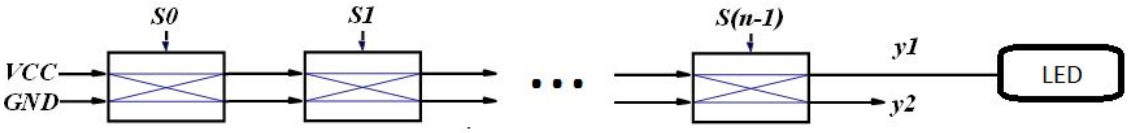
b) Projete este mesmo circuito usando a construção WHEN ELSE [**sem usar processo**].

c) Projete este mesmo circuito em VHDL usando a construção PROCESS.

d) A partir desse componente xbar implemente o circuito abaixo com número variável de estágios (utilize os comandos GENERIC e GENERATE). A simulação não é obrigatória.

e) Crie um novo projeto instanciando este componente com 5 estágios. Veja o netlist criado. Programe a placa para verificar o funcionamento, usando 5 switches – SW(0) até SW(4) - e um LED, sinal LEDR(0), como saída.

f) Repita (e) para 8 estágios, ou seja, sinal SW(0) até SW(7).



OBS – aplicação desse circuito: implementação de um número arbitrário de interruptores (S0..Sn-1) em um corredor, de modo que uma única mudança em qualquer deles muda o estado da iluminação (LED) de ligado para desligado ou vice-versa.

ENTREGAR:
a) VHDL (xbar_v1.vhd) e screenshot da simulação (xbar_v1.png).

b) VHDL (xbar_v2.vhd) e screenshot da simulação (xbar_v2.png).
c) VHDL (xbar_v3.vhd) e screenshot da simulação (xbar_v3.png).
d) VHDL (xbar_gen.vhd).
e) VHDL (xbar_stage_5.vhd) e screenshot da simulação (xbar_stage_5.png).
f) VHDL (lxbar_stage_8.vhd) e screenshot da simulação (xbar_stage_8.png).

2. A figura abaixo mostra um circuito multiplexador 4 para 1 projetado utilizando-se um decodificador 2 para 4 e portas lógicas. $w_{0..3}$ são as entradas, $s_{0..1}$ os sinais de seleção de entrada, *En* sinal para ligar e desligar o circuito (significa que todas as saídas do decodificador serão iguais à zero) e f a saída selecionada. Projete os circuitos abaixo em VHDL e verifique o funcionamento de todos os projetos com simulação.

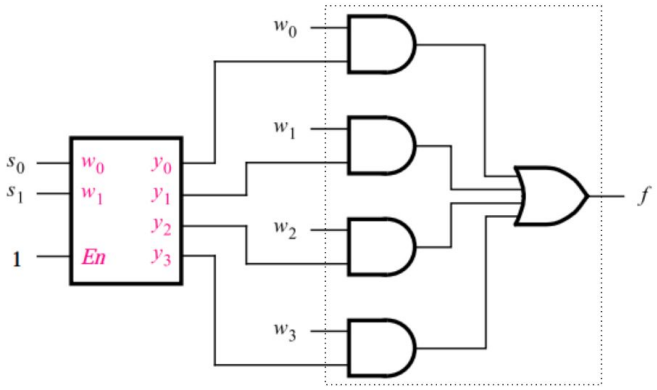


Figura 1: Mux 4-1 usando Dec2-4 e portas lógicas

a) Implemente o decodificador 2 para 4 da Figura 2 [**sem usar processo**].

b) Implemente o circuito dentro da caixa pontilhada na Figura 1, conforme o símbolo da Figura 3 [**sem usar processo**].

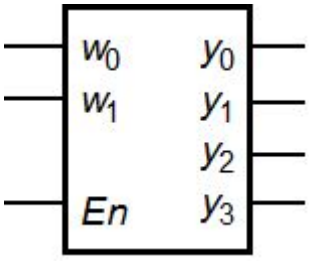


Figura 2: Dec2-4

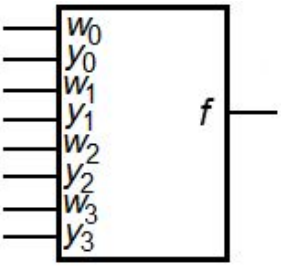


Figura 3: Lógica extra

c) A partir dos itens *a* e *b*, projete um multiplexador 4:1 como na Figura 1. Tabela de mapeamento do multiplexador:

Valor binário para a porta *sel*	**Saída esperada (porta selecionada)**
00	d0
01	d1

10	d2
11	d3

d) A partir do item c, implemente um multiplexador 16:1 em VHDL. Lembre-se que o quando o valor da entrada *sel* for igual a 0000, então a saída o bit menos significativo da porta *data*, ou seja *data*(0); e se sel for 1111, então a saída será o bit mais significativo, ou seja *data*(15). Não deve ser usado processo nem implementado de forma estrutural (utilizando portas lógicas).

ENTREGAR:
a) VHDL (dec2_to_4.vhd) e screenshot da simulação (dec2_to_4.png).
b) VHDL (extra_logic.vhd) e screenshot da simulação (extra_logic.png).
c) VHDL (mux4_to_1.vhd) e screenshot da simulação (mux4_to_1.png).
d) VHDL (mux16_to_1.vhd) e screenshot da simulação (mux16_to_1.png).