Instituto Federal de Santa Catarina
Área de Telecomunicações

SST20707– Síntese de Sistemas de Telecomunicações

Prof. Roberto de Matos
roberto.matos@ifsc.edu.br

São José, setembro de 2013.

Sistemas Digitais

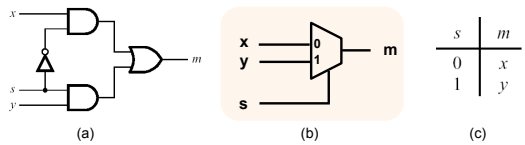
Circuito multiplexador - Mux

Objetivos do laboratório

1. Entender o conceito de "multiplexador".
2. Implementação de multiplexador em VHDL utilizando apenas funções booleanas (VHDL estrutural).
3. Implementação de multiplexador em VHDL utilizando *when* / *else* (VHDL comportamental).
4. Estudo de caso: uso de mux no projeto hierárquico do lab anterior.

Projeto de multiplexador - MUX 2x1

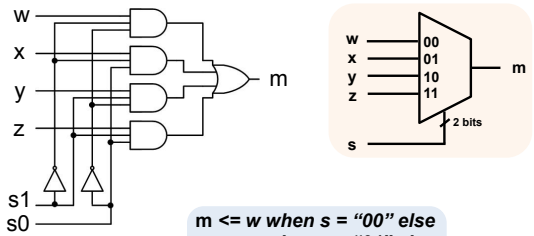
- **No circuito, se s = 0, a saída m será igual a entrada x.**
  Se s = 1, a saída m será igual a entrada y.



VHDL estrutural: *m <= (NOT (s) AND x) OR (s AND y)*

VHDL comportamental:
```
m <= x when s = '0' else
     y;
```


Projeto de multiplexador - MUX 4x1

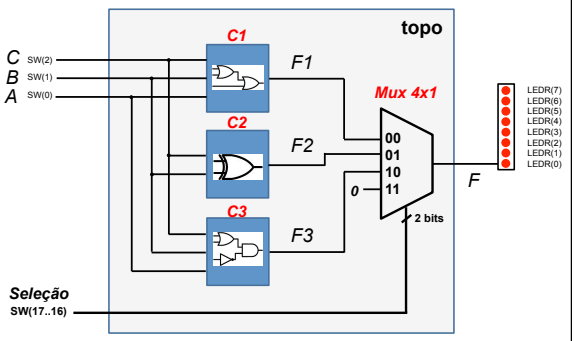



```
m <= w when s = "00" else
     x when s = "01" else
     y when s = "10" else
     z;
```

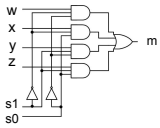
Tarefa a ser realizada na aula prática

PARTE I – Mux 4x1 em VHDL estrutural
PARTE II – Mux 4x1 em VHDL comportamental

Reutilizar os arquivos do lab anterior, e realizar as alterações indicadas a seguir (Mux 4x1 no lugar de C4):



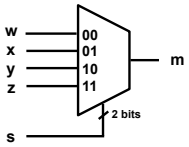
- **PARTE I – Projetar e implementar o MUX em VHDL estrutural:**

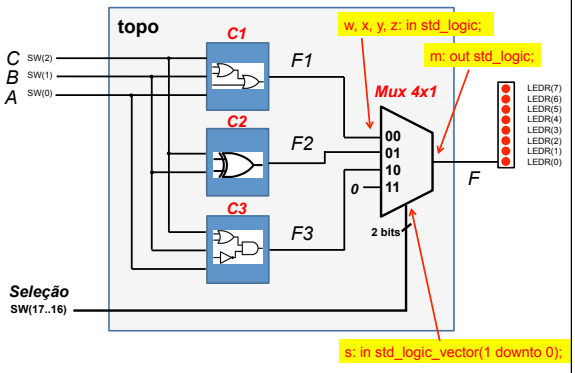



```
m <= (w and ((NOT (s1) AND (NOT(s0))) OR ...
```

- **PARTE II – Projetar e implementar o MUX em VHDL comportamental:**

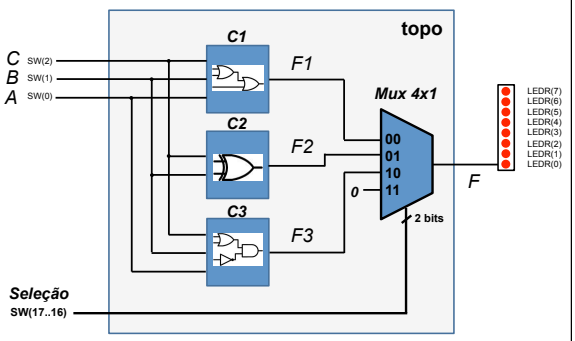
```
m <= w when s = "00" else
     x when s = "01" else
     y when s = "10" else
     z;
```



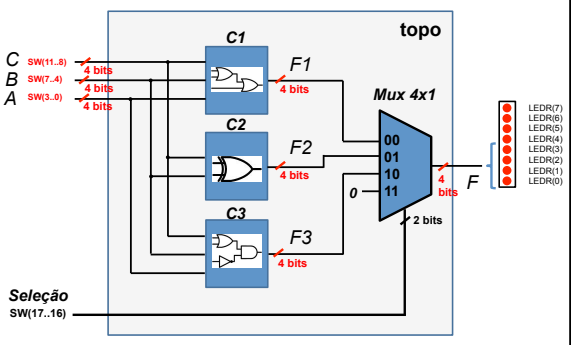


Barramentos em VHDL - "std_logic_vector"

No circuito do lab anterior, as entradas (A, B, C) e saídas (F) possuem o tamanho de 1 bit:



Assumir o mesmo circuito, porém com entradas e saídas de 4 bits:



Utilizando operandos de 4 bits

- Componente C1 realiza a operação *F1 = A or B or C*
- Componente C2 realiza a operação *F2 = B xor C*
- *Operandos de 1 bit - ex. A = 0, B = 1, C = 0*
- *Operandos de 4 bits - ex. A = 0101, B = 1000, C = 0001*

1 bit			4 bits		
0			0101		
1		0	1000		1000
0		1	0001		0001
F1 = 1		F2 = 1	F1 = 1101		F2 = 1001

Implementação de barramento em VHDL

std_logic ➔ fio

std_logic_vector ➔ vários fios (barramento)

- *std_logic_vector* é utilizado para implementar um "vetor" de sinais (vários fios, barramento).
- Por exemplo, a saída F deverá ser definida na *entity* como:

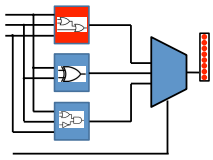
F: out std_logic_vector(3 downto 0);

Componente C1

```
library IEEE;
use IEEE.Std_Logic_1164.all;

entity C1 is
port (A: in std_logic_vector(3 downto 0);
      B: in std_logic_vector(3 downto 0);
      C: in std_logic_vector(3 downto 0);
      F: out std_logic_vector(3 downto 0)
     );
end C1;

architecture c1_estr of C1 is
begin
  F <= A or B or C;
end c1_estr;
```

4 bits

4 bits

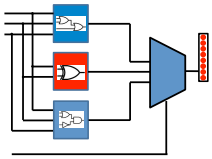


Componente C2

```
library IEEE;
use IEEE.Std_Logic_1164.all;

entity C2 is
port (A: in std_logic_vector(3 downto 0);
      B: in std_logic_vector(3 downto 0);
      F: out std_logic_vector(3 downto 0)
     );
end C2;

architecture c2_estr of C2 is
begin
  F <= A xor B;
end c2_estr;
```

4 bits

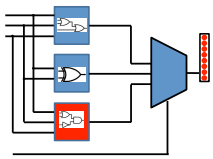
4 bits



Componente C3

```
library IEEE;
use IEEE.Std_Logic_1164.all;

entity C3 is
port (A: in std_logic_vector(3 downto 0);
      B: in std_logic_vector(3 downto 0);
      C: in std_logic_vector(3 downto 0);
      F: out std_logic_vector(3 downto 0)
     );
end C3;

architecture c3_estr of C3 is
begin
  -- ver lab. sobre componentes
end c3_estr;
```

4 bits

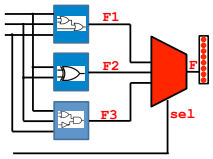


Componente Mux

```
library IEEE;
use IEEE.Std_Logic_1164.all;

entity Mux is
port (w: in std_logic_vector(3 downto 0);
      x: in std_logic_vector(3 downto 0);
      y: in std_logic_vector(3 downto 0);
      z: in std_logic_vector(3 downto 0);
      s: in std_logic_vector(1 downto 0);
      m: out std_logic_vector(3 downto 0)
     );
end Mux;

architecture mux_bhv of Mux is
begin
   -- ver lab. sobre Mux
end mux_bhv;
```

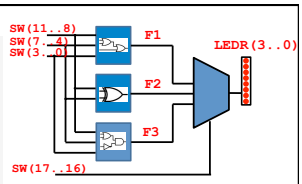
4 bits

2 bits



Componente Topo

```
library  ieee;
use  ieee.std_logic_1164.all;
entity topo is
 port ( SW : IN STD_LOGIC_VECTOR(17 downto 0);
        LEDR : OUT STD_LOGIC_VECTOR(17 downto 0)
 );
end topo;
architecture topo_estru of topo is
  signal F1, F2, F3: std_logic_vector(3 downto 0);
  component C1
    port (A : in std_logic_vector(3 downto 0);
          B : in std_logic_vector(3 downto 0);
          C : in std_logic_vector(3 downto 0);
          F : out std_logic_vector(3 downto 0));
  end component;
  component C2
    port (A : in std_logic_vector(3 downto 0);
          B : in std_logic_vector(3 downto 0);
          F : out std_logic_vector(3 downto 0));
  end component;
component C3
    port (A : in std_logic_vector(3 downto 0);
          B : in std_logic_vector(3 downto 0);
          C : in std_logic_vector(3 downto 0);
          F : out std_logic_vector(3 downto 0)
    );
  end component;
  -- INCLUIR AQUI O Mux
```

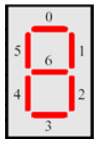


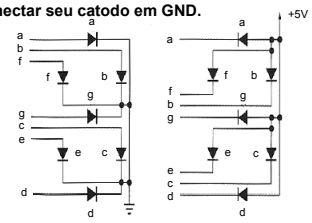

```
begin
  L0: C1 port map (SW(3 downto 0),
        SW(7 downto 4), SW(11 downto 8), F1);
  L1: C2 port map (SW(3 downto 0),
        SW(7 downto 4), SW(11 downto 8), F2);
  L2: C3 port map (SW(3 downto 0),
        SW(7 downto 4), SW(11 downto 8), F3);
  L3: Mux port map (F1, F2, F3,
        SW(17 downto 16), LEDR(3 downto 0));
end topo_estru;   -- END da architecture
```

Decodificadores em VHDL

Display de 7-segmentos

- Um display de 7-segmentos é composto por sete LEDs que podem ser ligados ou desligados de forma independente.
- Catodo comum - terminais catodo dos LEDs estão conectados a GND, e cada LED é ligado ao conectar seu anodo em Vcc.
- Anodo comum - terminais anodo dos LEDs estão conectados a Vcc, e cada LED é ligado ao conectar seu catodo em GND.

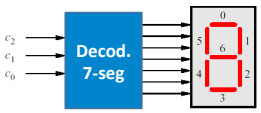




Catodo comum Anodo Comum

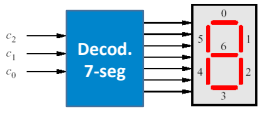
Decodificador de binário para 7-segmentos

- Placa DE2 possui 7 displays de 7-segmentos, todos do tipo anodo comum (LEDs acendem com GND, ou zero lógico).
- Para escrever um valor binário em um dos displays, é preciso realizar uma conversão do código binário para o código 7-segmentos.
- Os 3 bits de entrada do circuito "Decod. 7-seg" são *decodificados*, e a palavra de 7 bits gerada é enviada para o display de 7 segmentos.



Projeto de decodificador de binário para 7-segmentos

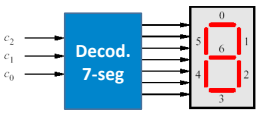
- Exemplos de valores em binários convertidos (decodificados) para 7-segmentos, visando escrita no display da placa DE2 anodo comum)



C2	C1	C0	7 bits *6543210*	Letra
0	0	0	1001111	I
0	0	1	0001110	F
0	1	0	0010010	S
0	1	1	1000110	C
1	1	1	1111111	

- Nesse exemplo, ao receber "000" na entrada, o decodificador gera o código equivalente ao acendimento da letra "I" no display 7-seg.
- Ao receber "111", todos os segmentos são desligados.
- Notar que por ser do tipo anodo comum, um "0" liga um segmento.

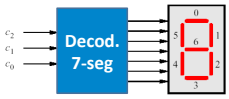
Projeto de decodificador "binário para 7-segmentos"



C2	C1	C0	7 bits 6543210	Letra
0	0	0	1001111	I
0	0	1	0001110	F
0	1	0	0010010	S
0	1	1	1000110	C
1	1	1	1111111	

- Um circuito para implementar a lógica do decodificador em questão poderia ser projetado utilizando vários métodos:
  - Soma de produtos:
    F(0) = C2' C1' C0' + C2C1C0
    F(1) = C2' C1' C0 + C2' C1C0' + C2' C1C0 + C2C1C0
    F(2) = ...
  - Análise comportamental:
    F = "1000001" quando C2C1C0 = "000" senão
    "0001110" quando C2C1C0 = "001" senão
    ... senão
    "1111111"

Componente Decod_IFSC

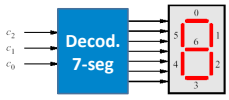



```
library IEEE;
use IEEE.Std_Logic_1164.all;

entity decodIFSC is
port (C:  in std_logic_vector(2 downto 0);
      F:  out std_logic_vector(6 downto 0)
     );
end decodIFSC;

architecture decod_bhv of decodIFSC is
begin
  F <= "1001111" when C = "000" else    -- I
       "0001110" when C = "001" else    -- F
       "0010010" when C = "010" else    -- S
       "1000110" when C = "011" else    -- C
       "1111111";
end decod_bhv;
```

3 bits

7 bits

Componente Decod_IFSC




```
library IEEE;
use IEEE.Std_Logic_1164.all;

entity decodIFSC is
port (C:  in std_logic_vector(2 downto 0);
      F:  out std_logic_vector(6 downto 0)
     );
end decodIFSC;

architecture decod_bhv of decodIFSC is
Begin
  with C select
  F <= "1001111" when "000",   -- I
       "0001110" when "001",   -- F
       "0010010" when "010",   -- S
       "1000110" when "011",   -- C
       "1111111" when others;
end decod_bhv;
```

3 bits

7 bits