



FEDERAL UNIVERSITY
OF SANTA CATARINA

Laboratório 4: Atribuições Condicionais e de Seleção

EEL5105 – Circuitos e Técnicas Digitais

Objetivos

- Entender o uso de **atribuições condicionais** (usando estrutura **when/else**) e **atribuições de seleção** (usando estrutura **with/select**) para facilitar o projeto de circuitos em **VHDL**.
- Trabalhar os conceitos de **multiplexador** e **decodificador**.
- Fazer **implementações** visando fixar os conceitos estudados.

Introdução

- **When/else**: permite a **atribuição condicional** de valores a um **sinal**.
 - Facilita o projeto, pois permite a descrição de um **circuito** de forma análoga à sua **tabela verdade**, sem a necessidade de **descrição booleana**.
 - **Não** implica execução **sequencial** de código.
 - Facilita também a **interpretação** do código.
- **Exemplo:**

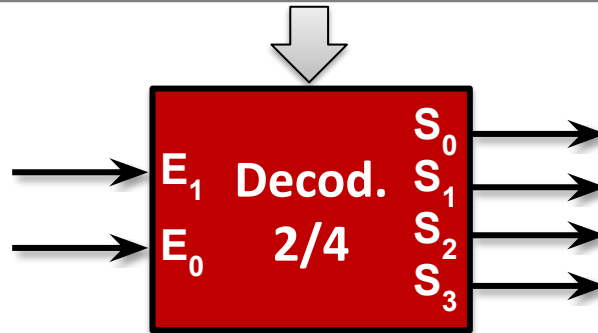
```
saida <= "0001" when entrada = "00" else  
        "0010" when entrada = "01" else  
        "0100" when entrada = "10" else  
        "1000";
```

Introdução

- **When/else**: permite a **atribuição condicional** de valores a um **sinal**.

• **Exemplo:**

```
saida <= "0001" when entrada = "00" else  
      "0010" when entrada = "01" else  
      "0100" when entrada = "10" else  
      "1000";
```



- Código **estrutural convencional**:

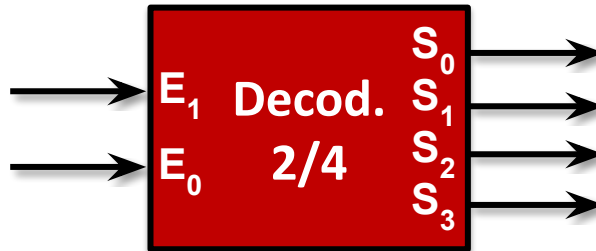
```
saida(0) <= (NOT entrada(1)) AND (NOT entrada(0));  
saida(1) <= (NOT entrada(1)) AND entrada(0);  
saida(2) <= entrada(1) AND (NOT entrada(0));  
saida(3) <= entrada(1) AND entrada(0);
```

Introdução

- **With/select**: *selected signal assignment*

Outra forma de atribuição em **VHDL** que permite a **seleção de valores para um sinal** baseado no valor de **outro sinal**.

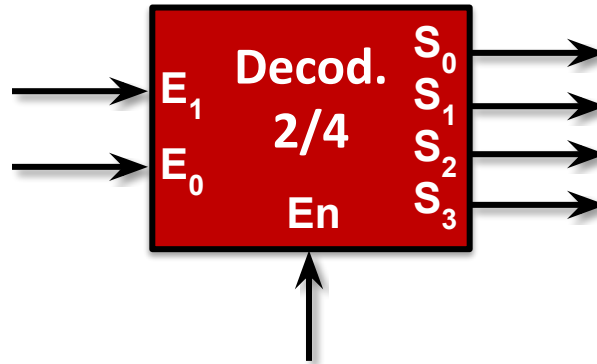
- Também não envolve execução sequencial de código.
- **Exemplo:**



```
with entrada select saida <= "0001" when "00",  
                             "0010" when "01",  
                             "0100" when "10",  
                             "1000" when others;
```

Introdução

- **When/Else** é um pouco mais flexível que **With/Select**
 - Exemplo: **Decodificador 2/4 com Enable**



```
saida <= "0000" when En = '0' else  
         "0001" when entrada = "00" else  
         "0010" when entrada = "01" else  
         "0100" when entrada = "10" else  
         "1000";
```

Introdução

- Exemplo 2:

```
saida <= w when sel = "00" else  
         x when sel = "01" else  
         y when sel = "10" else  
         z;
```

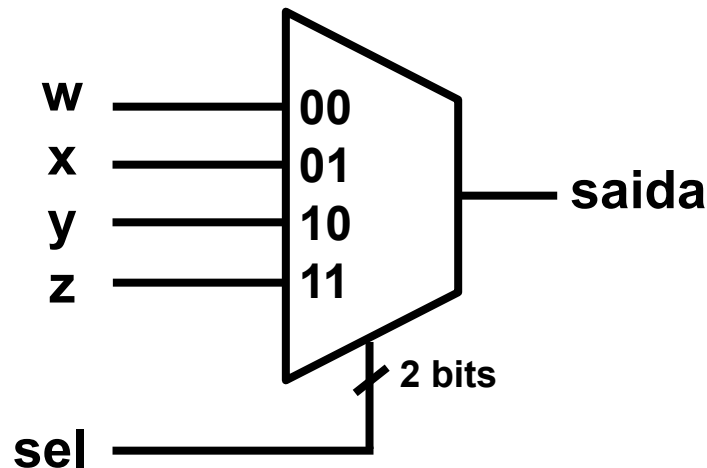
```
with entrada select saida <= w when "00",  
                                x when "01",  
                                y when "10",  
                                z when others;
```

Introdução

- Exemplo 2: **Multiplexador com 4 entradas**

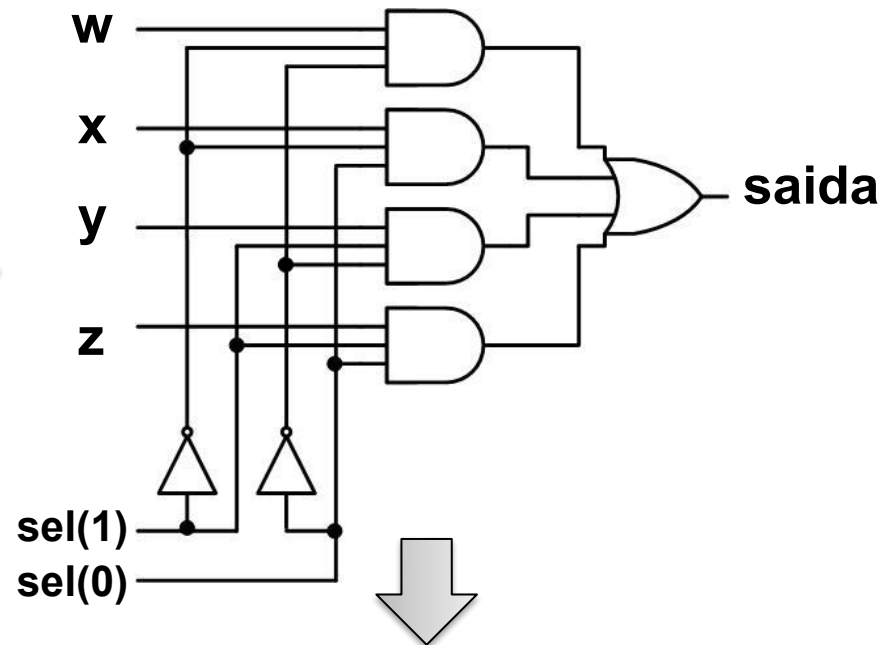
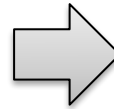
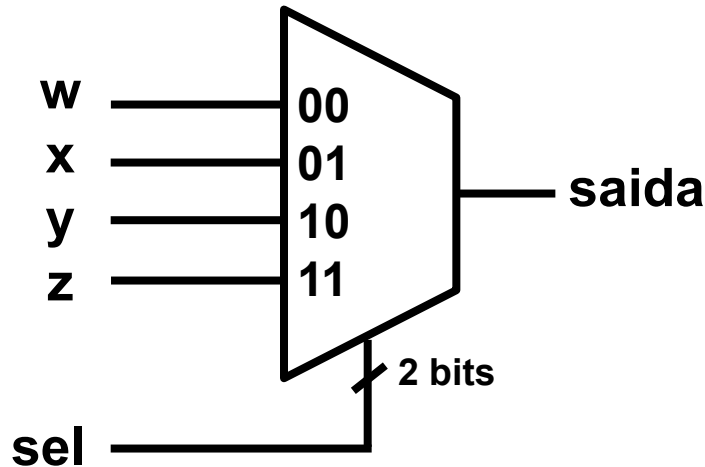
```
saida <= w when sel = "00" else  
        x when sel = "01" else  
        y when sel = "10" else  
        z;
```

```
with entrada select saida <= w when "00",  
                           x when "01",  
                           y when "10",  
                           z when others;
```



Introdução

- **Exemplo 2: multiplexador com 4 entradas**
 - Projeto com **when/else** ou **with/select** é bem mais simples do que o convencional:

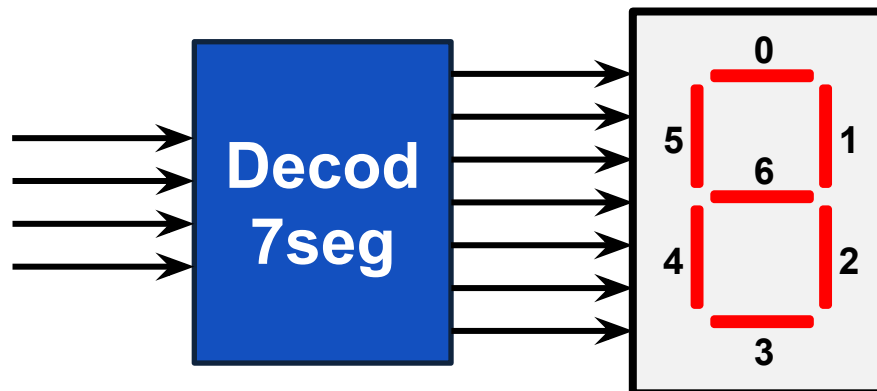


VHDL estrutural convencional:

```
saida <= (w and (not sel(1)) and (not sel(0))) or  
(x and (not sel(1)) and sel(0)) or  
(y and sel(1) and (not sel(0))) or  
(z and sel(1) and sel(0));
```

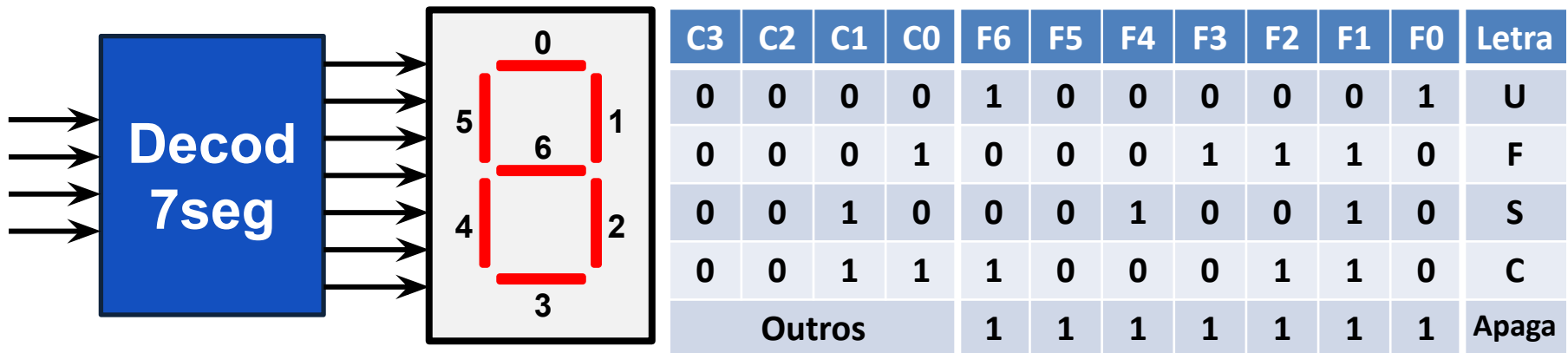
Introdução

- **Exemplo 3: decodificador para display de 7 segmentos**
 - O kit **DE2** possui 8 **displays** de 7 segmentos, todos do tipo anodo comum (**LEDs acendem com zero lógico**).
 - Para apresentar um número ou caractere em um display, é preciso realizar uma **conversão** para o código de 7 segmentos.
 - Para isso, utiliza-se um decodificador para 7 segmentos:



Introdução

- **Exemplo 3: decodificador para display de 7 segmentos**
 - Decodificador **UFSC** para 7-segmentos, visando escrita no display anodo comum do **DE2**
 - Anodo comum: segmento acende com nível lógico baixo



- Neste exemplo:
 - “0000” desenha “U” no display 7-seg.
 - Etc...
 - Qualquer valor maior que “0011”: todos os segmentos são desligados.

Introdução

- Decodificador **UFSC** (when/else):

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

entity decodUFSC is
port (C:  in std_logic_vector(3 downto 0);
      F:  out std_logic_vector(6 downto 0)
      );
end decodUFSC;

architecture decod of decodUFSC is
begin
    F <= "1000001" when C = "0000" else -- U
        "0001110" when C = "0001" else -- F
        "0010010" when C = "0010" else -- S
        "1000110" when C = "0011" else -- C
        "1111111";
end decod;
```

4 bits

7 bits

Introdução

- Decodificador **UFSC** (with/select):

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

entity decodUFSC is
port (C:  in std_logic_vector(3 downto 0);
      F:  out std_logic_vector(6 downto 0)
      );
end decodUFSC;

architecture decod of decodUFSC is
begin
  with C select
    F <= "1000001" when "0000", -- U
        "0001110"  when "0001", -- F
        "0010010"  when "0010", -- S
        "1000110"  when "0011", -- C
        "1111111"  when others;
end decod;
```

4 bits

7 bits

Tarefas

Tarefa 1

- Implementar um circuito decodificador de **4 bits** para **display de 7 segmentos**.



Entrada	Saída 6543210	Display
0000	1000000	0
0001	1111001	1
0010	0100100	2
0011	0110000	3
0100	0011001	4
0101	0010010	5
0110	0000010	6
0111	1111000	7
1000	0000000	8
...	...	9, A, b, C, d
1110	0000110	E
1111	0001110	F

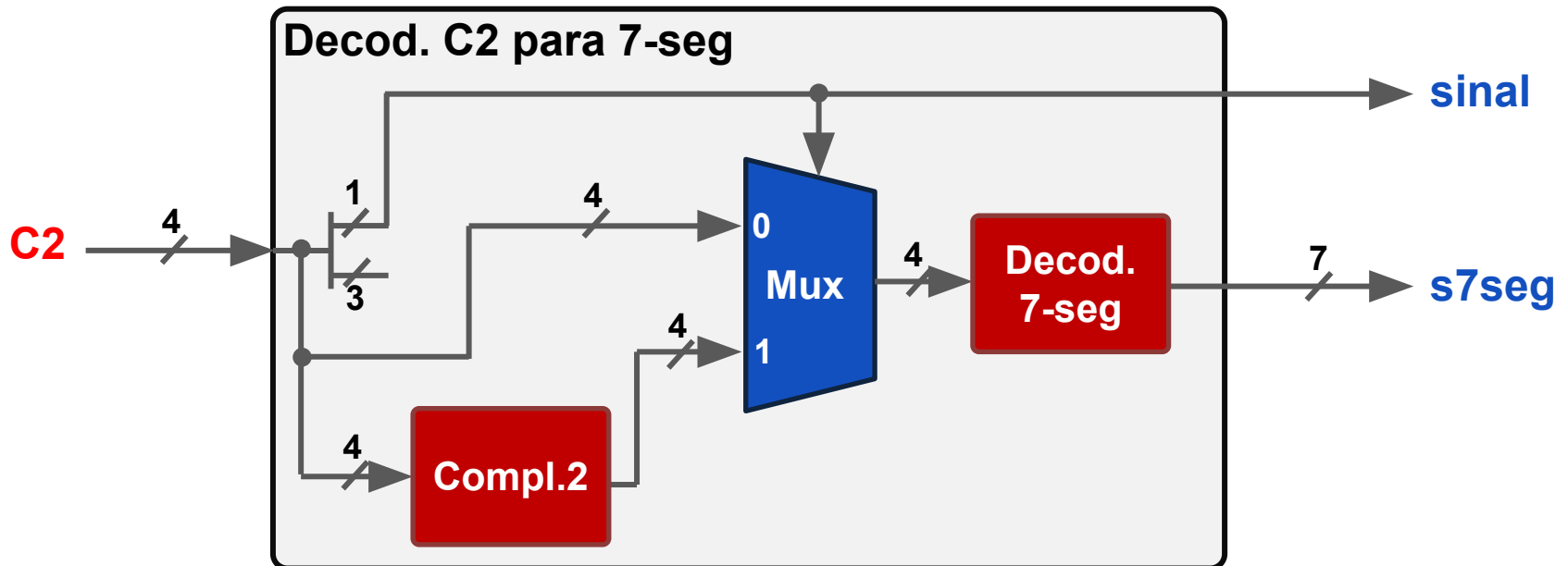
Tarefa 2

- Implementar um circuito decodificador de **Complemento de 2** com **4 bits** para **display de 7 segmentos**
 - Se o número de entrada for positivo, mostrar no display.
 - Se o número for negativo, calcular o seu complemento de 2 e mostrar no display.



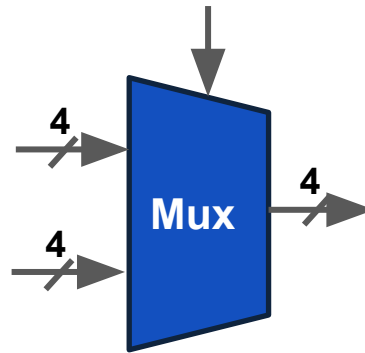
Tarefa 2

- Implementar um circuito decodificador de **Complemento de 2** com **4 bits** para **display de 7 segmentos**
 - Se o número de entrada for positivo, mostrar no display.
 - Se o número for negativo, calcular o seu complemento de 2 e mostrar no display.



Tarefa 2

- **Passo 1:** Criar arquivo **VHDL** com a descrição do **Multiplexador 2x1** para barramentos de 4 bits.
 - **Atenção:** nome do **arquivo** deve ser igual ao da **entity**



Tarefa 2

- **Passo 2:** Criar arquivo do tipo **VHDL** com a descrição do decodificador de **binário para 7 segmentos (hex)**

Decod.
7-seg

- **Dica:** usar **Tarefa 1**.

Entrada	Saída 6543210	Display
0000	1000000	0
0001	1111001	1
0010	0100100	2
0011	0110000	3
0100	0011001	4
0101	0010010	5
0110	0000010	6
0111	1111000	7
1000	0000000	8
...	...	9, A, b, C, d
1110	0000110	E
1111	0001110	F

Tarefa 2

- **Passo 3:** Criar arquivo do tipo **VHDL** com a descrição do bloco que calcula o **Complemento 2** de um número binário.



- Usar código **pronto**, o qual faz **inversão bit a bit** e **soma 1**:

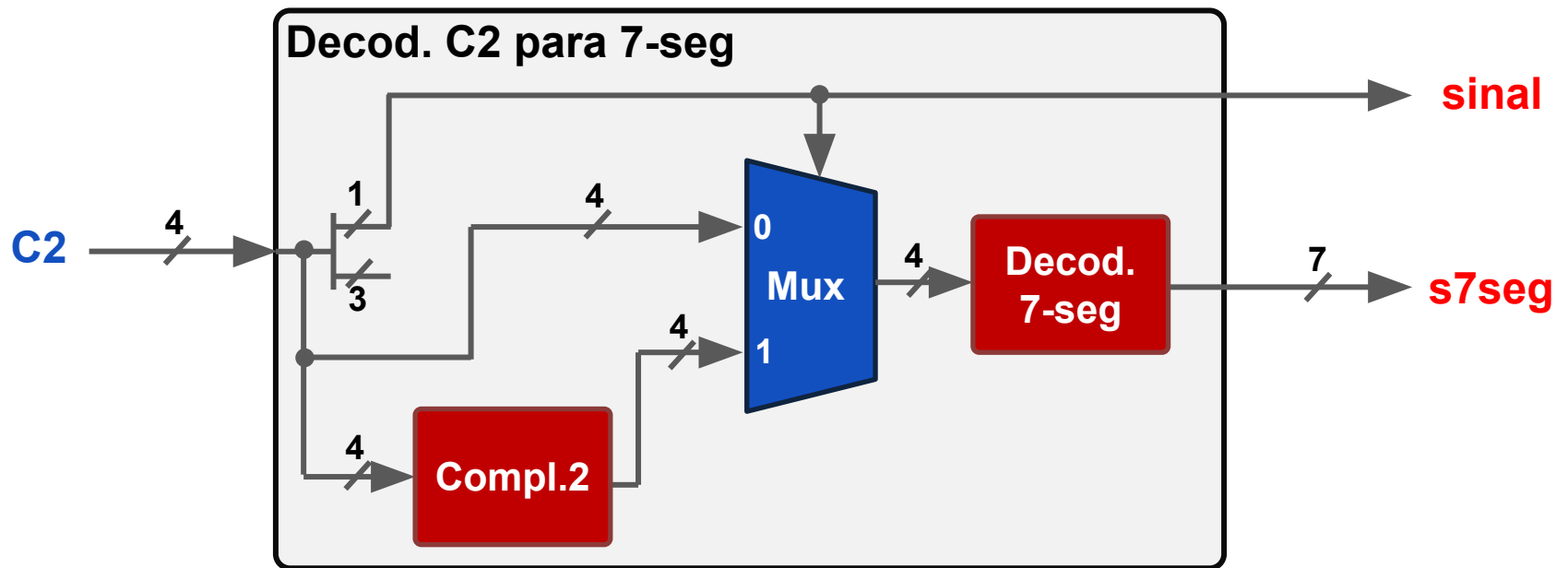
```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_unsigned.all;

entity Compl2 is
port (X:  in std_logic_vector(3 downto 0);
      Y:  out std_logic_vector(3 downto 0));
end Compl2;

architecture c22 of Compl2 is
begin
    Y <= (not X) + "0001";
end c22;
```

Tarefa 2

- **Passo 4:** Criar o **VHDL** do elemento de topo (**C2to7seg.vhd**) da arquitetura e conectar os diferentes blocos



Tarefa 2

- **Passo 5:** Conectar **C2to7seg.vhd** com o kit e testar o seu funcionamento.
 - Usar **SWs** como entradas, além de **LEDR** e **HEX0** como saídas.

```
SW: in std_logic_vector(17 downto 0)
LEDR: out std_logic_vector(17 downto 0)
HEX0: out std_logic_vector(6 downto 0)
HEX1: out std_logic_vector(6 downto 0)
```

- Opções:
 - Criar um arquivo **usertop.vhd** com as ports descritas acima, fazer declaração de component **C2to7seg** e **port maps** adequados.
 - Mudar nomes das ports de **C2to7seg.vhd** adequadamente e da entity/arquivo pra **usertop**.