Sistemas Digitais

P02 – Multiplexadores e bit_vector

Simulando Multiplexadores

MULTIPLEXADOR

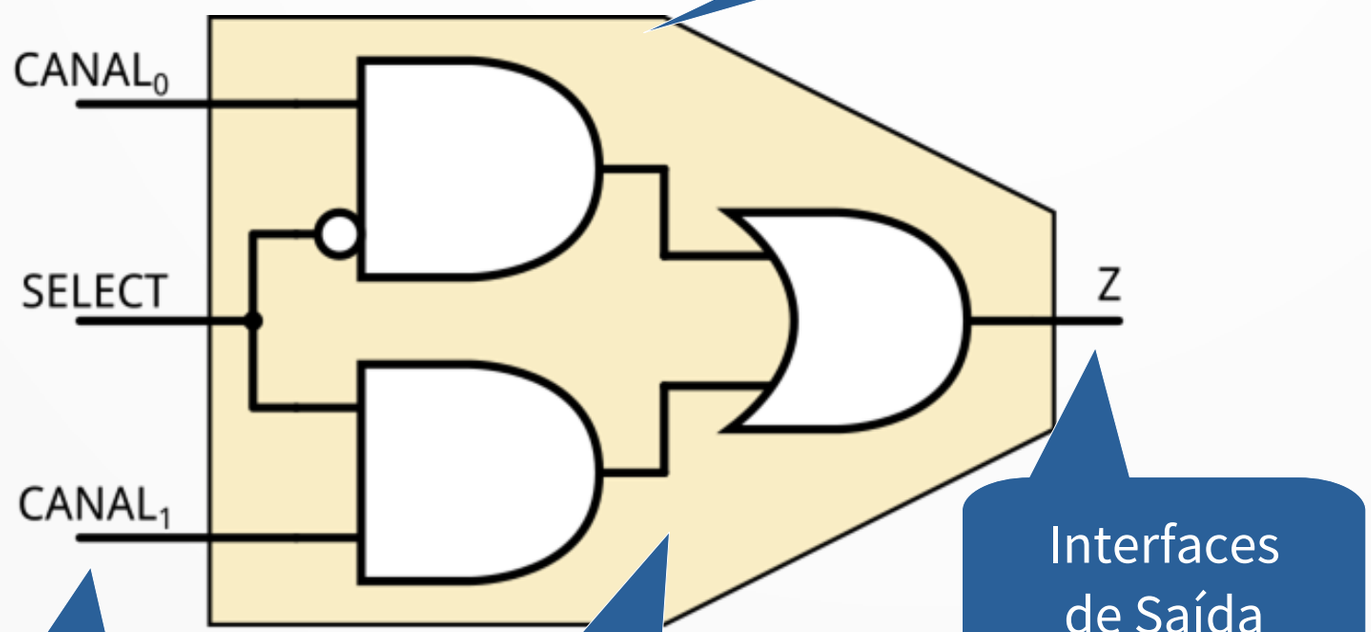
descrevendo o hardware comutador

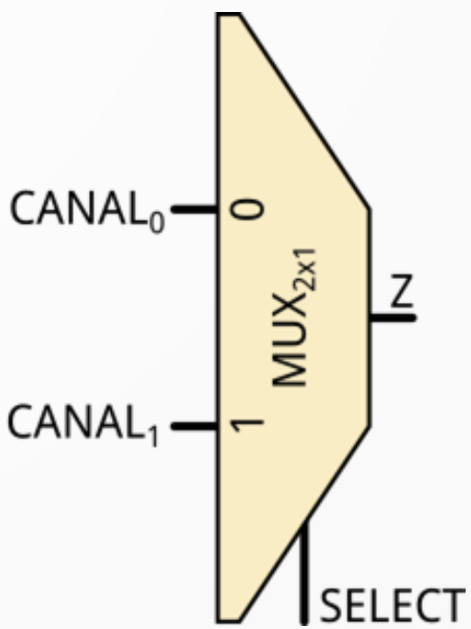
Multiplexador

- MUX_{2x1}
  - Entradas de 1 bit
- $Canal_0$ → C0
- $Canal_1$ → C1
- Seletor → SELECT
  - Saída de 1 bit
- Z

Multiplexador

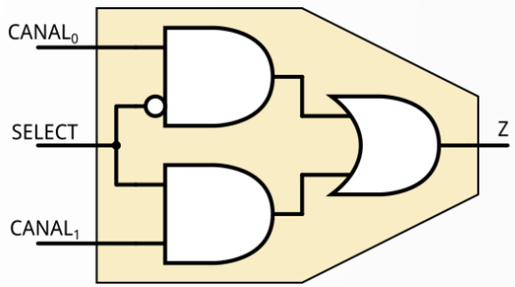
- MUX_{2x1}

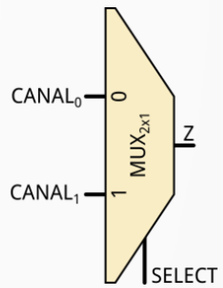




Multiplexador

- MUX_{2x1}





ENTIDADE:

```
entity mux2x1 is

port(

    c0  : in  bit;

    c1  : in  bit;

    sel : in  bit;

    z   : out bit;

);

end entity;
```

COMPORTAMENTO:

```
architecture comuta of mux2x1 is

begin

   z <= (c0 and not(sel)) or

        (c1 and sel     );

end architecture;
```

Multiplexador

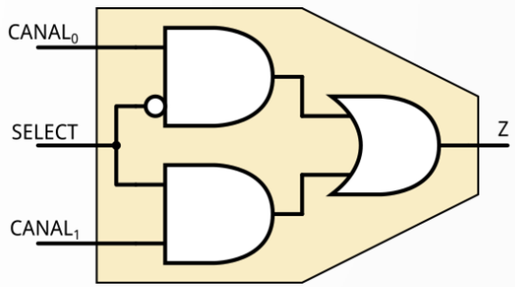
- MUX_{2x1}
  - Um pouco mais sobre VHDL

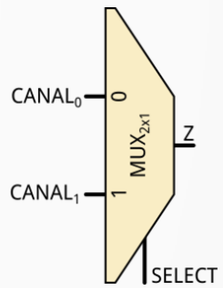
```
architecture comuta of mux2x1 is
begin
    zpl <= (c0 and not(sel)) or
           (c1 and sel     );
end architecture;
```

  - Seleção entre sinais
- Comando de atribuição condicional
- <receptor> <= <sinal> when <condição> else <sinal>;

Multiplexador

- MUX_{2x1}





ENTIDADE:

```
entity mux2x1 is

port(

    c0  : in  bit;

    c1  : in  bit;

    sel : in  bit;

    z   : out bit;

);

end entity;
```

COMPORTAMENTO:

```
architecture comuta of mux2x1b is

begin

    z <= c0 when sel = ‘1’ else c1;

end architecture;
```

Multiplexador

- MUX_{2x1}
  - Anexos mux2x1.vhdl e tb_mux2x1.vhdl

Multiplexador

- MUX_{2x4}
  - Entradas de 4 bit
- $Canal_0$ → C0 = {C03, C02, C01, C00}
- $Canal_1$ → C1 = {C13, C12, C11, C10}
- Seletor → SELECT
  - Saída de 1 bit
- Z = = {Z3, Z2, Z1, Z0}

Multiplexador

- MUX_{2x4_feio}

ENTIDADE:

```
entity mux2x1_feio is
    port(
        c03, c02, c01, c00  : in  bit;
        c13, c12, c11, c10  : in  bit;
        sel : in  bit;
        z3, z2, z1, z0  : out bit;
    );
end entity;
```

Multiplexador

- MUX2x4_feio

COMPORTAMENTO:

```
architecture comuta of mux2x1_feio is

begin

   -- usando portas lógicas

   z3 <= (c03 and not(sel)) or (c13 and sel); -- 4 x mux2x1

   Z2 <= (c02 and not(sel)) or (c12 and sel);

   z1 <= (c01 and not(sel)) or (c11 and sel);

   z0 <= (c00 and not(sel)) or (c10 and sel);


   -- ou

   z3 <= c03 when sel = '0' else c13;

            ...

end architecture;
```

Multiplexador

- MUX_{2x4_feio}
  - Anexos mux2x4_feio.vhdl e tb_mux2x4_feio.vhdl

Multiplexador

- MUX_{2x4}
  - Entradas de 4 bit
- $Canal_0$ → C0 = {C03, C02, C01, C00}
- $Canal_1$ → C1 = {C13, C12, C11, C10}
- Seletor → SELECT
  - Saída de 1 bit
- Z = = {Z3, Z2, Z1, Z0}
  - FEIO?

```
entity mux2x1_feio is
port(
    c03, c02, c01, c00  : in  bit;
    c13, c12, c11, c10  : in  bit;
    sel : in  bit;
    z3, z2, z1, z0  : out bit;
);
end entity;
```

Multiplexador

- MUX_{2x4}
  - Entradas de 4 bit
- $Canal_0$ → C0 = {C03, C02, C01, C00}
- $Canal_1$ → C1 = {C13, C12, C11, C10}
- Seletor → SELECT
  - Saída de 1 bit
- Z = = {Z3, Z2, Z1, Z0}

bit_vector

criando barramentos

Sinal: bit_vector

- Vetor unidimensional
  - Elementos do tipo bit
- Definição de MSb e LSb (direção e limites)

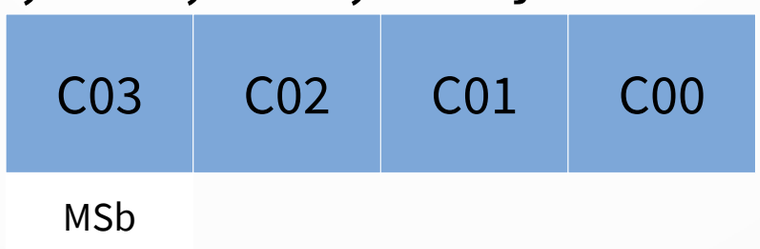
- MUX_{2x4}
  - Entradas de 4 bit
- Canal_ → C0 = {C03, C02, C01, C00}

Vetor de bits

C03	C02	C01	C00
MSb			

Sinal: bit_vector

- MUX_{2x4}
  - Entradas de 4 bit
- $Canal_0$ → C0 = {C03, C02, C01, C00}



- 4 sinais → bit_vector com 4 elementos bit:
- C0 : bit_v(3 do

Vetor de bits

limite superior

direção

limite inferior

Sinal: bit_vector

- Atribuição
  - Sinais bit
- Usa-se <= ‘<valorLógico>’
- Exemplo:
- sz <= ‘1’; -- aspas simples
  - Sinais bit_vector
- Usa-se <= “<listaValoresLógicos>”
- Exemplo:
- sz_v <= “10010110”; -- aspas duplas

Sinal: bit_vector

- Atribuição parcial
  - Considere o bit_vector

sz_v : bit_vector (7 downto 0);

e

sz_v <= “10010110”;

- atribuir único bit
- sz_v(3) <= ‘1’;
- resultado: “10011110”

Sinal: bit_vector

- Atribuição parcial
  - Considere o bit_vector

sz_v : bit_vector (7 downto 0);

e

sz_v <= “10010110”;

- atribuir parcial
- sz_v(6 downto 4) <= “110”;
- resultado: “1**110**1110”

Sinal: bit_vector

- Leitura e atribuição parcial
  - Considere os bit_vectors

sz_v : bit_vector (7 downto 0);

sx_v : bit_vector (7 downto 0);

e

sz_v <= “10010110”

sx_v <= “11000000”

- Leitura e atribuição parcial
- sx_v(3 downto 0) <= sz_v(7 downto 4);
- Resultado sx_v: “1100**1001**”

EXERCÍCIOS

Exercícios

- Descrever os Multiplexadores:
  - MUX_{2x1}
  - MUX_{2x8} (NEANDER)
  - MUX_{4x1}
  - MUX_{8x1}

- Criar os arquivos Testbench e simular.

Dúvidas?