Universidade Federal do Rio Grande do Norte
Centro de Tecnologia
Departamento de Engenharia de Computação e Automação
DCA0212.1 - Circuitos Digitais
Docente: Tiago Barros

Laboratório 3 – Multiplexadores e decodificadores

Objetivos:

1. Experimentar a descrição em VHDL de circuitos na forma comportamental;
2. Reforçar os conceitos de multiplexadores e decodificadores.
3. Pôr em prática conceitos aprendidos na disciplina Circuitos Digitais - Teoria.

Introdução:

Na aula de hoje iremos rever dois conceitos importantes de circuitos digitais: (1) multiplexadores e (2) decodificadores. Também serão apresentadas duas formas de implementações deste tipo de circuito combinacional: descrição comportamental e descrição por portas lógicas.

Multiplexadores:

Multiplexadores (ou MUX) são um bloco construtivo de nível mais elevado usado em circuitos digitais. Um multiplexador Mx1 tem M entradas e 1 saída, permitindo que apenas uma das entradas seja passada para a saída, através de bits de seleção. Pode-se chamar um multiplexador de **seletor** pois seleciona uma das entradas para ser passada para a saída.

Um multiplexador 2x1 tem duas entradas de dados I0 e I1, e um bit de seleção de entrada S0. Quando S0 = 0, a entrada I0 é replicada na saída, quando S0 = 1, o valor que passará para a saída será o da entrada I1.

O multiplexador 2x1 pode ser construído usando portas lógicas NOT, AND e OR. Na Figura 1(c), é ilustrado o projeto de um multiplexador usando essas portas, apresentando também o fluxo de dados (Figura 1(b)) quando o bit de entrada se altera.

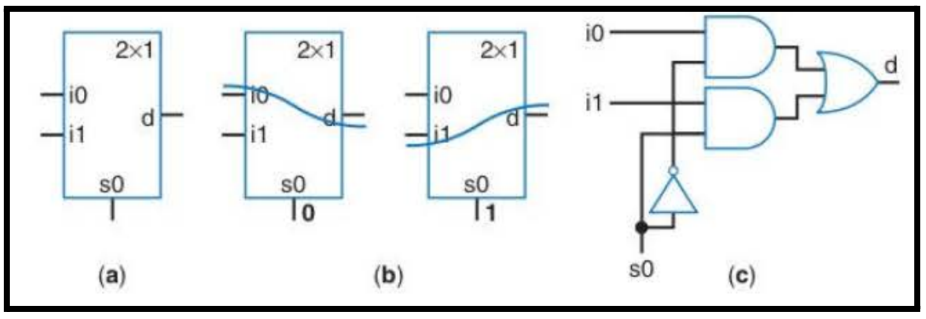

Figura 1: Representação de um Multiplexador 2x1.

Analisando o circuito mostrado na Figura 1(c), pode-se obter a expressão:

$$D = I0 \cdot S0' + I1 \cdot S0 \qquad (1)$$

A utilização de multiplexadores são as mais diversas, por exemplo como seletores de canais em televisões, seletores de canais de áudio, geradores de sinais. A representação gráfica de multiplexadores, está apresentada na Figura 2, bem como a tabela verdade.

Normalmente um MUX pode ter até 2^N entradas e necessita para a seleção do sinal de saída, que N sinais de seleção estejam presentes. Portanto, para dois sinais, é necessário um sinal de seleção, para quatro sinais, é necessário dois sinais para seleção, e assim por diante.

Um multiplexador 4x1 tem quatro entradas de dados A0, A1, A2 e A3, duas entradas de seleção S1 e S0 e uma saída de dados X (um multiplexador sempre terá apenas uma saída de dados, não importando quantas entradas). Um diagrama de blocos de um multiplexador 4x1 está mostrado na Figura 2.

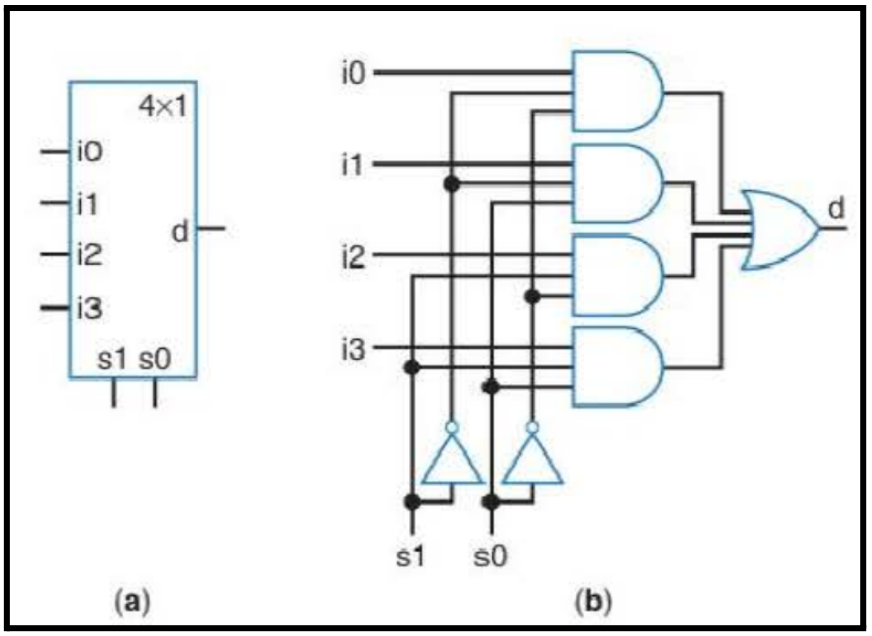

Figura 2: Representação de um multiplexador 4x1.

Multiplexadores mais complexos podem ser construídos utilizando multiplexadores 2 x 1, conforme visto na Figura 3.

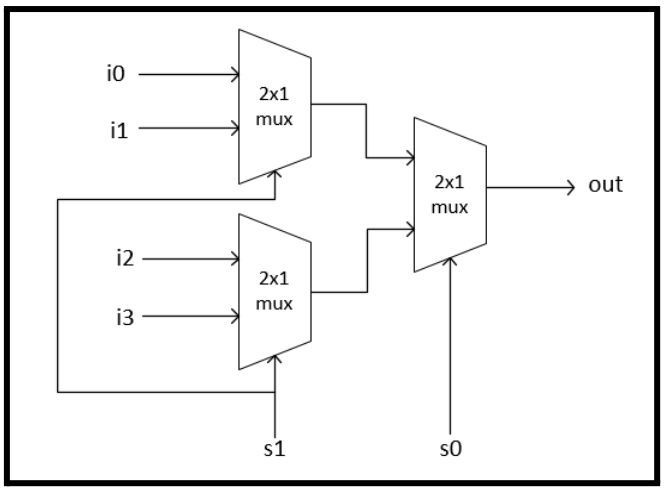

Figura 3: Multiplexador 4x1 utilizando multiplexadores 2x1

Descrição Comportamental de multiplexadores em VHDL:

A linguagem de descrição de hardware VHDL permite a utilização de comandos condicionais para construir uma declaração de arquitetura baseada no comportamento do módulo. Nesse tipo de descrição, não é necessário, obrigatoriamente, preocupar-se com portas lógicas. Pode-se implementar o comportamento dos circuitos lógicos por meio de algoritmos. No quadro 1, a funcionalidade da porta NOT é implementada por meio de sua descrição comportamental.

```
ENTITY PortaNot IS
PORT(A : IN BIT;
                S : OUT BIT);
END;
```

```
ARCHITECTURE behav OF PortaNot IS
BEGIN
WITH A SELECT
        S <= '1' WHEN '0',
             '0' WHEN '1';
END;
```

Quadro 1: Descrição comportamental da Porta NOT.

O comando WITH <variavel> SELECT funciona como um SWITCH...CASE da linguagem C++.

É possível implementar um MUX por meio da descrição comportamental do componente, conforme a descrição do quadro 2.

```
ENTITY Mux2x1 IS
PORT(I0,I1,s0 : IN BIT; -- No qual s0 é a porta de seleção
                 d : OUT BIT);
END;
ARCHITECTURE behav OF Mux2x1 IS
BEGIN
WITH s0 SELECT
  d <= I0 WHEN '0',
       I1 WHEN '1';
END;
```

Quadro 2: Descrição comportamental do MUX 2x1

Decodificadores:

Um decodificador é um bloco construtivo de nível mais elevado, comumente usados em circuitos digitais. Um **decodificador**, como o nome já diz, decodifica um número binário de N bits de entrada colocando exatamente uma das 2^n saídas do decodificador em 1. Como, por exemplo, o circuito representado pela Tabela 1.

Tabela 1: Decodificador de 2 bits.

Entradas		**Saídas**			
A_1	A_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

O circuito que descreve um decodificador 2x4 está representado na Figura 4.

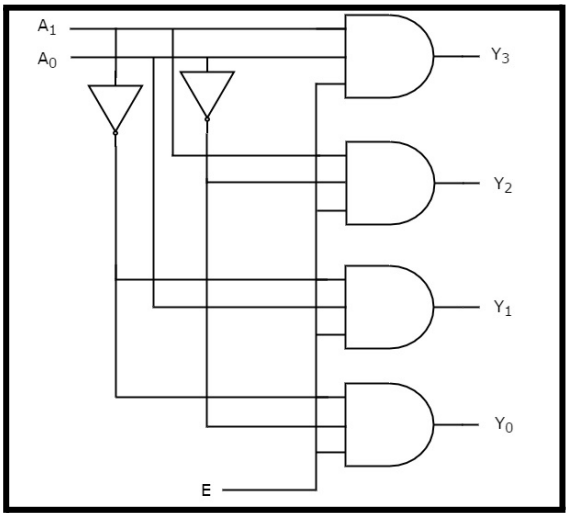

Figura 4: Circuito de um decodificador 2x4.

Note que na Figura 4 existe uma entrada adicional, identificada como **E**. Essa entrada é conhecida como **enable** (traduzida para português como *habilitar*) e é comum em decodificadores. Quando enable é 0, o decodificador coloca todas as saídas em 0, quando o enable é 1, o decodificador funciona conforme mostrado na Tabela 1.

Esse sinal de enable é especialmente útil quando se deseja que nenhuma das saídas sejam ativadas (enable = 0). Por outro lado, se a entrada de habilitação for ativada (enable = 1), pelo menos uma das saídas será 1.

Em VHDL pode-se criar decodificadores utilizando diferentes estratégias, como a descrição de circuitos por portas lógicas ou a descrição comportamental. Os quadros 3 e 4 mostram exemplos destes dois tipos de descrição VHDL. A Figura 6 apresenta formas de onda obtidas com a simulação de ambos os códigos.

```
ENTITY Decode IS
PORT(A1,A0,E: IN BIT;
                Y0,Y1,Y2,Y3 : OUT BIT);
END;
ARCHITECTURE behav OF Decode IS
BEGIN
Y0 <= E AND NOT(A1) AND NOT(A0);
Y1 <= E AND NOT(A1) AND A0;
Y2 <= E AND A1 AND NOT(A0);
Y3 <= E AND A1 AND A0;
END;
```

Quadro 3 - Estratégia de descrição via portas lógicas

```
ENTITY Decode_Comp IS
PORT(A1,A0,E: IN BIT;
                Y0,Y1,Y2,Y3 : OUT BIT);
END;

ARCHITECTURE behav OF Decode_Comp IS
SIGNAL out_aux: BIT_VECTOR(3 DOWNTO 0);
SIGNAL in_aux : BIT_VECTOR(2 DOWNTO 0);
BEGIN
in_aux <= E & A1 & A0 ; -- Usado para concatenar os sinais de entrada
WITH in_aux  SELECT
out_aux <= "0001" WHEN "100",
           "0010" WHEN "101",
           "0100" WHEN "110",
           "1000" WHEN "111",
           "0000" WHEN OTHERS;
Y0 <= out_aux(0);
Y1 <= out_aux(1);
Y2 <= out_aux(2);
Y3 <= out_aux(3);
END;
```

Quadro 4 - Descrição Comportamental do decodificador

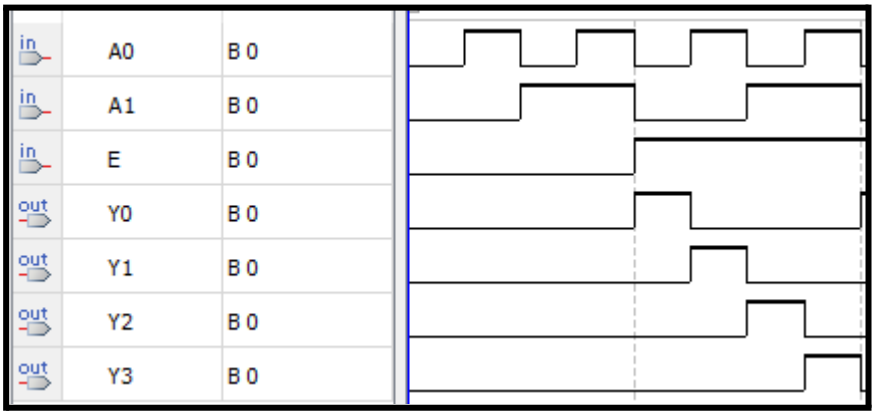

Figura 5 - Simulação do decodificador 2x4.

Note na Figura 5 que quando enable é 0, as saídas do Y0, Y1, Y2 e Y3 são 0. Quando o pino de enable (E) está ativo, as saídas se comportam conforme descrito na Tabela 1.

Circuitos digitais compostos por decodificadores são empregados em diversas aplicações, por exemplo: display de sete segmentos, contadores regressivos, sistemas de irrigação etc.

Exemplo de decodificador: display de 7 segmentos

Muitos eletrodomésticos exibem números para serem lidos por nós, humanos. Exemplos desses eletrodomésticos incluem relógios, fornos micro-ondas, geladeiras etc. Um dispositivo muito popular e simples, capaz de exibir um número de um único dígito é o display de sete segmentos, mostrado na Figura 6.

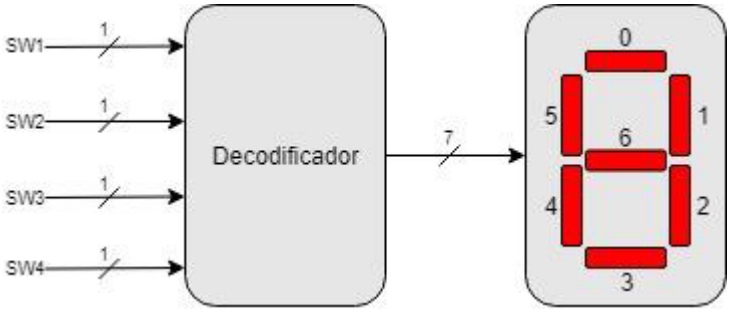

Figura 6: Display de sete segmentos.

Um display de sete segmentos funciona da seguinte forma:

- Quando deseja-se mostrar o número 1, apaga-se os pinos: 0, 3, 4, 5 e 6, deixando ligados apenas os pinos 1 e 2;
- Quando deseja-se mostrar o número 2, apaga-se os pinos: 2 e 5, deixando os restantes ligados.
- Por convenção, para os pinos de saída 0 a 6, adota-se nível lógico ‘1’ para indicar que o pino está apagado e nível lógico ‘0’ para indicar que o pino está aceso.

Pode-se projetar um decodificador BCD (Binary-coded decimal) que é responsável por receber comandos externos (representação binária) e mostrar o número em decimal no display de sete segmentos.

Atividades:

Entregar um relatório descrevendo a execução das tarefas 1 e 2. O relatório deve conter todos os códigos VHDL e formas de onda das simulações.

1) **Multiplexador**
   - **a)** Considerando entradas de dados com largura de 1 bit, implemente em VHDL, utilizando a descrição comportamental, um MUX 2x1. Apresente a simulação da implementação;
   - **b)** Considerando entradas de dados com largura de 1 bit, implemente em VHDL, utilizando a descrição por portas lógicas, um MUX 2x1. Apresente a simulação da implementação;
   - **c)** Considerando entradas de dados com largura de 1 bit, implemente em VHDL, utilizando a descrição comportamental ou com portas lógicas + componentes, um MUX 4x1 utilizando MUX 2x1. Apresente a simulação da implementação;
   - **d)** Considerando entradas de dados com largura de 4 bits, implemente em VHDL, utilizando a descrição por **portas lógicas**, um MUX 2x1. Apresente a simulação da implementação. Pode-se utilizar componentes para facilitar o projeto;

Dica:

A descrição de **sinais de seleção**, para multiplexadores com mais de duas entradas, pode ser auxiliada pelo(s):

- Uso do comando “x <= a & b”, o qual concatena em x os valores de a e b.
- Uso de vetores de bits, declarados com variáveis do tipo “BIT_VECTOR”.

2) Display de 7 segmentos

a) Monte a tabela verdade para um decodificador de 4 para 7 bits. Utilize nível lógico '1' para indicar que o pino está apagado e nível lógico '0' para indicar que o pino está aceso.

Exemplo:

Entradas				Saídas							
A4	A3	A2	A1	S6	S5	S4	S3	S2	S1	S0	Valor Visualizado
0	0	0	0	1	0	0	0	0	0	0	0
0	0	0	1	1	1	1	1	0	0	1	1
0	0	1	0								2
0	0	1	1								3
0	1	0	0								4
0	1	0	1								5
0	1	1	0								6
0	1	1	1								7
1	0	0	0								8
1	0	0	1								9

b) Projete este decodificador BCD em VHDL utilizando a descrição por portas lógicas.

c) Projete este decodificador BCD em VHDL utilizando a descrição comportamental.

d) Considere, em seu projeto, que a saída deve ter todos os pinos apagados para os casos em que as entradas não correspondam a uma saída que mostre um algarismo entre 0 e 9.