

# Aulas 12 e 13

## Demultiplexadores

---

## Autores

Gabriel A. F. Souza, Gustavo D. Colletta, Leonardo B. Zoccal, Odilon O. Dutra
---

Unifei
--------

## Histórico de Revisões

22 de fevereiro de 2025	1.0	Primeira versão do documento.
-------------------------	-----	-------------------------------

# Tópicos

---

- Introdução
- Descrição em Verilog
- Simulação e Verificação
- Atividades Hands-on



# Introdução

# Objetivos

---

- Compreender o conceito de circuitos demultiplexadores.
- Explorar suas aplicações na eletrônica digital.
- Apresentar estruturas para descrição em Verilog.

# Definição

---

- Definição: Um circuito demultiplexador (ou Demux) é um circuito lógico combinacional que recebe uma entrada e possui  $N$  linhas de seleção, permitindo que o valor de entrada seja direcionado a apenas uma das  $2^N$  saídas.
- Função: Distribuir um sinal de entrada para uma das várias saídas com base no valor de sinais de controle (ou seleção).

# O que são Demultiplexadores?

---

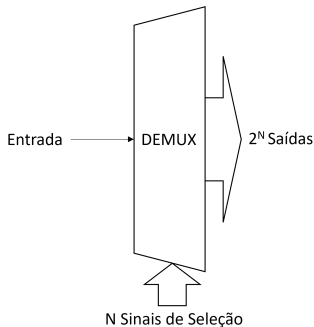


Figura 1: Circuito Demultiplexador

- Possui N entradas de seleção e uma entrada de dados.
- Possui  $2^N$  saídas, sendo que apenas uma reflete o valor de entrada, em função do sinal de seleção.



# Aplicações

---

- Distribuição de dados em redes: Em sistemas de comunicação, um único canal de transmissão pode ser usado para enviar dados a vários dispositivos diferentes.
- Multiplexação de tempo: Em circuitos digitais, um demux pode ser usado para compartilhar um único canal de comunicação entre vários dispositivos, alternando rapidamente entre eles.
- Endereçamento de memória: Em computadores, um demux pode ser usado para selecionar uma célula de memória específica ao receber um endereço binário.
- Escrita Condicional em Registradores: Em arquiteturas de processadores, um demux pode ser usado para escolher em qual registrador armazenar o resultado da ULA.

# Descrição em Verilog

# Tipos de Descrição

---

- **Fluxo de Dados (Dataflow)**
- **Estrutural (Structural)**
- **Comportamental (Behavioral)**

# Demultiplexadores

---

- Demultiplexador 1x2
  - Descrição por Fluxo de Dados (Expressões Lógicas)
  - Descrição Comportamental
- Demultiplexador 1x4
  - Descrição por Fluxo de Dados (Expressões Lógicas)
  - Descrição Comportamental

## Demultiplexador 1x2

---

- Direciona a entrada para uma das duas saídas

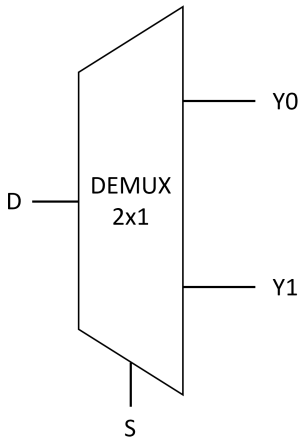


Figura 2: Exemplo de Demultiplexador 1x2

# Tabela Verdade

---

Tabela de um demultiplexador 1 para 2.

Entrada (D)	Seleção (S)	Saída Y0	Saída Y1
0	0	0	0
0	1	0	0
1	0	1	0
1	1	0	1

# Expressões Lógicas

---

A partir da tabela verdade, podemos escrever as expressões para as saídas  $Y_0$  e  $Y_1$ :

$$Y_0 = D \cdot \overline{S}$$

$$Y_1 = D \cdot S$$

# Descrição Fluxo de Dados

---

```
1  module demux_1x2 (  
2      input D,          // Entrada de dados  
3      input S,          // Seletor  
4      output Y0,        // Saída 0  
5      output Y1         // Saída 1  
6  );  
7  
8      // Implementação das expressões lógicas  
9      assign Y0 = D & ~S; // Y0 = D * S'  
10     assign Y1 = D & S;  // Y1 = D * S  
11  
12 endmodule
```



## Descrição Comportamental (Incorreto)

```
1 module demux_1x2_incomp (  
2     input  D,          // Entrada de dados  
3     input  S,          // Seletor  
4     output reg Y0,     // Saída 0  
5     output reg Y1      // Saída 1  
6 );  
7  
8     always @(*) begin  
9         case (S)  
10            1'b0:  Y0 = D;  // Se S = 0, Y0 recebe D  
11            1'b1:  Y1 = D;  // Se S = 1, Y1 recebe D  
12        endcase  
13    end  
14  
15 endmodule
```

**O bloco case apresenta atribuições incompletas. A ferramenta de síntese irá inferir elementos de memória indesejados**

## Descrição Comportamental (Correto)

```
1 module demux_1x2_comp (
2     input  D,          // Entrada de dados
3     input  S,          // Seletor
4     output reg Y0,      // Saída 0
5     output reg Y1      // Saída 1
6 );
7
8     always @(*) begin
9         case (S)
10            1'b0: begin
11                Y0 = D; // Se S = 0, Y0 recebe D
12                Y1 = 0; // Y1 recebe 0
13            end
14            1'b1: begin
15                Y0 = 0; // Y0 recebe 0
16                Y1 = D; // Se S = 1, Y1 recebe D
17            end
18        endcase
19    end
```

# Descrição Comportamental

---

- Observe que as saídas inativas receberam o valor lógico baixo seguindo a definição da tabela verdade.
- Um Demux também pode ter as saídas inativas em nível lógico alto, dependendo da especificação.

## Demultiplexador 1x4

---

- Direciona a entrada para uma das quatro saídas

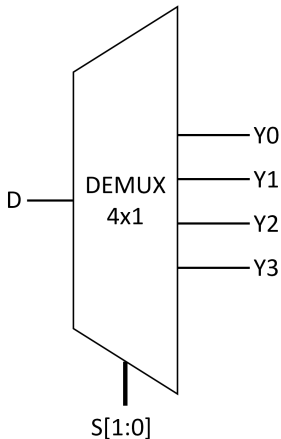


Figura 3: Exemplo de Demultiplexador 1x4

---

# Tabela Verdade

---

Tabela de um demultiplexador 1 para 4.

Entrada (D)	Seleção (S1,S0)	Y0	Y1	Y2	Y3
0	xx	0	0	0	0
1	00	1	0	0	0
1	01	0	1	0	0
1	10	0	0	1	0
1	11	0	0	0	1

# Expressões Lógicas

---

$$Y_0 = D \cdot \overline{S_1} \cdot \overline{S_0}$$

$$Y_1 = D \cdot \overline{S_1} \cdot S_0$$

$$Y_2 = D \cdot S_1 \cdot \overline{S_0}$$

$$Y_3 = D \cdot S_1 \cdot S_0$$

# Descrição Fluxo de Dados

---

```
1 module demux_1x4 (  
2     input D,           // Entrada de dados  
3     input S1,          // Bit de seleção mais significativo  
4     input S0,          // Bit de seleção menos significativo  
5     output Y0,         // Saída 0  
6     output Y1,         // Saída 1  
7     output Y2,         // Saída 2  
8     output Y3          // Saída 3  
9 );  
10  
11 // Implementação das expressões lógicas  
12 assign Y0 = D & ~S1 & ~S0; // Y0 = D * S1' * S0'  
13 assign Y1 = D & ~S1 & S0;  // Y1 = D * S1' * S0  
14 assign Y2 = D & S1 & ~S0;  // Y2 = D * S1 * S0'  
15 assign Y3 = D & S1 & S0;   // Y3 = D * S1 * S0  
16  
17 endmodule
```

# Descrição Comportamental

```
1  module demux_1x4_comp (
2      input D,           // Entrada de dados
3      input S1,          // Bit de seleção mais significativo
4      input S0,          // Bit de seleção menos significativo
5      output reg Y0,      // Saída 0
6      output reg Y1,      // Saída 1
7      output reg Y2,      // Saída 2
8      output reg Y3       // Saída 3
9  );
10
11  always @(*) begin
12      // Todas as saídas são zeradas por padrão
13      Y0 = 0; Y1 = 0; Y2 = 0; Y3 = 0;
14      case ({S1, S0})
15          2'b00: Y0 = D; // Se S1S0 = 00, Y0 recebe D
16          2'b01: Y1 = D; // Se S1S0 = 01, Y1 recebe D
17          2'b10: Y2 = D; // Se S1S0 = 10, Y2 recebe D
18          2'b11: Y3 = D; // Se S1S0 = 11, Y3 recebe D
19      endcase
20  end
21  endmodule
```



# Simulação e Verificação

# Testbench

---

```
1 module demux_1x4_tb();
2     reg D;
3     reg S1, S0;
4     wire Y0, Y1, Y2, Y3;
5
6     // Instanciação do módulo Demux 1x4
7     demux_1x4 uut (.D(D), .S1(S1), .S0(S0), .Y0(Y0), .Y1(Y1),
8         .Y2(Y2), .Y3(Y3));
9
10    // Bloco inicial para definir os estímulos de teste
11    initial begin
12        // Monitoramento dos sinais para depuração
13        $monitor("D=%b | S1=%b | S0=%b -> Y0=%b, Y1=%b, Y2=%b,
14            Y3=%b",
15                D, S1, S0, Y0, Y1, Y2, Y3);
```

# Testbench

---

```
15      // Testa todas as combinações de seletores com D = 0
      (todas as saídas devem ser 0)
16      D = 0; S1 = 0; S0 = 0; #10;
17      D = 0; S1 = 0; S0 = 1; #10;
18      D = 0; S1 = 1; S0 = 0; #10;
19      D = 0; S1 = 1; S0 = 1; #10;
20
21      // Testa todas as combinações de seletores com D = 1
      (apenas uma saída deve ser 1)
22      D = 1; S1 = 0; S0 = 0; #10;
23      D = 1; S1 = 0; S0 = 1; #10;
24      D = 1; S1 = 1; S0 = 0; #10;
25      D = 1; S1 = 1; S0 = 1; #10;
26
27      // Finaliza a simulação
28      $stop;
29  end
30
31  endmodule
```

# Forma de Onda

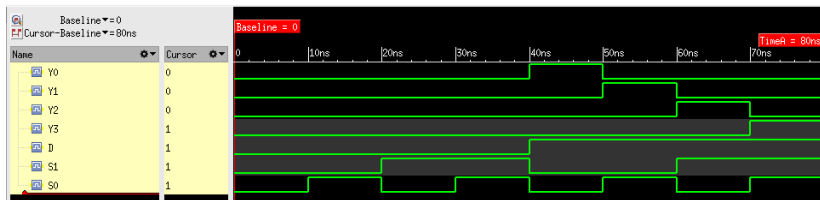


Figura 4: Resultado da Simulação

# Atividades Hands-on

# Atividade 1

---

- Monte uma descrição comportamental de um circuito demultiplexador 1x8.
- Elabore um Testbench e simule a operação do circuito.

## Atividade 2

---

- Monte uma descrição estrutural de um circuito demultiplexador 1x8 utilizando dois módulos de um demultiplexador 1x4 e um módulo de um demultiplexador 1x2.
- Elabore um Testbench e simule a operação do circuito.

## Atividade 3

---

- Elabore uma descrição parametrizável de um circuito demultiplexador  $1 \times N$ .
- Simule para  $N = 8$ ;