Aulas 10 e 11
Multiplexadores

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

Histórico de Revisões		
20 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 multiplexadores.
- Explorar suas aplicações na eletrônica digital.
- Apresentar estruturas para descrição em Verilog.

Definição

- Definição: Um circuito multiplexador (ou Mux) é um circuito lógico combinacional que recebe 2^N entradas e possui N linhas de seleção, permitindo que apenas uma das entradas seja direcionada à saída.
- Função: Selecionar uma das várias entradas de dados e a encaminhar para a saída com base no valor de sinais de controle (ou seleção).

Os multiplexadores estão entre os circuitos combinacionais mais comumente utilizados em sistemas digitais

O que são Multiplexadores?

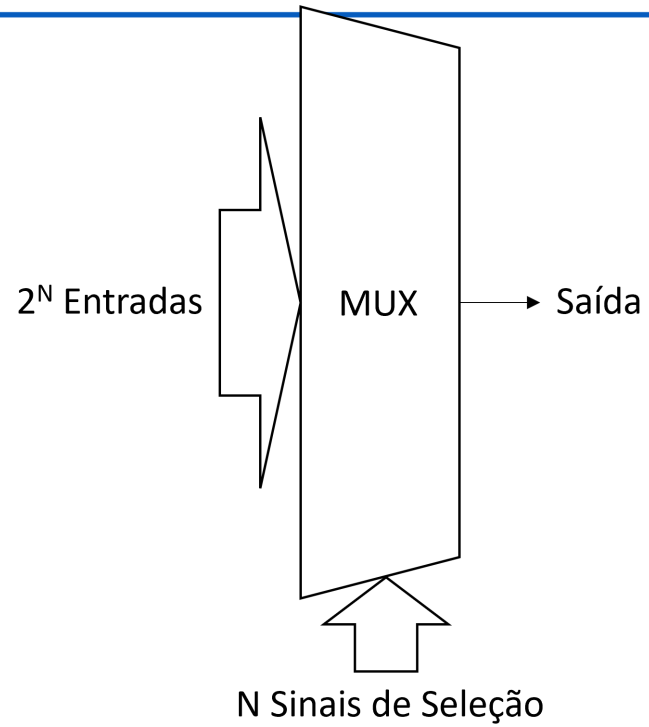

Figura 1: Circuito Multiplexador

- Possui N entradas de seleção e 2^N entradas de dados.
- Possui uma saída, que reflete a entrada selecionada.

Aplicações

- Seleção de sinais em diversos sistemas digitais: Permite a escolha de diferentes fontes de dados, como em processadores e circuitos de comunicação.
- Uso em barramentos de comunicação: Facilita a transmissão eficiente de dados entre diferentes dispositivos em um sistema.
- Controle de fluxo de dados em memórias e processadores: Utilizado para gerenciar o acesso à memória e a troca de informações dentro de uma CPU.
- Sistemas de telecomunicações: Multiplexadores são usados para combinar múltiplos sinais em uma única linha de transmissão.
- Conversores de dados: Utilizados em conversores digital-analógico (DAC) para selecionar diferentes valores de entrada.

Descrição em Verilog

Tipos de Descrição

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

Multiplexadores

- Multiplexador 2x1
  - Expressões Lógicas
  - Atribuição Condicional
  - Comportamental (If-Else)
  - Comportamental (Case)
- Multiplexador 4x1
  - Expressões Lógicas
  - Atribuição Condicional
  - Comportamental (If-Else)
  - Comportamental (Case)
- Multiplexador 8x1
  - Descrição Comportamental (Case)
- Multiplexador 4x1 (Barramento de Dados)
  - Descrição Comportamental

Multiplexador 2x1

- Seleciona uma saída a partir de 2 entradas

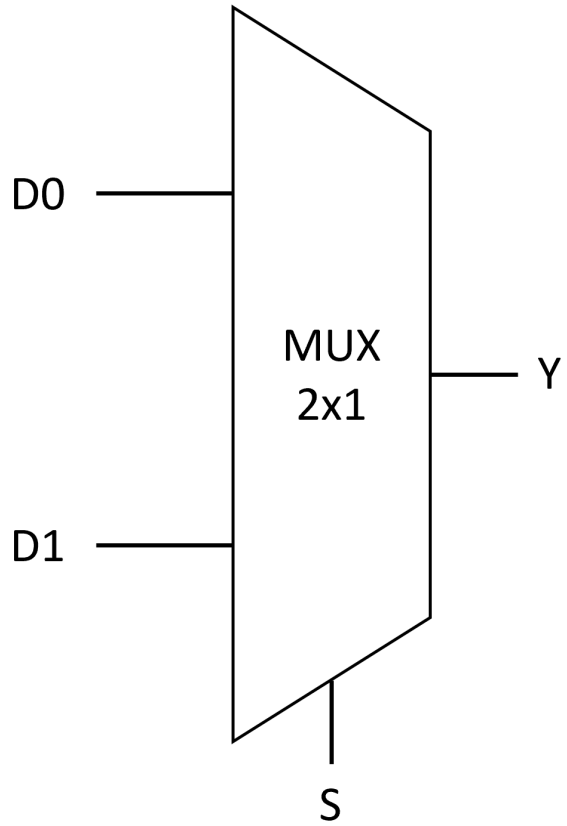

Figura 2: Exemplo de Multiplexador 2x1

Tabela Verdade

Tabela de um multiplexador 2 para 1.

Seleção (S)	Entrada D0	Entrada D1	Saída (Y)
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Expressão Lógica: $Y = (\overline{S} \cdot D_0) + (S \cdot D_1)$

Descrição Fluxo de Dados

```
1  module mux2x1 (
2      input S,        // Sinal de seleção
3      input D0,       // Entrada 0
4      input D1,       // Entrada 1
5      output Y        // Saída
6  );
7      // Implementação da expressão booleana: Y = (~S & D0) | (S
           & D1)
8      assign Y = (~S & D0) | (S & D1);
9
10 endmodule
```

Atribuição Condicional

```
1  module mux2x1_cond (
2      input S,       // Sinal de seleção
3      input D0,      // Entrada 0
4      input D1,      // Entrada 1
5      output Y       // Saída
6  );
7
8      assign Y = S ? D0 : D1;
9
10 endmodule
```

Descrição Comportamental (If-Else)

```
1  module mux2x1_if(
2      input S,        // Sinal de seleção
3      input D0,       // Entrada 0
4      input D1,       // Entrada 1
5      output reg Y        // Saída
6  );
7      always@(*) begin
8          if(S)
9              Y = D1;
10         else
11             Y = D0;
12     end
13
14 endmodule
```

Descrição Comportamental (Case)

```
1  module mux2x1_case(
2      input S,       // Sinal de seleção
3      input D0,      // Entrada 0
4      input D1,      // Entrada 1
5      output reg Y       // Saída
6  );
7      always@(*) begin
8          case(S)
9              1'b0: Y = D0;
10             1'b1: Y = D1;
11         endcase
12     end
13
14 endmodule
```

Multiplexador 4x1

- Seleciona uma saída a partir de 4 entradas

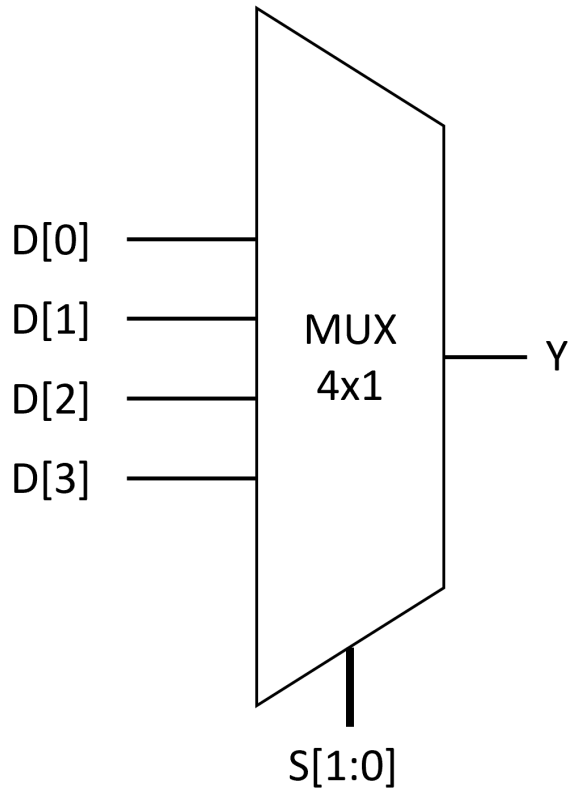

Figura 3: Exemplo de Multiplexador 4x1

Tabela Verdade

Tabela de um multiplexador 4 para 1.

S[1]	S[0]	D[3]	D[2]	D[1]	D[0]	Y
0	0	X	X	X	0	0
0	0	X	X	X	1	1
0	1	X	X	0	X	0
0	1	X	X	1	X	1
1	0	X	0	X	X	0
1	0	X	1	X	X	1
1	1	0	X	X	X	0
1	1	1	X	X	X	1

Expressão Lógica

$$Y = (\overline{S[1]} \cdot \overline{S[0]} \cdot D[0]) + (\overline{S[1]} \cdot S[0] \cdot D[1]) +$$
$$(S[1] \cdot \overline{S[0]} \cdot D[2]) + (S[1] \cdot S[0] \cdot D[3])$$

Descrição Fluxo de Dados

```
1  module mux4x1 (
2      input [1:0] S,       // Entradas de seleção S[1:0]
3      input [3:0] D,       // Entradas de dados D[3:0]
4      output Y             // Saída Y
5  );
6      // Implementação da expressão booleana:
7      assign Y = (~S[1] & ~S[0] & D[0]) |
8                 (~S[1] &  S[0] & D[1]) |
9                 ( S[1] & ~S[0] & D[2]) |
10                ( S[1] &  S[0] & D[3]);
11
12 endmodule
```

Atribuição Condicional

```
1  module mux4x1_cond (
2      input [1:0] S,       // Entradas de seleção S[1:0]
3      input [3:0] D,       // Entradas de dados D[3:0]
4      output Y             // Saída Y
5  );
6
7      assign Y =  (S == 2'b00) ? D[0] :
8                  (S == 2'b01) ? D[1] :
9                  (S == 2'b10) ? D[2] :
10                                D[3] ;
11
12     endmodule
```

Descrição Comportamental (If-Else)

```
1  module mux4x1_if(
2      input [1:0] S,       // Entradas de seleção S[1:0]
3      input [3:0] D,       // Entradas de dados D[3:0]
4      output reg Y              // Saída Y
5  );
6
7          always @(*) begin
8              if (S == 2'b00)
9                  Y = D[0];
10             else if (S == 2'b01)
11                 Y = D[1];
12             else if (S == 2'b10)
13                 Y = D[2];
14             else
15                 Y = D[3];
16         end
17
18     endmodule
```

Descrição Comportamental (Case)

```
1  module mux4x1_case(
2      input [1:0] S,       // Entradas de seleção S[1:0]
3      input [3:0] D,       // Entradas de dados D[3:0]
4      output reg Y              // Saída Y
5  );
6
7      always @(*) begin
8          case (S)
9                  2'b00: Y = D[0];
10                 2'b01: Y = D[1];
11                 2'b10: Y = D[2];
12                 2'b11: Y = D[3];
13             //default: Y = 1'b0;
14             // O uso do "default" é opcional nesse caso -->
                   Circuito sintetizado será o mesmo
15         endcase
16     end
17 endmodule
```

Multiplexador 8x1

- Seleciona uma saída a partir de 8 entradas

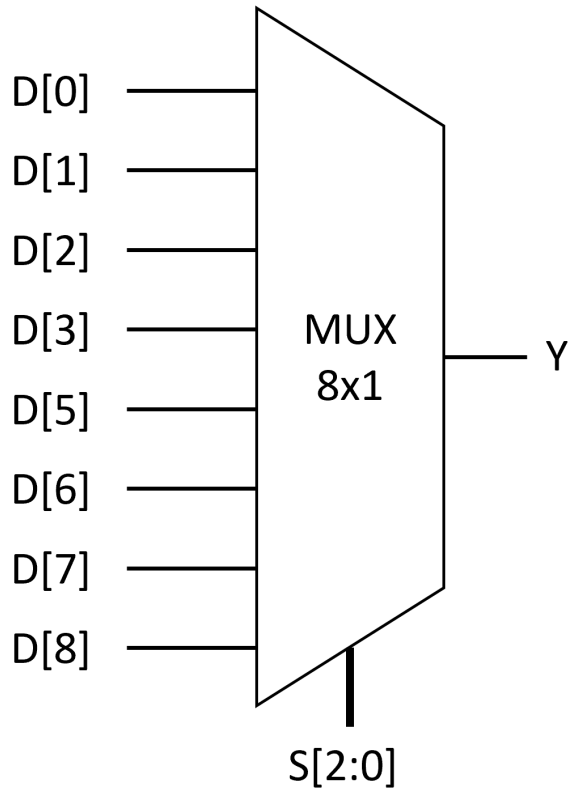

Figura 4: Exemplo de Multiplexador 8x1

Descrição Comportamental

```
1  module mux8x1 (
2      input [2:0] S,        // Entradas de seleção S[2:0]
3      input [7:0] D,        // Entradas de dados D[7:0]
4      output reg Y             // Saída Y
5  );
6
7      always @(*) begin
8          case (S)
9              3'b000: Y = D[0];
10             3'b001: Y = D[1];
11             3'b010: Y = D[2];
12             3'b011: Y = D[3];
13             3'b100: Y = D[4];
14             3'b101: Y = D[5];
15             3'b110: Y = D[6];
16             3'b111: Y = D[7];
17         endcase
18     end
19 endmodule
```

Multiplexador 4x1 (Barramento de Dados)

- Seleciona uma saída a partir de 4 entradas
- Tanto a saída quanto as entradas são barramentos de 3 bits

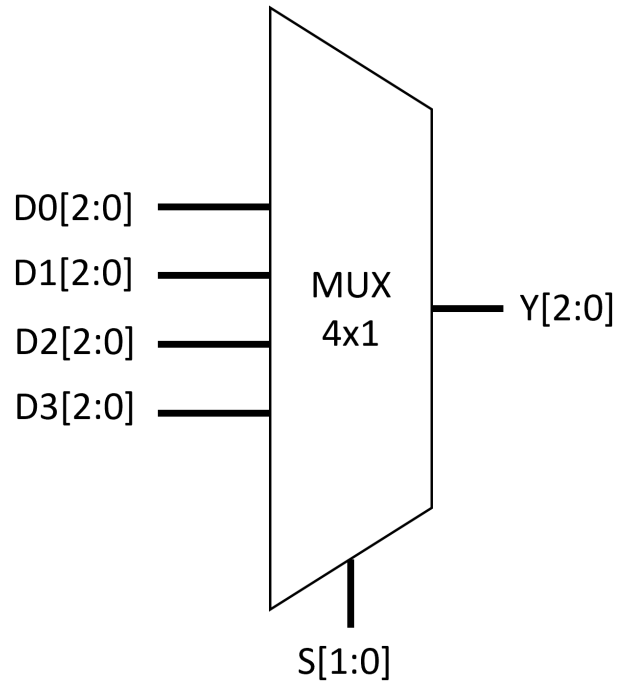

Figura 5: Exemplo de Multiplexador 4x1 com barramento de dados de 3 bits

Descrição Comportamental

```
1  module mux4x1_bus (
2      input wire [1:0] S,          // Entradas de seleção
3      input wire [2:0] D0, D1, D2, D3,    // Barramento de dados
4      output reg [2:0] Y           // Saída
5  );
6
7      always @(*) begin
8          case (S)
9              2'b00: Y = D0;
10             2'b01: Y = D1;
11             2'b10: Y = D2;
12             2'b11: Y = D3;
13         endcase
14     end
15
16 endmodule
```

Simulação e Verificação

Testbench

```
1  module mux4x1_bus_tb();
2      reg [1:0] S;              // Sinais de seleção
3      reg [2:0] D0, D1, D2, D3;     // Barramento de dados
4      wire [2:0] Y;             // Saída do MUX
5
6      // Instancia o DUT (Device Under Test)
7      mux4x1_bus uut (.S(S),.D0(D0), .D1(D1), .D2(D2),
           .D3(D3),.Y(Y));
8
9      initial begin
10         // Define os valores das entradas de dados
11         D0 = 3'b000;
12         D1 = 3'b101;
13         D2 = 3'b011;
14         D3 = 3'b111;
15
16         // Testa todas as combinações de seleção
```

Testbench

```
17          S = 2'b00; #10;  // Deve selecionar D0 = 000
18          S = 2'b01; #10;  // Deve selecionar D1 = 101
19          S = 2'b10; #10;  // Deve selecionar D2 = 011
20          S = 2'b11; #10;  // Deve selecionar D3 = 111
21
22          $stop;// Finaliza a simulação
23      end
24
25      // Monitor para observar os valores
26      initial begin
27          $monitor("Tempo = %0t | S = %b | Y = %b", $time, S, Y);
28      end
29
30  endmodule
```

Forma de Onda

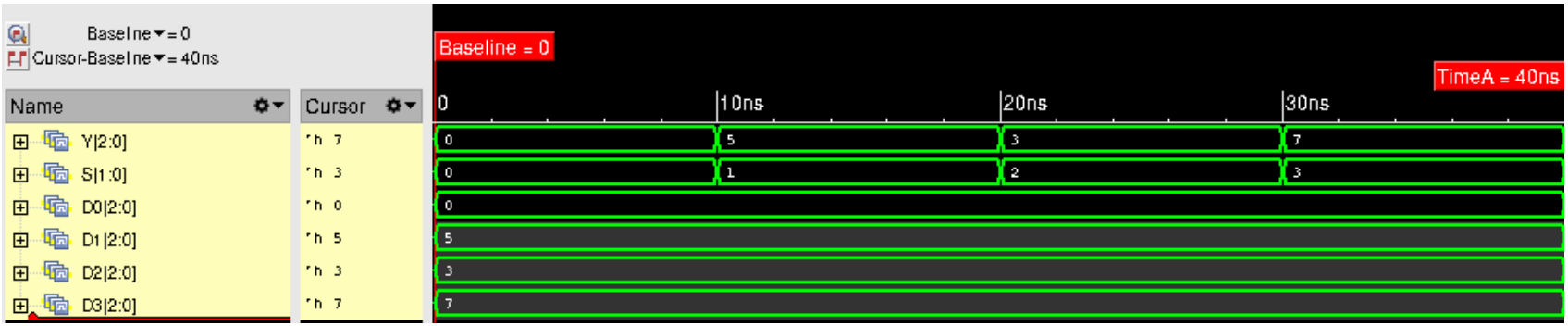

Figura 6: Resultado da Simulação

Atividades Hands-on

Atividade 1

- Monte uma descrição comportamental de um multiplexador 16x1.
- Elabore um Testbench e simule a operação do circuito.

Atividade 2

- Monte uma descrição estrutural de um multiplexador 8x1 utilizando dois módulos de um multiplexador 4x1 e um módulo de um multiplexador 2x1.
- Elabore um Testbench e simule a operação do circuito.

Atividade 3

- Elabore uma descrição parametrizável de um multiplexador 4x1 com barramentos de dados de N bits.
- Simule para N = 3;

Atividade 4

- Elabore uma descrição parametrizável de um multiplexador Nx1, com N entradas de dados de 1 bit e uma saída de 1 bit.
- Simule para N = 4;