Implementando funções lógicas com Mux

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

Histórico de Revisões		
10 de janeiro de 2025	1.0	Primeira versão do documento.

Tópicos

- Revisão sobre multiplexadores
- Implementação de funções lógicas
- Exemplo: Número de variáveis igual ao número de entradas de seleção
- Exemplo: Número de entradas maior que o número de entradas de seleção
- Exercícios

Revisão sobre multiplexadores

Multiplexadores em Circuitos Digitais

Um **multiplexador** (MUX) é um circuito combinacional que seleciona uma entre várias entradas de dados e direciona para uma única linha de saída. Ele é frequentemente chamado de “mux” e funciona como um “canal seletor” no processamento digital de sinais.

Estrutura Básica

Um multiplexador possui:

- 2^m **entradas de dados**: $D_0, D_1, \ldots, D_{2^m-1}$
- m **linhas de seleção**: $S_0, S_1, \ldots, S_{m-1}$
- **1 linha de saída**: Y

A linha de saída é determinada pelo valor das linhas de seleção.
Por exemplo:

- Se $S_1S_0 = 00$, a saída será $Y = D_0$.
- Se $S_1S_0 = 01$, a saída será $Y = D_1$.
- E assim por diante.

Tabela Verdade

A tabela verdade de um multiplexador 4x1 (4 entradas, 2 seletores) é apresentada abaixo:

S_1	S_0	Saída Y
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

O valor de Y é igual ao dado de entrada D_i, onde i é o valor binário das linhas de seleção S_1S_0.

Expressão Lógica

A saída de um multiplexador pode ser expressa por uma soma de produtos:

$$Y = \overline{S_1} \cdot \overline{S_0} \cdot D_0 + \overline{S_1} \cdot S_0 \cdot D_1 + S_1 \cdot \overline{S_0} \cdot D_2 + S_1 \cdot S_0 \cdot D_3$$

Cada termo da equação representa uma combinação de seleção e o respectivo dado de entrada.

Aplicações

Multiplexadores são usados em diversos sistemas digitais, incluindo:

- **Seleção de dados**: Combinar várias fontes de dados em uma linha.
- **Implementação de funções lógicas**: Qualquer função lógica pode ser implementada com um MUX.
- **Circuitos de comunicação**: Para compartilhar um único canal entre vários sinais.

Multiplexadores Cascateados

Multiplexadores podem ser combinados para criar circuitos maiores:

- Exemplo: Dois MUX 4x1 podem ser usados para implementar um MUX 8x1, adicionando uma linha de seleção extra.

Implementação de funções lógicas

Utilizando o Mux

A implementação de funções lógicas com **multiplexadores** é um processo que utiliza as entradas de seleção (S_2, S_1, S_0) para escolher diretamente as combinações de variáveis que definem a função lógica. Abaixo, explicamos o processo com um **multiplexador 8x1** (8 entradas de dados e 3 seletores).

Exemplo: Número de variáveis igual ao número de entradas de seleção

Função a ser implementada

Considere a função lógica de 3 variáveis A, B e C:

$$f(A, B, C) = \overline{A} \cdot B \cdot \overline{C} + A \cdot \overline{B} \cdot C + A \cdot B \cdot C$$

Passo-a-passo

❶ Representação da Função em Tabela Verdade

Primeiro, construímos a tabela verdade da função:

A	B	C	$f(A, B, C)$
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Passo-a-passo

❷ Mapeamento para o Multiplexador

As entradas de seleção do multiplexador (S_2, S_1, S_0) são mapeadas diretamente para as combinações das variáveis A, B e C:

- $S_2 = A$
- $S_1 = B$
- $S_0 = C$

Passo-a-passo

❷ Mapeamento para o Multiplexador

As entradas de dados (D_0 a D_7) são atribuídas com base na saída da função para cada linha da tabela verdade. Neste caso:

Entrada (D_i)	Valor
D_0	0
D_1	0
D_2	1
D_3	0
D_4	0
D_5	1
D_6	0
D_7	1

Passo-a-passo

❸ Equação do Multiplexador

O multiplexador realiza a função lógica:

$$f(A,B,C) = D_0 \cdot \overline{S_2} \cdot \overline{S_1} \cdot \overline{S_0} + D_1 \cdot \overline{S_2} \cdot \overline{S_1} \cdot S_0 + \cdots + D_7 \cdot S_2 \cdot S_1 \cdot S_0$$

Substituímos os valores de D_i:

$$f(A,B,C) = 0 \cdot \overline{A} \cdot \overline{B} \cdot \overline{C} + 0 \cdot \overline{A} \cdot \overline{B} \cdot C + 1 \cdot \overline{A} \cdot B \cdot \overline{C} + \cdots + 1 \cdot A \cdot B \cdot C$$

O MUX combina essas entradas automaticamente com base no seletor.

Diagrama esquemático

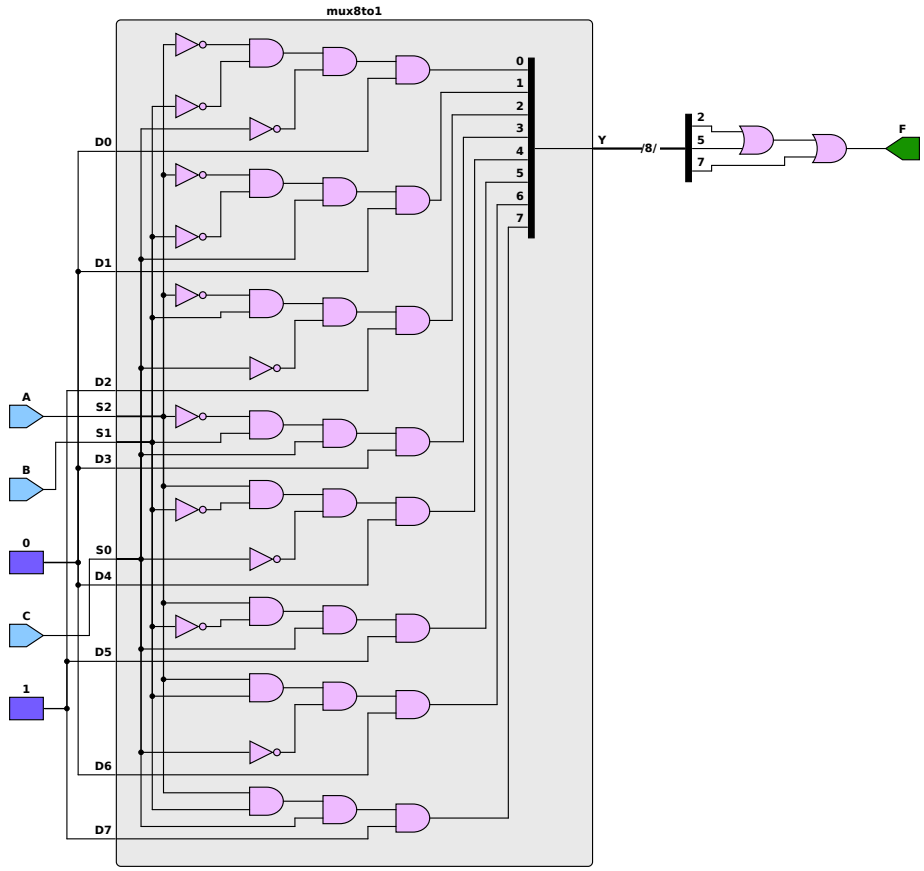

Figura 1: Esquemático da função lógica implementada

Implementação em Verilog

```
1  // Módulo principal para implementar f(A, B, C)
2  module logic_function (
3    input wire A, B, C,          // Entradas da função lógica
4    output wire F                // Saída da função lógica
5  );
6    wire [7:0] Y;                // Vetor para as saídas do MUX
7    wire D0, D1, D2, D3, D4, D5, D6, D7;  // Entradas do MUX
8    // Definição dos valores das entradas do multiplexador
9    assign D0 = 1'b0; // f(A, B, C) = 0 para ABC = 000
10   assign D1 = 1'b0; // f(A, B, C) = 0 para ABC = 001
11   assign D2 = 1'b1; // f(A, B, C) = 1 para ABC = 010
12   assign D3 = 1'b0; // f(A, B, C) = 0 para ABC = 011
13   assign D4 = 1'b0; // f(A, B, C) = 0 para ABC = 100
14   assign D5 = 1'b1; // f(A, B, C) = 1 para ABC = 101
15   assign D6 = 1'b0; // f(A, B, C) = 0 para ABC = 110
16   assign D7 = 1'b1; // f(A, B, C) = 1 para ABC = 111
```

Implementação em Verilog

```
17    // Instância do multiplexador 8x1
18    mux8to1 mux_inst (
19        .S2(A),
20        .S1(B),
21        .S0(C),
22        .D0(D0),
23        .D1(D1),
24        .D2(D2),
25        .D3(D3),
26        .D4(D4),
27        .D5(D5),
28        .D6(D6),
29        .D7(D7),
30        .Y(Y)
31    );
```

Implementação em Verilog

```
32    // Saída da função lógica conectada à porta OR
33    assign F = Y[2] | Y[5] | Y[7]; // OR das saídas selecionadas
34  endmodule
35  // Multiplexador 8x1 com entradas de dados como terminais
36  module mux8to1 (
37    input wire S2, S1, S0,         // Entradas de seleção (A, B, C)
38    input wire D0, D1, D2, D3,   // Entradas de dados
39    input wire D4, D5, D6, D7,
40    output wire [7:0] Y            // Saídas do multiplexador
41  );
42    assign Y[0] = (~S2 & ~S1 & ~S0 & D0);
43    assign Y[1] = (~S2 & ~S1 & S0 & D1);
44    assign Y[2] = (~S2 & S1 & ~S0 & D2);
45    assign Y[3] = (~S2 & S1 & S0 & D3);
46    assign Y[4] = (S2 & ~S1 & ~S0 & D4);
47    assign Y[5] = (S2 & ~S1 & S0 & D5);
48    assign Y[6] = (S2 & S1 & ~S0 & D6);
49    assign Y[7] = (S2 & S1 & S0 & D7);
50  endmodule
```

Exemplo: Número de entradas maior que o número de entradas de seleção

Função Lógica

A função lógica a ser implementada é:

$$f(A, B, C) = A \cdot \overline{B} \cdot \overline{C} + \overline{A} \cdot B \cdot \overline{C}$$

Estratégia de Decomposição

- O número de variáveis da função lógica é 3 (A, B, C).
- O MUX utilizado tem **2 entradas de seleção** (4 entradas).
- As variáveis B e C serão conectadas às entradas de seleção do MUX.
- A variável A será utilizada para determinar as entradas de dados do MUX.

A função é reescrita de forma a expressar as variáveis de entrada como:

- As combinações de B e C são utilizadas para selecionar as entradas do MUX.
- A variável A influencia os valores das entradas do MUX.

Tabela Verdade

A tabela verdade da função lógica é construída da seguinte forma:

A	B	C	$f(A,B,C)$
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

Configuração das Entradas do MUX

As entradas do MUX são configuradas de acordo com as combinações de B e C, considerando o impacto de A:

B	C	Entrada do MUX (I)
0	0	A
0	1	0
1	0	$\overline{A}$
1	1	0

Circuito Lógico

- **Entradas de Seleção**: B e C são conectadas às entradas de seleção do MUX.
- **Entradas de Dados**:
  - $I_0 = A$
  - $I_1 = 0$
  - $I_2 = \overline{A}$
  - $I_3 = 0$
- **Saída**: A saída do MUX é a função lógica $f(A, B, C)$.

Diagrama esquemático

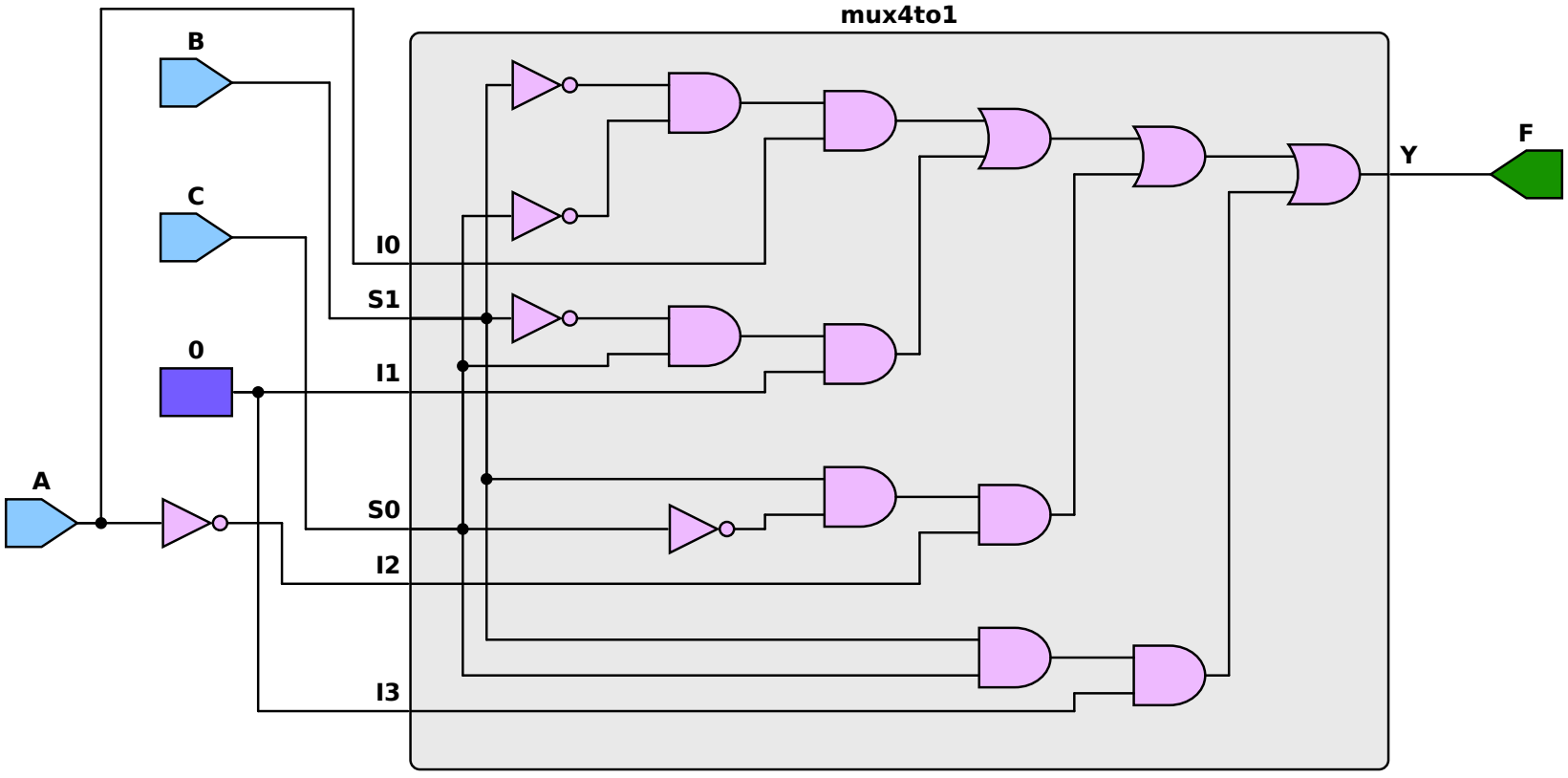

Figura 2: Esquemático da função implementada

Implementação em verilog

```
1  // Módulo principal para implementar f(A,B,C)
2  module logic_function (
3      input wire A, B, C,          // Entradas da função lógica
4      output wire F                // Saída da função lógica
5  );
6      wire I0, I1, I2, I3;         // Entradas de dados do MUX
7
8      // Definição das entradas do multiplexador
9      assign I0 = A;               // Para B=0, C=0
10     assign I1 = 1'b0;            // Para B=0, C=1
11     assign I2 = ~A;              // Para B=1, C=0
12     assign I3 = 1'b0;            // Para B=1, C=1
```

Implementação em verilog

```
13      // Instância do MUX 4x1
14      mux4to1 mux_inst (
15          .S1(B),
16          .S0(C),
17          .I0(I0),
18          .I1(I1),
19          .I2(I2),
20          .I3(I3),
21          .Y(F)                   // Saída do MUX é a função lógica
22      );
23  endmodule
```

Implementação em verilog

```
24 // Multiplexador 4x1 com entradas de dados determinadas pela
       decomposição
25 module mux4to1 (
26     input wire S1, S0,          // Entradas de seleção (B, C)
27     input wire I0, I1, I2, I3, // Entradas de dados
28     output wire Y               // Saída do multiplexador
29 );
30     assign Y = (~S1 & ~S0 & I0) |  // Seleção para I0
31                (~S1 & S0 & I1)  |  // Seleção para I1
32                (S1 & ~S0 & I2)  |  // Seleção para I2
33                (S1 & S0 & I3);    // Seleção para I3
34 endmodule
```

Explicação

1. **Redução de Variáveis**:
   - A variável A foi utilizada como entrada de dados do MUX.
   - As variáveis B e C são suficientes para controlar as entradas de seleção.
2. **Configuração do MUX**:
   - As entradas do MUX são configuradas para refletir os valores de $f(A, B, C)$ em cada combinação de B e C.
3. **Generalização**:
   - Para funções com mais variáveis do que entradas de seleção disponíveis, pode-se usar uma hierarquia de multiplexadores para implementar funções mais complexas.

Vantagens

1. **Simplicidade**: O multiplexador pode implementar qualquer função lógica diretamente a partir de sua tabela verdade.
2. **Flexibilidade**: Um único multiplexador pode ser usado para diversas funções apenas configurando as entradas.
3. **Escalabilidade**: Múltiplos multiplexadores podem ser cascateados para funções maiores.

Os multiplexadores são fundamentais no design digital, especialmente em circuitos compactos que exigem flexibilidade e eficiência na implementação.

Exercícios

Exercício 1

1. Descreva um multiplexador de duas linhas de seleção e 4 linhas de entrada.
   1. Utilize apenas escalares.
   2. Nomeie o módulo **multiplexador_4x1**.
2. Verifique o esquemático gerado.
3. Faça um arquivo de *testbench*, conforme mostra a figura, e teste o módulo.

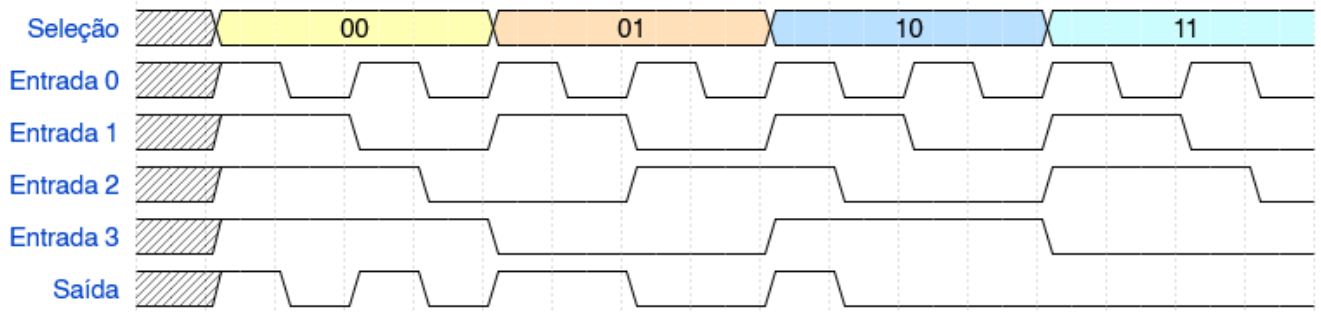

Figura 3: Formas de onda para o *testbench*

Exercício 2

1. Utilize o módulo **multiplexador_4x1** do exercício 1 para implementar a seguinte função lógica:

$$f(A, B) = \overline{A} \cdot \overline{B} + A \cdot \overline{B}$$

2. Obtenha a tabela verdade da função e teste a implementação através de um arquivo de *testbench*.

Exercício 3

1. Utilize o módulo **multiplexador** do exercício 1 para implementar a seguinte função lógica:

$$f(A, B, C) = A \cdot \overline{B} \cdot \overline{C} + \overline{A} \cdot B \cdot \overline{C}$$

2. Obtenha a tabela verdade da função e teste a implementação através de um arquivo de *testbench*.