Implementando funções lógicas com decodificador

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 Decodificadores em Circuitos Digitais
- Implementação de funções lógicas
- Exemplo: Número de variáveis igual ao número de entradas do decodificador
- Exemplo: Número de variáveis maior que o número de entradas do decodificador
- Exercícios

Revisão sobre Decodificadores em Circuitos Digitais

Definição

Um **decodificador** é um circuito combinacional que converte códigos binários de entrada em uma única saída ativa correspondente. Em termos simples, ele traduz um número binário em um formato que ativa uma das várias linhas de saída.

Características Principais

1. **Entradas e Saídas**:
   - Um decodificador com n entradas tem 2^n saídas.
   - Por exemplo, um decodificador de 2 entradas possui $2^2 = 4$ saídas.
2. **Ativação de Saídas**:
   - Apenas uma saída é ativada (lógica 1 ou lógica 0) para qualquer combinação de entrada.
   - O restante das saídas permanece inativo.
3. **Aplicações**:
   - Seleção de linhas de memória.
   - Endereçamento em sistemas computacionais.
   - Controle em circuitos sequenciais.
   - Conversão de códigos binários em representações humanas (como displays de 7 segmentos).

Tabela Verdade de um Decodificador 2x4

Entradas $(A_1 A_0)$	Saídas (Y_0, Y_1, Y_2, Y_3)
00	1000
01	0100
10	0010
11	0001

Implementação Lógica

1. Cada saída é gerada como uma expressão lógica que depende das combinações das entradas.
2. Fórmulas para o decodificador 2x4:
   - $Y_0 = \overline{A_1} \cdot \overline{A_0}$
   - $Y_1 = \overline{A_1} \cdot A_0$
   - $Y_2 = A_1 \cdot \overline{A_0}$
   - $Y_3 = A_1 \cdot A_0$

Essas expressões podem ser implementadas usando **portas AND, NOT** e **outras portas básicas**.

Símbolo e Representação

Em circuitos digitais, o decodificador é geralmente representado como um bloco com:

- Entradas rotuladas (como $A_0, A_1, ...$).
- Saídas numeradas ($Y_0, Y_1, Y_2, ...$).

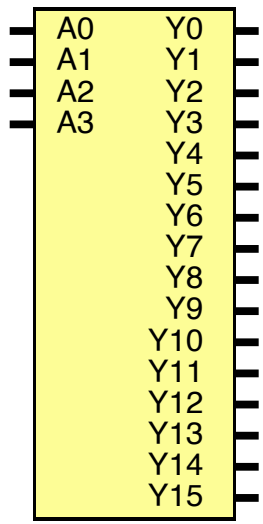

Figura 1: Símbolo de um decodificador de 4x16.

Variações

1. **Decodificadores com Habilitação**:
   - Além das entradas de dados, possuem uma entrada de habilitação (Enable). O circuito só funciona quando a habilitação está ativa.
2. **Decodificadores de Nível Superior**:
   - Decodificadores 3x8, 4x16, etc., seguem o mesmo princípio, mas lidam com mais entradas e saídas.

Resumo

Decodificadores são blocos essenciais em sistemas digitais para ativação seletiva de linhas ou elementos. Combinam simplicidade lógica e utilidade prática em diversos contextos, como endereçamento e controle de dispositivos.

Implementação de funções lógicas

Utilizando um decodificador

Decodificadores podem ser usados para implementar funções lógicas ao converter as entradas em linhas de saída únicas, que representam as combinações possíveis das variáveis de entrada. Cada linha de saída do decodificador equivale a um dos **minitermos** da tabela verdade da função. Esses minitermos podem ser combinados (usando portas OR, por exemplo) para implementar a função lógica desejada.

Passo-a-passo

1. **Entenda a função lógica** Considere a função lógica dada. Por exemplo:

$$F(A, B, C) = \Sigma(1, 3, 5, 7)$$

Aqui, $\Sigma(1, 3, 5, 7)$ indica que a função F é verdadeira (1) para as combinações de entrada 001, 011, 101, e 111.

Passo-a-passo

❷ **Escolha um decodificador apropriado**

- Para a função $F(A, B, C)$, temos 3 variáveis de entrada.
- Escolhemos um **decodificador 3-to-8 (3x8)**, que possui:
  - 3 entradas (A, B, C).
  - 8 saídas (Y_0 a Y_7), onde cada saída representa uma das 8 combinações possíveis das entradas.

Passo-a-passo

❸ **Identifique as saídas relevantes** Com base nos minitermos, as combinações da função $F(A, B, C)$ são:

$$F = 1 \text{ para } Y_1, Y_3, Y_5, Y_7$$

Isso significa que as saídas Y_1, Y_3, Y_5, Y_7 do decodificador devem ser combinadas.

Passo-a-passo

4. **Combine as saídas** Conecte as saídas relevantes do decodificador a uma **porta OR** para implementar a função. As saídas Y_1, Y_3, Y_5, Y_7 serão combinadas, pois qualquer uma delas ativa F.

Exemplo: Número de variáveis igual ao número de entradas do decodificador

Tabela verdade

A	B	C	Y_0	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7	F
0	0	0	1	0	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0	1
0	1	0	0	0	1	0	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0	1
1	0	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	0	0	0	1	0	0	1
1	1	0	0	0	0	0	0	0	1	0	0
1	1	1	0	0	0	0	0	0	0	1	1

Conexões

1. Conecte as variáveis A, B, C às entradas do decodificador.
2. As saídas do decodificador serão:
   - $Y_0 = \overline{A}\overline{B}\overline{C}$
   - $Y_1 = \overline{A}\overline{B}C$
   - $Y_2 = \overline{A}B\overline{C}$
   - ...
   - $Y_7 = ABC$
3. Conecte Y_1, Y_3, Y_5, Y_7 a uma porta OR:

$$F = Y_1 + Y_3 + Y_5 + Y_7$$

Diagrama

Um diagrama típico usando um decodificador pode ser descrito como:

1. Entradas A, B, C conectadas ao decodificador 3x8.
2. Linhas de saída Y_1, Y_3, Y_5, Y_7 conectadas à porta OR.
3. A saída da porta OR é a função $F(A, B, C)$.

Diagrama

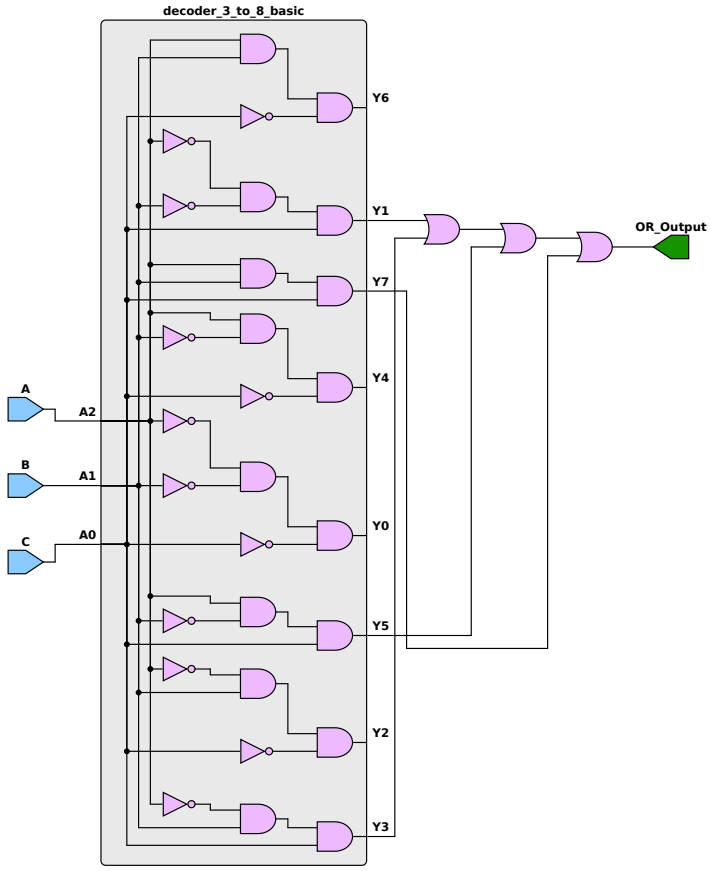

Figura 2: Esquemático da função implementada.

Vantagens de usar decodificadores

1. **Modularidade**: O decodificador abstrai a parte complexa da lógica, representando diretamente os minitermos.
2. **Facilidade de expansão**: Para mais entradas, basta escolher um decodificador com mais saídas.
3. **Eficiência em projetos grandes**: Reduz a necessidade de projetar circuitos lógicos complexos do zero.

Observação

Ao implementar funções lógicas com decodificadores, tenha em mente que o número de variáveis de entrada determina o tipo de decodificador necessário (n-to-2^n). Para funções maiores, múltiplos decodificadores podem ser usados em cascata.

Exemplo: Número de variáveis maior que o número de entradas do decodificador

Função de 4 variáveis com decodificador de 3 entradas

É possível implementar uma função lógica de 4 variáveis utilizando um decodificador de 3 entradas e 8 saídas. A abordagem consiste em usar o decodificador para mapear 3 das variáveis de entrada e utilizar a 4ª variável para modificar ou controlar a lógica das saídas.

Solução

1. **Usar o decodificador de 3 entradas para 8 saídas** para gerar todas as combinações possíveis de 3 bits de entrada, ou seja, A_0, A_1 e A_2.
2. **A 4ª variável** A_3 pode ser usada para habilitar ou desabilitar as saídas do decodificador, ou até modificar as saídas utilizando portas lógicas, como uma porta **OR** ou **AND**.

Raciocínio

- O decodificador de 3 entradas recebe as variáveis A_0, A_1 e A_2, e gera 8 saídas correspondentes às combinações possíveis dessas 3 entradas.
- A 4ª variável A_3 pode ser usada para selecionar ou controlar o uso dessas saídas.

Função Lógica

O decodificador de 3 entradas gera as saídas Y_0, Y_1, Y_2, Y_3, Y_4, Y_5, Y_6 e Y_7. A função lógica F que desejamos implementar pode ser expressa como:

$$F(A_3, A_2, A_1, A_0) = \begin{cases} (Y_1 + Y_3 + Y_5 + Y_7) & \text{se } A_3 = 1 \\ (Y_0 + Y_2 + Y_4 + Y_6) & \text{se } A_3 = 0 \end{cases}$$

Ou seja, a variável A_3 controla quais saídas do decodificador serão ativadas para formar a função lógica.

Função Lógica

Onde:

- $Y_0 = \overline{A_2} \cdot \overline{A_1} \cdot \overline{A_0}$
- $Y_1 = \overline{A_2} \cdot \overline{A_1} \cdot A_0$
- $Y_2 = \overline{A_2} \cdot A_1 \cdot \overline{A_0}$
- $Y_3 = \overline{A_2} \cdot A_1 \cdot A_0$
- $Y_4 = A_2 \cdot \overline{A_1} \cdot \overline{A_0}$
- $Y_5 = A_2 \cdot \overline{A_1} \cdot A_0$
- $Y_6 = A_2 \cdot A_1 \cdot \overline{A_0}$
- $Y_7 = A_2 \cdot A_1 \cdot A_0$

Função Lógica

Com o decodificador de 3 entradas e 8 saídas, a função lógica final será:

$$F = (Y_1 + Y_3 + Y_5 + Y_7) \cdot A_3 + (Y_0 + Y_2 + Y_4 + Y_6) \cdot \overline{A_3}$$

Essa equação descreve a função lógica que será implementada utilizando o decodificador e a porta OR. Ela utiliza a combinação das variáveis de entrada A_0, A_1, A_2 e A_3, controlando a ativação das saídas de acordo com o valor de A_3.

Código Verilog

```
1  module func2 (
2      input A3,      // Entrada A3 (4ª variável)
3      input A2,      // Entrada A2
4      input A1,      // Entrada A1
5      input A0,      // Entrada A0
6      output F       // Saída da função lógica
7  );
8      // Saídas do decodificador de 3 entradas
9      wire Y0, Y1, Y2, Y3, Y4, Y5, Y6, Y7;
```

Código Verilog

```
10      // Instância do decodificador de 3 para 8
11      decoder_3_to_8_basic decoder (
12          .A0(A0),   // Entrada A0
13          .A1(A1),   // Entrada A1
14          .A2(A2),   // Entrada A2
15          .Y0(Y0),   // Saída Y0
16          .Y1(Y1),   // Saída Y1
17          .Y2(Y2),   // Saída Y2
18          .Y3(Y3),   // Saída Y3
19          .Y4(Y4),   // Saída Y4
20          .Y5(Y5),   // Saída Y5
21          .Y6(Y6),   // Saída Y6
22          .Y7(Y7)    // Saída Y7
23      );
24      // Combinação das saídas utilizando a 4ª variável A3
25      // Aqui a lógica depende da sua função
26      assign F = (A3) ? (Y1 | Y3 | Y5 | Y7) : (Y0 | Y2 | Y4 | Y6);
27  endmodule
```

Diagrama esquemáticos

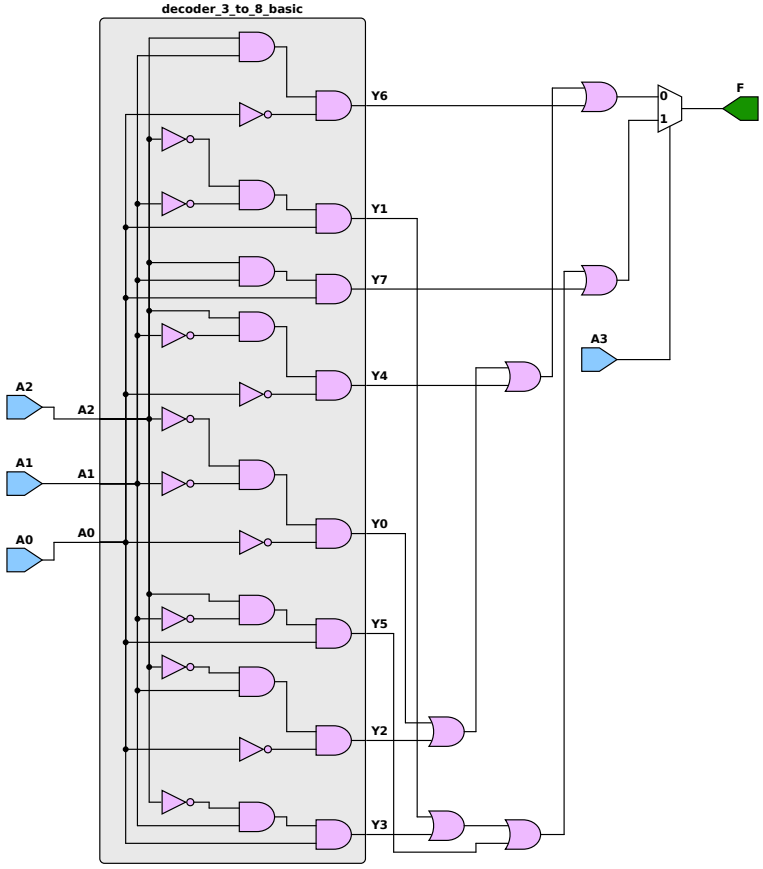

Figura 3: Esquemático do função lógica.

Exercícios

Exercício 1

1. Descreva um decodificador de duas linhas de entrada e 4 linhas de saída.
   1. Utilize apenas escalares.
   2. Nomeie o módulo **decodificador_2x4**.
2. Verifique o esquemático gerado.
3. Faça um arquivo de *testbench*, conforme mostra a figura, e teste o módulo.

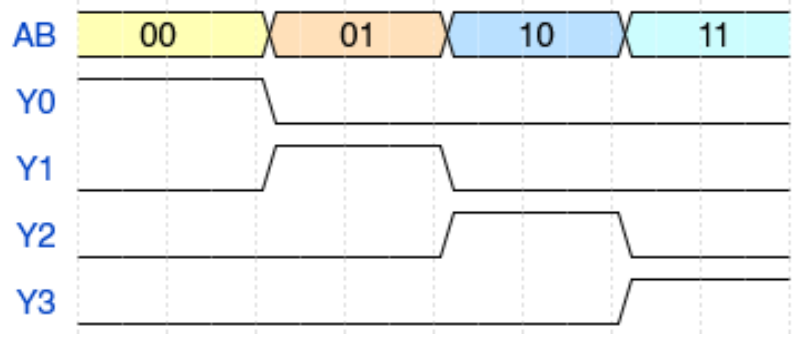

Figura 4: Formas de onda para o *testbench*

Exercício 2

1. Utilize o módulo **decodificador_2x4** 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 **decodificador_2x4** 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*.