Aulas 6 e 7
Codificadores

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

Histórico de Revisões		
16 de fevereiro de 2025	1.0	Primeira versão do documento.

Tópicos

- Introdução
- Descrição em Verilog
- Tipos de Codificador
- Simulação e Verificação
- Atividades Hands-on

Introdução

Objetivos

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

Definição

- Definição: Um circuito codificador é um dispositivo que converte informação de um formato para outro, especificamente de uma entrada ativa para um código de saída.
- Função: Reduzir a quantidade de sinais necessários para representar informação em sistemas digitais.

Conversores de Código vs Codificadores

- Conversores de Código: Traduzem uma informação de um código para outro,.
- Codificadores: Recebem apenas uma entrada ativa por vez, gerando código de saída correspondente.

Os codificadores podem ser considerados um caso particular de conversor de código

O que são Codificadores?

Figura 1: Estrutura de um Codificador

- Codifica M linhas de entrada em N linhas de saída.
- Apenas uma linha de entrada está ativa por vez.

Aplicações

- Teclados e interfaces de entrada
  - Em teclados de computador, cada tecla pressionada é convertida em um código binário antes de ser enviado para o processador.
- Processadores e arquiteturas de CPU
  - Codificadores de prioridade são usados em sistemas de interrupção de CPU para selecionar a tarefa mais urgente a ser processada.
- Redes de comunicação e segurança da informação
  - Codificadores são usados em sistemas de telecomunicação para selecioção de canais e representação em código binário
- Sensores e sistemas embarcados
  - Sensores em automação industrial e dispositivos IoT (Internet das Coisas) utilizam codificadores para organizar sinais múltiplos.

Descrição em Verilog

Tipos de Descrição

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

Tipos de Codificador

Codificadores

- Codificador Binário Simples
- Codificador de Prioridade
- Codificador de Teclado
  - Decimal para BCD
  - Decimal para Excesso de 3

Codificador Binário

- Converte M entradas em um código binário correspondente de N bits, sendo que:

$$N = log_2 M$$

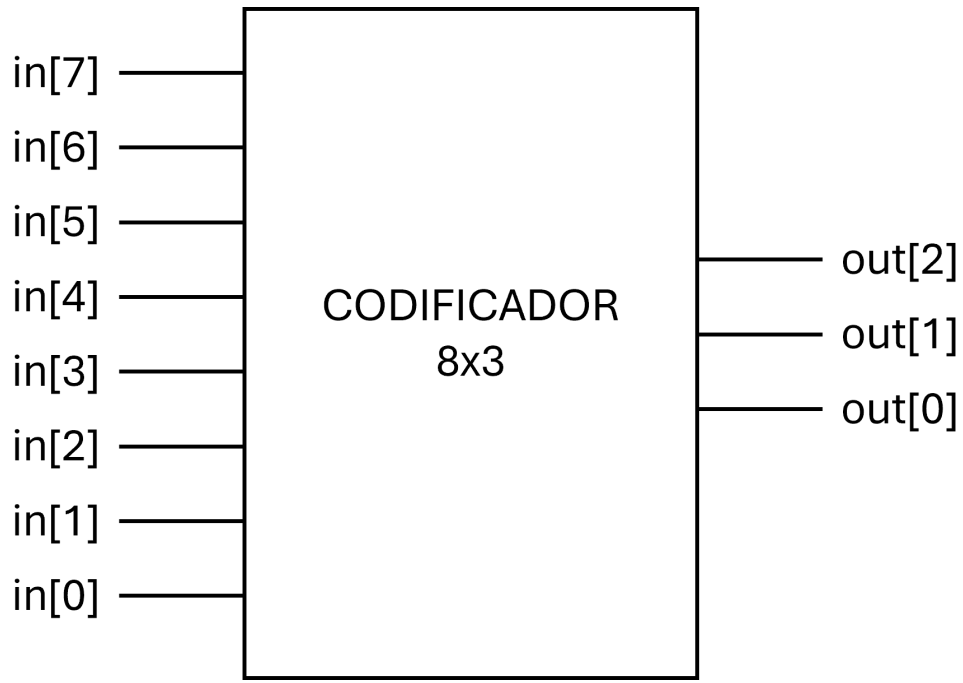

Figura 2: Exemplo de Codificador 8x3

Tabela Verdade

Tabela de um codificador binário de 8 para 3 linhas.

in[7:0]	out[2:0]
00000001	000
00000010	001
00000100	010
00001000	011
00010000	100
00100000	101
01000000	110
10000000	111

Expressões Lógicas

As saídas `out[2:0]` são obtidas a partir das entradas `in[7:0]` usando as seguintes expressões lógicas:

$$\mathsf{out}[2] = \mathsf{in}[4] + \mathsf{in}[5] + \mathsf{in}[6] + \mathsf{in}[7]$$

$$\mathsf{out}[1] = \mathsf{in}[2] + \mathsf{in}[3] + \mathsf{in}[6] + \mathsf{in}[7]$$

$$\mathsf{out}[0] = \mathsf{in}[1] + \mathsf{in}[3] + \mathsf{in}[5] + \mathsf{in}[7]$$

Descrição Fluxo de Dados

```
1  module Codificador_Binario (
2      input [7:0] in,
3      output [2:0] out
4  );
5      assign out[2] = in[4] | in[5] | in[6] | in[7];
6      assign out[1] = in[2] | in[3] | in[6] | in[7];
7      assign out[0] = in[1] | in[3] | in[5] | in[7];
8  endmodule
```

Descrição Comportamental

```
1  module Codificador_Binario2 (
2      input wire [7:0] in,
3      output reg [2:0] out
4  );
5      always @(*) begin
6          case (in)
7              8'b10000000: out = 3'b111;
8              8'b01000000: out = 3'b110;
9              8'b00100000: out = 3'b101;
10             8'b00010000: out = 3'b100;
11             8'b00001000: out = 3'b011;
12             8'b00000100: out = 3'b010;
13             8'b00000010: out = 3'b001;
14             8'b00000001: out = 3'b000;
15             default: out = 3'b000;
16         endcase
17     end
18 endmodule
```

Codificador de Prioridade

- Se duas ou mais entradas forem acionadas simultaneamente no codificador anterior, as saídas poderão apresentar códigos inválidos. (Inclusive com comportamentos diferentes a depender do tipo de descrição).
- Para evitar esse erro, deve-se ter uma prioridade nas entradas.
- Se mais de uma for acionada ao mesmo tempo, somente uma delas deve ter efeito na saída.

Tabela Verdade

Tabela de um codificador binário de prioridade de 8 para 3 linhas.

in[7:0]	out[2:0]
00000001	000
0000001x	001
000001xx	010
00001xxx	011
0001xxxx	100
001xxxxx	101
01xxxxxx	110
1xxxxxxx	111

Expressões Lógicas

$$out[2] = in[4] + in[5] + in[6] + in[7]$$

$$out[1] = (in[2] \cdot \overline{in[4]} \cdot \overline{in[5]}) + (in[3] \cdot \overline{in[4]} \cdot \overline{in[5]}) + in[6] + in[7]$$

$$out[0] = (in[1] \cdot \overline{in[2]} \cdot \overline{in[4]} \cdot \overline{in[6]}) + (in[3] \cdot \overline{in[4]} \cdot \overline{in[6]}) +$$

$$(in[5] \cdot \overline{in[6]}) + in[7]$$

Descrição Fluxo de Dados

```
1  module Codificador_Prioridade (
2      input  [7:0] in,
3      output [2:0] out
4  );
5      assign out[2] = in[4] | in[5] | in[6] | in[7];
6      assign out[1] = in[2] & ~in[4] & ~in[5] | in[3] & ~in[4] &
           ~in[5] | in[6] | in[7];
7      assign out[0] = in[1] & ~in[2] & ~in[4] & ~in[6] | in[3] &
           ~in[4] & ~in[6] | in[5] & ~in[6] | in[7];
8
9  endmodule
```

Descrição comportamental

- case: Compara todos os bits exatamente como estão escritos.
  - Não permite lidar com valores `x` para representar casos do tipo "não importa"
- casez: Trata os valores `z` como "não importa"
- casex: Trata os valores `x` e `z` como "não importa"

case

- A instrução `case` trata `x` e `z` exatamente como são.
- Uma expressão `case` contendo `x` ou `z` só corresponderá a um item que contenha `x` ou `z` nas mesmas posições de bits correspondentes.
- Se nenhum item `case` corresponder, então o item `default` será executado.

casez

- A instrução `casez` trata `z` como "não importa" (don't care).
- O valor `z` pode corresponder a qualquer valor de entrada 0, 1, `x` ou `z`.
- Bits `x` ainda são tratados como valores exatos.

casex

- A instrução `casex` trata tanto `x` quanto `z` como "não importa" (don't care).
- Os valores `x` e `z` podem corresponder a qualquer valor de entrada 0, 1, `x` ou `z`.

Descrição Comportamental

```
1  module Codificador_Prioridade2 (
2      input wire [7:0] in,
3      output reg [2:0] out
4  );
5      always @(*) begin
6          casex (in)
7              8'b1xxxxxxx: out = 3'b111;
8              8'b01xxxxxx: out = 3'b110;
9              8'b001xxxxx: out = 3'b101;
10             8'b0001xxxx: out = 3'b100;
11             8'b00001xxx: out = 3'b011;
12             8'b000001xx: out = 3'b010;
13             8'b0000001x: out = 3'b001;
14             default: out = 3'b000;
15         endcase
16     end
17 endmodule
```

Codificador de Teclado para BCD

- Um codificador de teclado BCD recebe como entrada sinais correspondentes às teclas pressionadas em um teclado numérico.
- Gera uma saída que representa o valor decimal correspondente à tecla pressionada.
- É usado em dispositivos como calculadoras e sistemas de entrada de dados, onde números decimais precisam ser convertidos para o formato binário.

Codificador de Teclado para BCD

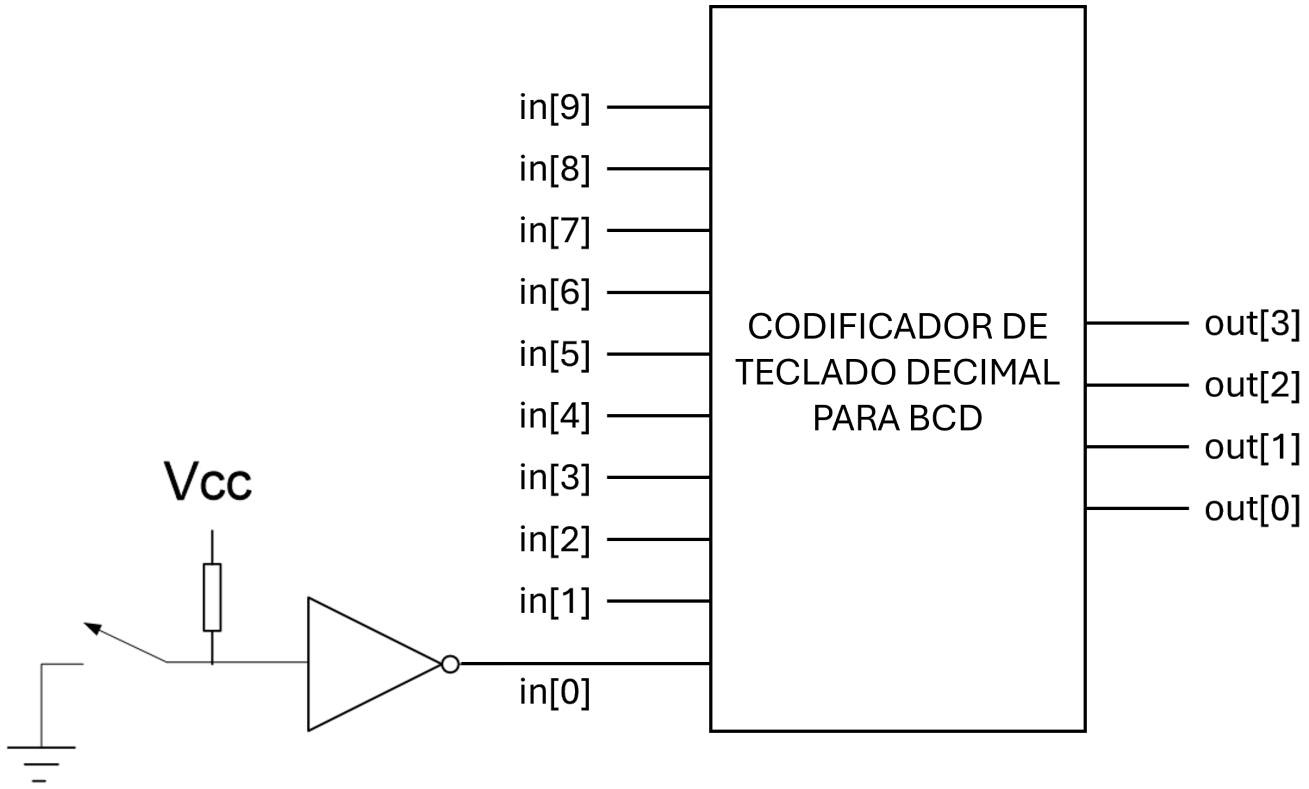

Figura 3: Codificador de Teclado Decimal para código BCD8421

Tabela Verdade

Tabela de um codificador de teclado decimal para BDC.

in[9:0]	out[3:0]
0000000001	0000
0000000010	0001
0000000100	0010
0000001000	0011
0000010000	0100
0000100000	0101
0001000000	0110
0010000000	0111
0100000000	1000
1000000000	1001

Expressões Lógicas

$$out[3] = in[8] + in[9]$$

$$out[2] = in[4] + in[5] + in[6] + in[7]$$

$$out[1] = in[2] + in[3] + in[6] + in[7]$$

$$out[0] = in[1] + in[3] + in[5] + in[7] + in[9]$$

Descrição Fluxo de Dados

```
1  module Codificador_Teclado_BCD (
2      input  [9:0] in,   // Entradas decimais (somente um bit
           ativo por vez)
3      output [3:0] out   // Saída BCD correspondente
4  );
5      // Expressões lógicas para cada bit da saída BCD
6      assign out[3] = in[8] | in[9];
7      assign out[2] = in[4] | in[5] | in[6] | in[7];
8      assign out[1] = in[2] | in[3] | in[6] | in[7];
9      assign out[0] = in[1] | in[3] | in[5] | in[7] | in[9];
10
11 endmodule
```

Descrição Comportamental

```
1  module Codificador_Teclado_BCD2 (
2      input  wire [9:0] in,   // Entradas decimais (somente um
           bit ativo por vez)
3      output reg  [3:0] out   // Saída BCD correspondente
4  );
5      always @(*) begin
6          case (in)
7              10'b0000000001: out = 4'b0000; // 0
8              10'b0000000010: out = 4'b0001; // 1
9              10'b0000000100: out = 4'b0010; // 2
10             10'b0000001000: out = 4'b0011; // 3
11             10'b0000010000: out = 4'b0100; // 4
12             10'b0000100000: out = 4'b0101; // 5
13             10'b0001000000: out = 4'b0110; // 6
14             10'b0010000000: out = 4'b0111; // 7
15             10'b0100000000: out = 4'b1000; // 8
16             10'b1000000000: out = 4'b1001; // 9
17             default:        out = 4'b0000; // Caso inválido
18         endcase
19     end
20 endmodule
```

Simulação e Verificação

Testbench

```
1  module Codificador_Teclado_BCD_tb();
2      reg  [9:0] in;
3      wire [3:0] out;
4
5      Codificador_Teclado_BCD2 uut (.in(in), .out(out));  //
           Instanciação do módulo
6      initial begin
7          $monitor("in = %b, out = %b", in, out);
8          in = 10'b0000000001; #10; // 0 -> 0000
9          in = 10'b0000000010; #10; // 1 -> 0001
10         in = 10'b0000000100; #10; // 2 -> 0010
11         in = 10'b0000001000; #10; // 3 -> 0011
12         in = 10'b0000010000; #10; // 4 -> 0100
13         in = 10'b0000100000; #10; // 5 -> 0101
14         in = 10'b0001000000; #10; // 6 -> 0110
15         in = 10'b0010000000; #10; // 7 -> 0111
16         in = 10'b0100000000; #10; // 8 -> 1000
17         in = 10'b1000000000; #10; // 9 -> 1001
18         $stop;
19     end
20 endmodule
```

Forma de Onda

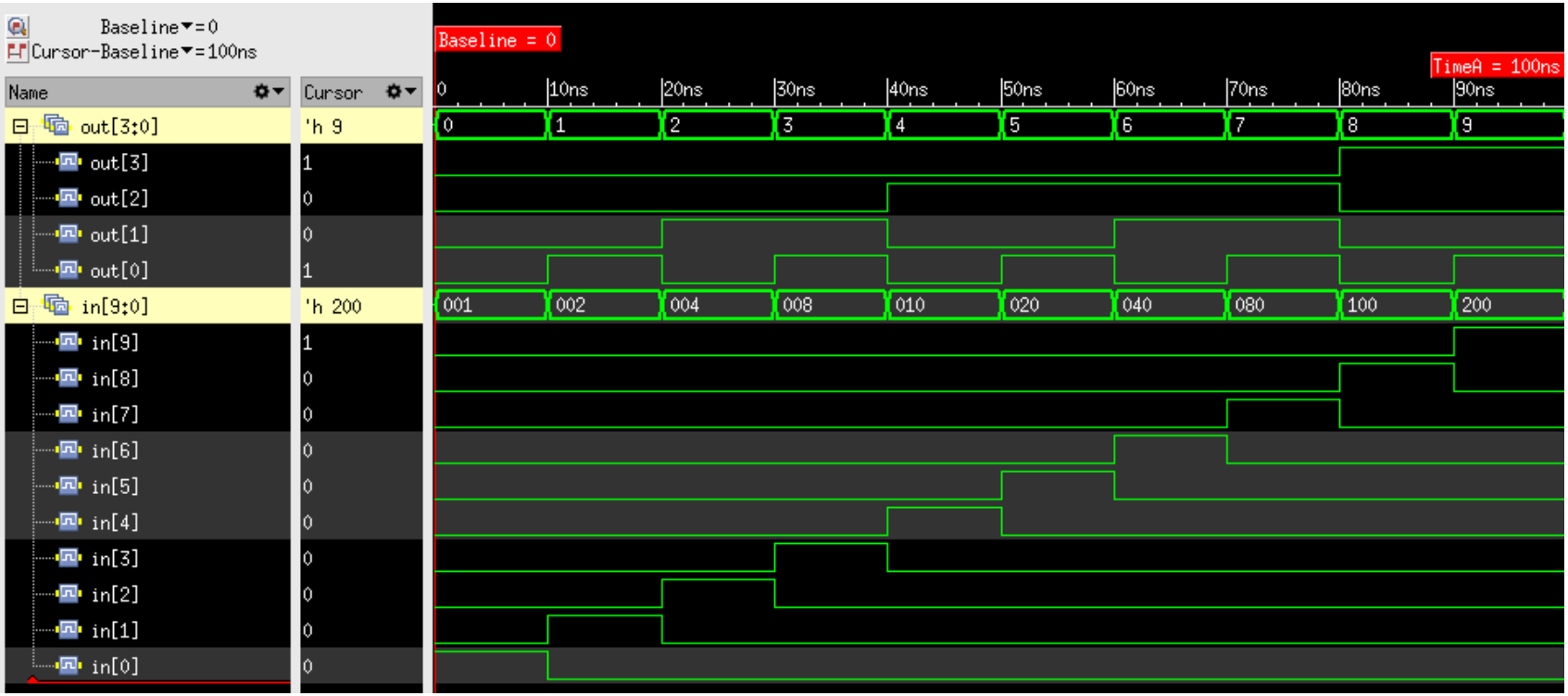

Figura 4: Resultado da Simulação

Atividades Hands-on

Atividade 1

- Monte uma descrição comportamental de um codificador binário de prioridade (16 para 4).
- Elabore um Testbench e simule a operação do circuito.

Atividade 2

- Monte uma descrição comportamental de um codificador de teclado decimal para Excesso de 3
- Elabore um Testbench e simule a operação do circuito.

Atividade 3

- Monte uma descrição estrutural de um codificador binário simples 16 para 4 utilizando dois módulos de um codificador 8 para 3 e mais a lógica adicional que for necessária.
- Essa lógica adicional pode ser descrita no formato de fluxo de dados diretamente no módulo topo do circuito.
- Elabore um Testbench e simule a operação do circuito.

Atividade 4

- Elabore uma descrição parametrizável de um codificador binário simples, com M bits de entrada e N bits de saída, sendo que:

$$N = log_2(M)$$

- Elabore um testbench e simule para M = 16;