Lógica Digital (1001351)

Circuitos Combinacionais: Conversores de códigos

Prof. Edilson Kato
kato@ufscar.br

Prof. Maurício Figueiredo
mauricio@ufscar.br

Prof. Ricardo Menotti
menotti@ufscar.br

Prof. Roberto Inoue
rsinoue@ufscar.br

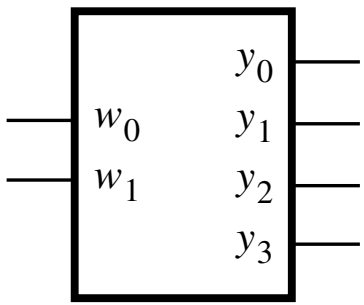
Departamento de Computação
Universidade Federal de São Carlos

Atualizado em: 18 de abril de 2019

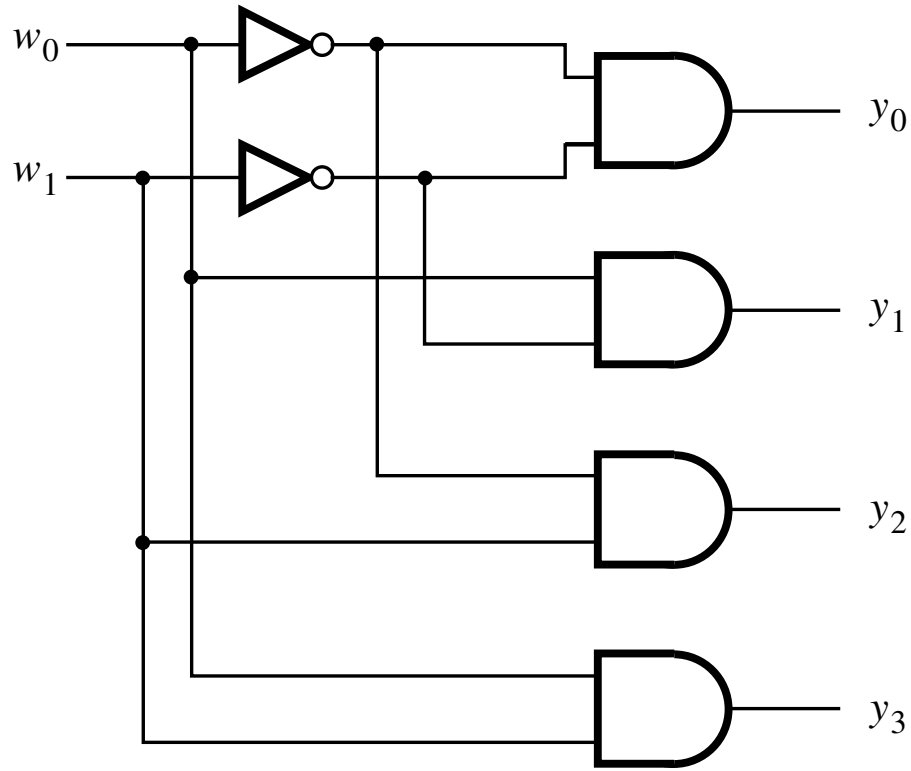
Decodificadores

w_1	w_0	y_0	y_1	y_2	y_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

(a) Truth table



(b) Graphical symbol



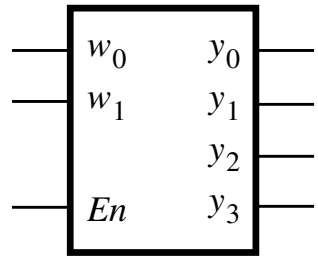
(c) Logic circuit

Figure 4.13 A 2-to-4 decoder.

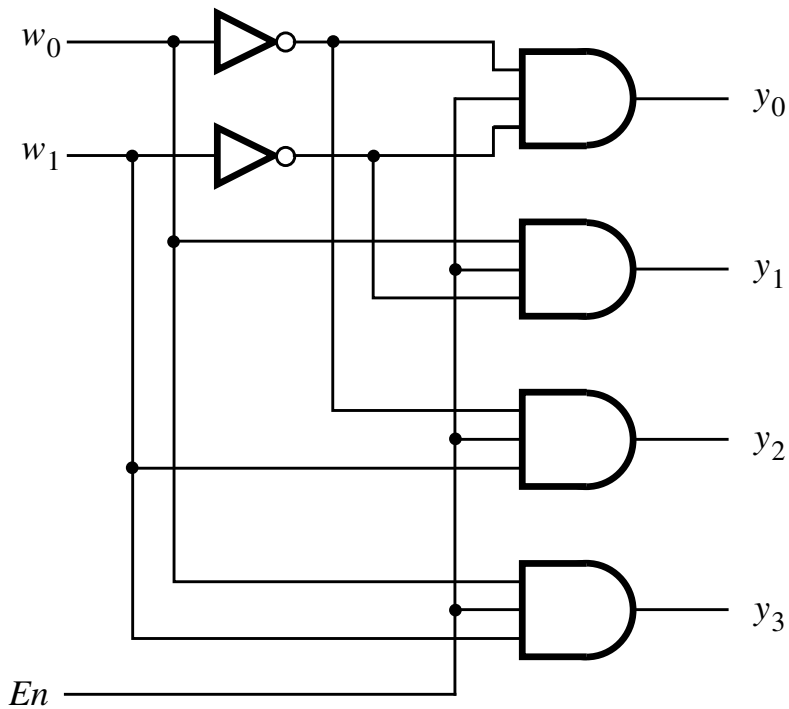
Decodificadores

En	w_1	w_0	y_0	y_1	y_2	y_3
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1
0	x	x	0	0	0	0

(a) Truth table



(b) Graphical symbol



(c) Logic circuit

figure4.31.v

```
1  module dec2to4 (W, En, Y);
2    input [1:0]W;
3    input En;
4    output reg [0:3] Y;
5
6    always @(W, En)
7      case ({En,W})
8        3'b100: Y = 4'b1000;
9        3'b101: Y = 4'b0100;
10       3'b110: Y = 4'b0010;
11       3'b111: Y = 4'b0001;
12       default: Y = 4'b0000;
13     endcase
14 endmodule
```

figure4.32.v

```
1  module dec2to4 (W, En, Y);
2    input [1:0] W;
3    input En;
4    output reg [0:3] Y;
5
6    always @(W, En)
7    begin
8      if (En == 0)
9        Y = 4'b0000;
10     else
11       case (W)
12         0: Y = 4'b1000;
13         1: Y = 4'b0100;
14         2: Y = 4'b0010;
15         3: Y = 4'b0001;
16       endcase
17   end
18 endmodule
```

figure4.37.v

```
1  module dec2to4 (W, En, Y);
2    input [1:0] W;
3    input En;
4    output reg [0:3] Y;
5    integer k;
6
7    always @(W, En)
8      for (k = 0; k <= 3; k = k+1)
9        if ((W == k) && (En == 1))
10         Y[k] = 1;
11       else
12         Y[k] = 0;
13 endmodule
```

Decodificadores

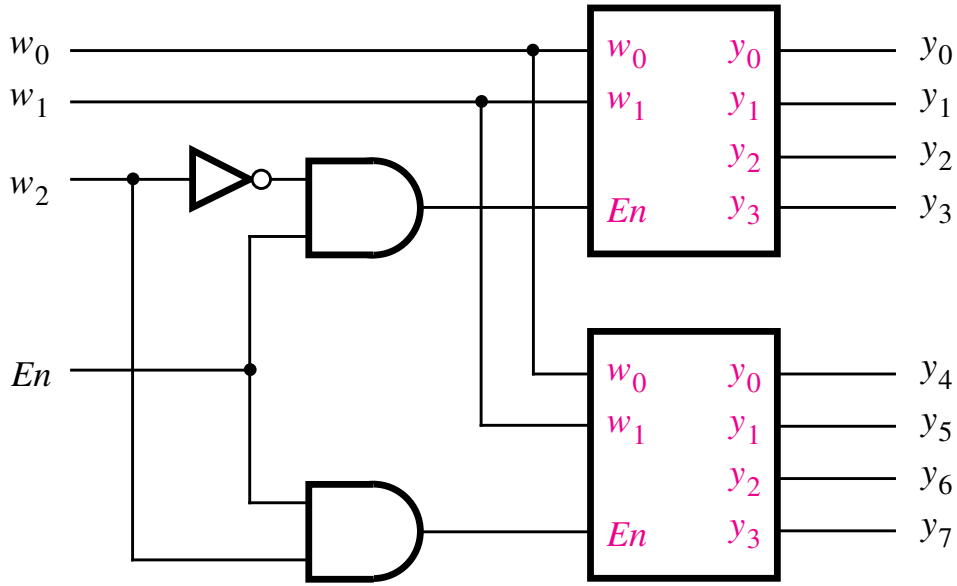


Figure 4.15 A 3-to-8 decoder using two 2-to-4 decoders.

Decodificadores

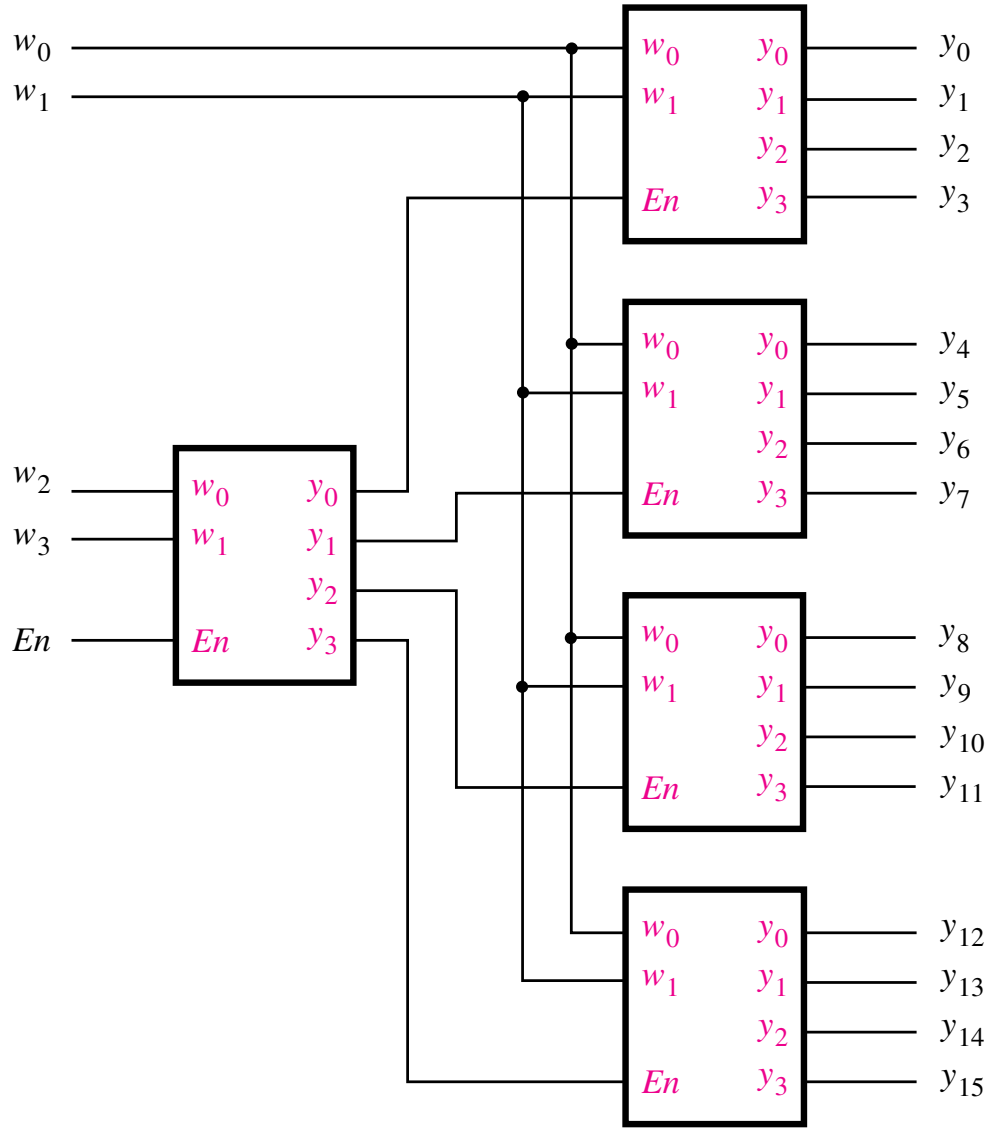


Figure 4.16 A 4-to-16 decoder built using a decoder tree.

figure4.33.v

```
1  module dec4to16 (W, En, Y);
2    input [3:0] W;
3    input En;
4    output [0:15] Y;
5    wire [0:3] M;
6
7    dec2to4 Dec1 (W[3:2], M[0:3], En);
8    dec2to4 Dec2 (W[1:0], Y[0:3], M[0]);
9    dec2to4 Dec3 (W[1:0], Y[4:7], M[1]);
10   dec2to4 Dec4 (W[1:0], Y[8:11], M[2]);
11   dec2to4 Dec5 (W[1:0], Y[12:15], M[3]);
12 endmodule
```

Decodificadores

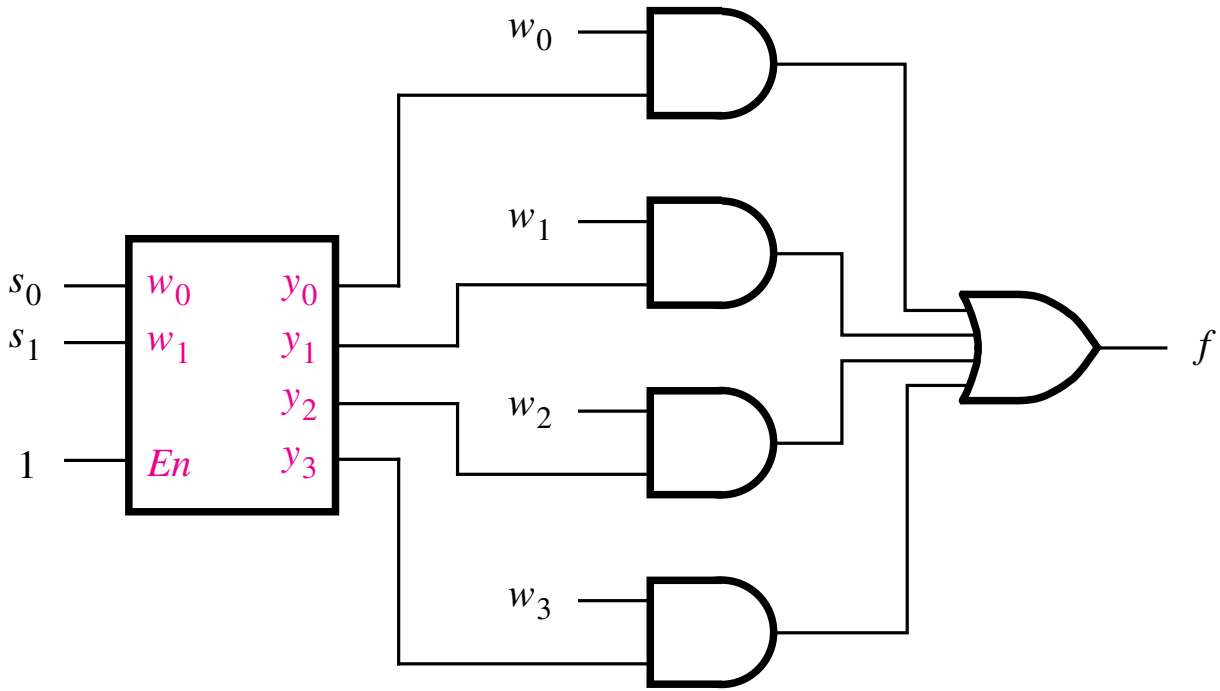


Figure 4.17 A 4-to-1 multiplexer built using a decoder.

Decodificadores

$$f(w_1, w_2, w_3) = \sum m(0, 1, 3, 4, 6, 7)$$

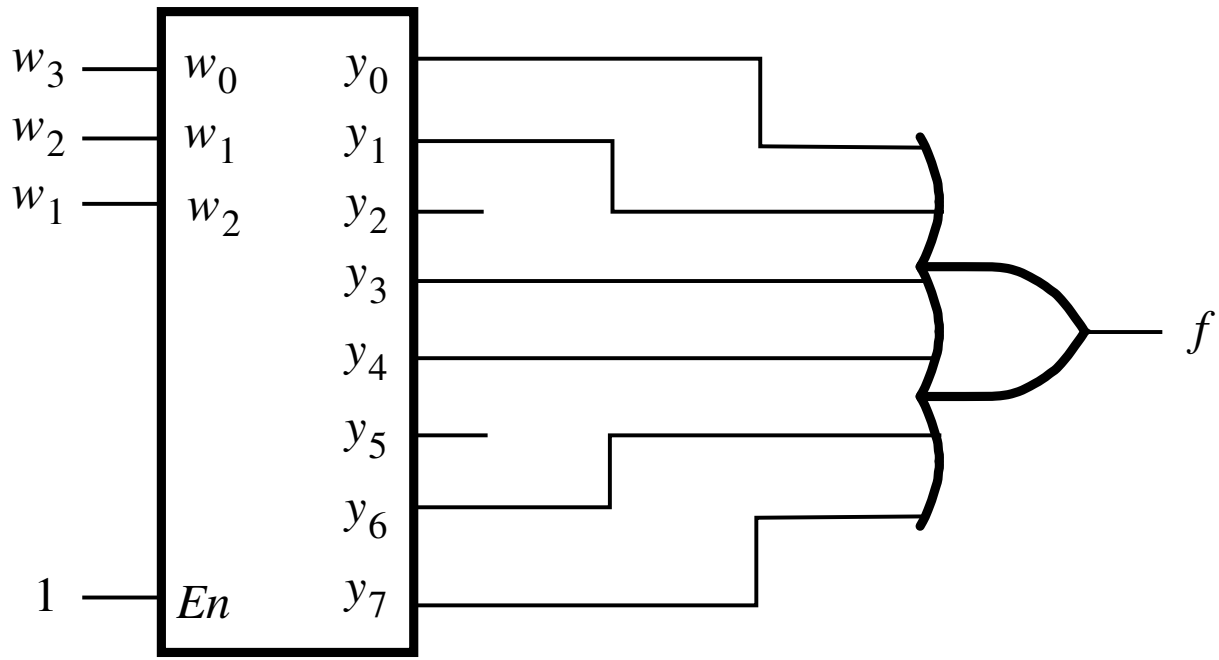
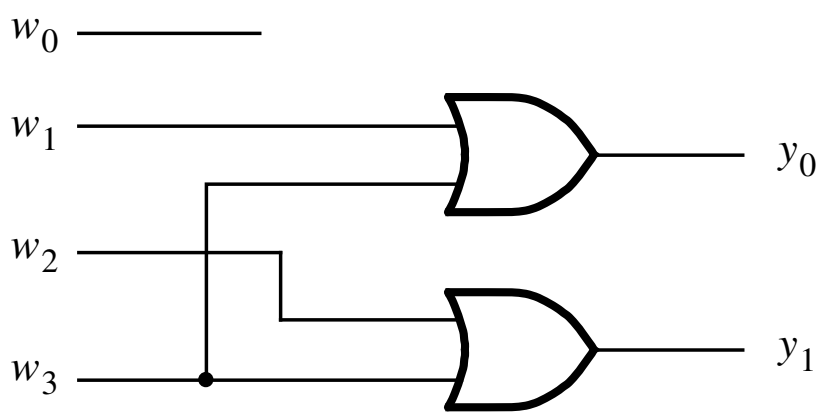


Figure 4.44 Circuit for Example 4.24.

Codificadores

w_3	w_2	w_1	w_0	y_1	y_0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

(a) Truth table



(b) Circuit

Figure 4.19 A 4-to-2 binary encoder.

Codificadores

w_3	w_2	w_1	w_0	y_1	y_0	z
0	0	0	0	d	d	0
0	0	0	1	0	0	1
0	0	1	x	0	1	1
0	1	x	x	1	0	1
1	x	x	x	1	1	1

Figure 4.20 Truth table for a 4-to-2 priority encoder.

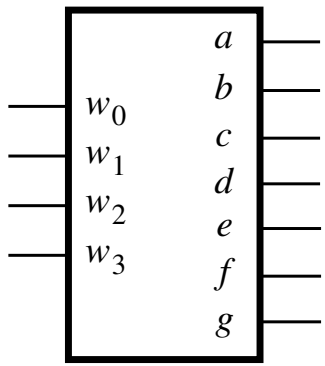
figure4.36.v

```
1  module priority (W, Y, z);
2    input [3:0] W;
3    output reg [1:0] Y;
4    output reg z;
5
6    always @(W)
7    begin
8      z = 1;
9      casex (W)
10       4'b1xxx: Y = 3;
11       4'b01xx: Y = 2;
12       4'b001x: Y = 1;
13       4'b0001: Y = 0;
14       default:
15       begin
16         z = 0;
17         Y = 2'bx;
18       end
19     endcase
20   end
21 endmodule
```

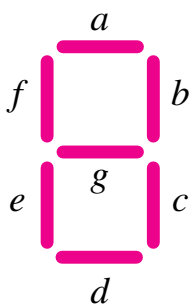
figure4.38.v

```
1  module priority (W, Y, z);
2    input [3:0] W;
3    output reg [1:0] Y;
4    output reg z;
5    integer k;
6
7    always @(W)
8    begin
9      Y = 2'bx;
10     z = 0;
11     for (k = 0; k < 4; k = k+1)
12       if (W[k])
13       begin
14         Y = k;
15         z = 1;
16       end
17   end
18 endmodule
```

Conversores de Códigos



(a) Code converter



(b) 7-segment display

w_3	w_2	w_1	w_0	a	b	c	d	e	f	g
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1
1	0	1	0	1	1	1	0	1	1	1
1	0	1	1	0	0	1	1	1	1	1
1	1	0	0	1	0	0	1	1	1	0
1	1	0	1	0	1	1	1	1	0	1
1	1	1	0	1	0	0	1	1	1	1
1	1	1	1	1	0	0	0	1	1	1

(c) Truth table

Figure 4.21 A hex-to-7-segment display code converter.

figure4.34.v

```
1  module seg7 (hex, leds);
2    input [3:0] hex;
3    output reg [1:7] leds;
4    always @(hex)
5      case (hex)     //abcdefg
6         0: leds = 7'b1111110;
7         1: leds = 7'b0110000;
8         2: leds = 7'b1101101;
9         3: leds = 7'b1111001;
10        4: leds = 7'b0110011;
11        5: leds = 7'b1011011;
12        6: leds = 7'b1011111;
13        7: leds = 7'b1110000;
14        8: leds = 7'b1111111;
15        9: leds = 7'b1111011;
16       10: leds = 7'b1110111; // A
17       11: leds = 7'b0011111; // b
18       12: leds = 7'b1001110; // C
19       13: leds = 7'b0111101; // d
20       14: leds = 7'b1001111; // E
21       15: leds = 7'b1000111; // F
22     endcase
23 endmodule
```

Bibliografia

- Brown, S. & Vranesic, Z. - Fundamentals of Digital Logic with Verilog Design, 3rd Ed., Mc Graw Hill, 2009

Lógica Digital (1001351)

Circuitos Combinacionais: Conversores de códigos

Prof. Edilson Kato
kato@ufscar.br

Prof. Maurício Figueiredo
mauricio@ufscar.br

Prof. Ricardo Menotti
menotti@ufscar.br

Prof. Roberto Inoue
rsinoue@ufscar.br

Departamento de Computação
Universidade Federal de São Carlos

Atualizado em: 18 de abril de 2019