

Lógica Digital (1001351)

Circuitos Combinacionais: Multiplexadores

Prof. Edilson Kato
kato@ufscar.br

Prof. Maurício Figueiredo
mauricio@ufscar.br

Prof. Ricardo Menotti
menotti@ufscar.br

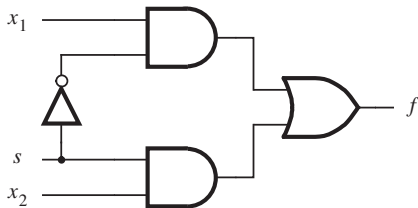
Prof. Roberto Inoue
rsinoue@ufscar.br

Departamento de Computação
Universidade Federal de São Carlos

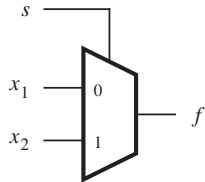
Atualizado em: 22 de abril de 2019



Multiplexador 2-para-1



(b) Circuit



(c) Graphical symbol

s	$f(s, x_1, x_2)$
0	x_1
1	x_2

(d) More compact truth-table representation

Figure 2.33 Implementation of a multiplexer.

figure4.23.v

```
1 module mux2to1 (w0, w1, s, f);  
2     input w0, w1, s;  
3     output f;  
4  
5     assign f = s ? w1 : w0;  
6 endmodule
```

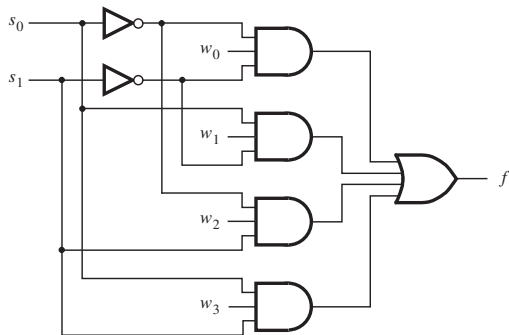
figure4.24.v

```
1 module mux2to1 (w0, w1, s, f);  
2   input w0, w1, s;  
3   output reg f;  
4  
5   always @(w0, w1, s)  
6     f = s ? w1 : w0;  
7 endmodule
```

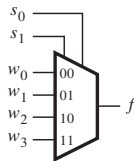
figure4.26.v

```
1 module mux2to1 (w0, w1, s, f);  
2   input w0, w1, s;  
3   output reg f;  
4  
5   always @(w0, w1, s)  
6     if (s==0)  
7       f = w0;  
8     else  
9       f = w1;  
10 endmodule
```

Multiplexador 4-para-1



(c) Circuit



(a) Graphical symbol

s_1	s_0	f
0	0	w_0
0	1	w_1
1	0	w_2
1	1	w_3

(b) Truth table

Figure 4.2 A 4-to-1 multiplexer.

Multiplexador 4-para-1

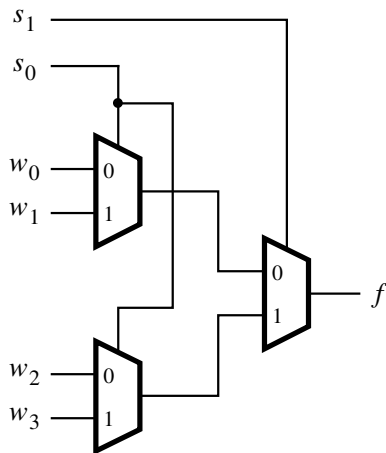


Figure 4.3 Using 2-to-1 multiplexers to build a 4-to-1 multiplexer.

figure4.25.v

```
1 module mux4to1 (w0, w1, w2, w3, S, f);  
2   input w0, w1, w2, w3;  
3   input [1:0] S;  
4   output f;  
5  
6   assign f = S[1] ? (S[0] ? w3 : w2) : (S[0] ? w1 : w0);  
7 endmodule
```


figure4.27.v

```
1 module mux4to1 (w0, w1, w2, w3, S, f);
2   input w0, w1, w2, w3;
3   input [1:0] S;
4   output reg f;
5
6   always @(*)
7     if (S == 2'b00)
8       f = w0;
9     else if (S == 2'b01)
10      f = w1;
11    else if (S == 2'b10)
12      f = w2;
13    else if (S == 2'b11)
14      f = w3;
15 endmodule
```

figure4.28.v

```
1 module mux4to1 (W, S, f);
2   input [0:3] W;
3   input [1:0] S;
4   output reg f;
5
6   always @(W, S)
7     if (S == 0)
8       f = W[0];
9     else if (S == 1)
10      f = W[1];
11    else if (S == 2)
12      f = W[2];
13    else if (S == 3)
14      f = W[3];
15 endmodule
```

figure4.30.v

```
1 module mux4to1 (W, S, f);
2   input [0:3] W;
3   input [1:0] S;
4   output reg f;
5
6   always @(W, S)
7     case (S)
8       0: f = W[0];
9       1: f = W[1];
10      2: f = W[2];
11      3: f = W[3];
12    endcase
13 endmodule
```

Multiplexador 16-para-1

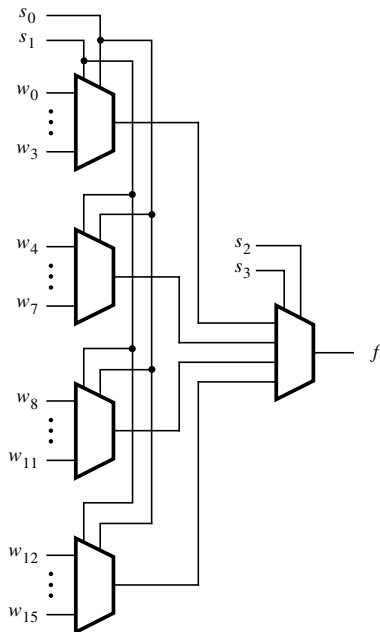


figure4.29.v

```
1 module mux16to1 (W, S, f);
2   input [0:15] W;
3   input [3:0] S;
4   output f;
5   wire [0:3] M;
6
7   mux4to1 Mux1 (W[0:3], S[1:0], M[0]);
8   mux4to1 Mux2 (W[4:7], S[1:0], M[1]);
9   mux4to1 Mux3 (W[8:11], S[1:0], M[2]);
10  mux4to1 Mux4 (W[12:15], S[1:0], M[3]);
11  mux4to1 Mux5 (M[0:3], S[3:2], f);
12 endmodule
```

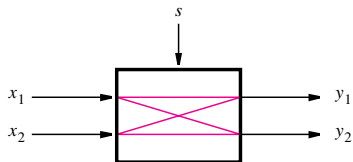
figure4.42.v

```
1 module mux16to1 (W, S16, f);
2   input [0:15] W;
3   input [3:0] S16;
4   output reg f;
5
6   always @(W, S16)
7   case (S16[3:2])
8     0: mux4to1 (W[0:3], S16[1:0], f);
9     1: mux4to1 (W[4:7], S16[1:0], f);
10    2: mux4to1 (W[8:11], S16[1:0], f);
11    3: mux4to1 (W[12:15], S16[1:0], f);
12  endcase
13  // Task that specifies a 4-to-1 multiplexer
14  task mux4to1;
15    input [0:3] X;
16    input [1:0] S4;
17    output reg g;
18    case (S4)
19      0: g = X[0];
20      1: g = X[1];
21      2: g = X[2];
22      3: g = X[3];
23    endcase
24  endtask
25 endmodule
```

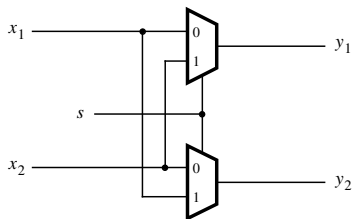
figure4.43.v

```
1  module mux16to1 (W, S16, f);
2      input  [0:15]W;
3      input  [3:0] S16;
4      output reg f;
5      // Function that specifies a 4-to-1 multiplexer
6      function mux4to1;
7          input [0:3] X;
8          input [1:0] S4;
9          case (S4)
10             0: mux4to1 = X[0];
11             1: mux4to1 = X[1];
12             2: mux4to1 = X[2];
13             3: mux4to1 = X[3];
14         endcase
15     endfunction
16
17     always @(W, S16)
18         case (S16[3:2])
19             0: f = mux4to1 (W[0:3], S16[1:0]);
20             1: f = mux4to1 (W[4:7], S16[1:0]);
21             2: f = mux4to1 (W[8:11], S16[1:0]);
22             3: f = mux4to1 (W[12:15], S16[1:0]);
23         endcase
24 endmodule
```

Crossbar 2x2



(a) A 2x2 crossbar switch



(b) Implementation using multiplexers

Figure 4.5 A practical application of multiplexers.

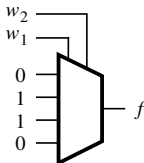
Síntese de funções lógicas usando multiplexadores

Teorema de Shannon:

$$f(w_1, w_2, \dots, w_n) = \overline{w_1} \cdot f(0, w_2, \dots, w_n) + w_1 \cdot f(1, w_2, \dots, w_n)$$

Síntese de funções lógicas usando multiplexadores

w_1	w_2	f
0	0	0
0	1	1
1	0	1
1	1	0

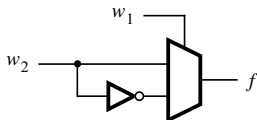


(a) Implementation using a 4-to-1 multiplexer

w_1	w_2	f
0	0	0
0	1	1
1	0	1
1	1	0

w_1	f
0	w_2
1	\bar{w}_2

(b) Modified truth table



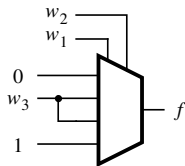
(c) Circuit

Figure 4.6 Synthesis of a logic function using multiplexers.

Síntese de funções lógicas usando multiplexadores

w_1	w_2	w_3	f		w_1	w_2
0	0	0	0	}	0	0
0	0	1	0		0	1
0	1	0	0	}	1	0
0	1	1	1		1	1
1	0	0	0	}		
1	0	1	1			
1	1	0	1	}		
1	1	1	1			

(a) Modified truth table



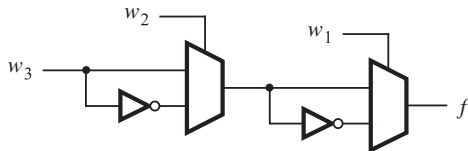
(b) Circuit

Figure 4.7 Implementation of the three-input majority function using a 4-to-1 multiplexer.

Síntese de funções lógicas usando multiplexadores

w_1	w_2	w_3	f
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

(a) Truth table



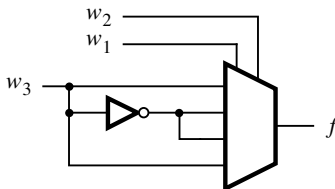
(b) Circuit

Figure 4.8 Three-input XOR implemented with 2-to-1 multiplexers.

Síntese de funções lógicas usando multiplexadores

w_1	w_2	w_3	f
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

(a) Truth table



(b) Circuit

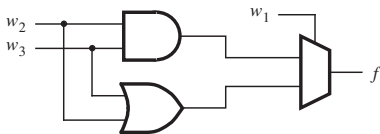
Figure 4.9 Three-input XOR implemented with a 4-to-1 multiplexer.

Síntese de funções lógicas usando multiplexadores

w_1	w_2	w_3	f
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

w_1	f
0	$w_2 w_3$
1	$w_2 + w_3$

(a) Truth table



(b) Circuit

Figure 4.10 The three-input majority function implemented using a 2-to-1 multiplexer.

Bibliografia

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

Lógica Digital (1001351)

Circuitos Combinacionais: Multiplexadores

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: 22 de abril de 2019

