Lógica Digital (1001351)

Outros Circuitos Combinacionais

Prof. Ricardo Menotti
menotti@ufscar.br

Prof. Luciano de Oliveira Neris
lneris@ufscar.br

Atualizado em: 1 de abril de 2024

Departamento de Computação
Centro de Ciências Exatas e de Tecnologia
Universidade Federal de São Carlos

Comparadores

Comparador

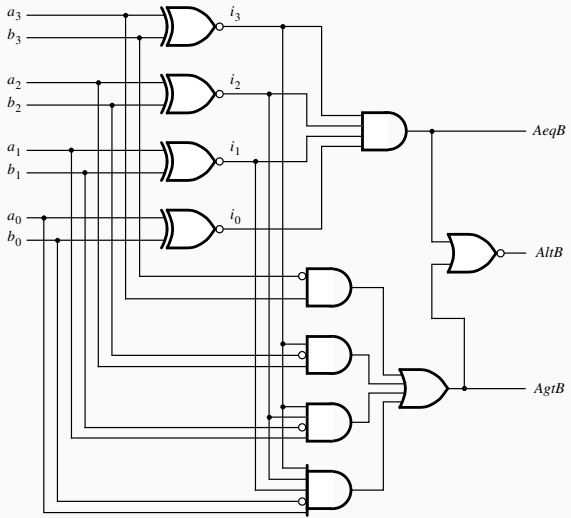


Figure 4.22 A four-bit comparator circuit.

figure4.40.v

```
1  module compare (A, B, AeqB, AgtB, AltB);
2    input [3:0] A, B;
3    output reg AeqB, AgtB, AltB;
4
5    always @(A, B)
6    begin
7      AeqB = 0;
8      AgtB = 0;
9      AltB = 0;
10     if(A == B)
11       AeqB = 1;
12     else if (A > B)
13       AgtB = 1;
14     else
15       AltB = 1;
16   end
17
18 endmodule
```

Comparador por subtração

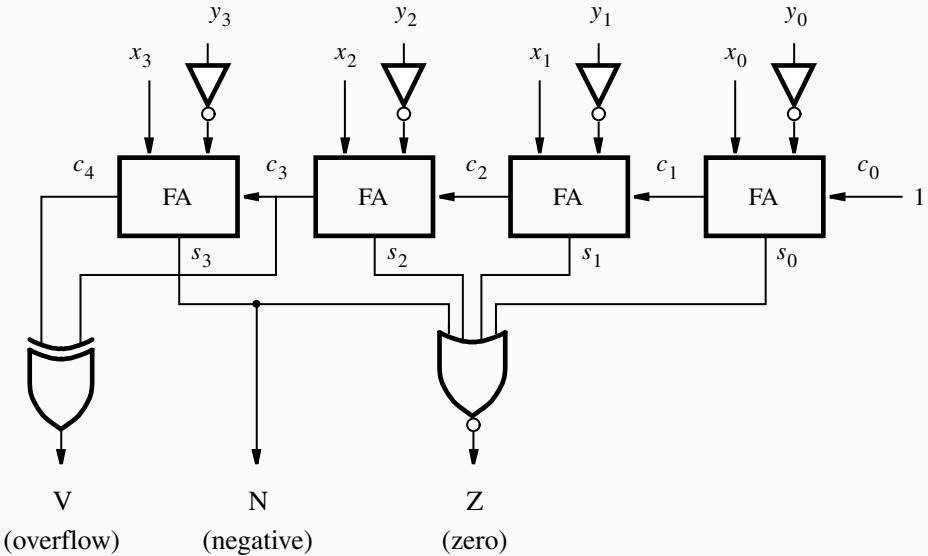


figure3.46.v

```
1  module comparator (X, Y, V, N, Z);
2    input [3:0] X, Y;
3    output V, N, Z;
4    wire [3:0] S;
5    wire [4:1] C;
6
7    fulladd stage0 (1'b1, X[0], ~Y[0], S[0], C[1]);
8    fulladd stage1 (C[1], X[1], ~Y[1], S[1], C[2]);
9    fulladd stage2 (C[2], X[2], ~Y[2], S[2], C[3]);
10   fulladd stage3 (C[3], X[3], ~Y[3], S[3], C[4]);
11   assign V  = C[4] ^ C[3];
12   assign N = S[3];
13   assign Z = !S;
14
15 endmodule
16
17 module fulladd (Cin, x, y, s, Cout);
18   input Cin, x, y;
19   output s, Cout;
20
21   assign s = x ^ y ^ Cin,
22        Cout = (x & y) | (x & Cin) | (y & Cin);
23
24 endmodule
```

figure3.47.v

```
1  module comparator (X, Y, V, N, Z);
2    parameter n = 32;
3    input [n-1:0] X, Y;
4    output reg V, N, Z;
5    reg [n-1:0] S;
6    reg [n:0] C;
7    integer k;
8
9    always @(X, Y)
10   begin
11     C[0] = 1'b1;
12     for (k = 0; k < n;  k = k + 1)
13     begin
14       S[k] = X[k] ^ ~Y[k] ^ C[k];
15       C[k+1] = (X[k] & ~Y[k]) | (X[k] & C[k]) | (~Y[k] & C[k]);
16     end
17     V  = C[n] ^ C[n-1];
18     N = S[n-1];
19     Z = !S;
20   end
21
22 endmodule
```

Deslocadores

Deslocamento (*shift*)

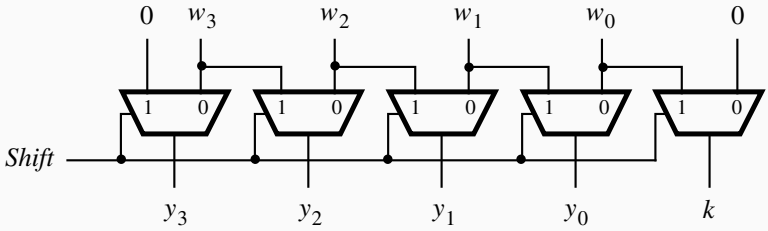


Figure 4.50 A shifter circuit.

figure4.53.v

```
1  module shifter (W, Shift, Y , k);
2    input [3:0] W;
3    input Shift;
4    output reg [3:0] Y;
5    output reg k;
6
7    always @(W, Shift)
8    begin
9      if (Shift)
10     begin
11       Y[3] = 0;
12       Y[2:0] = W[3:1];
13       k = W[0];
14     end
15     else
16     begin
17       Y = W;
18       k = 0;
19     end
20   end
21
22 endmodule
```

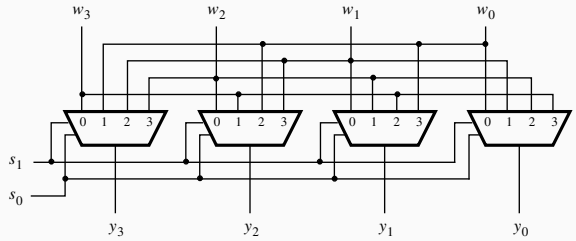
figure4.54.v

```
1  module shifter (W, Shift, Y , k);
2    input [3:0] W;
3    input Shift;
4    output reg [3:0] Y;
5    output reg k;
6
7    always @(W, Shift)
8    begin
9      if (Shift)
10     begin
11       Y = W >> 1;
12       k = W[0];
13     end
14     else
15     begin
16       Y = W;
17       k = 0;
18     end
19   end
20
21 endmodule
```

Deslocamento (*rotate*)

s_1	s_0	y_3	y_2	y_1	y_0
0	0	w_3	w_2	w_1	w_0
0	1	w_0	w_3	w_2	w_1
1	0	w_1	w_0	w_3	w_2
1	1	w_2	w_1	w_0	w_3

(a) Truth table



(b) Circuit

Figure 4.51 A barrel shifter circuit.

figure4.55.v

```
1 module barrel (W, S, Y);
2   input [3:0] W;
3   input [1:0] S;
4   output [3:0] Y;
5   wire [3:0] T;
6
7   assign {T, Y} = {W, W} >> S;
8
9 endmodule
```

ULA

Table 4.1 The functionality of the 74381 ALU.

Operation	Inputs $s_2\ s_1\ s_0$	Outputs F
Clear	0 0 0	0 0 0 0
B−A	0 0 1	$B - A$
A−B	0 1 0	$A - B$
ADD	0 1 1	$A + B$
XOR	1 0 0	A XOR B
OR	1 0 1	A OR B
AND	1 1 0	A AND B
Preset	1 1 1	1 1 1 1

Table 4.1 The functionality of the 74381 ALU.

Operation	Inputs $s_2\ s_1\ s_0$	Outputs F
Clear	0 0 0	0 0 0 0
B−A	0 0 1	$B - A$
A−B	0 1 0	$A - B$
ADD	0 1 1	$A + B$
XOR	1 0 0	A XOR B
OR	1 0 1	A OR B
AND	1 1 0	A AND B
Preset	1 1 1	1 1 1 1

```
1  // 74381 ALU
2  module alu (S, A, B, F);
3    input [2:0] S;
4    input [3:0] A, B;
5    output reg [3:0] F;
6
7    always @(S, A, B)
8      case (S)
9        0: F = 4'b0000;
10       1: F = B - A;
11       2: F = A - B;
12       3: F = A + B;
13       4: F = A ^ B;
14       5: F = A | B;
15       6: F = A & B;
16       7: F = 4'b1111;
17   endcase
18
19 endmodule
```

Verilog

Operadores em Verilog (1/2)

Table 4.2 Verilog operators.

Operator type	Operator symbols	Operation performed	Number of operands
Bitwise	~	1's complement	1
	&	Bitwise AND	2
	|	Bitwise OR	2
	^	Bitwise XOR	2
	~ ^ or ^ ~	Bitwise XNOR	2
Logical	!	NOT	1
	&&	AND	2
	||	OR	2
Reduction	&	Reduction AND	1
	~&	Reduction NAND	1
	|	Reduction OR	1
	~ |	Reduction NOR	1
	^	Reduction XOR	1
	~^ or ^ ~	Reduction XNOR	1

. . .

Operadores em Verilog (2/2)

. . .

Arithmetic	+	Addition	2
	−	Subtraction	2
	−	2's complement	1
	*	Multiplication	2
	/	Division	2
Relational	>	Greater than	2
	<	Less than	2
	>=	Greater than or equal to	2
	<=	Less than or equal to	2
Equality	==	Logical equality	2
	! =	Logical inequality	2
Shift	>>	Right shift	2
	<<	Left shift	2
Concatenation	{,}	Concatenation	Any number
Replication	{{}}	Replication	Any number
Conditional	?:	Conditional	3

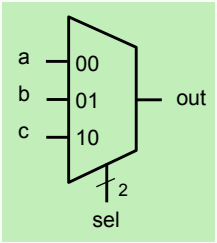
Precedencia dos Operadores em Verilog

Table 4.3 Precedence of Verilog operators.

Operator type	Operator symbols	Precedence
Complement	! ~ –	Highest precedence
Arithmetic	* / + –	
Shift	<< >>	
Relational	< <= > >=	
Equality	== !=	
Reduction	& ~& ^ ~^ | ~|	
Logical	&& ||	
Conditional	?:	Lowest precedence

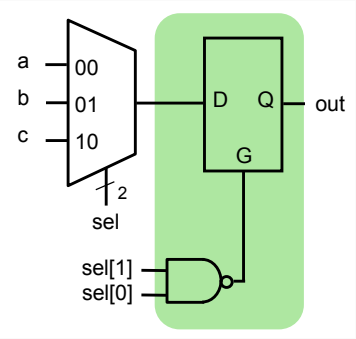
Erros comuns

Perigo!: Especificação Incompleta




```
1  module mux3to1(a, b, c, sel, out);
2    input [1:0] sel;
3    input a, b, c;
4    output out;
5    reg out;
6
7    always @(a or b or c or sel)
8    begin
9      case (sel)
10       2'b00: out = a;
11       2'b01: out = b;
12       2'b10: out = c;
13     endcase
14   end
15 endmodule
```

Perigo!: Especificação Incompleta

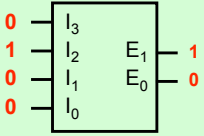


Evitando a Especificação Incompleta

```
always @(a or b or c or sel)
begin
  out = 1'bx
  case (sel)
    2'b00: out = a;
    2'b01: out = b;
    2'b10: out = c;
  endcase
end
```

```
always @(a or b or c or sel)
begin
  case (sel)
    2'b00: out = a;
    2'b01: out = b;
    2'b10: out = c;
    default: out = 1'bx
  endcase
end
```

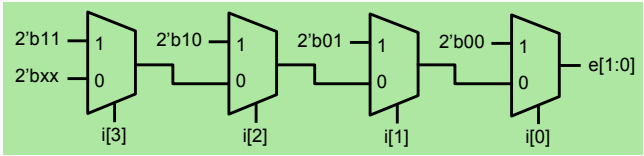
Perigo!: Prioridade Indesejada



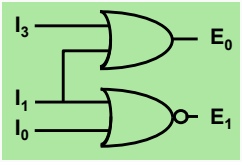
I_3 I_2 I_1 I_0	E_1 E_0
0 0 0 1	0 0
0 0 1 0	0 1
0 1 0 0	1 0
1 0 0 0	1 1
all others	X X

```
1  module binary_encoder(i, e);
2    input [3:0] i;
3    output [1:0] e;
4    reg e;
5
6    always @(i)
7    begin
8      if (i[0])
9        e = 2'b00;
10     else if (i[1])
11       e = 2'b01;
12     else if (i[2])
13       e = 2'b10;
14     else if (i[3])
15       e = 2'b11;
16     else
17       e = 2'bxx;
18   end
19 endmodule
```

Perigo!: Prioridade Indesejada



Evitando Prioridade Indesejada




```
1  module binary_encoder(i, e);
2    input [3:0] i;
3    output [1:0] e;
4    reg e;
5
6    always @(i)
7    begin
8      if (i == 4'b0001)
9        e = 2'b00;
10     else if (i == 4'b0010)
11       e = 2'b01;
12     else if (i == 4'b0100)
13       e = 2'b10;
14     else if (i == 4'b1000)
15       e = 2'b11;
16     else
17       e = 2'bxx;
18   end
19 endmodule
```

Bibliografia

Bibliografia

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

Lógica Digital (1001351)

Outros Circuitos Combinacionais

Prof. Ricardo Menotti
menotti@ufscar.br

Prof. Luciano de Oliveira Neris
lneris@ufscar.br

Atualizado em: 1 de abril de 2024

Departamento de Computação
Centro de Ciências Exatas e de Tecnologia
Universidade Federal de São Carlos