

RESUMO MULTIPLEXADORES

```
//-----  
// 2x1 MUX using if  
//-----  
module mux2x1i ( out, in0, in1, sel );  
input in0, in1, sel ;  
output out;  
reg out;  
  
always @ ( sel or in0 or in1 )  
begin : MUX  
out = sel;  
if ( sel == 1'b0 )  
begin out = in0; end  
else  
if ( sel == 1'b1 )  
begin out = in1; end  
end  
endmodule // mux2x1_using_if  
  
//-----  
// 2x1 MUX using case  
//-----  
module mux2x1c ( out, in0, in1, sel );  
input in0, in1, sel ;  
output out;  
reg out;  
  
always @ ( sel or in0 or in1 )  
begin : MUX  
case ( sel )  
1'b0 : out = in0;  
1'b1 : out = in1;  
endcase  
end  
endmodule // mux2x1_using_case
```

```
//-----  
// 2x1 MUX using gates  
//-----  
module mux2x1g ( out, in0, in1, sel );  
input in0, in1, sel ;  
output out;  
wire notsel, s0, s1;  
  
not NOT1 ( notsel, sel );  
and AND1 ( s0, in0, notsel );  
and AND2 ( s1, in1, sel );  
or OR1 ( out, s0, s1 );  
endmodule // mux2x1_using_gates  
  
//-----  
// 2x1 MUX using vector  
//-----  
module mux2x1v ( out, in, sel );  
input [1:0] in;  
input sel ;  
output out;  
wire notsel, s0, s1;  
  
not NOT1 ( notsel, sel );  
and AND1 ( s0, in[0], notsel );  
and AND2 ( s1, in[1], sel );  
or OR1 ( out, s0, s1 );  
endmodule // mux2x1_using_vector
```

```
//-----  
// test MUXes  
//-----  
module testmux;  
reg a, b, c;  
reg [1:0] d;  
wire s1, s2, s3, s4, s5;  
  
mux2x1a MUX1 ( s1, a, b, c );  
mux2x1i MUX2 ( s2, a, b, c );  
mux2x1c MUX3 ( s3, a, b, c );  
mux2x1g MUX4 ( s4, a, b, c );  
mux2x1p MUX5 ( s5, a, b, c );  
mux2x1v MUX6 ( s6, d, c );  
  
initial  
begin: init  
a = 0; b = 1;  
d[0] = 1'b1;  
d[1] = 1'b0;  
end  
  
initial  
begin: main  
$display ( "Test MUX selections" );  
$display ( "time a b c s1 s2 s3  
s4 s5 s6" );  
$monitor ( "%2d %b %b %b %3b  
%4b %4b %4b %4b %4b",  
$time, a, b, c, s1, s2, s3, s4, s5, s6 );  
#1 c = 0;  
#1 c = 1;  
#1 a = 1; b = 0;  
#1 c = 0;  
end  
  
endmodule // testmux
```

RESUMO DEMULTIPLEXADORES

```
//-----  
// 1x2 DEMUX using if  
//-----  
module demux1x2i ( out0, out1, in, sel );  
input in, sel;  
output out0, out1;  
reg out0, out1;  
  
always @ ( sel or in )  
begin : DEMUX  
if ( sel == 1'b0 )  
begin  
out0 = in; out1 = 1'b0;  
end  
else  
if ( sel == 1'b1 )  
begin  
out0 = 1'b0; out1 = in;  
end  
end  
  
endmodule // demux1x2_using_if
```

```
//-----  
// 1x2 DEMUX using case  
//-----  
module demux1x2c ( out0, out1, in, sel );  
input in, sel;  
output out0, out1;  
reg out0, out1;  
  
always @ ( sel or in )  
begin : DEMUX  
case ( sel )  
1'b0 : begin out0 = in; out1 = 1'b0; end  
1'b1 : begin out0 = 1'b0; out1 = in; end  
endcase  
end  
endmodule // demux1x2_using_case
```

```
//-----  
// 1x2 DEMUX using gates  
//-----  
module demux1x2g ( out0, out1, in, sel );  
input in, sel;  
output out0, out1;  
wire notsel;  
  
not NOT1 ( notsel, sel );  
and AND1 ( out0, in, notsel );  
and AND2 ( out1, in, sel );  
endmodule // demux1x2_using_gates
```

```
//-----  
// 1x2 DEMUX using vector  
//-----  
module demux1x2v ( out, in, sel );  
input in;  
input sel;  
output [0:1] out;  
wire notsel;  
wire [0:1] out;  
  
not NOT1 ( notsel, sel );  
and AND1 ( out[0], in, notsel );  
and AND2 ( out[1], in, sel );  
//assign out[0] = 0;  
//assign out[1] = 1;  
endmodule // mux1x2_using_vector
```

```
//-----  
// test DEMUXes  
//-----  
module testdemux;  
reg a, b, c;  
reg [1:0] d;  
wire s1, s2, s3, s4, s5, s7, s8, s9, s0;  
wire [0:1] s;  
  
demux1x2a DEMUX1 ( s1, s2, a, c );  
demux1x2i DEMUX2 ( s3, s4, a, c );  
demux1x2c DEMUX3 ( s5, s6, a, c );  
demux1x2g DEMUX4 ( s7, s8, a, c );  
demux1x2v DEMUX5 ( s, a, c );  
  
initial  
begin: init  
a = 0;  
end  
  
initial  
begin: main  
$display ( "Test DEMUX selections" );  
$display ( "time a c s1 s2 s3 s4  
s5 s6 s7 s8 s01" );  
$monitor ( "#%2d %b %b %3b %4b  
%4b %4b %4b %4b %4b %4b %4b",  
$time, a, c, s1, s2, s3, s4, s5, s6, s7, s8, s );  
#1 c = 0;  
#1 c = 1;  
#1 a = 1;  
#1 c = 0;  
end  
  
endmodule // testdemux
```

RESUMO GERADORES DE CLOCK

```
// -----  
// -- test clock generator (1)  
// -----  
  
module clock ( clock );  
output clock;  
reg clock;  
  
initial  
begin  
clock = 1'b0;  
end  
  
always  
begin  
#12 clock = ~clock;  
end  
  
endmodule // clock ( )  
  
module testclock01a;  
  
wire clock;  
clock clk ( clock );  
  
initial begin  
$dumpfile ( "testclock01a.vcd" );  
$dumpvars;  
  
#120 $finish;  
end  
  
endmodule // testclock1a ( )
```

```
// -----  
// -- test clock generator (2)  
// -----  
  
//^include "clock.v"  
  
module clock ( clock );  
output clock;  
reg clock;  
  
initial  
begin  
clock = 1'b0;  
end  
  
always  
begin  
#12 clock = ~clock;  
end  
endmodule  
  
module pulse ( signal, clock );  
input clock;  
output signal;  
reg signal;  
  
always @ ( clock )  
begin  
signal = 1'b1;  
#3 signal = 1'b0;  
#3 signal = 1'b1;  
#3 signal = 1'b0;  
end  
endmodule // pulse
```

```
module trigger ( signal, on, clock );  
input on, clock;  
output signal;  
reg signal;  
  
always @ ( negedge clock & on )  
begin  
#60 signal = 1'b1;  
#60 signal = 1'b0;  
end  
endmodule // trigger  
  
module testclock01b;  
  
wire clock;  
clock clk ( clock );  
  
reg p;  
  
wire p1,t1;  
  
pulse pulse1 ( p1, clock );  
trigger trigger1 ( t1, p, clock );  
  
initial begin  
p = 1'b0;  
end  
  
initial begin  
$dumpfile ( "testclock01b.vcd" );  
$dumpvars ( 1, clock, p1, p, t1 );  
  
#060 p = 1'b1;  
#120 p = 1'b0;  
#180 p = 1'b1;  
#240 p = 1'b0;  
#300 p = 1'b1;  
#360 p = 1'b0;  
#376 $finish;  
end  
  
endmodule // testclock01b
```