

Code for 1:2 demultiplexer

```
1 // Code your testbench here
2 // or browse Examples
3 module tb_demux_1to2;
4 reg a,sel;
5 wire y0,y1;
6
7 demux_1to2 uut (.a(a),.sel(sel),.y0(y0),
8 .y1(y1));
9
10 initial begin
11     $dumpfile("demux_1to2.vcd");
12     $dumpvars(0,tb_demux_1to2);
13     $monitor("Time = %0t | a = %b | sel =
14 %b | y0 = %b | y1 = %b",$time,a,sel,y0,
15 y1);
16
17     a = 0;sel = 0;#10;
18     a = 1;sel = 0;#10;
19     a = 1;sel = 1;#10;
20     a = 0;sel = 1;#10;
21     #10;
22
23     $finish;
24 end
25 endmodule
```

```
1 // Code your design here
2 module demux_1to2 (
3     input a,
4     input sel,
5     output reg y0,
6     output reg y1
7 );
8 always @(*) begin
9     case(sel)
10         1'b0:begin
11             y0 = a;
12             y1 = 0;
13         end
14         1'b1:begin
15             y0 = 0;
16             y1 = a;
17         end
18     endcase
19 end
20 endmodule
```

Output:

Time = 0 | a = 0 | sel = 0 | y0 = 0 | y1 = 0
Time = 10 | a = 1 | sel = 0 | y0 = 1 | y1 = 0
Time = 20 | a = 1 | sel = 1 | y0 = 0 | y1 = 1
Time = 30 | a = 0 | sel = 1 | y0 = 0 | y1 = 0

Code for 1:4 demultiplexer

```
1 // Code your testbench here
2 // or browse Examples
3 module tb_demux_1x4;
4
5 reg a;
6 reg [1:0]sel;
7 wire y0,y1,y2,y3;
8
9 demux_1x4 uut
10 (.a(a),.sel(sel),.y0(y0),.y1(y1),.y2(y2),
11 y3(y3));
12
13 initial begin
14     $display("Time\t a sel\t y0 y1 y2
15 y3");
16
17     $monitor("%0t\t %b %b\t %b %b %b",
18 $time, a,sel, y0,y1,y2,y3);
19
20     // Test cases
21     a = 0; sel = 2'b00;#10;
22     a = 1; sel = 2'b00;#10;
23     a = 1; sel = 2'b01;#10;
24     a = 1; sel = 2'b10;#10;
25     a = 1; sel = 2'b11;#10;
26     a = 0; sel = 2'b11;#10;
27 #10;
28
29     $finish;
30 end
31
32 endmodule
```

```
1 // Code your design here
2
3 module demux_1x4 (
4     input wire a,
5     input wire [1:0]sel,
6     output reg y0,y1,y2,y3
7 );
8
9 always @ ( a or sel) begin
10     // Default outputs
11     y0 = 0; y1 = 0; y2 = 0; y3 = 0;
12
13     case (sel)
14         2'b00:y0 = a;
15         2'b01:y1 = a;
16         2'b10:y2 = a;
17         2'b11:y3 = a;
18     endcase
19 end
20
21 endmodule
```

Output:

Time	a	sel	y0	y1	y2	y3
0	0	00	0	0	0	0
10	1	00	1	0	0	0
20	1	01	0	1	0	0
30	1	10	0	0	1	0
40	1	11	0	0	0	1
50	0	11	0	0	0	0

Code for 1:8 demultiplexer

```
1 // Code your testbench here
2 // or browse Examples
3 module tb_demux_1to8;
4     reg din;
5     reg [2:0]sel;
6     wire [7:0]out;
7
8     demux_1to8
9         uut(.din(din),.sel(sel),.out(out))
10    );
11
12    initial begin
13        $dumpfile("demux_1to8.vcd");
14        $dumpvars(0, tb_demux_1to8);
15
16        din = 1;
17        sel = 3'b000; #10;
18        sel = 3'b001; #10;
19        sel = 3'b010; #10;
20        sel = 3'b011; #10;
21        sel = 3'b100; #10;
22        sel = 3'b101; #10;
23        sel = 3'b110; #10;
24        sel = 3'b111; #10;
25
26        din =0; sel =3'b101;#10;
27
28        $finish;
29    end
30
31    initial begin
32        $monitor("Time=%0t | din=%b |
33        sel=%b | out=%b", $time, din, sel, out);
34    end
35 endmodule
```

```
1 // Code your design here
2 module demux_1to8(
3     input din,
4     input [2:0]sel,
5     output reg [7:0]out
6 );
7     always @(*) begin
8         out = 8'b00000000;
9         out[sel] =din;
10    end
11 endmodule
```

Output:

Time=0 | din=1 | sel=000 | out=00000001
Time=10 | din=1 | sel=001 | out=00000010
Time=20 | din=1 | sel=010 | out=00000100
Time=30 | din=1 | sel=011 | out=00001000
Time=40 | din=1 | sel=100 | out=00010000
Time=50 | din=1 | sel=101 | out=00100000
Time=60 | din=1 | sel=110 | out=01000000
Time=70 | din=1 | sel=111 | out=10000000
Time=80 | din=0 | sel=101 | out=00000000

Code for 4:2 Encoder

```
1 // Code your testbench here
2 // or browse Examples
3 module tb_encoder_4to2;
4   reg [3:0]din;
5   wire [1:0]out;
6
7   encoder_4to2 uut (.din(din),.out(out))
8 );
9
10 initial begin
11   $dumpfile("encoder_4to2.vcd");
12   $dumpvars(0, tb_encoder_4to2);
13
14   din = 4'b0001:#10;
15   din = 4'b0010:#10;
16   din = 4'b0100:#10;
17   din = 4'b1000:#10;
18   din = 4'b0000:#10;
19
20   $finish;
21 end
22
23 initial begin
24   $monitor("Time=%0t | din=%b |
25 out=%b", $time,din,out);
26 end
endmodule
```

```
1 // Code your design here
2 module encoder_4to2 (
3   input [3:0]din,
4   output reg [1:0]out
5 );
6   always @(*) begin
7     case (din)
8       4'b0001:out = 2'b00;
9       4'b0010:out = 2'b01;
10      4'b0100:out = 2'b10;
11      4'b1000:out = 2'b11;
12
13    endcase
14  end
15 endmodule
```

Output:

Time=0 | din=0001 | out=00

Time=10 | din=0010 | out=01

Time=20 | din=0100 | out=10

Time=30 | din=1000 | out=11

Code for 8:3 Encoder

```
1 // Code your testbench here
2 // or browse Examples
3 module tb_encoder_8to3;
4   reg [7:0]din;
5   wire [2:0]out;
6
7   encoder_8to3 uut (.din(din),.out(out))
8   );
9
10 initial begin
11   $dumpfile("encoder_8to3.vcd");
12   $dumpvars(0,tb_encoder_8to3);
13
14   din = 8'b00000001:#10;
15   din = 8'b00000010:#10;
16   din = 8'b00000100:#10;
17   din = 8'b00001000:#10;
18   din = 8'b00010000:#10;
19   din = 8'b00100000:#10;
20   din = 8'b10000000:#10;
21   din = 8'b00000000:#10;
22
23   $finish;
24 end
25
26
27 initial begin
28   $monitor("Time=%0t | din=%b |
29 out=%b", $time,din,out);
30 end
endmodule
```

```
1 // Code your design here
2 module encoder_8to3 (
3   input [7:0]din,
4   output reg [2:0]out
5 );
6   always @(*) begin
7     case (din)
8       8'b00000001:out = 3'b000;
9       8'b00000010:out = 3'b001;
10      8'b00000100:out = 3'b010;
11      8'b00001000:out = 3'b011;
12      8'b00010000:out = 3'b100;
13      8'b00100000:out = 3'b101;
14      8'b01000000:out = 3'b110;
15      8'b10000000:out = 3'b111;
16
17     endcase
18   end
19 endmodule
```

Output:

Time=0 | din=00000001 | out=000
Time=10 | din=00000010 | out=001
Time=20 | din=00000100 | out=010
Time=30 | din=00001000 | out=011
Time=40 | din=00010000 | out=100
Time=50 | din=00100000 | out=101
Time=60 | din=01000000 | out=110
Time=70 | din=10000000 | out=111
Time=80 | din=00000000 | out=xxx

Code for 16:4 Encoder

```
1 // Code your testbench here
2 // or browse Examples
3 module tb_encoder_16to4;
4   reg [15:0]din;
5   wire [3:0]out;
6
7   encoder_16to4 uut (
8     .din(din),.out(out)
9   );
10
11   initial begin
12     $dumpfile("encoder_16to4.vcd");
13     $dumpvars(0,tb_encoder_16to4);
14
15     din = 16'b0000000000000001;#10;
16     din = 16'b0000000000000010;#10;
17     din = 16'b000000000000000100;#10;
18     din = 16'b0000000000000001000;#10;
19     din = 16'b00000000000000010000;#10;
20     din = 16'b000000000000000100000;#10;
21     din = 16'b0000000000000001000000;#10;
22     din = 16'b00000000000000010000000;#10;
23     din = 16'b000000000000000100000000;#10;
24     din = 16'b0000000000000001000000000;#10;
25     din = 16'b00000000000000010000000000;#10;
26     din = 16'b000000000000000100000000000;#10;
27     din = 16'b0000000000000001000000000000;#10;
28     din = 16'b0100000000000000;#10;
29     din = 16'b1000000000000000;#10;
30
31
32     $finish;
33   end
34
35   initial begin
36     $monitor("Time=%0t | din=%b |
37     out=%b", $time,din,out);
38   end
39 endmodule
```

```
3   input [15:0]din,
4   output reg [3:0]out
5 );
6   always @(*) begin
7     case (din)
8       16'b0000000000000001:out =
9         4'b0000;
10      16'b0000000000000010:out =
11        4'b0001;
12      16'b000000000000000100:out =
13        4'b0010;
14      16'b0000000000000001000:out =
15        4'b0011;
16      16'b00000000000000010000:out =
17        4'b0100;
18      16'b000000000000000100000:out =
19        4'b0101;
20      16'b0000000000000001000000:out =
21        4'b0110;
22      16'b00000000000000010000000:out =
23        4'b0111;
24      16'b000000000000000100000000:out =
25        4'b1000;
26      16'b0000000000000001000000000:out =
27        4'b1001;
28      16'b00000000000000010000000000:out =
29        4'b1010;
30      16'b00001000000000000000:out =
31        4'b1011;
32      16'b00010000000000000000:out =
33        4'b1100;
34      16'b00100000000000000000:out =
35        4'b1101;
36      16'b01000000000000000000:out =
37        4'b1110;
38      16'b10000000000000000000:out =
39        4'b1111;
40
41     endcase
42   end
43 endmodule
```

Output:

Time=0 | sel=0000 | out=0000000000000001
Time=10 | sel=0001 | out=0000000000000010
Time=20 | sel=0010 | out=000000000000000100
Time=30 | sel=0011 | out=0000000000000001000
Time=40 | sel=0100 | out=00000000000010000
Time=50 | sel=0101 | out=00000000000100000
Time=60 | sel=0110 | out=0000000001000000
Time=70 | sel=0111 | out=0000000010000000
Time=80 | sel=1000 | out=0000000100000000
Time=90 | sel=1001 | out=0000001000000000
Time=100 | sel=1010 | out=0000010000000000
Time=110 | sel=1011 | out=0000100000000000
Time=120 | sel=1100 | out=0001000000000000
Time=130 | sel=1101 | out=0010000000000000
Time=140 | sel=1110 | out=0100000000000000
Time=150 | sel=1111 | out=1000000000000000

Code for 3:8 decoder

```
1 // Code your testbench here
2 // or browse Examples
3 module tb_decoder_3to8;
4     reg [2:0]sel;
5     wire [7:0]out;
6
7     decoder_3to8 uut (.sel(sel),.out(out)
8 );
9
10    initial begin
11        $dumpfile("decoder_3to8.vcd");
12        $dumpvars(0,tb_decoder_3to8);
13
14        sel = 3'b000;#10;
15        sel = 3'b001;#10;
16        sel = 3'b010;#10;
17        sel = 3'b011;#10;
18        sel = 3'b100;#10;
19        sel = 3'b101;#10;
20        sel = 3'b110;#10;
21        sel = 3'b111;#10;
22
23        $finish;
24    end
25
26    initial begin
27        $monitor("Time=%0t | sel=%b |
28 out=%b", $time,sel,out);
29    end
30 endmodule
```

```
1 // Code your design here
2 module decoder_3to8 (
3     input [2:0]sel,
4     output reg [7:0]out
5 );
6     always @(*) begin
7         out = 8'b00000000;
8         out[sel] = 1'b1;
9     end
10 endmodule
```

Output:

Time=0 | sel=000 | out=00000001
Time=10 | sel=001 | out=00000010
Time=20 | sel=010 | out=00000100
Time=30 | sel=011 | out=00001000
Time=40 | sel=100 | out=00010000
Time=50 | sel=101 | out=00100000
Time=60 | sel=110 | out=01000000
Time=70 | sel=111 | out=10000000