# Code for 1:2 demultiplexer

```
// Code your testbench here
// or browse Examples
module tb_demux_1to2;
reg a,sel;
wire y0,y1;

demux_1to2 uut (.a(a),.sel(sel),.y0(y0),
.y1(y1));

initial begin
    $dumpfile("demux_1to2.vcd");
    $dumpvars(0,tb_demux_1to2);
    $monitor("Time = %0t | a = %b | sel =
%b | y0 = %b | y1 = %b",$time,a,sel,y0,
y1);

    a = 0;sel = 0;#10;
    a = 1;sel = 0;#10;
    a = 1;sel = 1;#10;
    a = 0;sel = 1;#10;
    #10;

    $finish;
end
endmodule
```

```
// Code your design here
module demux_1to2 (
    input a,
    input sel,
    output reg y0,
    output reg y1
);
always @(*) begin
    case(sel)
        1'b0:begin
            y0 = a;
            y1 = 0;
        end
        1'b1:begin
            y0 = 0;
            y1 = a;
        end
    endcase
end
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

```
// Code your testbench here
// or browse Examples
module tb_demux_1x4;

reg a;
reg [1:0]sel;
wire y0,y1,y2,y3;

demux_1x4 uut
(.a(a),.sel(sel),.y0(y0),.y1(y1),.y2(y2),.y3(y3)
);

initial begin
    $display("Time\t a sel\t y0 y1 y2 y3");

  $monitor("%0t\t %b %b\t %b  %b  %b  %b", $time, a,sel, y0,y1,y2,y3);

    // Test cases
    a = 0; sel = 2'b00;#10;
    a = 1; sel = 2'b00;#10;
    a = 1; sel = 2'b01;#10;
    a = 1; sel = 2'b10;#10;
    a = 1; sel = 2'b11;#10;
    a = 0; sel = 2'b11;#10;
#10;

    $finish;
end

endmodule
```

```
// Code your design here

module demux_1x4 (
    input wire a,
    input wire [1:0]sel,
    output reg y0,y1,y2,y3
);

always @ (a or sel) begin
    // Default outputs
    y0 = 0; y1 = 0; y2 = 0; y3 = 0;

    case (sel)
        2'b00:y0 = a;
        2'b01:y1 = a;
        2'b10:y2 = a;
        2'b11:y3 = a;
    endcase
end

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

```
// Code your testbench here
// or browse Examples
module tb_demux_1to8;
    reg din;
    reg [2:0]sel;
    wire [7:0]out;

  demux_1to8
uut(.din(din),.sel(sel),.out(out)
    );

    initial begin
        $dumpfile("demux_1to8.vcd");
        $dumpvars(0, tb_demux_1to8);

        din = 1;
        sel = 3'b000; #10;
        sel = 3'b001; #10;
        sel = 3'b010; #10;
        sel = 3'b011; #10;
        sel = 3'b100; #10;
        sel = 3'b101; #10;
        sel = 3'b110; #10;
        sel = 3'b111; #10;

        din =0; sel =3'b101;#10;

        $finish;
    end

    initial begin
        $monitor("Time=%0t | din=%b |
sel=%b | out=%b", $time, din, sel, out);
    end
endmodule
```

```
// Code your design here
module demux_1to8(
    input din,
    input [2:0]sel,
    output reg [7:0]out
);
    always @(*) begin
        out = 8'b00000000;
        out[sel] =din;
    end
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

```
// Code your testbench here
// or browse Examples
module tb_encoder_4to2;
    reg [3:0]din;
    wire [1:0]out;

    encoder_4to2 uut (.din(din),.out(out)
    );

    initial begin
        $dumpfile("encoder_4to2.vcd");
        $dumpvars(0, tb_encoder_4to2);

        din = 4'b0001;#10;
        din = 4'b0010;#10;
        din = 4'b0100;#10;
        din = 4'b1000;#10;
        din = 4'b0000;#10;

        $finish;
    end

    initial begin
        $monitor("Time=%0t | din=%b |
out=%b", $time,din,out);
    end
endmodule
```

```
// Code your design here
module encoder_4to2 (
  input [3:0]din,
    output reg [1:0]out
);
    always @(*) begin
        case (din)
            4'b0001:out = 2'b00;
            4'b0010:out = 2'b01;
            4'b0100:out = 2'b10;
            4'b1000:out = 2'b11;

        endcase
    end
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

```
// Code your testbench here
// or browse Examples
module tb_encoder_8to3;
    reg [7:0]din;
    wire [2:0]out;

    encoder_8to3 uut (.din(din),.out(out)
    );

    initial begin
        $dumpfile("encoder_8to3.vcd");
        $dumpvars(0,tb_encoder_8to3);

        din = 8'b00000001;#10;
        din = 8'b00000010;#10;
        din = 8'b00000100;#10;
        din = 8'b00001000;#10;
        din = 8'b00010000;#10;
        din = 8'b00100000;#10;
        din = 8'b01000000;#10;
        din = 8'b10000000;#10;
        din = 8'b00000000;#10;

        $finish;
    end

    initial begin
        $monitor("Time=%0t | din=%b |
out=%b", $time,din,out);
    end
endmodule
```

```
// Code your design here
module encoder_8to3 (
    input [7:0]din,
    output reg [2:0]out
);
    always @(*) begin
        case (din)
            8'b00000001:out = 3'b000;
            8'b00000010:out = 3'b001;
            8'b00000100:out = 3'b010;
            8'b00001000:out = 3'b011;
            8'b00010000:out = 3'b100;
            8'b00100000:out = 3'b101;
            8'b01000000:out = 3'b110;
            8'b10000000:out = 3'b111;

        endcase
    end
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

```
// Code your testbench here
// or browse Examples
module tb_encoder_16to4;
    reg [15:0]din;
    wire [3:0]out;

    encoder_16to4 uut (
.din(din),.out(out)
    );

    initial begin
        $dumpfile("encoder_16to4.vcd");
        $dumpvars(0,tb_encoder_16to4);

        din = 16'b0000000000000001;#10;
        din = 16'b0000000000000010;#10;
        din = 16'b0000000000000100;#10;
        din = 16'b0000000000001000;#10;
        din = 16'b0000000000010000;#10;
        din = 16'b0000000000100000;#10;
        din = 16'b0000000001000000;#10;
        din = 16'b0000000010000000;#10;
        din = 16'b0000000100000000;#10;
        din = 16'b0000001000000000;#10;
        din = 16'b0000010000000000;#10;
        din = 16'b0000100000000000;#10;
        din = 16'b0001000000000000;#10;
        din = 16'b0010000000000000;#10;
        din = 16'b0100000000000000;#10;
        din = 16'b1000000000000000;#10;


        $finish;
    end

    initial begin
      $monitor("Time=%0t | din=%b |
out=%b", $time,din,out);
    end
endmodule
```

```
    input [15:0]din,
    output reg [3:0]out
);
    always @(*) begin
        case (din)
            16'b0000000000000001:out =
4'b0000;
            16'b0000000000000010:out =
4'b0001;
            16'b0000000000000100:out =
4'b0010;
            16'b0000000000001000:out =
4'b0011;
            16'b0000000000010000:out =
4'b0100;
            16'b0000000000100000:out =
4'b0101;
            16'b0000000001000000:out =
4'b0110;
            16'b0000000010000000:out =
4'b0111;
            16'b0000000100000000:out =
4'b1000;
            16'b0000001000000000:out =
4'b1001;
            16'b0000010000000000:out =
4'b1010;
            16'b0000100000000000:out =
4'b1011;
            16'b0001000000000000:out =
4'b1100;
            16'b0010000000000000:out =
4'b1101;
            16'b0100000000000000:out =
4'b1110;
            16'b1000000000000000:out =
4'b1111;

        endcase
    end
endmodule
```

# Output:

```
Time=0   | sel=0000 | out=0000000000000001
Time=10  | sel=0001 | out=0000000000000010
Time=20  | sel=0010 | out=0000000000000100
Time=30  | sel=0011 | out=0000000000001000
Time=40  | sel=0100 | out=0000000000010000
Time=50  | sel=0101 | out=0000000000100000
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

```
// Code your testbench here
// or browse Examples
module tb_decoder_3to8;
    reg [2:0]sel;
    wire [7:0]out;

    decoder_3to8 uut (.sel(sel),.out(out)
    );

    initial begin
        $dumpfile("decoder_3to8.vcd");
        $dumpvars(0,tb_decoder_3to8);

        sel = 3'b000;#10;
        sel = 3'b001;#10;
        sel = 3'b010;#10;
        sel = 3'b011;#10;
        sel = 3'b100;#10;
        sel = 3'b101;#10;
        sel = 3'b110;#10;
        sel = 3'b111;#10;

        $finish;
    end

    initial begin
        $monitor("Time=%0t | sel=%b |
out=%b", $time,sel,out);
    end
endmodule
```

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