Program 7 : Design Verilog program to implement types of D-Multiplexer.

Program 1:8 demux

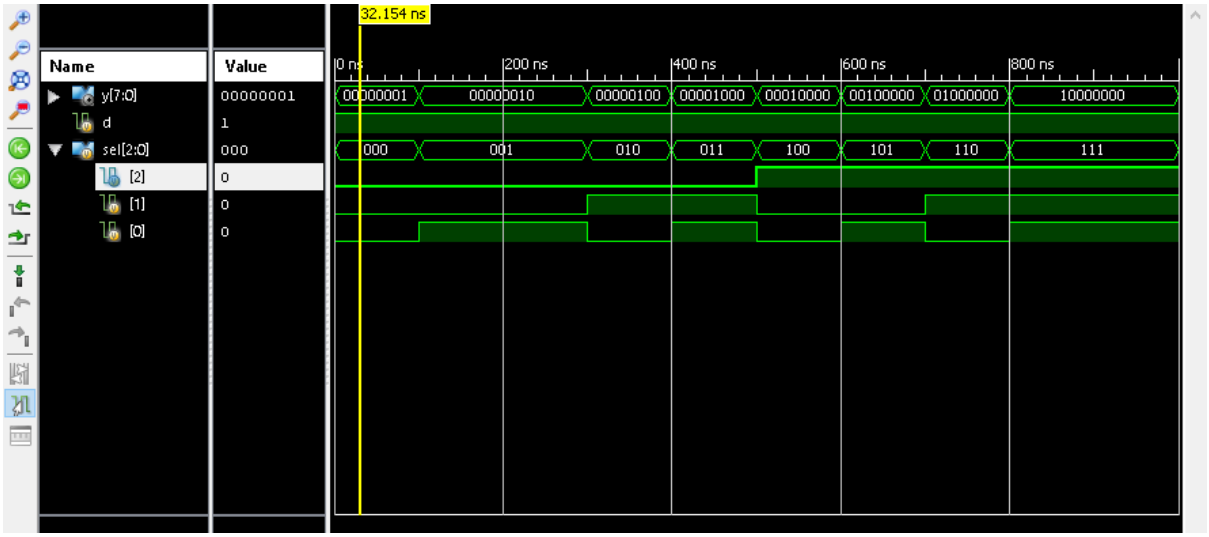
```
module demux_1_8(
    input d,
    input [2:0] sel,
    output reg[7:0] y  );
always@(d or sel)
begin
y=8'b0;
case(sel)
3'b000:y[0]=d;
3'b001:y[1]=d;
3'b010:y[2]=d;
3'b011:y[3]=d;
3'b100:y[4]=d;
3'b101:y[5]=d;
3'b110:y[6]=d;
3'b111:y[7]=d;
default:y=8'b0;
endcase
end
endmodule
```

Test Benches

```
module t_demux_1_8;
// Inputs
reg d;
```

```
reg [2:0] sel;
// Outputs
wire [7:0] y;
// Instantiate the Unit Under Test (UUT)
demux_1_8 uut (
        .d(d),
        .sel(sel),
        .y(y)
);
initial begin
        // Initialize Inputs
        d = 1;
        sel = 3'b000;
        // Wait 100 ns for global reset to finish
                #100;       sel = 3'b001;
          #100;         sel = 3'b010;
          #100;         sel = 3'b011;
          #100;         sel = 3'b100;
          #100;         sel = 3'b101;
          #100;         sel = 3'b110;
          #100;         sel = 3'b111;
        end
    endmodule
```

Output



Program 1:4 Demux

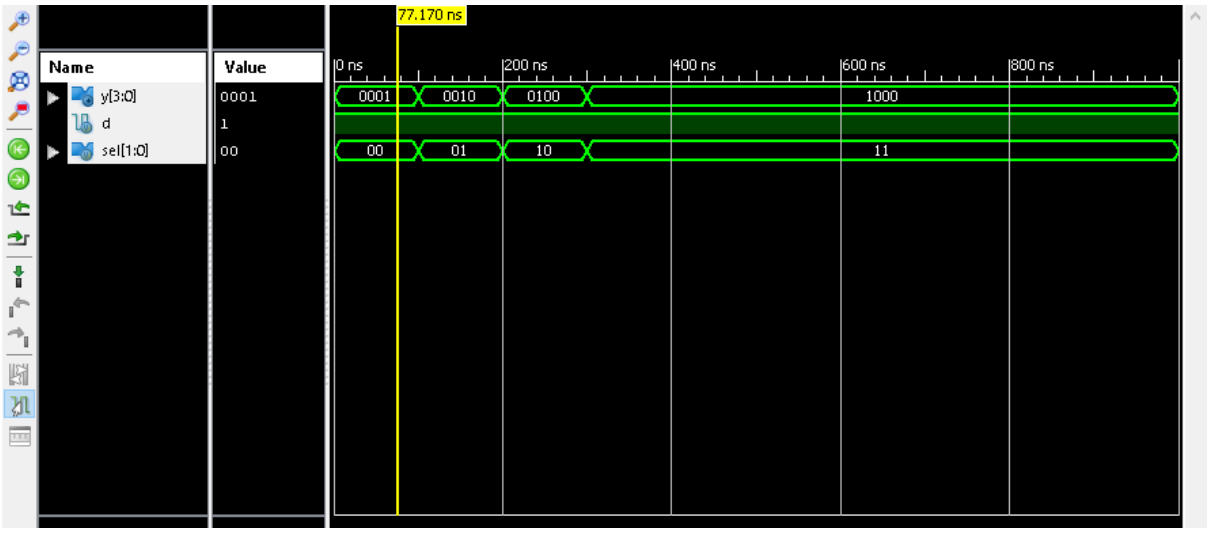
```
module demux_1_4(
    input d,
    input [1:0] sel,
    output  reg [3:0]y  );
always@(d or sel)
begin
y=4'b0;
case(sel)
2'b00:y[0]=d;
2'b001:y[1]=d;
2'b10:y[2]=d;
2'b11:y[3]=d;
default:y=4'b0;
endcase
end
endmodule
```

Test Benches :

```
module T_demux_1_4;
// Inputs
reg d;
reg [1:0] sel;

// Outputs
wire [3:0] y;
// Instantiate the Unit Under Test (UUT)
demux_1_4 uut (
        .d(d),
        .sel(sel),
        .y(y)
);
initial begin
        // Initialize Inputs
        d = 1;
        sel = 2'b00;
        // Wait 100 ns for global reset to finish
        #100;sel = 2'b01;
        #100;sel = 2'b10;
        // Add stimulus here
#100;sel = 2'b11;
end
  endmodule
```

Output:


77.170 ns
Name
Value
y[3:0]
d
sel[1:0]
0001
1
00
0 ns
200 ns
400 ns
600 ns
800 ns
0001
0010
0100
1000
00
01
10
11