Lab Program 6.

Design Verilog Program for implement different types of multiplexer like 2:1,4:1 and 8:1.

Program for 2: 1 mux

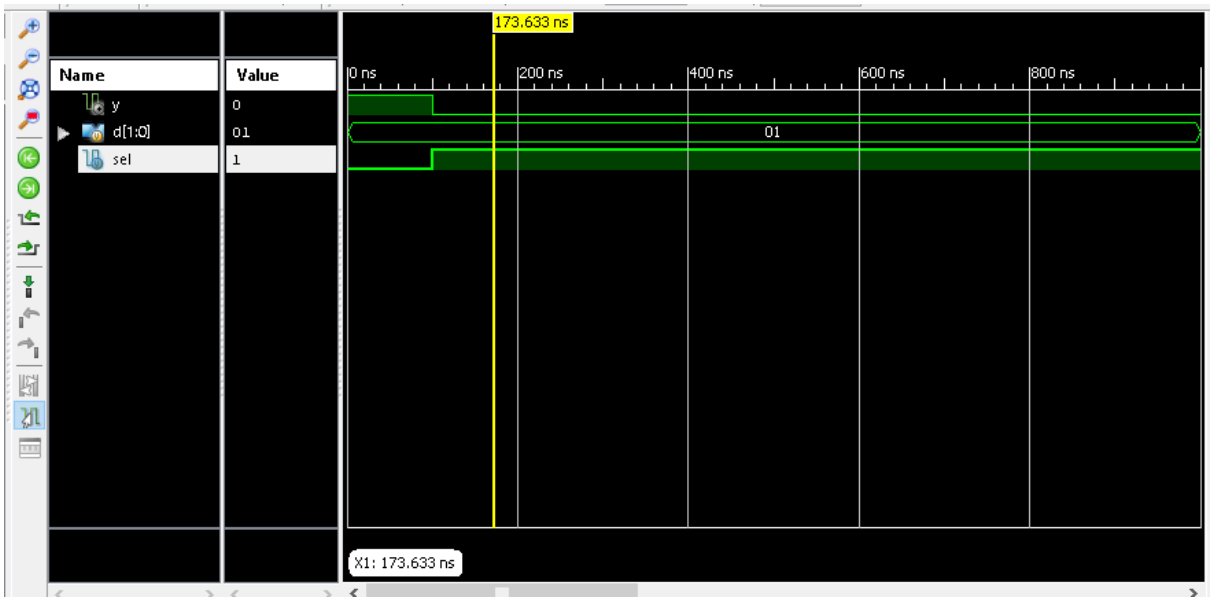
```
module Lab_program_mux_2_1(
   input[1:0] d,
   input sel,
   output reg y
   );
always@(sel,d)
begin
case(sel)
1'b0: y=d[0];
1'b1:y=d[1];
default:y=0;
endcase
end
endmodule
```

Test benches:

```
module T_lab_Program_mux2_1;
// Inputs
reg [1:0] d;
reg sel;
// Outputs
wire y;
// Instantiate the Unit Under Test (UUT)
Lab_program_mux_2_1 uut (
```

```
        .d(d),
        .sel(sel),
        .y(y)
);
initial begin
        // Initialize Inputs
        d = 1'b1;
        sel = 1'b0;
        // Wait 100 ns for global reset to finish
        #100;sel=1'b1;
                // Add stimulus here
end
    endmodule
```

OUTPUT



Program for multiplexer 4:1

```
module Lab_mux_4_1(
    input [3:0] d,
    input [1:0] sel,
    output reg y
    );
always@ (sel,d)
begin
case(sel)
2'b00:y=d[0];
2'b01:y=d[1];
2'b10:y=d[2];
2'b11:y=d[3];
default:y=0;
endcase
end

endmodule
```

Test Benches:

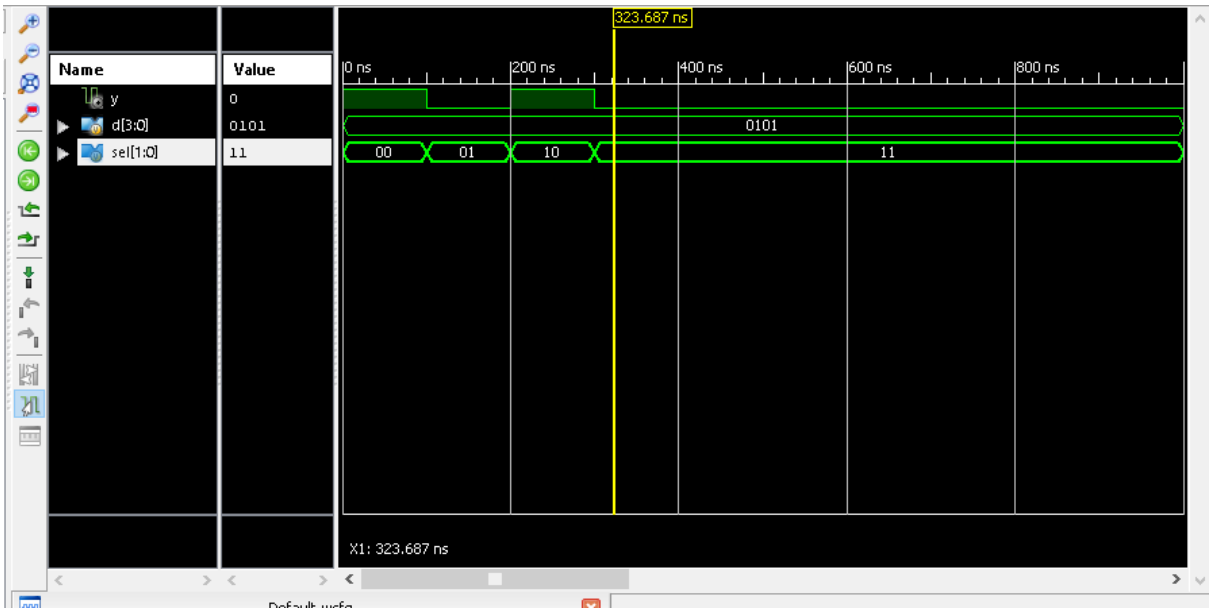
```
module T_mux_4_1;

// Inputs
reg [3:0] d;
reg [1:0] sel;

// Outputs
```

```
wire y;
// Instantiate the Unit Under Test (UUT)
Lab_mux_4_1 uut (
        .d(d),
        .sel(sel),
        .y(y)
);
initial begin
        // Initialize Inputs
        d = 4'b0101;
        sel = 2'b00;
        // Wait 100 ns for global reset to finish
        #100;sel=2'b01;
        #100;sel=2'b10;
                        #100;sel=2'b11;
                        // Add stimulus here
end
    endmodule
```

Output



Program 8:1 MUX

```
module Lab_program_mux8_1( input [7:0] d,
    input [2:0] sel,
    output reg y);
always@(sel,d)
begin
case(sel)
3'b000 :y=d[0];
3'b001 :y=d[1];
3'b010 :y=d[2];
3'b011 :y=d[3];
3'b100 :y=d[4];
3'b101 :y=d[5];
3'b110 :y=d[6];
3'b111 :y=d[7];
default: y=0;
```

```
endcase
end
endmodule
```

Test Benches:

```
module T_mux_8_1;

// Inputs
reg [7:0] d;
reg [2:0] sel;

// Outputs
wire y;
// Instantiate the Unit Under Test (UUT)
Lab_program_mux2_1 uut (
        .d(d),
        .sel(sel),
        .y(y)
);
initial begin
        // Initialize Inputs
        d = 8'b10110011;
        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;
        // Add stimulus here
end
    endmodule
```

OUTPUT