

LAB 6: Designing of Multiplexer, Demultiplexer, Decoder, Encoder

Question 1: Write a Verilog code to design a 4:1 Multiplexer and verify the same.

design.sv



```

1 module mux_4to1_case ( input [3:0] a,
2                       input [3:0] b,
3                       input [3:0] c,
4                       input [3:0] d,
5                       input [1:0] sel,
6                       output reg [3:0] out);
7     always @ (a or b or c or d or sel) begin
8         case (sel)
9             2'b00 : out <= a;
10            2'b01 : out <= b;
11            2'b10 : out <= c;
12            2'b11 : out <= d;
13        endcase
14    end
15 endmodule

```

testbench.sv

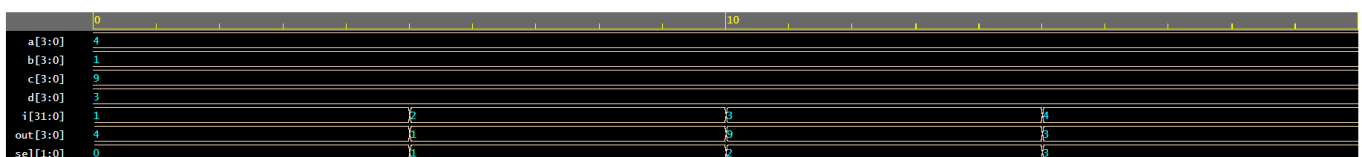


SV/Verilog Tes

```

1 module tb_4to1_mux;
2
3     reg [3:0] a;
4     reg [3:0] b;
5     reg [3:0] c;
6     reg [3:0] d;
7     wire [3:0] out;
8     reg [1:0] sel;
9     integer i;
10
11     mux_4to1_case mux0 ( .a (a),
12                         .b (b),
13                         .c (c),
14                         .d (d),
15                         .sel (sel),
16                         .out (out));
17
18     initial
19     begin
20         $monitor ("%0t sel=0x%0h a=0x%0h b=0x%0h c=0x%0h d=0x%0h
21 out=0x%0h", $time, sel, a, b, c, d, out);
22         sel <= 0;
23         a <= $random;
24         b <= $random;
25         c <= $random;
26         d <= $random;
27         $dumpfile("dump.vcd");
28         $dumpvars(1);
29         for (i = 1; i < 4; i=i+1) begin
30             #5 sel <= i;
31         end
32         #5 $finish;
33     end
34 endmodule

```



Question 2: Write a Verilog code to design an 8:1 Multiplexer and verify the same.

design sv



```

1 module Moltiplexer(d0,d1,d2,d3,d4,d5,d6,d7,sel,out);
2   input d0,d1,d2,d3,d4,d5,d6,d7;
3   input [2:0] sel;
4   output reg out;
5   always@(sel)
6     begin
7       case(sel)
8         3'b000:out=d0;
9         3'b001:out=d1;
10        3'b010:out=d2;
11        3'b011:out=d3;
12        3'b100:out=d4;
13        3'b101:out=d5;
14        3'b110:out=d6;
15        3'b111:out=d7;
16      endcase
17    end
18 endmodule

```

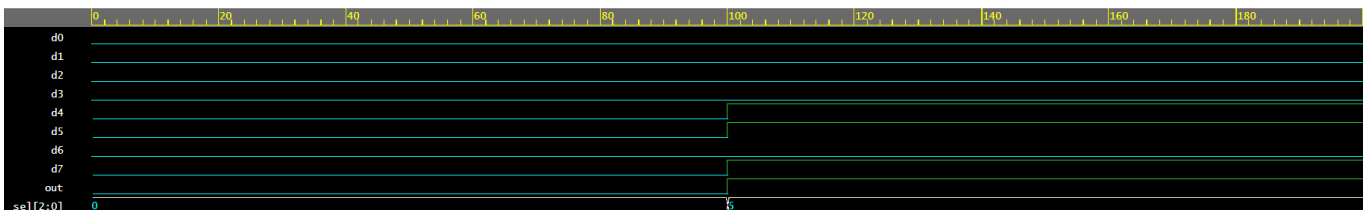
testbench sv



```

1 module TestModule;
2
3   reg d0, d1, d2, d3, d4, d5, d6, d7;
4   reg [2:0] sel;
5   wire out;
6
7   Moltiplexer uut(.d0(d0),.d1(d1),.d2(d2),.d3(d3),.d4(d4),.d5(d5),.d6(d6),.d7(d7),.sel(sel),.out(out));
8
9   initial
10    begin
11      d0 = 0;
12      d1 = 0;
13      d2 = 0;
14      d3 = 0;
15
16      d4 = 0;
17      d5 = 0;
18      d6 = 0;
19      d7 = 0;
20      sel = 0;
21      #100;
22      d0 = 0;
23      d1 = 0;
24      d2 = 0;
25      d3 = 0;
26      d4 = 1;
27      d5 = 1;
28      d6 = 0;
29      d7 = 1;
30      sel = 5;
31      #100;
32    end
33    initial
34      begin
35        $dumpfile("dump.vcd");
36        $dumpvars(1);
37      end
38  endmodule

```



Question 3: Write a Verilog code to design a 1:4 Demultiplexer and verify the same.

design sv



```

1 module Demultiplexer1to4case (output reg [3:0] Y,
2                               input [1:0] A,
3                               input din);
4     always @(Y, A) begin
5         case (A)
6             2'b00 : begin Y[0] = din; Y[3:1] = 0; end
7             2'b01 : begin Y[1] = din; Y[0] = 0; end
8             2'b10 : begin Y[2] = din; Y[1:0] = 0; end
9             2'b11 : begin Y[3] = din; Y[2:0] = 0; end
10        endcase
11    end
12 endmodule

```

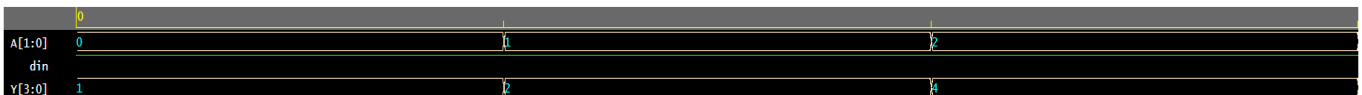
testbench sv



```

1 module Demultiplexer1to4case_tb;
2 wire [3:0] Y;
3 reg [1:0] A;
4 reg din;
5 Demultiplexer1to4case I0 (Y, A, din);
6 initial begin
7     din = 1;
8     A = 2'b00;
9     #1 A = 2'b01;
10    #1 A = 2'b10;
11    #1 A = 2'b11;
12 end
13 initial begin
14     $monitor("%t| Din = %d| A[1] = %d| A[0] = %d| Y[0] = %d| Y[1] = %d| Y[2] = %d| Y[3] = %d",
15             $time, din, A[1], A[0], Y[0], Y[1], Y[2], Y[3]);
16     $dumpfile("dump.vcd");
17     $dumpvars(1);
18 end
19 endmodule

```

**Question 4: Write a Verilog code to design a 1:8 Demultiplexer and verify the same.**

design sv



```

1 module Demultiplexer(in,s0,s1,s2,d0,d1,d2,d3,d4,d5,d6,d7);
2     input in,s0,s1,s2;
3     output d0,d1,d2,d3,d4,d5,d6,d7;
4     assign d0=(in & ~s2 & ~s1 & ~s0),
5            d1=(in & ~s2 & ~s1 & s0),
6            d2=(in & ~s2 & s1 & ~s0),
7            d3=(in & ~s2 & s1 & s0),
8            d4=(in & s2 & ~s1 & ~s0),
9            d5=(in & s2 & ~s1 & s0),
10           d6=(in & s2 & s1 & ~s0),
11           d7=(in & s2 & s1 & s0);
12 endmodule

```

testbench.sv

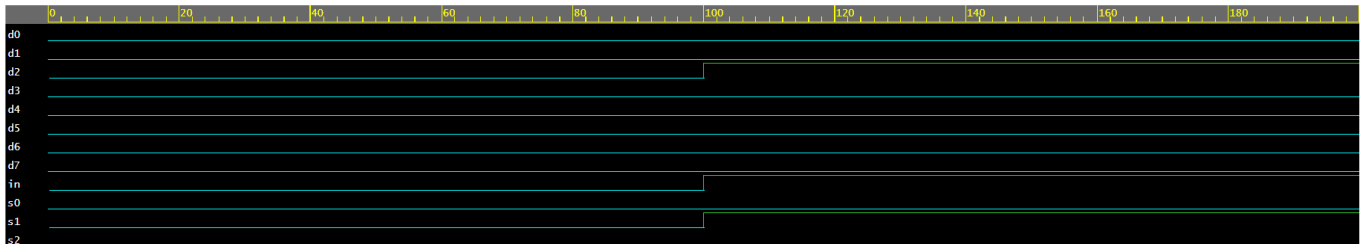


SV/Verilog Testbench

```

1 module TestModule;
2   reg in;
3   reg s0;
4   reg s1;
5   reg s2;
6
7   wire d0;
8   wire d1;
9   wire d2;
10  wire d3;
11  wire d4;
12  wire d5;
13  wire d6;
14  wire d7;
15
16  Demultiplexer d(.in(in), .s0(s0), .s1(s1), .s2(s2), .d0(d0), .d1(d1),
17  .d2(d2), .d3(d3), .d4(d4), .d5(d5), .d6(d6), .d7(d7));
18  initial
19    begin
20      in = 0;
21      s0 = 0;
22      s1 = 0;
23      s2 = 0;
24      #100;
25      in = 1;
26      s0 = 0;
27      s1 = 1;
28      s2 = 0;
29      #100;
30    end
31  initial
32    begin
33      $dumpfile("dump.vcd");
34      $dumpvars(1);
35    end
36 endmodule

```



Question 5: Write a Verilog code to design a 3:8 Decoder and verify the same.

design.sv



```

1 module encoder(Do, Din);
2   input [7:0]Din;
3   output [2:0]Do;
4
5   or(Do[0], Din[4], Din[5], Din[6], Din[7]);
6   or(Do[1], Din[2], Din[3], Din[6], Din[7]);
7   or(Do[2], Din[1], Din[3], Din[5], Din[7]);
8 endmodule
9

```

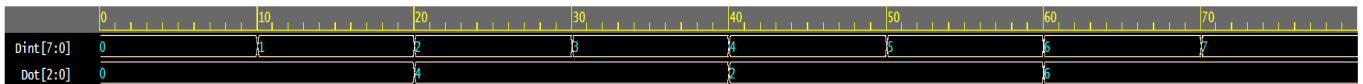
testbench.vv



```

1 module encoder_tb_v;
2   reg [7:0] Dint;
3   wire [2:0] Dot;
4
5   encoder uut(.Do(Dot),.Din(Dint));
6   initial
7     begin
8       Dint = 8'b00000000; #10;
9       Dint = 8'b00000001; #10;
10      Dint = 8'b00000010; #10;
11      Dint = 8'b00000011; #10;
12      Dint = 8'b00000100; #10;
13      Dint = 8'b00000101; #10;
14      Dint = 8'b00000110; #10;
15      Dint = 8'b00000111; #10;
16    end
17  initial
18    begin
19      $dumpfile("dump.vcd");
20      $dumpvars(1);
21    end
22 endmodule

```



Question 6: Write a Verilog code to design a 8:3 Encoder and verify the same.

design.vv



```

1 module encoder83(din, do0, do1, do2);
2   input [7:0]din;
3   output do0, do1, do2;
4   or(do0[0], din[4], din[5], din[6], din[7]);
5   or(do1[1], din[2], din[3], din[6], din[7]);
6   or(do2[2], din[1], din[3], din[5], din[7]);
7 endmodule

```

testbench.vv



```

1 module encoder83_tb;
2   reg [7:0] Din_t;
3   wire DO0, DO1, DO2;
4   encoder83 e1(.din(Din_t), .do0(DO0), .do1(DO1), .do2(DO2));
5   initial
6     begin
7       Din_t = 8'b00000000; #10;
8       Din_t = 8'b00000001; #10;
9       Din_t = 8'b00000010; #10;
10      Din_t = 8'b00000011; #10;
11      Din_t = 8'b00000100; #10;
12      Din_t = 8'b00000101; #10;
13      Din_t = 8'b00000110; #10;
14      Din_t = 8'b00000111; #10;
15    end
16  initial
17    begin
18      $dumpfile("dump.vcd");
19      $dumpvars(1);
20    end
21 endmodule

```

