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

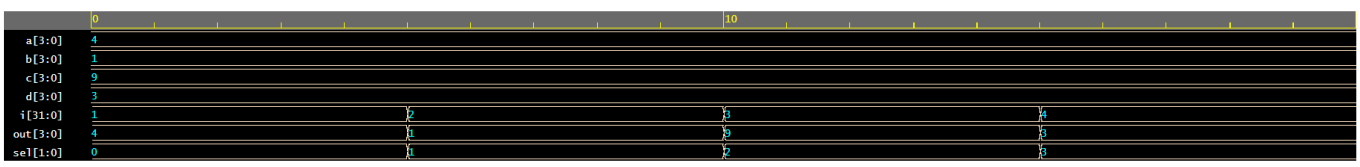
Question 1: Write a Verilog code to design a 4:1 Multiplexer and verify the same.

design.sv

```
module mux_4to1_case ( input [3:0] a,
                       input [3:0] b,
                       input [3:0] c,
                       input [3:0] d,
                       input [1:0] sel,
                       output reg [3:0] out);
   always @ (a or b or c or d or sel) begin
      case (sel)
         2'b00 : out <= a;
         2'b01 : out <= b;
         2'b10 : out <= c;
         2'b11 : out <= d;
      endcase
   end
endmodule
```

testbench.sv

```
module tb_4to1_mux;

   reg [3:0] a;
   reg [3:0] b;
   reg [3:0] c;
   reg [3:0] d;
   wire [3:0] out;
   reg [1:0] sel;
   integer i;

   mux_4to1_case  mux0 (   .a (a),
                           .b (b),
                           .c (c),
                           .d (d),
                           .sel (sel),
                           .out (out));

   initial
     begin
       $monitor ("[%0t] sel=0x%0h a=0x%0h b=0x%0h c=0x%0h d=0x%0h
out=0x%0h", $time, sel, a, b, c, d, out);
       sel <= 0;
      a <= $random;
      b <= $random;
      c <= $random;
      d <= $random;
       $dumpfile("dump.vcd");
      $dumpvars(1);
      for (i = 1; i < 4; i=i+1) begin
         #5 sel <= i;
      end
      #5 $finish;
    end
endmodule
```



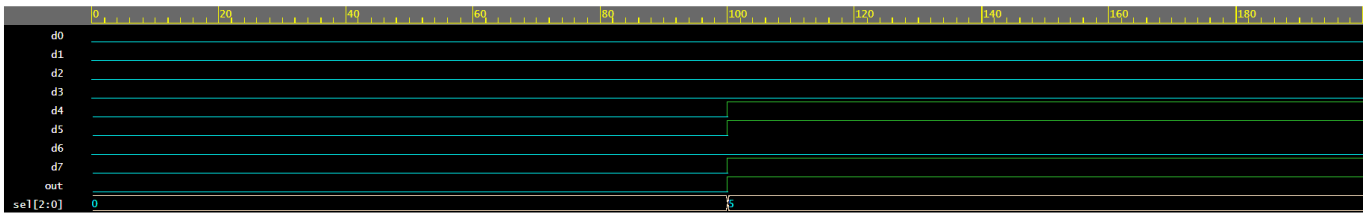
Question 2: Write a Verilog code to design an 8:1 Multiplexer and verify the same.

design.sv

```
module Mulitplexer(d0,d1,d2,d3,d4,d5,d6,d7,sel,out);
  input d0,d1,d2,d3,d4,d5,d6,d7;
  input [2:0] sel;
  output reg out;
  always@(sel)
    begin
      case(sel)
        3'b000:out=d0;
        3'b001:out=d1;
        3'b010:out=d2;
        3'b011:out=d3;
        3'b100:out=d4;
        3'b101:out=d5;
        3'b110:out=d6;
        3'b111:out=d7;
      endcase
    end
endmodule
```

testbench.sv

```
module TestModule;

reg d0, d1, d2, d3, d4, d5, d6, d7;
reg [2:0] sel;
wire out;

Mulitplexer uut(.d0(d0),.d1(d1),.d2(d2),.d3(d3),.d4(d4),.d5(d5),.d6(d6),.d7(d7),.sel(sel),.out(out));

initial
  begin
    d0 = 0;
    d1 = 0;
    d2 = 0;
    d3 = 0;

    d4 = 0;
    d5 = 0;
    d6 = 0;
    d7 = 0;
    sel = 0;
    #100;
    d0 = 0;
    d1 = 0;
    d2 = 0;
    d3 = 0;
    d4 = 1;
    d5 = 1;
    d6 = 0;
    d7 = 1;
    sel = 5;
    #100;
  end
  initial
    begin
      $dumpfile("dump.vcd");
      $dumpvars(1);
    end
endmodule
```



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

design.sv

```
module Demultiplexer1to4case (output reg [3:0] Y,
                              input [1:0] A,
                              input din);
  always @(Y, A) begin
    case (A)
        2'b00 : begin Y[0] = din; Y[3:1] = 0; end
        2'b01 : begin Y[1] = din; Y[0] = 0;   end
        2'b10 : begin Y[2] = din; Y[1:0] = 0; end
        2'b11 : begin Y[3] = din; Y[2:0] = 0; end
    endcase
end
endmodule
```

testbench.sv

```
module Demultiplexer1to4case_tb;
wire [3:0] Y;
reg [1:0] A;
reg din;
Demultiplexer1to4case I0 (Y, A, din);
initial begin
    din = 1;
    A = 2'b00;
#1  A = 2'b01;
#1  A = 2'b10;
#1  A = 2'b11;
end
initial begin
    $monitor("%t| Din = %d| A[1] = %d| A[0] = %d| Y[0] = %d| Y[1] = %d| Y[2] = %d| Y[3] = %d",
              $time, din, A[1], A[0], Y[0], Y[1], Y[2], Y[3]);
    $dumpfile("dump.vcd");
      $dumpvars(1);
end
endmodule
```

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

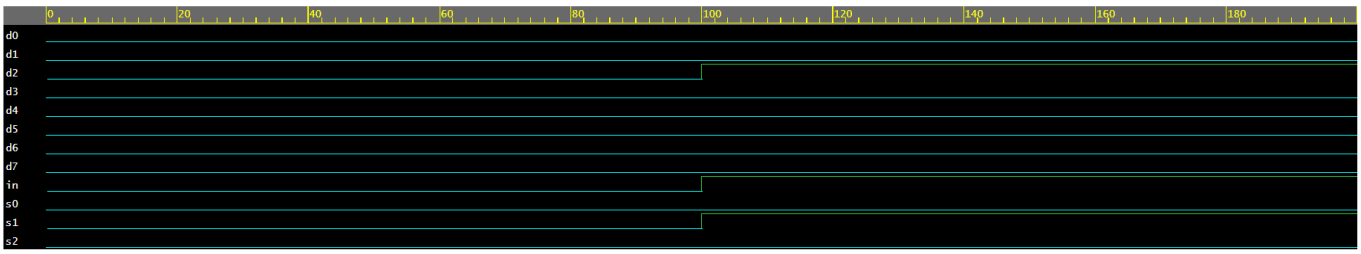
design.sv

```
module Demultiplexer(in,s0,s1,s2,d0,d1,d2,d3,d4,d5,d6,d7);
  input in,s0,s1,s2;
  output d0,d1,d2,d3,d4,d5,d6,d7;
  assign d0=(in & ~s2 & ~s1 &~s0),
    d1=(in & ~s2 & ~s1 &s0),
    d2=(in & ~s2 & s1 &~s0),
    d3=(in & ~s2 & s1 &s0),
    d4=(in & s2 & ~s1 &~s0),
    d5=(in & s2 & ~s1 &s0),
    d6=(in & s2 & s1 &~s0),
    d7=(in & s2 & s1 &s0);
endmodule
```

testbench.sv

SV/Verilog Testbench

```
module TestModule;
  reg in;
  reg s0;
  reg s1;
  reg s2;

  wire d0;
  wire d1;
  wire d2;
  wire d3;
  wire d4;
  wire d5;
  wire d6;
  wire d7;

  Demultiplexer d(.in(in), .s0(s0), .s1(s1), .s2(s2), .d0(d0), .d1(d1),
.d2(d2), .d3(d3), .d4(d4), .d5(d5), .d6(d6), .d7(d7));
  initial
    begin
      in = 0;
      s0 = 0;
      s1 = 0;
      s2 = 0;
      #100;
      in = 1;
      s0 = 0;
      s1 = 1;
      s2 = 0;
      #100;
    end
  initial
    begin
      $dumpfile("dump.vcd");
      $dumpvars(1);
    end
  endmodule
```



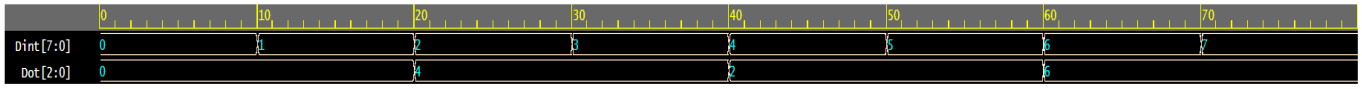
Question 5: Write a Verilog code to design a 3:8 Decoder and verify the same.

design.sv

```
module encoder(Do, Din);
  input [7:0]Din;
  output [2:0]Do;

  or(Do[0], Din[4], Din[5], Din[6], Din[7]);
  or(Do[1], Din[2], Din[3], Din[6], Din[7]);
  or(Do[2], Din[1], Din[3], Din[5], Din[7]);
endmodule

```

testbench.sv

```
module encoder_tb_v;
  reg [7:0] Dint;
  wire [2:0] Dot;

    encoder uut(.Do(Dot),.Din(Dint));
    initial
      begin
        Dint = 8'b00000000; #10;
        Dint = 8'b00000001; #10;
        Dint = 8'b00000010; #10;
        Dint = 8'b00000011; #10;
        Dint = 8'b00000100; #10;
        Dint = 8'b00000101; #10;
        Dint = 8'b00000110; #10;
        Dint = 8'b00000111; #10;
    end
  initial
    begin
      $dumpfile("dump.vcd");
      $dumpvars(1);
    end
endmodule
```



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

design.sv

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

testbench.sv

```
module encoder83_tb;
  reg [7:0] Din_t;
  wire DO0, DO1, DO2;
  encoder83 e1(.din(Din_t), .do0(DO0), .do1(DO1), .do2(DO2));
    initial
      begin
        Din_t = 8'b00000000; #10;
        Din_t = 8'b00000001; #10;
        Din_t = 8'b00000010; #10;
        Din_t = 8'b00000011; #10;
        Din_t = 8'b00000100; #10;
        Din_t = 8'b00000101; #10;
        Din_t = 8'b00000110; #10;
        Din_t = 8'b00000111; #10;
    end
  initial
    begin
      $dumpfile("dump.vcd");
      $dumpvars(1);
    end
endmodule
```