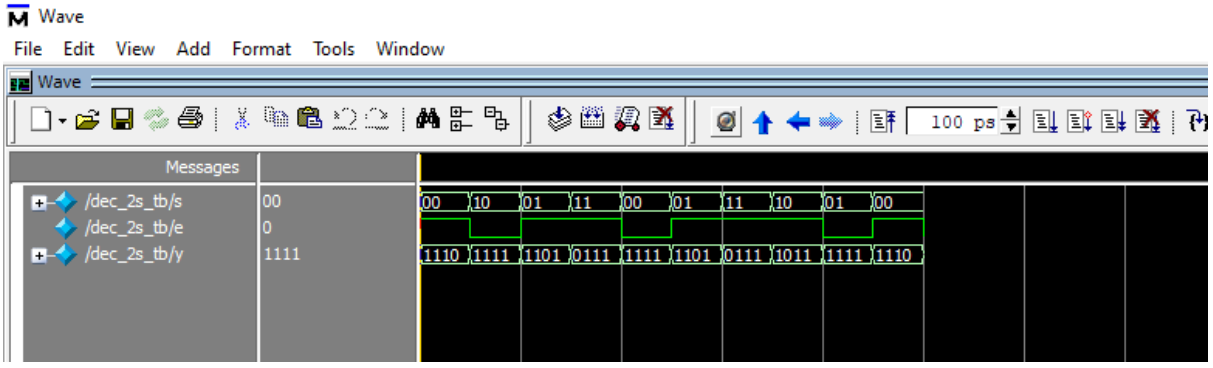
Exercitiul 1:

D:/ModelsimProjects/dec_2s.v

```
1   module dec_2s (
2     input [1:0]s,
3     input e,
4     output [3:0] y
5     );
6
7     assign y[0] = ~(e & ~s[1] & ~s[0]);
8     assign y[1] = ~(e & ~s[1] & s[0]);
9     assign y[2] = ~(e & s[1] & ~s[0]);
10    assign y[3] = ~(e & s[1] & s[0]);
11  endmodule
12
```

D:/ModelsimProjects/dec_2s_tb.v

```
1   module dec_2s_tb (
2     output reg [1:0] s,
3     output reg e,
4     output [3:0] y
5     );
6
7     dec_2s DUT (
8         .s(s),
9         .e(e),
10        .y(y)
11        );
12
13    initial begin
14    e = 1'd1;
15    #10 e = 1'd0;
16    #10 e = 1'd1;
17    #20 e = 1'd0;
18    #10 e = 1'd1;
19    #30 e = 1'd0;
20    #10 e = 1'd1;
21    #10 e = 1'd0;
22    end
23
24    initial begin
25    s = 2'h0;
26    #10 s = 2'h2;
27    #10 s = 2'h1;
28    #10 s = 2'h3;
29    #10 s = 2'h0;
30    #10 s = 2'h1;
31    #10 s = 2'h3;
32    #10 s = 2'h2;
33    #10 s = 2'h1;
34    #10 s = 2'h0;
35  end
36  endmodule
```



Exercitiul 2:

D:/ModelsimProjects/mux_2s.v

```
Ln#
 1  module mux_2s # (
 2      parameter w = 4
 3    ) (
 4        input [w-1:0] d0,d1,d2,d3,
 5        input [1:0] s,
 6        output reg [w-1:0] o
 7        );
 8
 9      always @ (*) begin
10        case(s)
11          2'b00: o <= d0;
12          2'b01: o <= d1;
13          2'b10: o <= d2;
14          2'b11: o <= d3;
15        endcase
16      end
17    endmodule
18
```

Exercitiul 3:

D:/ModelsimProjects/cntr.v

```
Ln#
 1  module rgst # ( parameter w=8, parameter iv={w{1'b0}} ) (
 2    input [w-1:0] d,
 3    input clk, clr, rst_b, c_up,
 4    output reg [w-1:0] q
 5    );
 6
 7  always @ ( posedge clk, negedge rst_b ) begin
 8    if( !rst_b )
 9      q <= iv;
10
11    else if ( clr )
12      q <= iv;
13
14    else if ( c_up )
15      q <= d;
16    end
17  endmodule
18
```

Exercitiul 4:

D:/ModelsimProjects/regfl_4x8.v

```
module regfl_4x8 (
  input clk ,
  input rst_b ,
  input [7:0] wr_data ,
  input [1:0] wr_addr ,
  input [1:0] rd_addr ,
  input wr_e ,
  input rd_e ,
  output [7:0] rd_data
  );

  wire [3:0] w;
  wire [7:0] q0, q1, q2, q3;

  dec_2x4 decoder (
          .s(wr_addr),
          .e(wr_e),
          .y(w)
        );
  rgst # ( .w(8) ) register1 (
                .d(wr_data),
                .ld(w[0]),
                .clk(clk),
                .rst_b(rst_b),
                .clr(rd_e),
                .q(q0)
              );

  rgst # ( .w(8) ) register2 (
                .d(wr_data),
                .ld(w[1]),
                .clk(clk),
                .rst_b(rst_b),
                .clr(rd_e),
                .q(q1)
              );

  rgst # ( .w(8) ) register3 (
                .d(wr_data),
                .ld(w[2]),
                .clk(clk),
                .rst_b(rst_b),
                .clr(rd_e),
                .q(q2)
              );

  rgst # ( .w(8) ) register4 (
                .d(wr_data),
                .ld(w[3]),
                .clk(clk),
                .rst_b(rst_b),
                .clr(rd_e),
                .q(q3)
              );

  mux_2s # ( .w(8) ) multiplexor (
                .s(rd_addr),
                .d0(q0),
                .d1(q1),
                .d2(q2),
                .d3(q3),
                .o(rd_data)
                );
endmodule
```

D:/ModelsimProjects/dec_2x4.v

```
module dec_2x4 (
  input [1:0] s,
  input e,
  output reg [3:0] y
  );

always @ (*)

  case ( {e, s} )
    3'b100 : y = 4'b1110;
    3'b101 : y = 4'b1101;
    3'b110 : y = 4'b1011;
    3'b111 : y = 4'b0111;
    3'b0?? : y = 4'b1111;
  endcase
endmodule
```

D:/ModelsimProjects/mux_2s.v

```
Ln#
 1  module mux_2s # (
 2      parameter w = 4
 3    ) (
 4        input [w-1:0] d0,d1,d2,d3,
 5        input [1:0] s,
 6        output reg [w-1:0] o
 7        );
 8
 9      always @ (*) begin
10        case(s)
11          2'b00: o <= d0;
12          2'b01: o <= d1;
13          2'b10: o <= d2;
14          2'b11: o <= d3;
15        endcase
16      end
17    endmodule
18
```

D:/ModelsimProjects/rgst.v

```
Ln#
 1  module rgst # ( parameter w=8, parameter iv={w{1'b0}} ) (
 2    input [w-1:0] d,
 3    input clk, clr, rst_b, ld,
 4    output reg [w-1:0] q
 5    );
 6
 7  always @ ( posedge clk, negedge rst_b ) begin
 8    if( !rst_b )
 9      q <= iv;
10
11    else if ( clr )
12      q <= iv;
13
14    else if ( ld )
15      q <= d;
16    end
17  endmodule
18
19
```

Testbench :

D:/ModelsimProjects/regfl_4x8_tb.v

```
Ln#
 1  module regfl_4x8_tb(
 2    output reg clk ,
 3    output reg rst_b ,
 4    output reg [7:0] wr_data ,
 5    output reg [1:0] wr_addr ,
 6    output reg [1:0] rd_addr ,
 7    output reg wr_e ,
 8    output reg rd_e ,
 9    output [7:0] rd_data);
10
11  regfl_4x8 DUT(.clk(clk), .rst_b(rst_b), .wr_data(wr_data), .wr_addr(wr_addr), .rd_addr(rd_addr), .wr_e(wr_e), .rd_e(rd_e), .rd_data(rd_data));
12  initial begin
13    clk=1'b0;
14    forever #10 clk=~clk;
15  end
16
17  initial begin
18    rst_b=0;
19    #5 rst_b=1;
20  end
21
22  initial begin
23    wr_addr=2'h0;
24    #20 wr_addr=2'h2;
25    #20 wr_addr=2'h1;
26    #20 wr_addr=2'h3;
27    #20 wr_addr=2'h0;
28    #20 wr_addr=2'h1;
29    #20 wr_addr=2'h3;
30    #20 wr_addr=2'h2;
31    #20 wr_addr=2'h3;
32  end
33
34  initial begin
35    wr_data=8'ha2;
36    #20 wr_data=8'h2e;
37    #20 wr_data=8'h98;
38    #20 wr_data=8'h55;
39    #20 wr_data=8'h20;
40    #20 wr_data=8'hff;
41    #20 wr_data=8'hc7;
42    #20 wr_data=8'hb5;
43    #20 wr_data=8'h91;
44  end
45
```

```
46    initial begin
47      wr_e=1'b1;
48      #40 wr_e=1'b0;
49      #20 wr_e=1'b1;
50      #80 wr_e=1'b0;
51      #20 wr_e=1'b1;
52    end
53
54    initial begin
55      rd_addr=2'h3;
56      #20 rd_addr=2'h0;
57      #20 rd_addr=2'h1;
58      #20 rd_addr=2'h2;
59      #20 rd_addr=2'h3;
60      #20 rd_addr=2'h0;
61      #20 rd_addr=2'h1;
62      #20 rd_addr=2'h2;
63      #20 rd_addr=2'h3;
64    end
65
66    initial begin
67      rd_e=1'b1;
68      #20 rd_e=1'b0;
69      #40 rd_e=1'b1;
70      #40 rd_e=1'b0;
71      #20 rd_e=1'b1;
72    end
73
74    endmodule
75
```