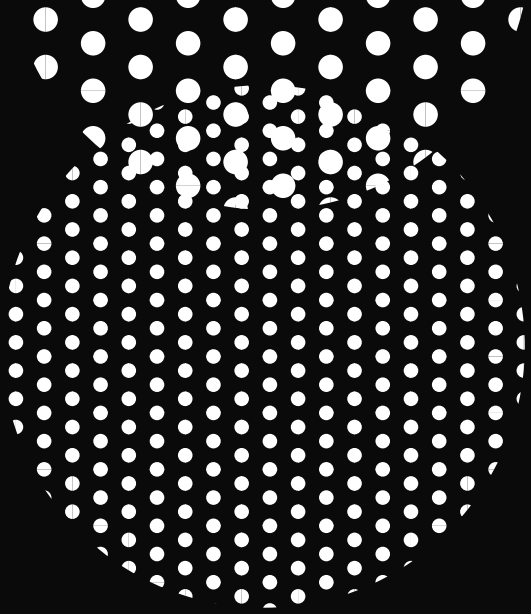


Lab5_top.v | slow_clkgen.v | scroll.v | convert.v | seg7_driver.v

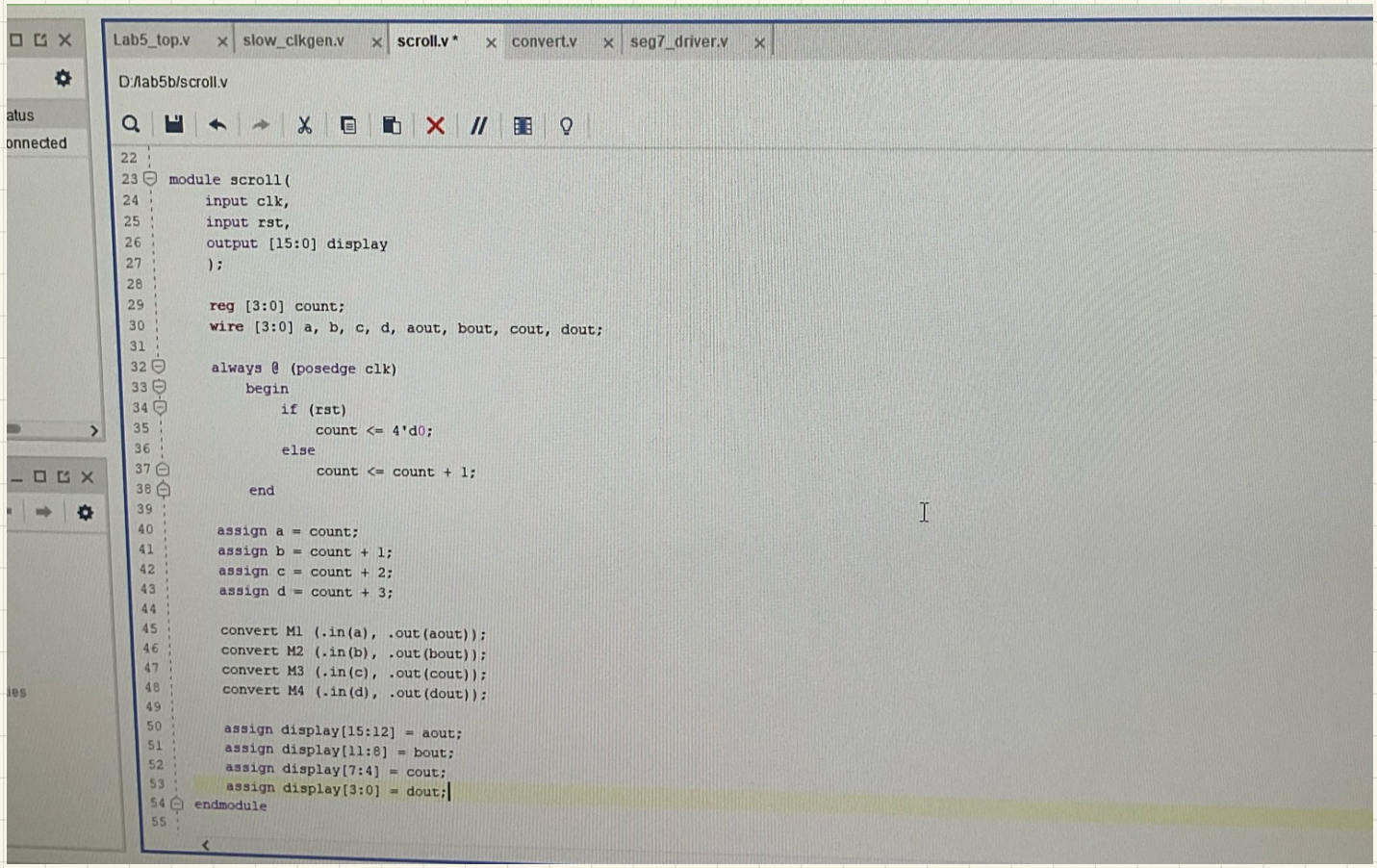
D:/lab5b/Lab5_top.v

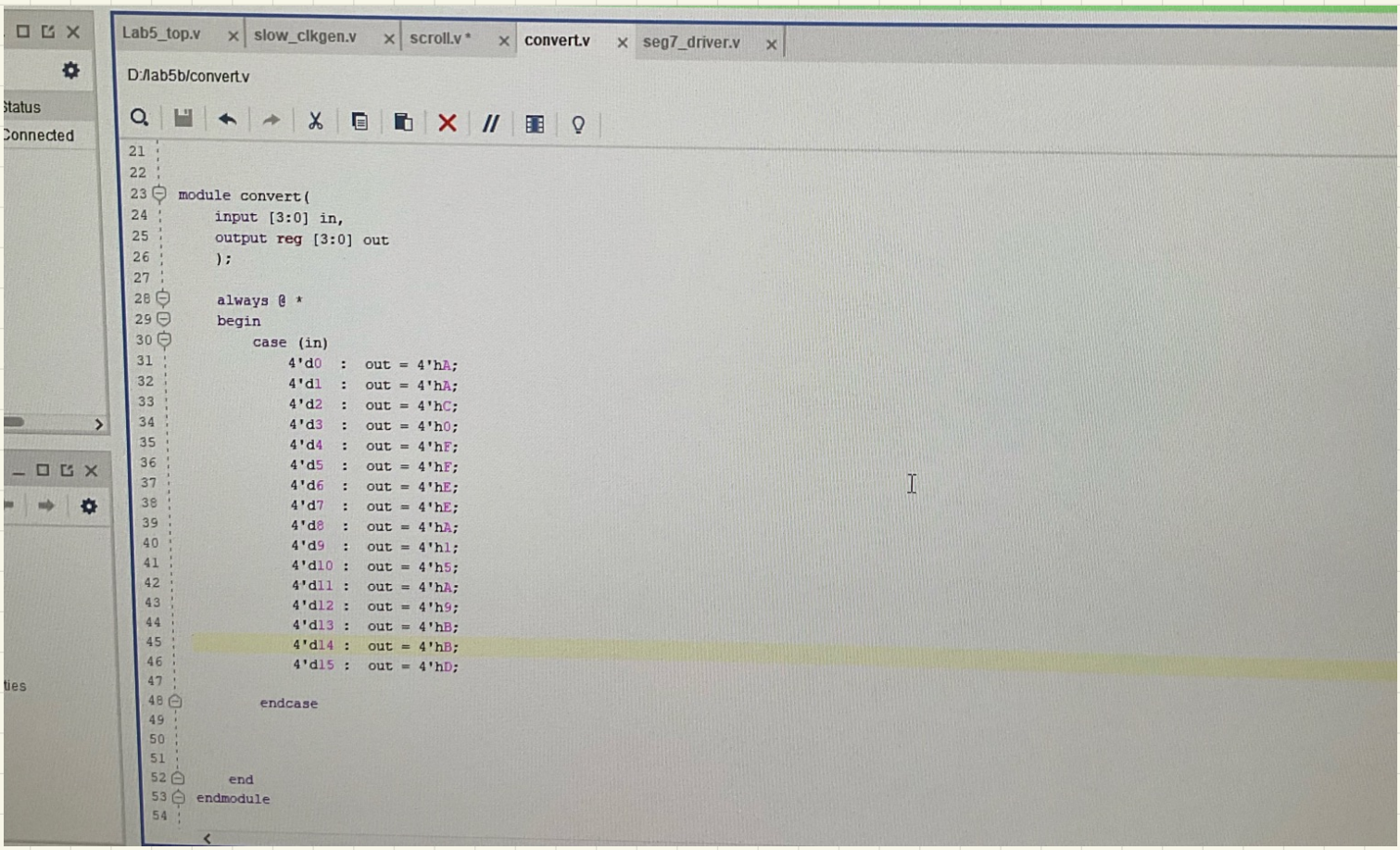
```
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//  Inputs:  100MHz system clock (clk); active high reset (rst)
//  Outputs: Active low 7 segment value (seg_L); Active low anode driver (anode_L)
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////

// The top level module. It implements the Verilog system shown in Figure 1.

module Lab5_top(input clk, rst, sel, output [6:0] seg_L, output [3:0] anode_L);

    // declare necessary wires here
    wire clk_out;
    wire [15:0] value;


    slow_clkgen M1 (.clk(clk), .rst(rst), .clk_out(clk_out));
    scroll M2 (.clk(clk_out), .rst(rst), .display(value));

    // instantiate modules here
    //scroll uut1
    //slow_clkgen uut2
    seg7_driver uut3 (.clk(clk), .rst(rst), .sel(sel), .value(value), .anode_d(4'b0000), .seg_L(seg_L), .anode_L(anode_L));

endmodule

```

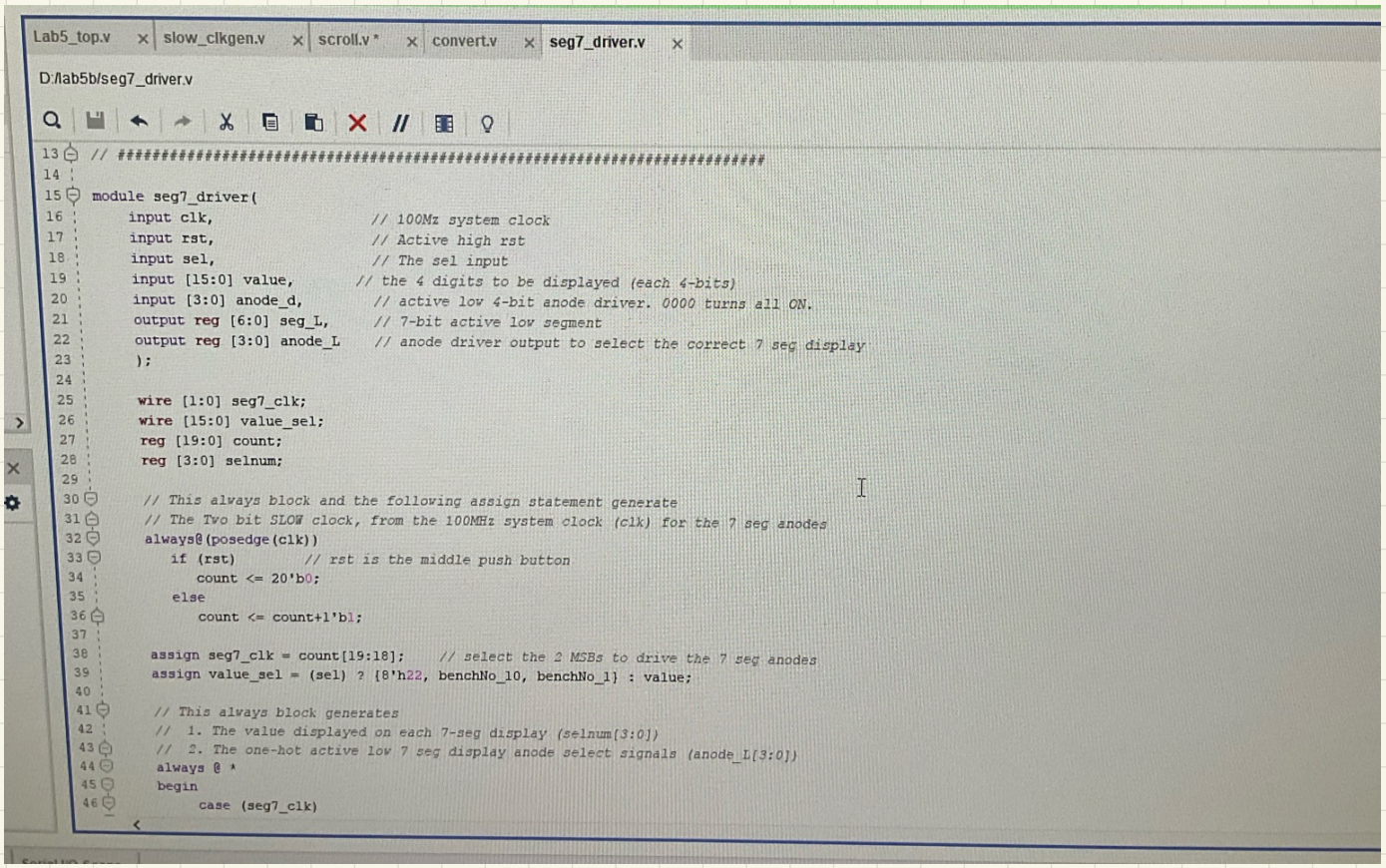
Lab5_top.v | slow_clkgen.v | scroll.v | convert.v | seg7_driver.v

D:/lab5b/slow_clkgen.v

```
// divides the 100Mz system clock (clk) by 2^25 to give around 1/3 s period

module slow_clkgen (input clk, rst, output clk_out);

    reg [24:0] counter;

    always @(posedge clk)
    begin
        if (rst)
            counter <= 25'd0;
        else
            counter <= counter + 1'b1;
    end


    assign clk_out = counter[24];

endmodule
```


Lab5_top.v
slow_clkgen.v
scroll.v *
convert.v
seg7_driver.v
D:/lab5b/scroll.v
22
23 module scroll(
24 input clk,
25 input rst,
26 output [15:0] display
27);
28
29 reg [3:0] count;
30 wire [3:0] a, b, c, d, aout, bout, cout, dout;
31
32 always @ (posedge clk)
33 begin
34 if (rst)
35 count <= 4'd0;
36 else
37 count <= count + 1;
38 end
39
40 assign a = count;
41 assign b = count + 1;
42 assign c = count + 2;
43 assign d = count + 3;
44
45 convert M1 (.in(a), .out(aout));
46 convert M2 (.in(b), .out(bout));
47 convert M3 (.in(c), .out(cout));
48 convert M4 (.in(d), .out(dout));
49
50 assign display[15:12] = aout;
51 assign display[11:8] = bout;
52 assign display[7:4] = cout;
53 assign display[3:0] = dout;
54 endmodule
55

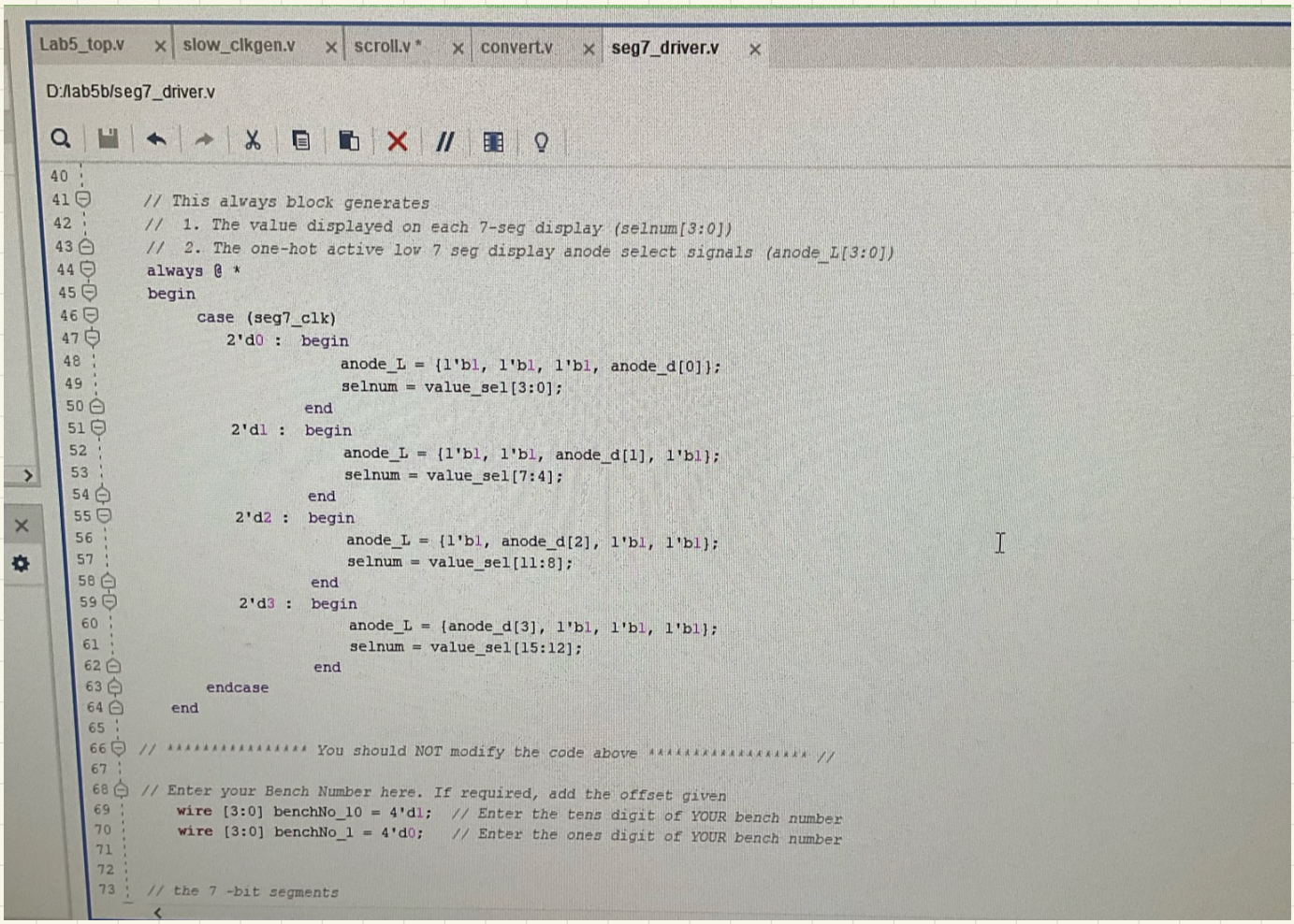

Lab5_top.v
slow_clkgen.v
scroll.v *
convert.v
seg7_driver.v
D:/lab5b/convert.v
21
22
23 module convert(
24 input [3:0] in,
25 output reg [3:0] out
26);
27
28 always @ *
29 begin
30 case (in)
31 4'd0 : out = 4'hA;
32 4'd1 : out = 4'hA;
33 4'd2 : out = 4'hC;
34 4'd3 : out = 4'h0;
35 4'd4 : out = 4'hF;
36 4'd5 : out = 4'hF;
37 4'd6 : out = 4'hE;
38 4'd7 : out = 4'hE;
39 4'd8 : out = 4'hA;
40 4'd9 : out = 4'h1;
41 4'd10 : out = 4'h5;
42 4'd11 : out = 4'hA;
43 4'd12 : out = 4'h9;
44 4'd13 : out = 4'hB;
45 4'd14 : out = 4'hB;
46 4'd15 : out = 4'hD;
47
48 endcase
49
50
51
52 end
53 endmodule
54



Lab5_top.v | slow_clkgen.v | scroll.v * | convert.v | seg7_driver.v

D:/lab5b/seg7_driver.v

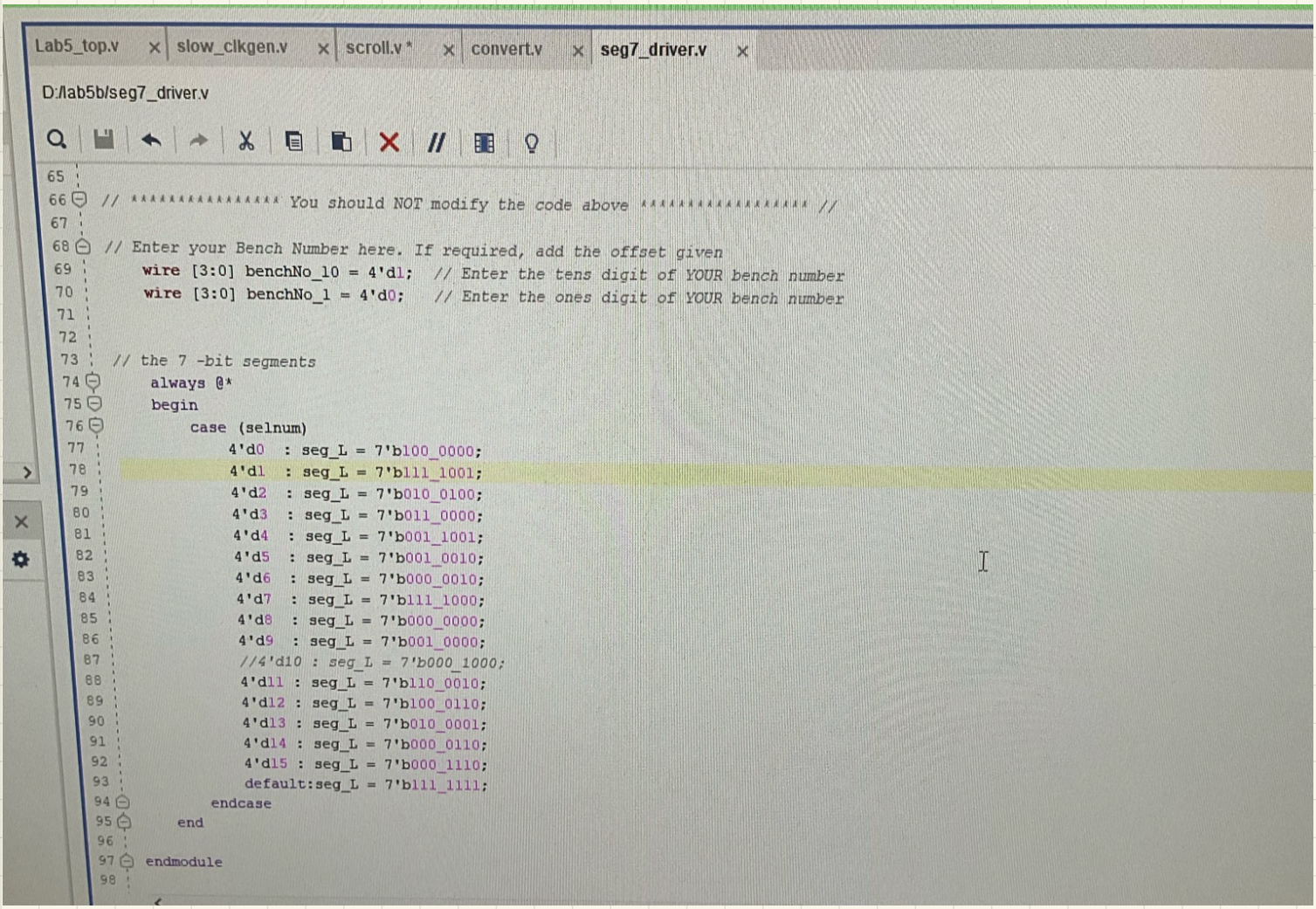
```
13  // ######################################################################
14
15  module seg7_driver(
16      input clk,                  // 100Mz system clock
17      input rst,                  // Active high rst
18      input sel,                  // The sel input
19      input [15:0] value,        // the 4 digits to be displayed (each 4-bits)
20      input [3:0] anode_d,        // active low 4-bit anode driver. 0000 turns all ON.
21      output reg [6:0] seg_L,     // 7-bit active low segment
22      output reg [3:0] anode_L    // anode driver output to select the correct 7 seg display
23      );
24
25      wire [1:0] seg7_clk;
26      wire [15:0] value_sel;
27      reg [19:0] count;
28      reg [3:0] selnum;
29
30      // This always block and the following assign statement generate
31      // The Two bit SLOW clock, from the 100MHz system clock (clk) for the 7 seg anodes
32      always@(posedge(clk))
33         if (rst)        // rst is the middle push button
34            count <= 20'b0;
35         else
36            count <= count+1'b1;
37
38      assign seg7_clk = count[19:18];     // select the 2 MSBs to drive the 7 seg anodes
39      assign value_sel = (sel) ? {8'h22, benchNo_10, benchNo_1} : value;
40
41      // This always block generates
42      //  1. The value displayed on each 7-seg display (selnum[3:0])
43      //  2. The one-hot active low 7 seg display anode select signals (anode_L[3:0])
44      always @ *
45      begin
46          case (seg7_clk)
```



Lab5_top.v | slow_clkgen.v | scroll.v * | convert.v | seg7_driver.v

D:/lab5b/seg7_driver.v

```
40
41      // This always block generates
42      //  1. The value displayed on each 7-seg display (selnum[3:0])
43      //  2. The one-hot active low 7 seg display anode select signals (anode_L[3:0])
44      always @ *
45      begin
46          case (seg7_clk)
47             2'd0 :  begin
48                         anode_L = {1'b1, 1'b1, 1'b1, anode_d[0]};
49                         selnum = value_sel[3:0];
50                     end
51              2'd1 :  begin
52                         anode_L = {1'b1, 1'b1, anode_d[1], 1'b1};
53                         selnum = value_sel[7:4];
54                     end
55              2'd2 :  begin
56                         anode_L = {1'b1, anode_d[2], 1'b1, 1'b1};
57                         selnum = value_sel[11:8];
58                     end
59              2'd3 :  begin
60                         anode_L = {anode_d[3], 1'b1, 1'b1, 1'b1};
61                         selnum = value_sel[15:12];
62                     end
63          endcase
64      end
65
66  // ***************** You should NOT modify the code above ****************** //
67
68  // Enter your Bench Number here. If required, add the offset given
69      wire [3:0] benchNo_10 = 4'd1;  // Enter the tens digit of YOUR bench number
70      wire [3:0] benchNo_1 = 4'd0;   // Enter the ones digit of YOUR bench number
71
72
73  // the 7 -bit segments
```


Lab5_top.v
slow_clkgen.v
scroll.v *
convert.v
seg7_driver.v
D:/lab5b/seg7_driver.v
// ********** You should NOT modify the code above ********** //
// Enter your Bench Number here. If required, add the offset given
wire [3:0] benchNo_10 = 4'd1; // Enter the tens digit of YOUR bench number
wire [3:0] benchNo_1 = 4'd0; // Enter the ones digit of YOUR bench number
// the 7 -bit segments
always @*
begin
case (selnum)
4'd0 : seg_L = 7'b100_0000;
4'd1 : seg_L = 7'b111_1001;
4'd2 : seg_L = 7'b010_0100;
4'd3 : seg_L = 7'b011_0000;
4'd4 : seg_L = 7'b001_1001;
4'd5 : seg_L = 7'b001_0010;
4'd6 : seg_L = 7'b000_0010;
4'd7 : seg_L = 7'b111_1000;
4'd8 : seg_L = 7'b000_0000;
4'd9 : seg_L = 7'b001_0000;
//4'd10 : seg_L = 7'b000_1000;
4'd11 : seg_L = 7'b110_0010;
4'd12 : seg_L = 7'b100_0110;
4'd13 : seg_L = 7'b010_0001;
4'd14 : seg_L = 7'b000_0110;
4'd15 : seg_L = 7'b000_1110;
default:seg_L = 7'b111_1111;
endcase
end
endmodule