ASSIGNMENT: 3

ES 204: Digital Systems

Prof. Joycee Mekie

Indian Institute of Technology, Gandhinagar

Aditya N. Mehta

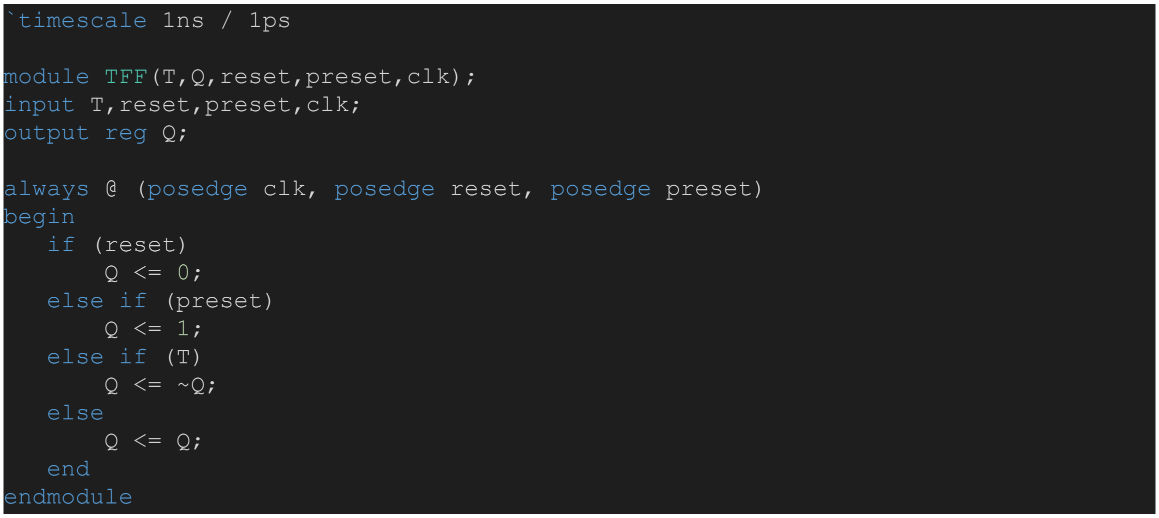
22110017

Hrriday V. Ruparel

22110099

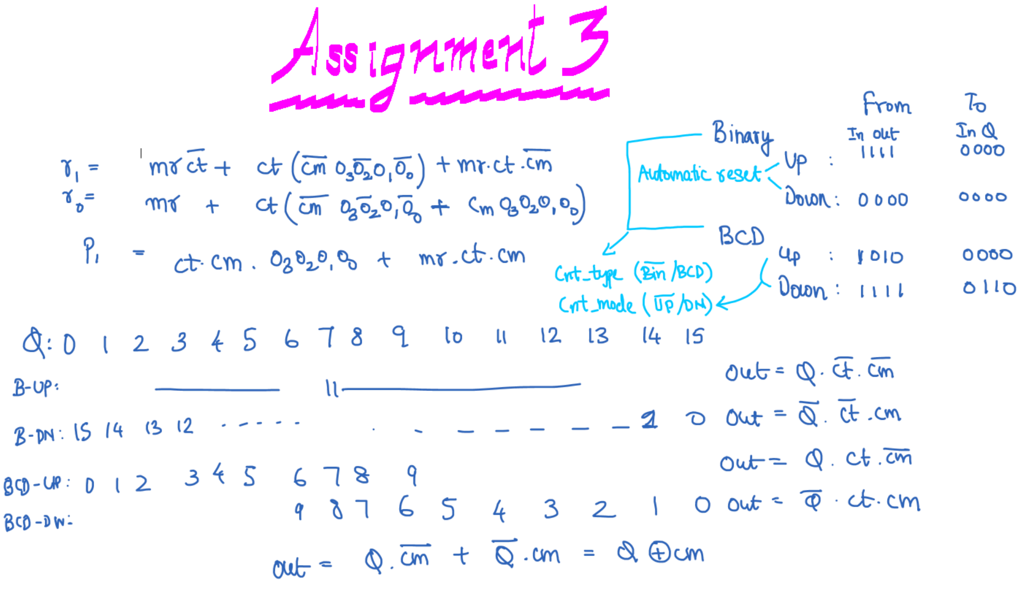
4-bit Synchronous Up/Down BCD/Binary counter with Univeral Shift Register

A) Toggle Flip Flop:



```verilog
`timescale 1ns / 1ps

module TFF(T,Q,reset,preset,clk);
input T,reset,preset,clk;
output reg Q;

always @ (posedge clk, posedge reset, posedge preset)
begin
   if (reset)
       Q <= 0;
   else if (preset)
       Q <= 1;
   else if (T)
       Q <= ~Q;
   else
       Q <= Q;
   end
endmodule
```

B) Structural Explanation of Counter Module :



Assignment 3

$r_1 = mr\overline{ct} + ct(\overline{cm}\,O_3\overline{O_2}O_1\overline{O_0}) + mr.ct.\overline{cm}$

$r_0 = mr + ct(\overline{cm}\,O_3\overline{O_2}O_1\overline{O_0} + cm\,O_3O_2O_1O_0)$

$P_1 = ct.cm.O_3O_2O_1O_0 + mr.ct.cm$

Automatic reset

		From In out	To In Q
Binary	Up :	1111	0000
	Down :	0000	0000
BCD	Up :	1010	0000
	Down :	1111	0110

Cnt_type ($\overline{Bin}$/BCD)

Cnt_mode ($\overline{Up}$/DN)

Q: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

B-UP: ———— 11 ————————— $out = Q.\overline{ct}.\overline{cm}$

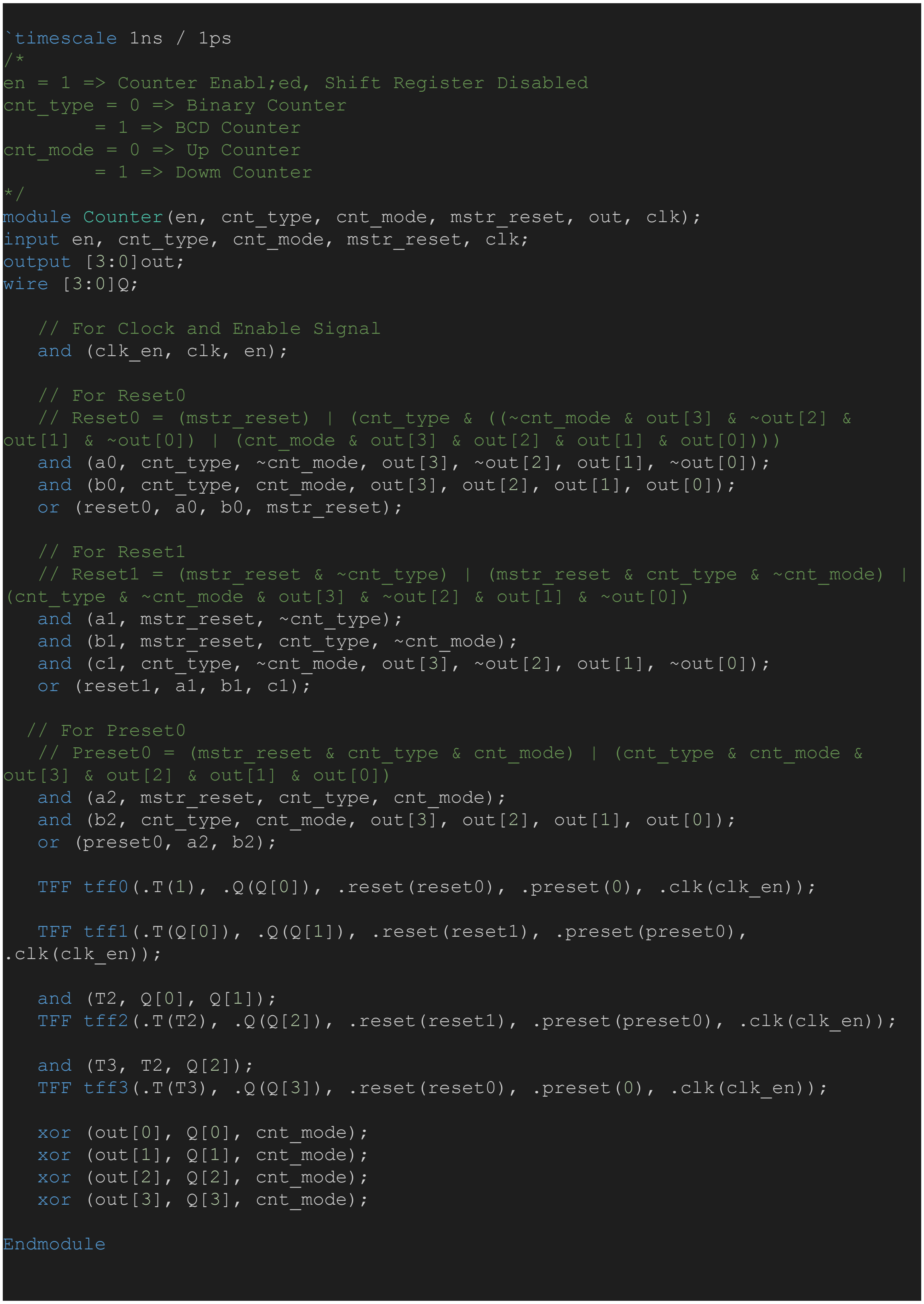
B-DN: 15 14 13 12 . - - - - . . - — — — — — 1 0 $out = \overline{Q}.\overline{ct}.cm$

BCD-UP: 0 1 2 3 4 5 6 7 8 9 $out = Q.ct.\overline{cm}$

BCD-DW: 9 8 7 6 5 4 3 2 1 0 $out = \overline{Q}.ct.cm$

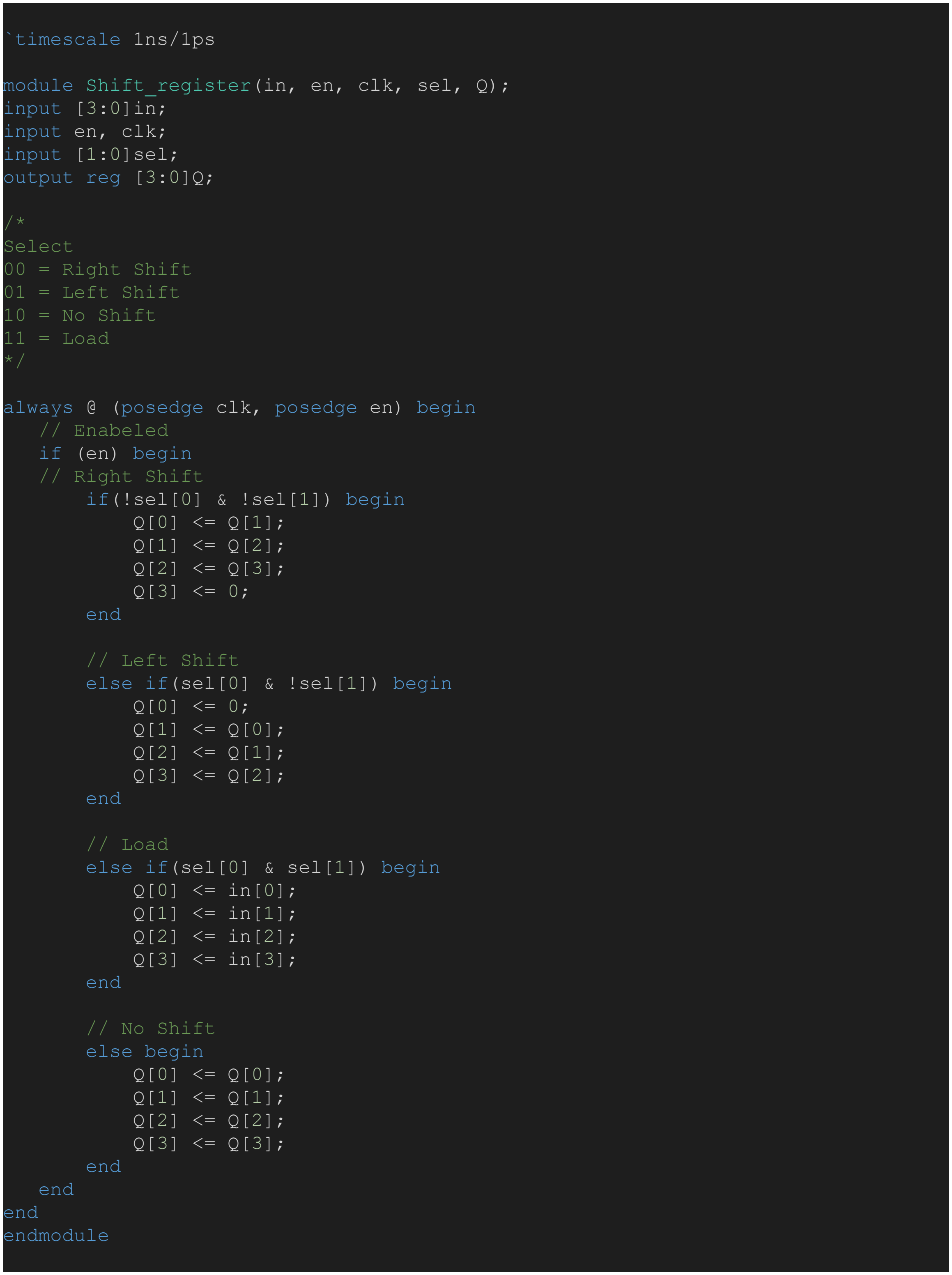
$out = Q.\overline{cm} + \overline{Q}.cm = Q \oplus cm$

C) Structural Verilog Code for Counter:



```
`timescale 1ns / 1ps
/*
en = 1 => Counter Enabl;ed, Shift Register Disabled
cnt_type = 0 => Binary Counter
        = 1 => BCD Counter
cnt_mode = 0 => Up Counter
        = 1 => Dowm Counter
*/
module Counter(en, cnt_type, cnt_mode, mstr_reset, out, clk);
input en, cnt_type, cnt_mode, mstr_reset, clk;
output [3:0]out;
wire [3:0]Q;

   // For Clock and Enable Signal
   and (clk_en, clk, en);

   // For Reset0
   // Reset0 = (mstr_reset) | (cnt_type & ((~cnt_mode & out[3] & ~out[2] &
out[1] & ~out[0]) | (cnt_mode & out[3] & out[2] & out[1] & out[0])))
   and (a0, cnt_type, ~cnt_mode, out[3], ~out[2], out[1], ~out[0]);
   and (b0, cnt_type, cnt_mode, out[3], out[2], out[1], out[0]);
   or (reset0, a0, b0, mstr_reset);

   // For Reset1
   // Reset1 = (mstr_reset & ~cnt_type) | (mstr_reset & cnt_type & ~cnt_mode) |
(cnt_type & ~cnt_mode & out[3] & ~out[2] & out[1] & ~out[0])
   and (a1, mstr_reset, ~cnt_type);
   and (b1, mstr_reset, cnt_type, ~cnt_mode);
   and (c1, cnt_type, ~cnt_mode, out[3], ~out[2], out[1], ~out[0]);
   or (reset1, a1, b1, c1);

  // For Preset0
   // Preset0 = (mstr_reset & cnt_type & cnt_mode) | (cnt_type & cnt_mode &
out[3] & out[2] & out[1] & out[0])
   and (a2, mstr_reset, cnt_type, cnt_mode);
   and (b2, cnt_type, cnt_mode, out[3], out[2], out[1], out[0]);
   or (preset0, a2, b2);

   TFF tff0(.T(1), .Q(Q[0]), .reset(reset0), .preset(0), .clk(clk_en));

   TFF tff1(.T(Q[0]), .Q(Q[1]), .reset(reset1), .preset(preset0),
.clk(clk_en));

   and (T2, Q[0], Q[1]);
   TFF tff2(.T(T2), .Q(Q[2]), .reset(reset1), .preset(preset0), .clk(clk_en));

   and (T3, T2, Q[2]);
   TFF tff3(.T(T3), .Q(Q[3]), .reset(reset0), .preset(0), .clk(clk_en));

   xor (out[0], Q[0], cnt_mode);
   xor (out[1], Q[1], cnt_mode);
   xor (out[2], Q[2], cnt_mode);
   xor (out[3], Q[3], cnt_mode);

Endmodule
```

D) Verilog Code for Universal Shift Register:



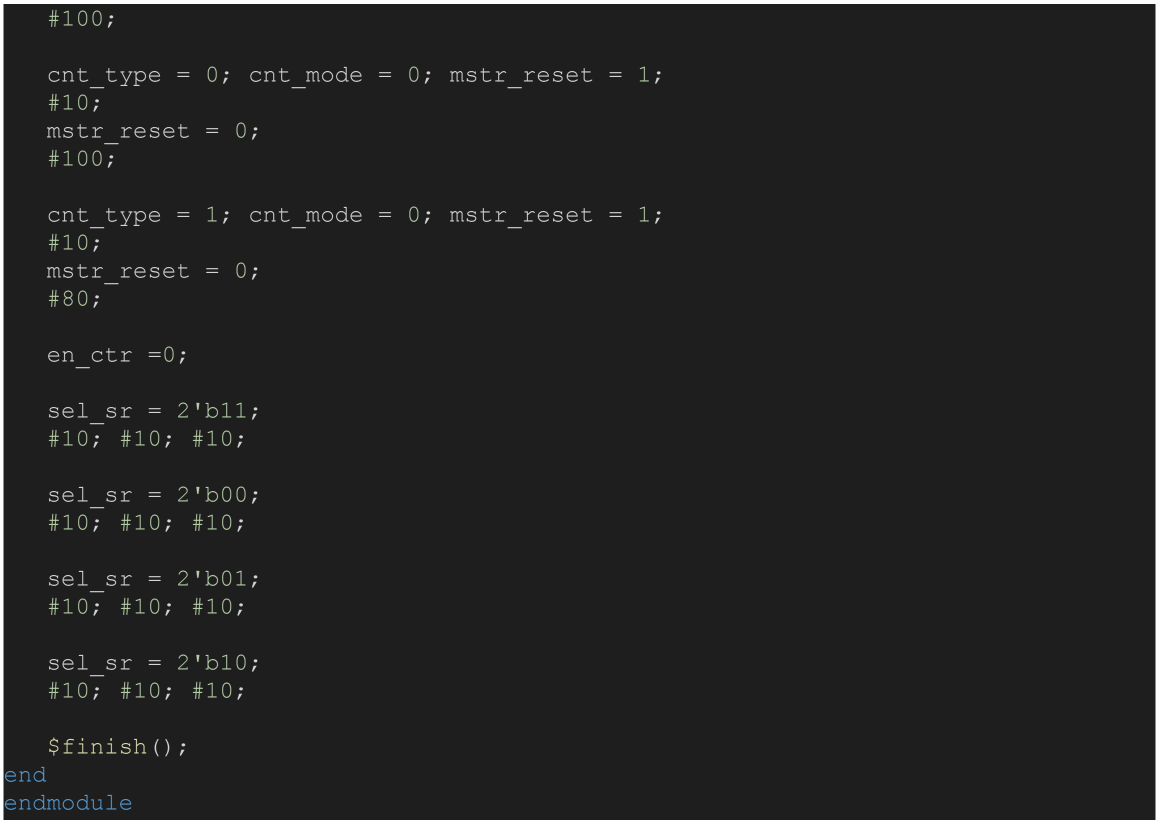
```
`timescale 1ns/1ps

module Shift_register(in, en, clk, sel, Q);
input [3:0]in;
input en, clk;
input [1:0]sel;
output reg [3:0]Q;

/*
Select
00 = Right Shift
01 = Left Shift
10 = No Shift
11 = Load
*/

always @ (posedge clk, posedge en) begin
   // Enabeled
   if (en) begin
   // Right Shift
       if(!sel[0] & !sel[1]) begin
           Q[0] <= Q[1];
           Q[1] <= Q[2];
           Q[2] <= Q[3];
           Q[3] <= 0;
       end

       // Left Shift
       else if(sel[0] & !sel[1]) begin
           Q[0] <= 0;
           Q[1] <= Q[0];
           Q[2] <= Q[1];
           Q[3] <= Q[2];
       end

       // Load
       else if(sel[0] & sel[1]) begin
           Q[0] <= in[0];
           Q[1] <= in[1];
           Q[2] <= in[2];
           Q[3] <= in[3];
       end

       // No Shift
       else begin
           Q[0] <= Q[0];
           Q[1] <= Q[1];
           Q[2] <= Q[2];
           Q[3] <= Q[3];
       end
   end
end
endmodule
```

E) Verilog Code for Final Module :

```
`timescale 1ns/1ps

module Lab_3(en_ctr, cnt_type, cnt_mode, mstr_reset, clk, sel_sr, out_sr,
out_ctr);

input en_ctr, cnt_type, cnt_mode, mstr_reset, clk;
input [1:0]sel_sr;
output [3:0]out_sr;
output [3:0]out_ctr;

Counter uut1(.en(en_ctr), .cnt_type(cnt_type), .cnt_mode(cnt_mode),
.mstr_reset(mstr_reset), .out(out_ctr), .clk(clk));
Shift_register uut2(.in(out_ctr), .en(~en_ctr), .clk(clk), .sel(sel_sr),
.Q(out_sr));

endmodule
```

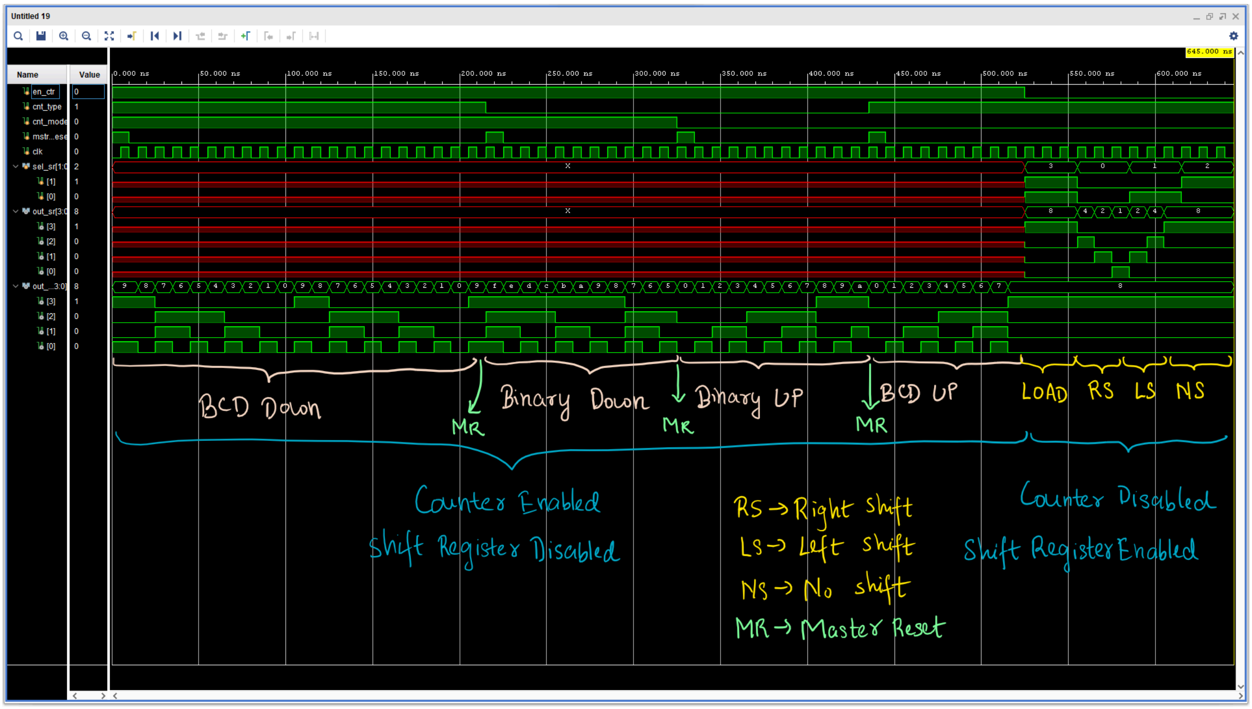
F) Verilog Code for Final Module Testbench:

```
`timescale 1ns/1ps

module Lab_3_tb ();
reg en_ctr, cnt_type, cnt_mode, mstr_reset;
reg clk;
reg [1:0] sel_sr;
wire [3:0]out_sr;
wire [3:0]out_ctr;

/*
Select
00 = Right Shift
01 = Left Shift
10 = No Shift
11 = Load
*/

Lab_4 dut (en_ctr, cnt_type, cnt_mode, mstr_reset, clk, sel_sr, out_sr,
out_ctr);

initial begin
   clk = 0;
   forever #5 clk = ~clk;
end

initial begin
//    $monitor($time, " en_ctr = %b, cnt_type = %b, cnt_mode = %b, mstr_reset =
%b, sel_str = %b, out = %b", en_ctr,cnt_type,cnt_mode,mstr_reset,sel_sr,out);
   en_ctr = 1; cnt_type = 1; cnt_mode = 1; mstr_reset = 1;
   #10;
   mstr_reset = 0;
   #205;

   cnt_type = 0; cnt_mode = 1; mstr_reset = 1;
   #10;
   mstr_reset = 0;
```



```
    #100;

    cnt_type = 0; cnt_mode = 0; mstr_reset = 1;
    #10;
    mstr_reset = 0;
    #100;

    cnt_type = 1; cnt_mode = 0; mstr_reset = 1;
    #10;
    mstr_reset = 0;
    #80;

    en_ctr =0;

    sel_sr = 2'b11;
    #10; #10; #10;

    sel_sr = 2'b00;
    #10; #10; #10;

    sel_sr = 2'b01;
    #10; #10; #10;

    sel_sr = 2'b10;
    #10; #10; #10;

    $finish();
end
endmodule
```

G) Final Stimulation:


Untitled 19
Name
Value
en_ctr 0
cnt_type 1
cnt_mode 0
mstr_reset 0
clk 0
sel_sr[1:0] 2
[1] 1
[0] 0
out_sr[3:0] 8
[3] 1
[2] 0
[1] 0
[0] 0
out_...3:0] 8
[3] 1
[2] 0
[1] 0
[0] 0
645.000 ns
0.000 ns
50.000 ns
100.000 ns
150.000 ns
200.000 ns
250.000 ns
300.000 ns
350.000 ns
400.000 ns
450.000 ns
500.000 ns
550.000 ns
600.000 ns
BCD Down
MR
Binary Down
MR
Binary UP
MR
BCD UP
LOAD
RS
LS
NS
Counter Enabled
Shift Register Disabled
RS → Right Shift
LS → Left Shift
NS → No shift
MR → Master Reset
Counter Disabled
Shift Register Enabled