P1: Universal Shift Register Code:

```
`timescale 1ns/10ps
module usr(Data_Out,MSB_out, LSB_out,Data_In,MSB_In, LSB_In,s1,s0,clk,rst);
output Data_Out;
output MSB_out, LSB_out;
input [3:0] Data_In;
input MSB_In, LSB_In;
input s1,s0,clk,rst;
reg [3:0] Data_Out;

        assign MSB_out = Data_Out[3];
        assign LSB_out = Data_Out[0];

        always @ (posedge clk)begin
         if(rst == 1'b1) Data_Out <= 0;
         else case({s1,s0})
          0:    Data_Out <= Data_Out;                          //hold
          1:    Data_Out <= {MSB_In, Data_Out[3:1]};    //Serial shift from MSB
          2:    Data_Out <= {Data_Out[2:0], LSB_In};    //Serial shift from LSB
          3:    Data_Out <= Data_In;                           //Parallel load
         endcase
        end
endmodule
```

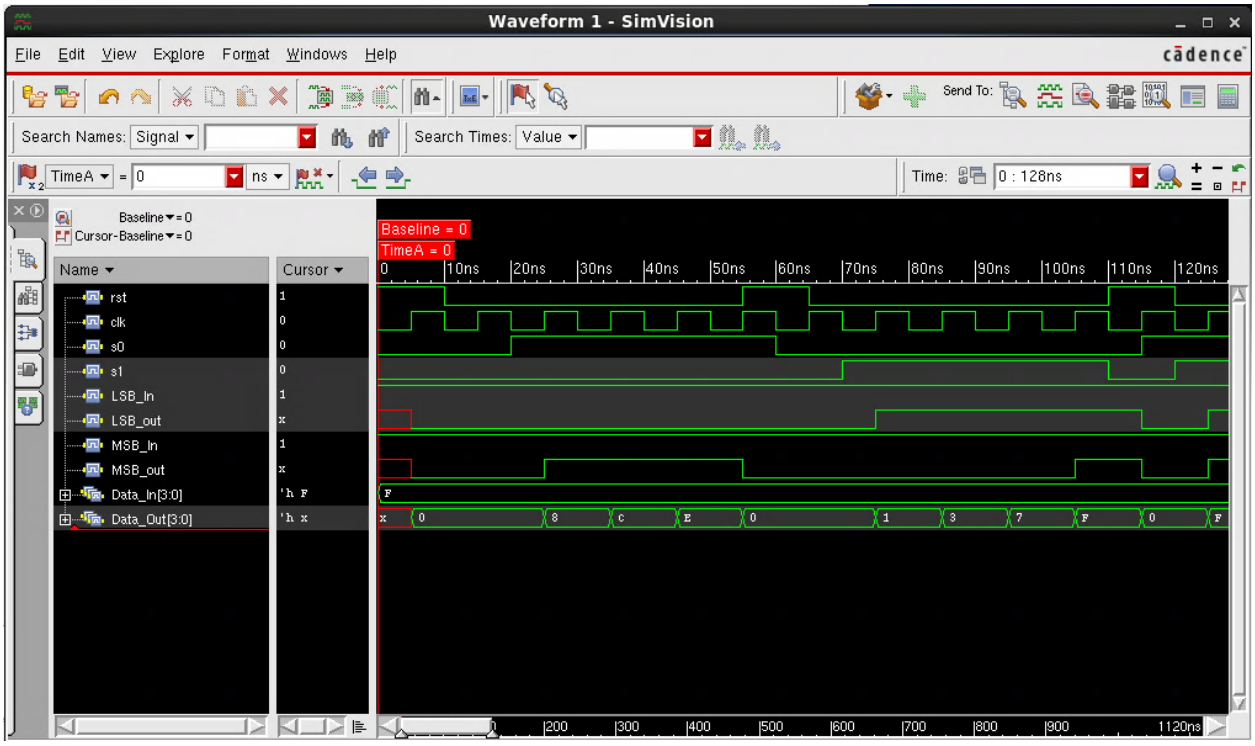
Testbench:

```
`timescale 1ns/10ps
module usr_t();
        wire [3:0] Data_out;
        wire MSB_out, LSB_out;
        reg  [3:0] Data_In;
        reg  MSB_In, LSB_In;
        reg  s1,s0,clk,rst;

        usr T1(.Data_Out(Data_Out),
                .MSB_out(MSB_out),
                .LSB_out(LSB_out),
                .Data_In(Data_In),
                .MSB_In(MSB_In),
                .LSB_In(LSB_In),
                .s1(s1),
                .s0(s0),
                .clk(clk),
                .rst(rst));

        initial
                clk=1'b0;
                always #5 clk=~clk;

        initial
                begin
                $display ("time\t clk\t rst\t s1\t s0\t out\t");
                $monitor ("%g\t %b\t %b\t %b\t %b\t %b\t", $time,clk,rst,s1,s0,Data_Out);
                #0      rst=1'b1;
                #10     rst=1'b0;
                #45     rst=1'b1;
                #10     rst=1'b0;
                #45     rst=1'b1;
                #10     rst=1'b0;
                #1000   $finish;
                end

        initial
                begin
                #0      s0=1'b0;
                #20     s0=1'b1;
                #40     s0=1'b0;
                #55     s0=1'b1;
                #95     s0=1'b0;
                end
        initial
                begin
                #0      s1=1'b0;
                #70     s1=1'b1;
                #40     s1=1'b0;
                #10     s1=1'b1;
                #90     s1=1'b0;
                end
        initial
                begin
                #0      LSB_In=1'b1;
                        MSB_In=1'b1;
                        Data_In=4'b1111;
                end

        initial
                begin
                        $shm_open ("mywave.db");
                        $shm_probe (usr_t,"AS");
                        $shm_save;
                end
endmodule
```

Output result:

time	clk	rst	s1	s0	out
0	0	1	0	0	xxxx
5	1	1	0	0	0000
10	0	0	0	0	0000
15	1	0	0	0	0000
20	0	0	0	1	0000
25	1	0	0	1	1000
30	0	0	0	1	1000
35	1	0	0	1	1100
40	0	0	0	1	1100
45	1	0	0	1	1110
50	0	0	0	1	1110
55	1	1	0	1	0000
60	0	1	0	0	0000
65	1	0	0	0	0000
70	0	0	1	0	0000
75	1	0	1	0	0001
80	0	0	1	0	0001
85	1	0	1	0	0011
90	0	0	1	0	0011
95	1	0	1	0	0111
100	0	0	1	0	0111
105	1	0	1	0	1111
110	0	1	0	0	1111

Waveform result:



Problem-2

8bit counter with negedge, active-low enable. Made by 2 4-bit counter, connected by RCO.

Verilog code

```
`timescale 1ns/10ps
module ebcounter(out,ebout,outh,rco,reset,enable,clk);
output out;
output outh;
output rco;
output ebout;
input clk,reset,enable;
reg [3:0] out;
reg [3:0] outh;
reg [1:0] rco;
wire [7:0] ebout;

assign ebout = {outh,out};

always @(negedge clk)
if (reset)
        out<=1'b0;
else
        if (enable==1'b0)
        out<=out+1;
always @(posedge clk)
if (out==4'b1111)
        rco<=1'b1;
else
        rco<=1'b0;
always @(negedge clk)
if (reset)
        outh<=1'b0;
else
        if(rco==1'b1)
        outh<=outh+1;
endmodule
```

Testbench

```
`timescale 1ns/10ps
module ebcounter_t();
        wire [3:0] out;
        wire [3:0] outh;
        wire [1:0] rco;
        wire [7:0] ebout;
        reg clk,reset,enable;

        ebcounter T1(.out(out),
                        .ebout(ebout),
                        .outh(outh),
                        .rco(rco),
                        .clk(clk),
                        .enable(enable),
                        .reset(reset));
        initial
                clk = 1'b1;
                always #5 clk=~clk;
        initial
                begin
                $display ("time\t clk\t reset\t enable\t 8counter\t");
                $monitor ("%g\t %b\t %b\t %b\t %b\t", $time,clk,reset,enable,ebout);
                #0 reset=1'b1;
                        enable=1'b1;
                #10 reset=1'b0;
                        enable=1'b0;
                #1800 $finish;
                end
        initial
                begin
                        $shm_open ("mywave.db");
                        $shm_probe (ebcounter_t,"AS");
                        $shm_save;
                end
endmodule
```

Output result

time	clk	reset	enable	8counter
0	1	1	1	xxxxxxxx
5	0	1	1	00000000
10	1	0	0	00000000
15	0	0	0	00000001
20	1	0	0	00000001
25	0	0	0	00000010
30	1	0	0	00000010
35	0	0	0	00000011
40	1	0	0	00000011
45	0	0	0	00000100
50	1	0	0	00000100
55	0	0	0	00000101
60	1	0	0	00000101
65	0	0	0	00000110
70	1	0	0	00000110
75	0	0	0	00000111
80	1	0	0	00000111
85	0	0	0	00001000
90	1	0	0	00001000
95	0	0	0	00001001
100	1	0	0	00001001
105	0	0	0	00001010
110	1	0	0	00001010
115	0	0	0	00001011
120	1	0	0	00001011
125	0	0	0	00001100
130	1	0	0	00001100
135	0	0	0	00001101
140	1	0	0	00001101
145	0	0	0	00001110
145	0	0	0	00001110
150	1	0	0	00001110
155	0	0	0	00001111
160	1	0	0	00001111
165	0	0	0	00010000
170	1	0	0	00010000
175	0	0	0	00010001
180	1	0	0	00010001
185	0	0	0	00010010
190	1	0	0	00010010
195	0	0	0	00010011
200	1	0	0	00010011
205	0	0	0	00010100
210	1	0	0	00010100
215	0	0	0	00010101
220	1	0	0	00010101
225	0	0	0	00010110
230	1	0	0	00010110
235	0	0	0	00010111
240	1	0	0	00010111
245	0	0	0	00011000
250	1	0	0	00011000
255	0	0	0	00011001
260	1	0	0	00011001
265	0	0	0	00011010
270	1	0	0	00011010
275	0	0	0	00011011
280	1	0	0	00011011
285	0	0	0	00011100
290	1	0	0	00011100
295	0	0	0	00011101

Waveform