BITS-Pilani Dubai Campus
I Sem 2021-22
Digital Design Laboratory / ECE/INSTR/CS F215
Submission Report
Experiment No.- 9 (Sequence Detector and Shift Register)

Name: V N Suchir Vangaveeti **ID Number: 2020A7PS0018U**

Hardware runs

Run 1: Parallel-in parallel-out

Truth Table

S0=1, S1=1,

Input	Clk	Output
A B C D		$Q_AQ_BQ_CQ_D$
0000	1	0000
1010	1	1010
1111	1	1111

Run 2: Serial-in Parallel-out (Right Shift)

First load the ‘0000’ to the O/P, with help of Parallel Load.

Truth Table

S0=1, S1=0,

Shift Right (I/P) Pin No.-2	No. of Pulses(n)	$Q_AQ_BQ_CQ_D$
1	1	1000
1	2	1100
1	3	1110
1	4	1111
0	5	0111
0	6	0011
0	7	0001
0	8	0000
1	9	1000

Experiment No.- 9 (Sequence Detector and Shift Register)

Name: V N Suchir Vangaveeti **ID Number: 2020A7PS0018U**

Run 3: Serial-in Parallel-out (Left Shift)

First load the '0000' to the O/P, with help of Parallel Load.

Truth Table

S0=0, S1=1,

Shift Left (I/P) Pin No.-7	**No. of Pulses(n)**	**$Q_AQ_BQ_CQ_D$**
1	1	0001
1	2	0011
1	3	0111
1	4	1111
0	5	1110
0	6	1100
0	7	1000
0	8	0000
1	9	0001

Software runs

Run 4:

Sequence detector for '110': Write the Verilog code and testbench for the sequence detector as Moore machine.

```
module detector_110 (input x, clk, rst, output reg z);
parameter [1:0] S0 = 2'b00, S1 = 2'b01, S2 = 2'b10, S3 = 2'b11;
reg [1:0] present_state;
always @ (posedge clk, posedge rst) begin
z <= (present_state == S3) ? 1 : 0;
if (rst) begin
z <= 1'b0;
present_state <= S0;
end
else begin
case (present_state)

S0: begin
if (x) present_state <= #1 S1;
else present_state <= #1 S0;
end

S1: begin
```

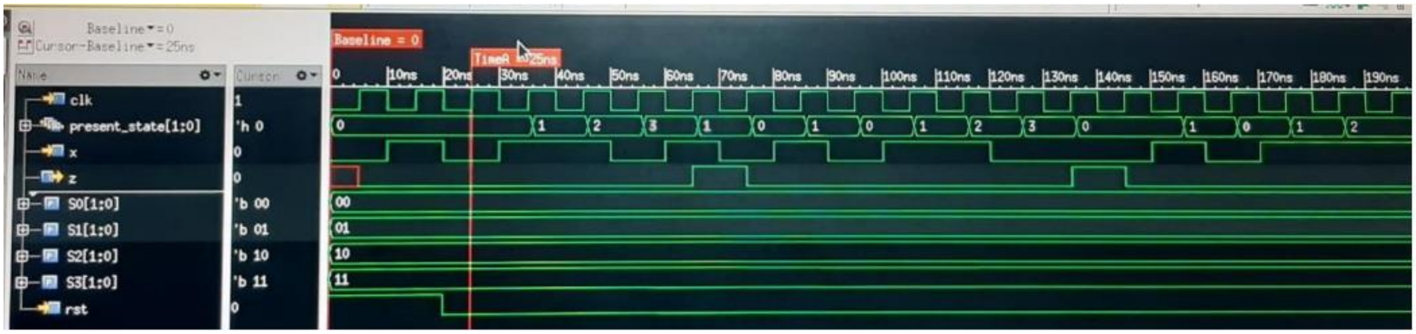
```
 if (x) present_state <= #1  S2;
else present_state <= #1 S0;
end

S2: begin
if (x) present_state <= #1 S2;
else present_state <= #1 S3;
end

S3: begin
if (x) present_state <= #1 S1;
else present_state  <= #1 S0;
end
endcase
end // end of else begin

z<= (present_state == S3) ? 1 : 0;
end // end of always begin
endmodule
//Testbench
module t_exp9a;
wire z;
reg x,clk, rst;
initial #200 $finish;
initial begin clk = 0; forever #5 clk = ~clk; end

initial begin
x = 0; rst = 1;
#10 x=1;
#10 x= 0; rst =0;
#10 x= 1;
#10 x= 1;
#10 x= 0;
#10 x= 1;
#10 x= 0;
#10 x= 1;
#10 x= 0;
#10 x= 1;
#10 x= 1;
#10 x= 0;
#10 x= 0;
#10 x= 0;
#10 x= 1;
#10 x= 0;
#10 x= 1;
end
detector_110  u(x,clk,rst,z);
endmodule
```

Experiment No.- 9 (Sequence Detector and Shift Register)

Name: V N Suchir Vangaveeti **ID Number: 2020A7PS0018U**

Q: Paste the Image of your **Simvision** window where you get the waveforms for the above code.

A:



Baseline = 0
Cursor-Baseline = 25ns

Name	Cursor
clk	1
present_state[1:0]	'h 0
x	0
z	0
S0[1:0]	'b 00
S1[1:0]	'b 01
S2[1:0]	'b 10
S3[1:0]	'b 11
rst	0

Q. Complete the state table below from the waveform observed.

State Table

Present state		Next state	Output
state [1:0]	**x_in**	**next_state [1:0]**	**y_out**
00	0	00	0
00	1	01	0
01	0	00	0
01	1	10	0
10	0	11	0
10	1	10	0
11	0	00	1
11	1	01	1

Run 5:

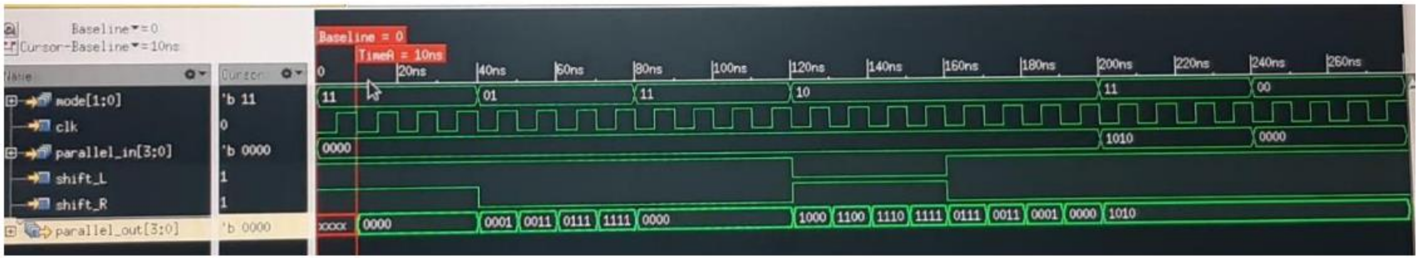
4-bit universal shift register: Write Verilog code and testbench for 4-bit universal shift register.

```
module uni_shift_reg (input shift_L, shift_R, clk, input [3:0] parallel_in, input [1:0] mode, output reg [3:0]
parallel_out);
        always @ (negedge clk) begin
        case (mode)
        0: parallel_out <= parallel_out; // for mode 0 data retain as it is
        1: parallel_out <= {parallel_out [2:0], shift_L}; // for mode 1 shift left
        2: parallel_out <= {shift_R, parallel_out [3:1] }; // for mode 2 shift right
        3: parallel_out <= parallel_in; // for mode 3 parallel data in
        default: parallel_out <= 4'bx;
        endcase
        end
endmodule

// Test Bench
module tb;
reg shift_L, shift_R, clk;
reg [1:0] mode;
reg [3:0] parallel_in;
wire [3:0] parallel_out;

        initial
        begin
        clk = 1'b0;
        repeat (300) # 5 clk = ~clk; // clock is of time period 10 and 50% duty cycle.
        end
        initial
 begin
        #00  mode = 2'b11; parallel_in=4'b0000; shift_L=1; shift_R=1;
        #40  mode =2'b01;  shift_L=1;shift_R=0;parallel_in =4'b0000;
        #40  mode =2'b11; parallel_in=4'b0000;
        #40  mode =2'b10;  shift_R=1;shift_L=0;
        #40  mode =2'b10;  shift_R=0;shift_L=1;
        #40  mode =2'b11; parallel_in =4'b1010;
        #40  mode =2'b00; parallel_in =4'b0000;
        #40 $stop;

        end
uni_shift_reg u(shift_L, shift_R, clk, parallel_in,mode, parallel_out);
endmodule
```

Experiment No.- 9 (Sequence Detector and Shift Register)

Name: V N Suchir Vangaveeti **ID Number: 2020A7PS0018U**

Q: Paste the Image of your **Simvision** window where you get the waveforms for the above code.
A:



Baseline = 0
Cursor-Baseline = 10ns

Name	Cursor
mode[1:0]	'b 11
clk	0
parallel_in[3:0]	'b 0000
shift_L	1
shift_R	1
parallel_out[3:0]	'b 0000

Q. Complete the truth table below from the waveform observed

Truth Table

Mode	No. of Clk Pulses (n)	Parallel I/P	Shift L (I/P)	Shift R (I/P)	Parallel O/P
11	1	0000	1	1	0000
11	2	0000	1	1	0000
11	3	0000	1	1	0000
11	4	0000	1	1	0000
01	5	0000	1	0	0001
01	6	0000	1	0	0011
01	7	0000	1	0	0111
01	8	0000	1	0	1111
11	9	0000	1	0	0000
11	10	0000	1	0	0000
11	11	0000	1	0	0000
11	12	0000	1	0	0000
10	13	0000	0	1	1000
10	14	0000	0	1	1100
10	15	0000	0	1	1110
10	16	0000	0	1	1111
10	17	0000	1	0	0111
10	18	0000	1	0	0011
10	19	0000	1	0	0001
10	20	0000	1	0	0000
11	21	1010	1	0	1010
11	22	1010	1	0	1010
11	23	1010	1	0	1010
11	24	1010	1	0	1010
00	25	0000	1	0	1010
00	26	0000	1	0	1010
00	27	0000	1	0	1010

Experiment No.- 9 (Sequence Detector and Shift Register)

Name: V N Suchir Vangaveeti **ID Number: 2020A7PS0018U**

Assignment All assignments are to be submitted strictly before start of next lab session through online only. Late assignments will not be entertained and will be awarded '0' marks.

1. Write the Verilog code and testbench for serial in serial out shift register.

Ans: Link1: https://www.edaplayground.com/x/eFGM

2. Write the Verilog code and testbench for serial in parallel out shift register.

Ans: Link2: https://www.edaplayground.com/x/r3r5

Self-Practice and self-evaluation (Very Important)

4-bit shift register: Below is the code for shift register using blocking statements. Write the testbench for the same and identify the issues when using blocking statements.

```
module shift_reg (output reg A, input E, clk, rst);
reg B, C, D;
always @ (posedge clk, posedge rst) begin
        if (rst == 1'b1) begin
         A=0;
         B = 0;
         C = 0;
         D = 0;
        end
        else begin
        A = B;
        B = C;
        C = D;
        D = E;
        end
end
endmodule
```