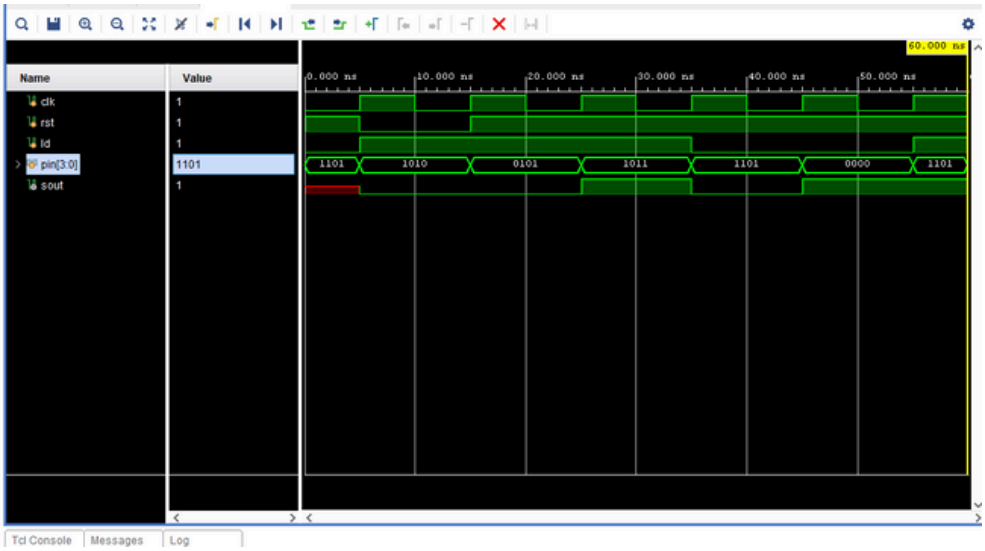
46.Parllel In Serial Out Shift Register(PISO)

Verilog Code:

```
module piso(clk,rst,ld,pin,sout );
input clk,rst,ld;
input[3:0] pin;
output sout;
reg[3:0]q;
always @(posedge clk)
begin
if(!rst)
q<=4'b0000;
else if(ld)
q<=pin;
else
q<=(q<<1);
end
assign sout=q[3];
endmodule
```

TestBench:

```
module tb6();
reg clk,rst,ld;
reg [3:0]pin;
wire sout;
piso uut(clk,rst,ld,pin,sout);
initial
begin
clk=0;
forever #5 clk=~clk;
end
initial
begin
rst=1;ld=0; pin=4'b1101; #5
rst=0;ld=1; pin=4'b1010; #10
rst=1; pin=4'b0101; #10
pin=4'b1011; #10
pin=4'b1101;ld=0; #10
pin=4'b0000; #10
pin=4'b1101;ld=1; #5
$finish;
end
endmodule
```



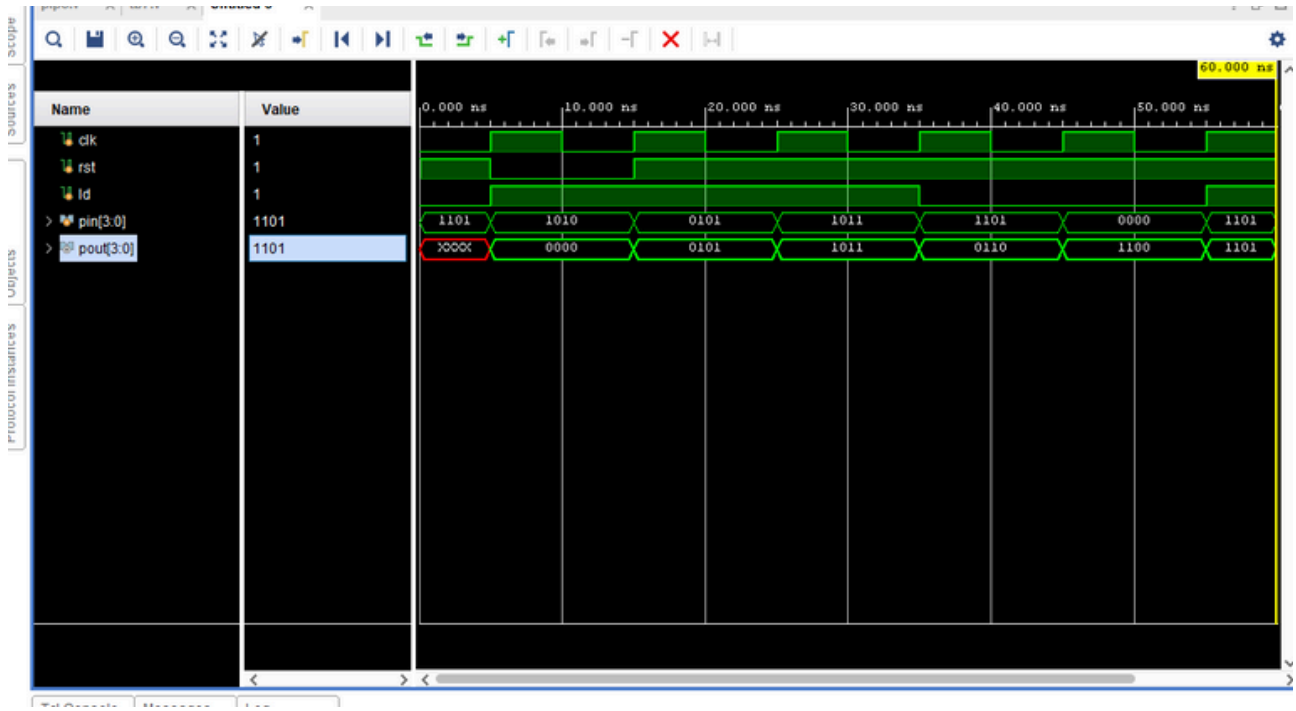
47.Parllel In Parllel Out Shift Register(PIPO)

Verilog Code:

```
module pipo(clk,rst,ld,pin,pout );
input clk,rst,ld;
input[3:0] pin;
output reg[3:0] pout;
always @(posedge clk)
begin
if(!rst)
pout<=4'b0000;
else if(ld)
pout<=pin;
else
pout<=(pout<<1);
end
endmodule
```

TestBench:

```
module tb7();
reg clk,rst,ld;
reg [3:0]pin;
wire [3:0]pout;
pipo uut(clk,rst,ld,pin,pout);
initial
begin
clk=0;
forever #5 clk=~clk;
end
initial
begin
rst=1;ld=0; pin=4'b1101; #5
rst=0;ld=1; pin=4'b1010; #10
rst=1; pin=4'b0101; #10
pin=4'b1011; #10
pin=4'b1101;ld=0; #10
pin=4'b0000; #10
pin=4'b1101;ld=1; #5
$finish;
end
endmodule
```



48.Master-Slave D Flipflop

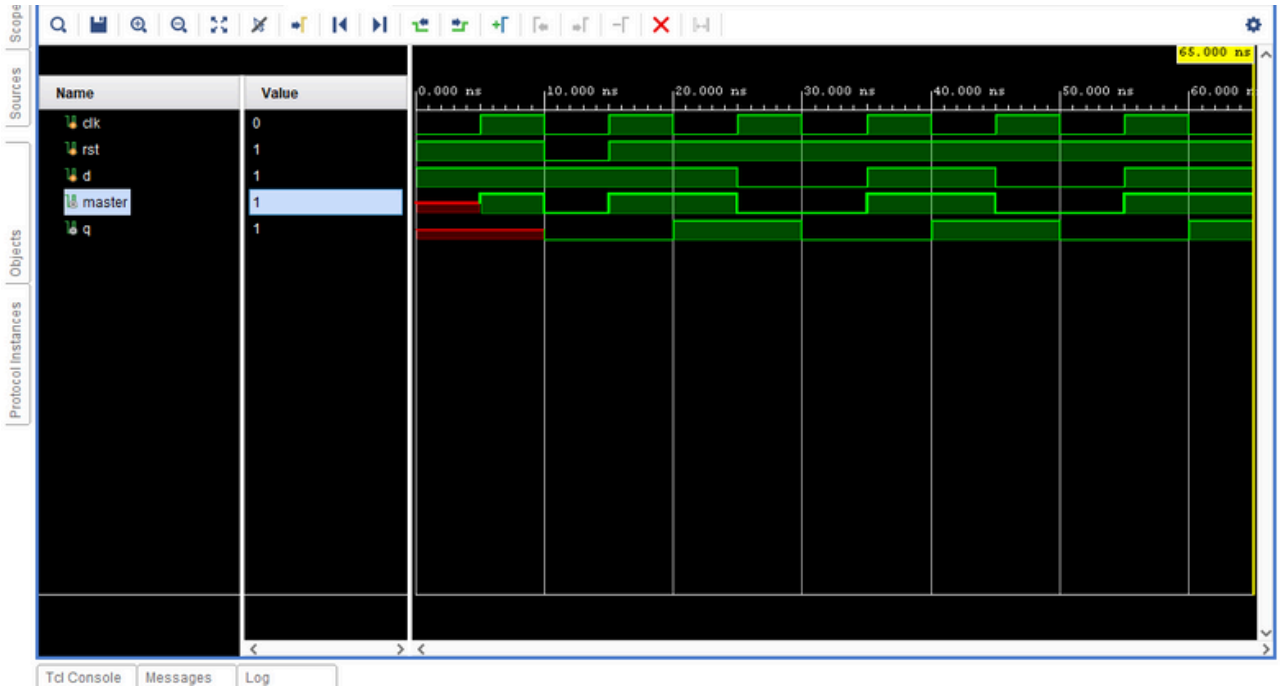
Verilog Code:

```
module master_slave_ff(clk,rst,d,master,q);
input clk,d,rst;
output reg q;
output reg master;
always@(posedge clk,negedge rst)
begin
if(!rst)
master<=0;
else
master<=d;
end
always@(negedge clk,negedge rst)
begin
if(!rst)
q<=0;
else
q<=master;
end
endmodule
```

TestBench:

```
module tb9();
reg clk,rst,d;
wire master,q;
```

```
master_slave_ff uut(clk,rst,d,master,q);
 initial begin
 clk=0;
 forever#5 clk=~clk;
end
initial begin d=1;rst=1; #10
d=1;rst=0; #5
d=1;rst=1; #10
d=0;rst=1; #10
d=1;rst=1; #10
d=0;rst=1; #10
d=1;rst=1; #10
$finish;
end
endmodule
```



49.Sequence Detector using Mealy FSM

Verilog Code:

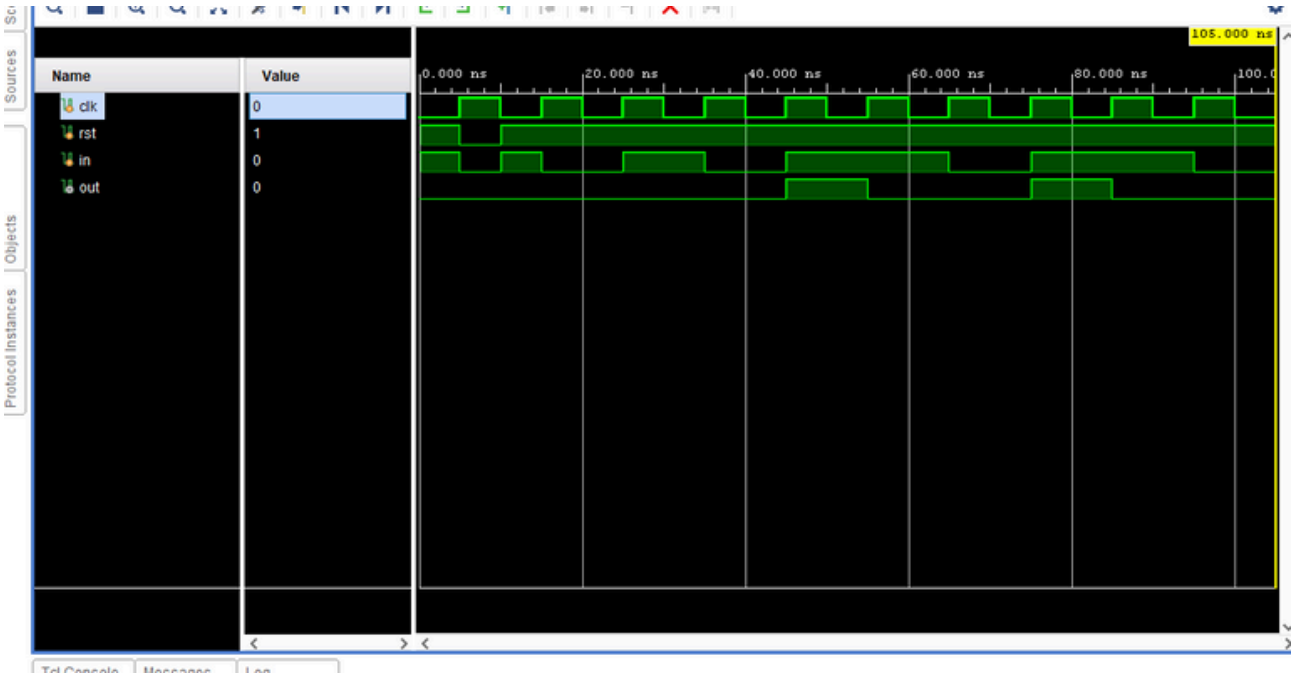
```
module seq_det_mealy ( clk,rst, in,out);
input clk,rst, in;
output reg out;
reg [1:0] state, next_st;
parameter s0 = 2'b00, s1 = 2'b01,s2 = 2'b10,s3 = 2'b11;
always @(posedge clk or negedge rst)
```

```
begin
if (!rst)
state <= s0;
else
state <= next_st;
end
always @(*) begin
case (state)
s0: begin
next_st = in ? s1 : s0;
out = 0;
end
s1: begin
next_st = in ? s1 : s2;
out = 0;
end
s2: begin
next_st = in ? s3 : s0;
out = 0;
end
s3: begin
next_st = in ? s1 : s2;
out = in ? 1 : 0;
end
default: begin
next_st = s0;
out = 0;
end
endcase
end
endmodule
```

TestBench:

```
module tb10();
reg clk,rst, in;
wire out;
seq_det_mealy uut(clk,rst,in,out);
initial
begin
clk=0;
forever#5 clk=~clk;
end
initial
begin
in=1; rst=1; #5
```

```
in=0;rst=0;#5
in=1; rst=1; #5
in=0;       #10
in=1;       #10
in=0;       #10
in=1;       #10
in=1;       #10
in=0;       #10
in=1;       #10
in=1;       #10
in=0;       #10
$finish; end
endmodule
```



50.Sequence Detector using Moore FSM

Verilog Code:

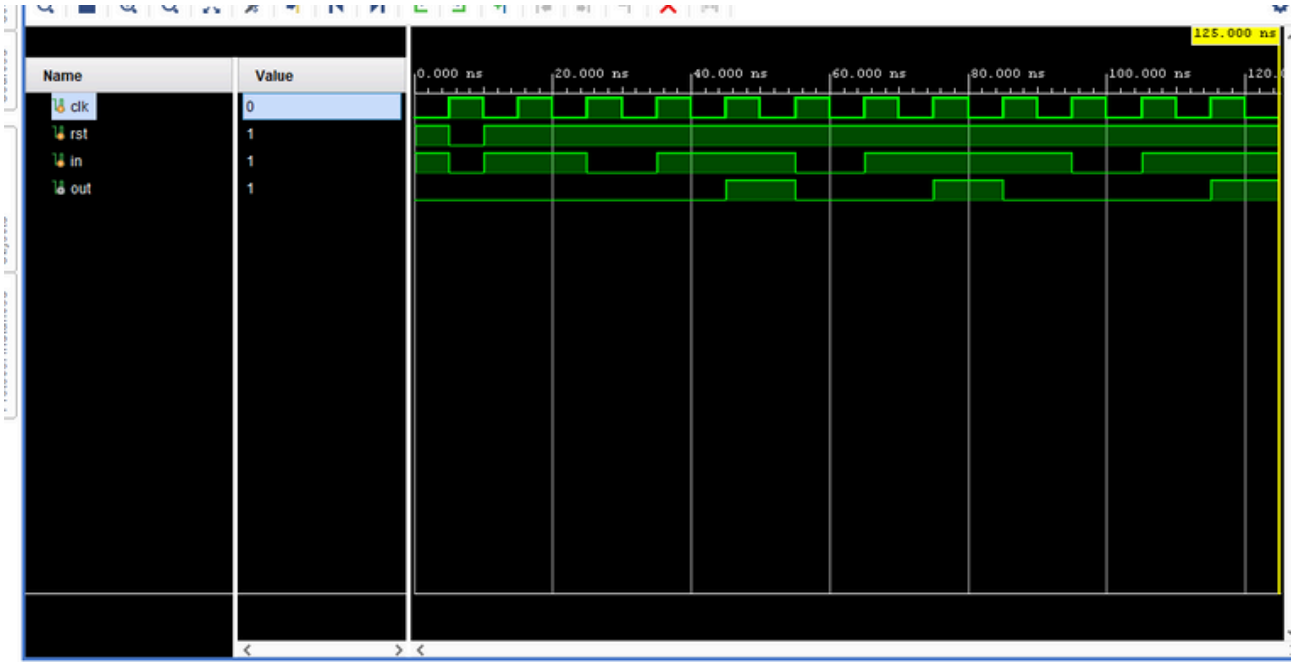
```
module seq_det_moore(clk,rst,in,out);
input clk,rst,in;
output reg out;
reg [2:0] state, next_st;
parameter s0 = 3'b000, s1 = 3'b001,s2 = 3'b010,s3 = 3'b011,s4=3'b100;
always @(posedge clk or negedge rst)
begin
if (!rst)
state <= s0;
else
```

```
state <= next_st;
end
always @(*) begin
case (state) s0:
begin next_st = in ? s1 : s0;
out = 0;
end s1: begin next_st = in ? s1 : s2;
out = 0;
 end s2: begin
next_st = in ? s3 : s0;
out = 0;
end
s3: begin
next_st = in ? s4 : s2;
out = 0;
end
s4: begin
next_st = in ? s1 : s2;
out = 1;
end
default: begin
next_st = s0;
out = 0;
end
endcase
end
endmodule
```

TestBench:

```
module tb11();
reg clk,rst, in;
wire out;
seq_det_moore uut(clk,rst,in,out);
initial
begin
clk=0;
forever#5 clk=~clk;
end
initial
begin
in=1; rst=1; #5
```

in=0;rst=0;#5
in=1; rst=1; #15
in=0; #10 in=1;
#10 in=1; #10
in=0; #10 in=1;
#10 in=1; #20
in=0; #10 in=1;
#10 in=1; #10
$finish; end
endmodule



Thankyou