College of Engineering, Pune

Dept. of Electronics and Telecommunication Engineering

Course: RTL Simulation and Synthesis Lab (FY VLSI & ES)

Year : 2020-21

RTL Lab 4,5 Report:

Mis: 122035014

Name: Swapnil Hanmant Phalke



1011 sequence detector using moore machine non-overlapping

```
Code:-
                                                    next_state = S4;
module Moore_sequence(X,clk,reset,Y);
                                                   end
input X,clk,reset;
                                                   S4:begin
output reg Y;
                                                   if(X==1'b0)
parameter S0=3'b000,
                                                         next_state = S0;
 S1=3'b001,
                                                   else
 S2=3'b011,
                                                    next_state = S1;
 S3=3'b010,
 S4=3'b110;
                                                   end
reg [2:0] current_state, next_state;
                                                   default:next_state = S0;
always @(negedge reset,posedge clk)
                                                   endcase
begin
                                                  end
if(!reset)
                                                  always @(current_state)
current_state <= S0;</pre>
                                                  begin
else
                                                   case(current_state)
current_state <= next_state;</pre>
                                                     S0:Y=0;
end
                                                          S1:Y=0;
always @(current_state,X)
                                                          S2:Y=0;
begin
                                                          S3:Y=0;
case(current_state)
                                                          S4:Y=1;
S0:begin
                                                          default Y=0;
if(X==1'b1)
                                                   endcase
       next_state = S1;
                                                  end
 else
                                                  endmodule
 next_state = S0;
end
                                                  testbench:
S1:begin
if(X==1'b0)
                                                  module qw;
       next_state = S2;
 else
                                                         // Inputs
 next_state = S1;
                                                         reg x;
end
                                                         reg clk;
S2:begin
                                                         reg rst;
if(X==1'b0)
       next_state = S0;
                                                         // Outputs
 else
```

 $next_state = S3;$

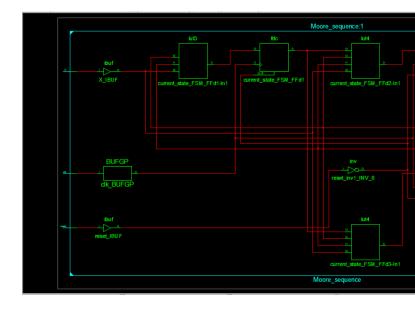
 $next_state = S2;$

end

else

S3:begin if(X==1'b0)

```
wire y;
       // Instantiate the Unit Under Test
(UUT)
       mealy uut (
              .x(x),
              .clk(clk),
              .rst(rst),
              .y(y)
       );
       initial begin
       clk = 1;
forever #60 clk=~clk;
end
              initial begin// Initialiye
Inputs
              x = 0;
              rst = 0;
              // Wait 100 ns for global rst
to finish
              #100;
              // Add stimulus here
rst=1;
x = 1;
#100;
x=0;
#100;
x = 1;
#100;
x = 1;
#100;
              // Add stimulus here
       end
endmodule
```

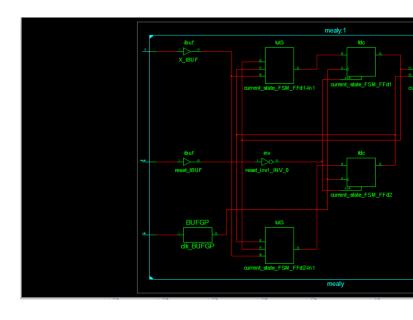


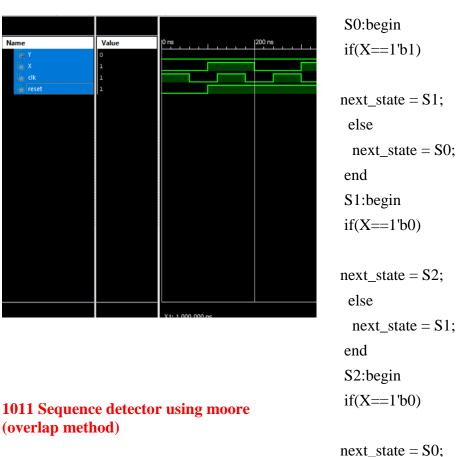


1011 Sequence detector using mealy machine

```
Code:-
module mealy(X,clk,reset,Y);
input X,clk,reset;
output reg Y;
parameter S0=2'b00,
 S1=2'b01,
 S2=2'b10,
 S3=2'b11;
reg [1:0] current_state, next_state;
always @(negedge reset,posedge clk)
begin
if(!reset)
current_state <= S0;
else
current_state <= next_state;</pre>
end
always @(current_state,X)
begin
case(current_state)
S0:begin
if(X==1'b1)
next_state = S1;
 else
 next_state = S0;
Y=0;
end
S1:begin
if(X==1'b0)
 next_state = S2;
 else
 next_state = S1;
Y=0;
end
S2:begin
```

```
if(X==1'b0)
next_state = S0;
 else
 next_state = S3;
Y=0;
end
S3:begin
if(X==1'b0)
next_state = S2;
 else
 next state = S0;
Y=1;
end
default:next\_state = S0;
endcase
end
endmodule
```





```
else
Code-
                                                       next_state = S3;
module Moore_sequence(X,clk,reset,Y);
                                                      end
input X,clk,reset;
                                                      S3:begin
output reg Y;
                                                      if(X==1'b0)
parameter S0=3'b000,
 S1=3'b001,
                                                     next_state = S2;
 S2=3'b011,
                                                      else
 S3=3'b010,
 S4=3'b110;
                                                       next_state = S4;
                                                      end
reg [2:0] current_state, next_state;
                                                      S4:begin
always @(negedge reset,posedge clk)
                                                      if(X==1'b0)
begin
if(!reset)
                                                     next_state = S2;
current_state <= S0;</pre>
                                                      else
else
                                                       next_state = S1;
current_state <= next_state;</pre>
end
```

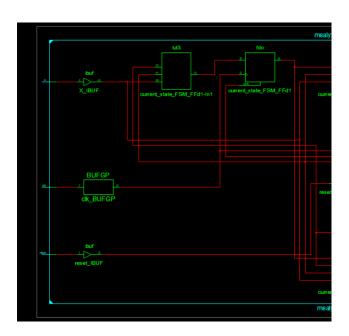
end

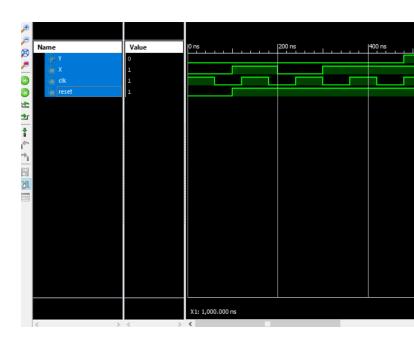
always @(current_state,X)

case(current_state)

begin

```
default:next_state = S0;
endcase
end
always @(current_state)
begin
case(current_state)
S0:Y=0;
S1:Y=0;
S2:Y=0;
S3:Y=0;
S4:Y=1;
default Y=0;
endcase
end
endmodule
```





Write a code for Mealy FSM to avoid Glitches	end
Gritches	C:
Code:	if(!x)
	begin
module Mealy(clk,x,z,rst);	ns<=A;
input clk,rst,x;	end
output reg z;	else
parameter A=2'b00,B=2'b01,C=2'b11,D=2'b10;	begin
reg [1:0] ps,ns;	ns<=D;
always @(posedge clk,negedge rst)	end
begin	D:
if(!rst)	if(!x)
ps<=A;	begin
else	ns<=C;
ps<=ns;	end
end	else
	begin
always @(ps,x)	ns<=A;
begin	end
case(ps)	default:
A:	ns<=0;
if(!x)	
begin	endcase
ns<=A;	end
end	
else	always @(posedge clk ,negedge rst)
begin	begin
ns<=B;	if(!rst)
end	z<=1'b0;
B:	else
if(!x)	begin
begin	z<=1'b0;
ns<=C;	case(ps)
end	A:
else	if(!x)
begin	z<=0;
ns<=B;	else

z<=0;

B:

if(!x)

z<=0;

else

z<=0;

C:

if(!x)

z<=0;

else

z<=0;

D:

if(!x)

z<=0;

else

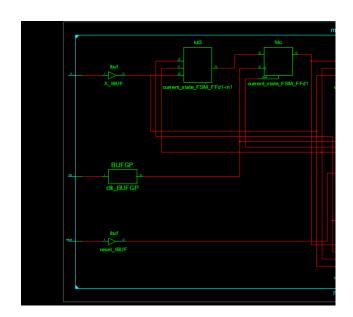
z<=1;

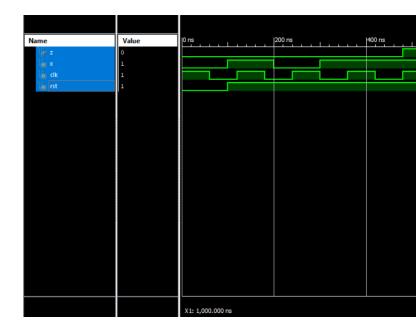
endcase

end

end

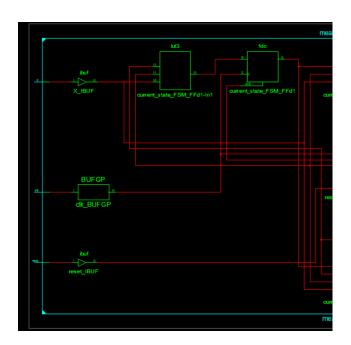
endmodule

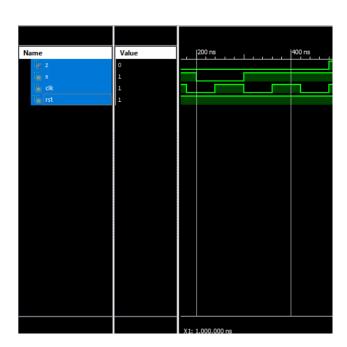




Mealy with one always block	begin
<pre>module mealy_lalwaysblock(clk,x,z,rst);</pre>	ns=B;
input clk,rst,x;	ps=ns;
output reg z;	z=0;
parameter	end
A=2'b00,B=2'b01,C=2'b11,D=2'b10;	C: if(!x)
reg [1:0] ps,ns;	begin
always @(posedge clk,negedge rst)	ns=A;
begin	ps=ns;
if(!rst)	
begin	z=0;
ps=A;	end
z=0;	else
end	begin
else	ns=D;
begin	ps=ns;
case(ps)	z=0;
A: $if(!x)$	end
begin	D: if(!x)
ns=A;	begin
ps=ns;	ns=C;
-	ns=C; ps=ns;
z=0;	ps=ns;
z=0; end	ps=ns; z=0;
z=0; end else	ps=ns; z=0; end
z=0; end else begin	ps=ns; z=0; end else
z=0; end else begin ns=B;	ps=ns; z=0; end else begin
z=0; end else begin ns=B; ps=ns;	ps=ns; z=0; end else begin ns=A;
z=0; end else begin ns=B; ps=ns; z=0;	ps=ns; z=0; end else begin ns=A; ps=ns;
z=0; end else begin ns=B; ps=ns; z=0; end	ps=ns; z=0; end else begin ns=A; ps=ns; z=1;
z=0; end else begin ns=B; ps=ns; z=0; end B: if(!x)	ps=ns; z=0; end else begin ns=A; ps=ns; z=1; end
z=0; end else begin ns=B; ps=ns; z=0; end B: if(!x) begin	ps=ns; z=0; end else begin ns=A; ps=ns; z=1; end default:
z=0; end else begin ns=B; ps=ns; z=0; end B: if(!x) begin ns=C;	ps=ns; z=0; end else begin ns=A; ps=ns; z=1; end default: begin
z=0; end else begin ns=B; ps=ns; z=0; end B: if(!x) begin	ps=ns; z=0; end else begin ns=A; ps=ns; z=1; end default: begin ns=A;
z=0; end else begin ns=B; ps=ns; z=0; end B: if(!x) begin ns=C; ps=ns;	ps=ns; z=0; end else begin ns=A; ps=ns; z=1; end default: begin ns=A; z=0;
z=0; end else begin ns=B; ps=ns; z=0; end B: if(!x) begin ns=C; ps=ns; z=0;	ps=ns; z=0; end else begin ns=A; ps=ns; z=1; end default: begin ns=A; z=0; end
z=0; end else begin ns=B; ps=ns; z=0; end B: if(!x) begin ns=C; ps=ns;	ps=ns; z=0; end else begin ns=A; ps=ns; z=1; end default: begin ns=A; z=0;

end endmodule





Moore using 2 always blocks.

```
module moore_2alwaysblock(clk,rst,x,y);
input rst,clk,x;
output reg y;
reg [2:0]ps,ns;
parameter s0=3'b000, s1=3'b001, s2=3'b010,
```

```
s3=3'b011, s4=3'b100;
always @(posedge clk,negedge rst)
begin
if(!rst)
ps<=s0;
else
ps<=ns;
end
always @(ps,x)
begin
case(ps)
3'b000: begin
y <= 1'b0;
if(!x)
ns \le s0;
 else
 ns <= s1;
 end
3'b001: begin
y <= 1'b0;
if(!x)
ns \le s2;
 else
 ns <= s1;
 end
3'b010: begin
y <= 1'b0;
if(!x)
ns \le s0;
 else
ns <= s3;
end
3'b011: begin
y <= 1'b0; if(!x)
ns \le s2;
else
ns <= s4;
```

end

3'b100: begin
y<=1'b1;
if(!x)
ns <=s0;
else
ns<=s1;
end
default:
begin
y<=1'b0;
ns<=s0;
end
endcase

end

endmodule

