30 Days of RTL Coding

-By T Tushar Shenoy

Day 21

Problem Statement: Implementing 101 Sequence Detector Moore machine.

Theory:

A sequence detector is a sequential state machine that takes an input string of bits and generates an output 1 whenever the target sequence has been detected. In a Moore machine, output depends only on the present state and not dependent on the input (x). Hence in the diagram, the output is written with the states.

Sequence detector is of two types:

1. Overlapping
2. Non-Overlapping

In an overlapping sequence detector, the last bit of one sequence becomes the first bit of the next sequence. However, in a non-overlapping sequence detector, the last bit of one sequence does not become the first bit of the next sequence.

Examples:

For non overlapping case

Input :0110101011001

Output:0000100010000

For overlapping case

Input :0110101011001

Output:0000101010000

Moore Machine 101 Sequence Detector non Overlapping

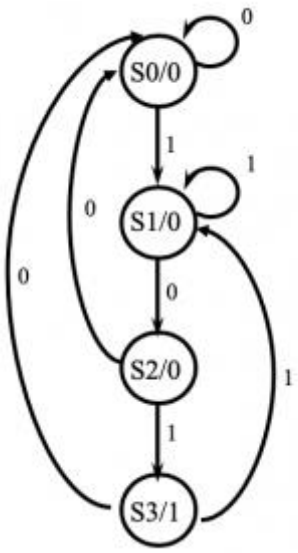


FIG: State Diagram of Moore Machine 101 Sequence Detector non Overlapping

Verilog Code:

```
//Verilog Code for 101 Sequence Detector (Moore Machine non Overlapping)
module fsm_detector_moore(
    input in,
    input reset,
    input clk,
    output reg out
    );

    parameter s0=2'b00;
    parameter s1=2'b01;
    parameter s2=2'b10;
    parameter s3=2'b11;
```

```
reg [1:0]cst;
reg [1:0]nst;
always@(posedge clk)
 begin
   if(reset)
     begin
      out=1'b0;
      cst=1'b0;
      nst=1'b0;
     end
   else
     begin
     cst=nst; //remember
      case(cst)

     s0: if(in)
          begin
           out=1'b0;
           nst=s1;
          end
           else
           begin
           out=1'b0;
           nst=s0;
           end
      s1: if(in)
          begin
           out=1'b0;
```

```
      nst=s1;
     end
      else
      begin
      out=1'b0;
      nst=s2;
      end
s2: if(in)
     begin
      out=1'b1;
      nst=s3;
     end
      else
      begin
      out=1'b0;
      nst=s0;
      end
s3: if(in)
     begin
      out=1'b0;
      nst=s1;
     end
      else
      begin
      out=1'b0;
      nst=s0;
      end
endcase
```

```
        end
    end

endmodule
```

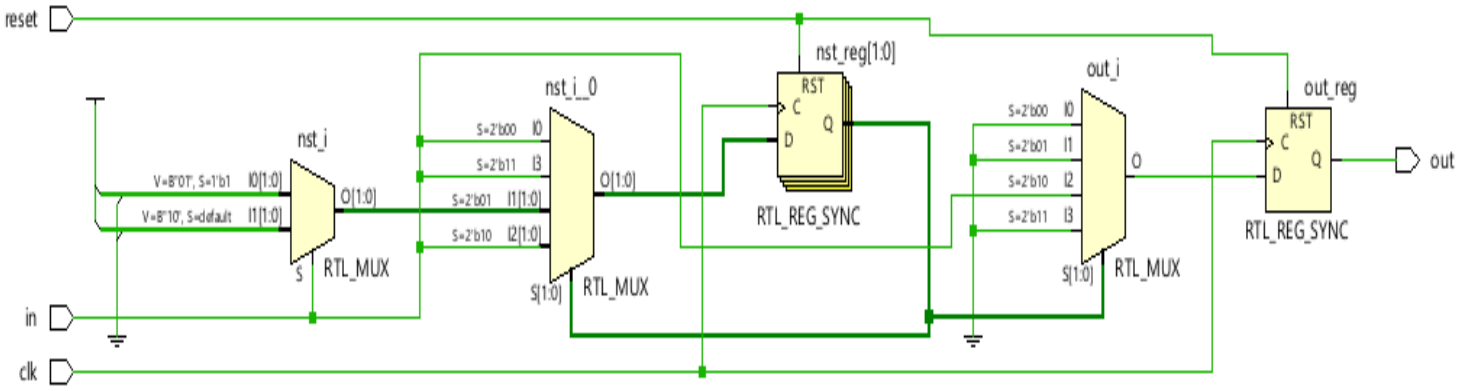
Testbench Code:

```
//Testbench Code for 101 Sequence Detector (Moore Machine non overlapping)
module fsm_detector_moore_tb();

reg clk,in,reset;
wire out;

fsm_detector_moore dut(in,reset,clk,out);

initial begin

 clk=1'b0;
 in=1'b0;
 reset=1'b0;
#5;
reset=1'b1;
#5;
reset=1'b0;

#10 in=1'b0;
```

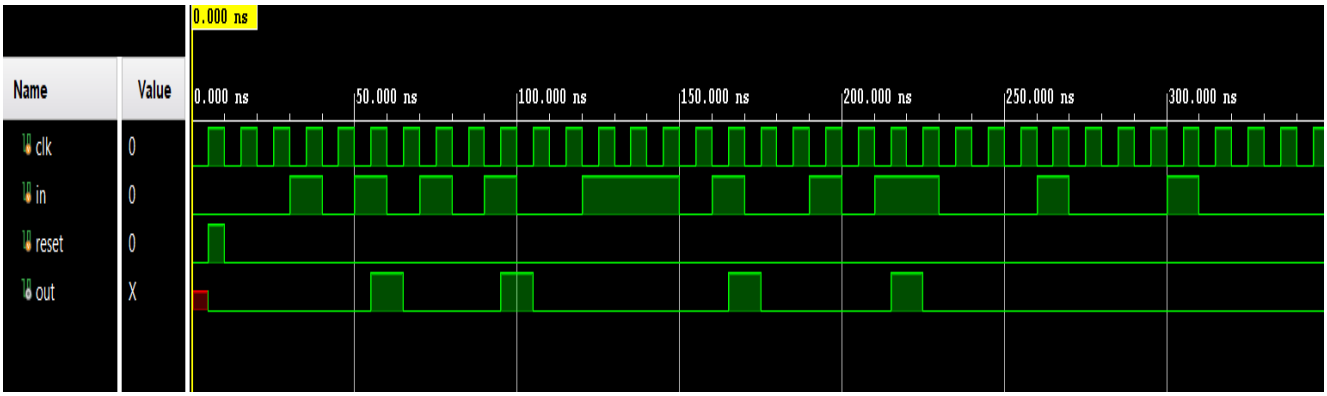
```
#10 in=1'b1;
#10 in=1'b0;
#10 in=1'b1;
#10 in=1'b0;
#10 in=1'b1;
#10 in=1'b0;
#10 in=1'b1;
#10 in=1'b0;
#10 in=1'b0;
#10 in=1'b1;
#10 in=1'b1;
#10 in=1'b1;
#10 in=1'b0;
#10 in=1'b1;
#10 in=1'b0;
#10 in=1'b0;
#10 in=1'b1;
#10 in=1'b0;
#10 in=1'b1;
#10 in=1'b1;
#10 in=1'b0;
#10 in=1'b0;
#10 in=1'b0;
#10 in=1'b1;
#10 in=1'b0;
#10 in=1'b0;
#10 in=1'b0;
#10 in=1'b1;
```

```
#10 in=1'b0;
#10 in=1'b0;
#10 in=1'b0;
#10 in=1'b0;
#10 $finish;
end
always #5 clk=~clk;
endmodule
```

Schematic:



Simulation Output:



Moore Machine 101 Sequence Detector Overlapping

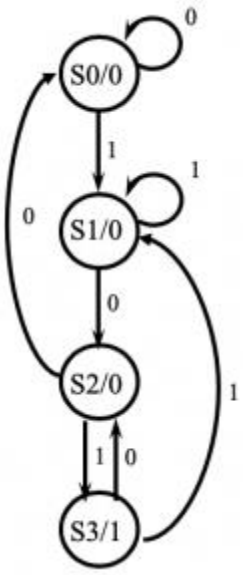


FIG: State Diagram of Moore Machine 101 Sequence Detector Overlapping

Verilog Code:

```
//Verilog Code for 101 Sequence Detector (Moore Machine Overlapping)
module fsm_detector_mooreo(
    input in,
    input reset,
    input clk,
    output reg out
    );

    parameter s0=2'b00;
    parameter s1=2'b01;
    parameter s2=2'b10;
    parameter s3=2'b11;
```

```
reg [1:0]cst;
reg [1:0]nst;
always@(posedge clk)
 begin
   if(reset)
     begin
      out=1'b0;
      cst=1'b0;
      nst=1'b0;
     end
   else
     begin
     cst=nst; //remember
      case(cst)

     s0: if(in)
          begin
           out=1'b0;
           nst=s1;
          end
           else
           begin
           out=1'b0;
           nst=s0;
           end
      s1: if(in)
          begin
           out=1'b0;
```

```
      nst=s1;
     end
      else
      begin
      out=1'b0;
      nst=s2;
      end
s2: if(in)
     begin
      out=1'b1;
      nst=s3;
     end
      else
      begin
      out=1'b0;
      nst=s0;
      end
s3: if(in)
     begin
      out=1'b0;
      nst=s1;
     end
      else
      begin
      out=1'b0;
      nst=s2;
      end
endcase
```

```
        end
    end

endmodule
```

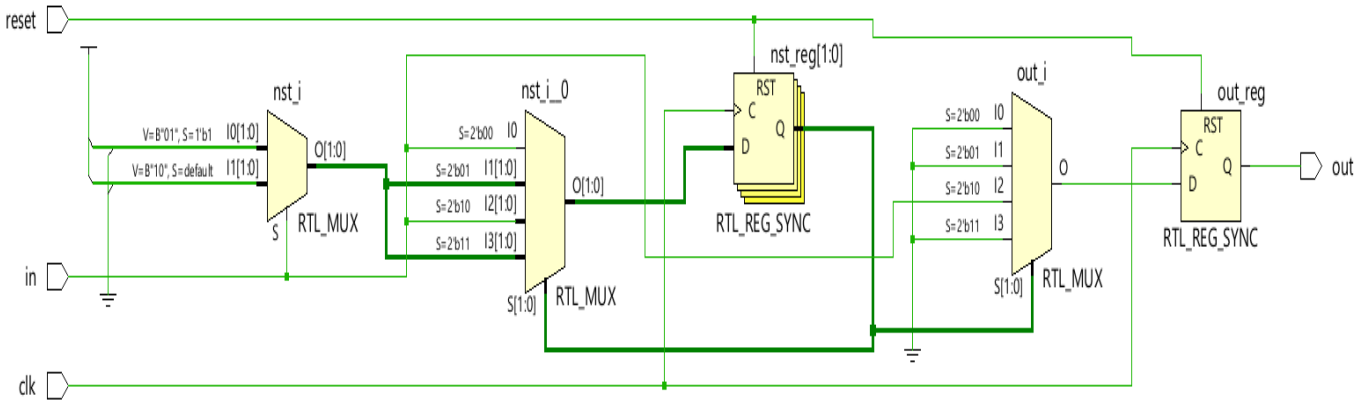
Testbench Code:

```
//Testbench Code for 101 Sequence Detector (Moore Machine Overlapping)
module fsm_detector_mooreo_tb();

reg clk,in,reset;
wire out;

fsm_detector_mooreo dut(in,reset,clk,out);

initial begin

 clk=1'b0;
 in=1'b0;
 reset=1'b0;
#5;
reset=1'b1;
#5;
reset=1'b0;

#10 in=1'b0;
```

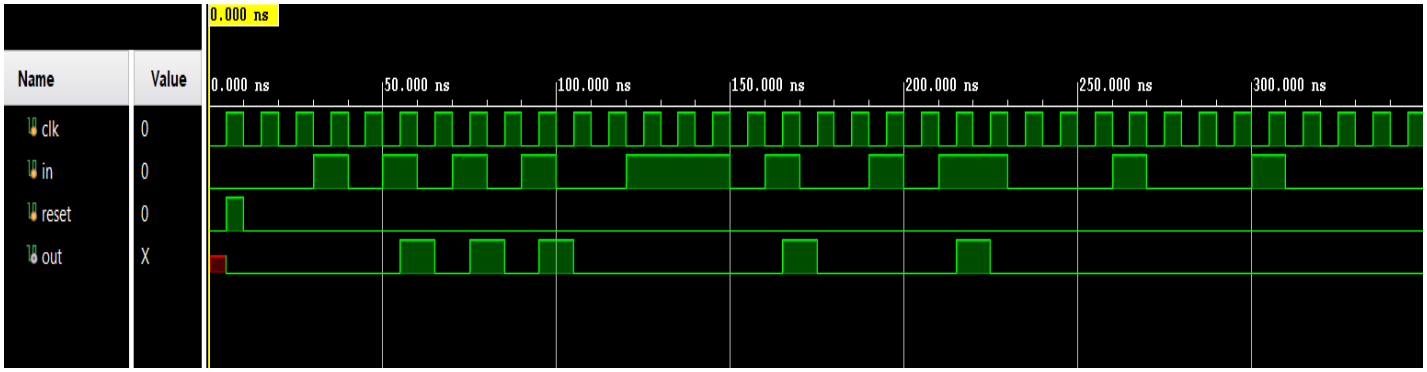
```
#10 in=1'b1;
#10 in=1'b0;
#10 in=1'b1;
#10 in=1'b0;
#10 in=1'b1;
#10 in=1'b0;
#10 in=1'b1;
#10 in=1'b0;
#10 in=1'b0;
#10 in=1'b1;
#10 in=1'b1;
#10 in=1'b1;
#10 in=1'b0;
#10 in=1'b1;
#10 in=1'b0;
#10 in=1'b0;
#10 in=1'b1;
#10 in=1'b0;
#10 in=1'b1;
#10 in=1'b1;
#10 in=1'b0;
#10 in=1'b0;
#10 in=1'b0;
#10 in=1'b1;
#10 in=1'b0;
#10 in=1'b0;
#10 in=1'b0;
#10 in=1'b1;
```

```
#10 in=1'b0;
#10 in=1'b0;
#10 in=1'b0;
#10 in=1'b0;
#10 $finish;
end
always #5 clk=~clk;
endmodule
```

Schematic:



Simulation Output:



GitHub Repository URL: https://github.com/tusharshenoy/RTL-Day-21-Moore-Machine