LOG BOOK [Week 11]
Name: TABINA KAMAL
Matric No.: 208651
Assigned Date: 06 January 2021

LAB 7: MODELSIM-INTEL SIMULATOR USING VERILOG TESTBENCH

1.0 Objectives

- Implementing a testbench and simulating the design using ModelSim software
- Performing basic simulation on an 8-bit binary up counter
- Designing and simulating a 16-bit full adder

2.0 Results and Simulation

A. Basic Simulation

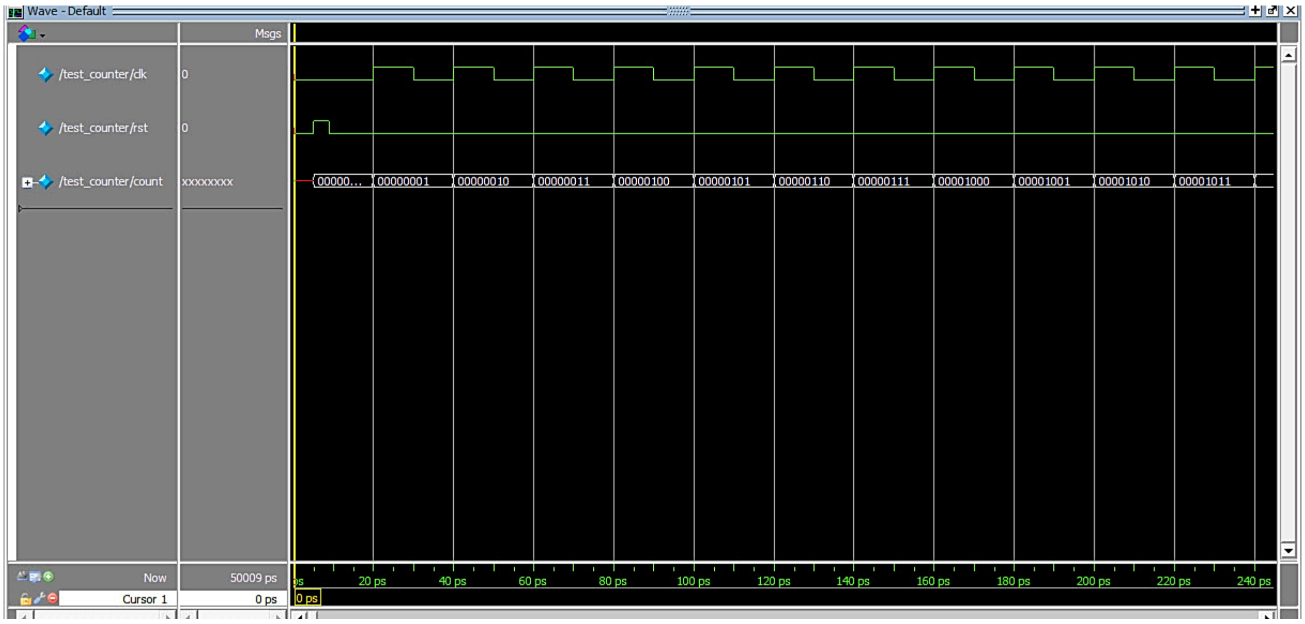
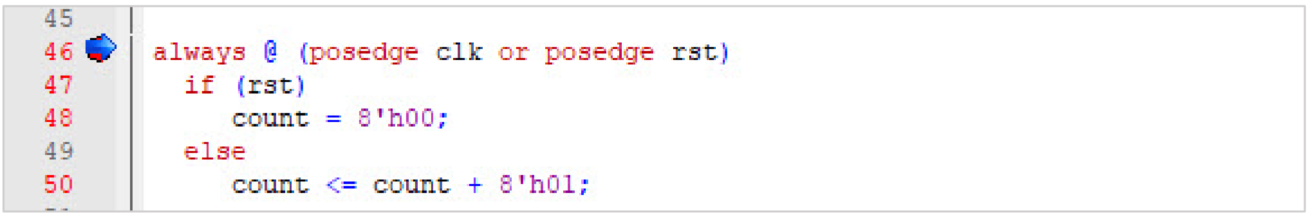


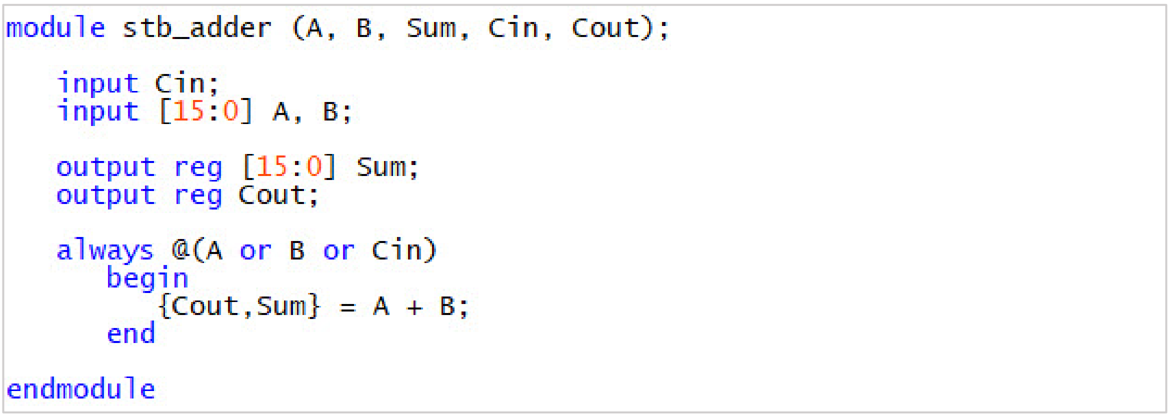
Figure 1: ModelSim waveform simulation for 8-bit binary counter



```
always @ (posedge clk or posedge rst)
  if (rst)
     count = 8'h00;
  else
     count <= count + 8'h01;
```

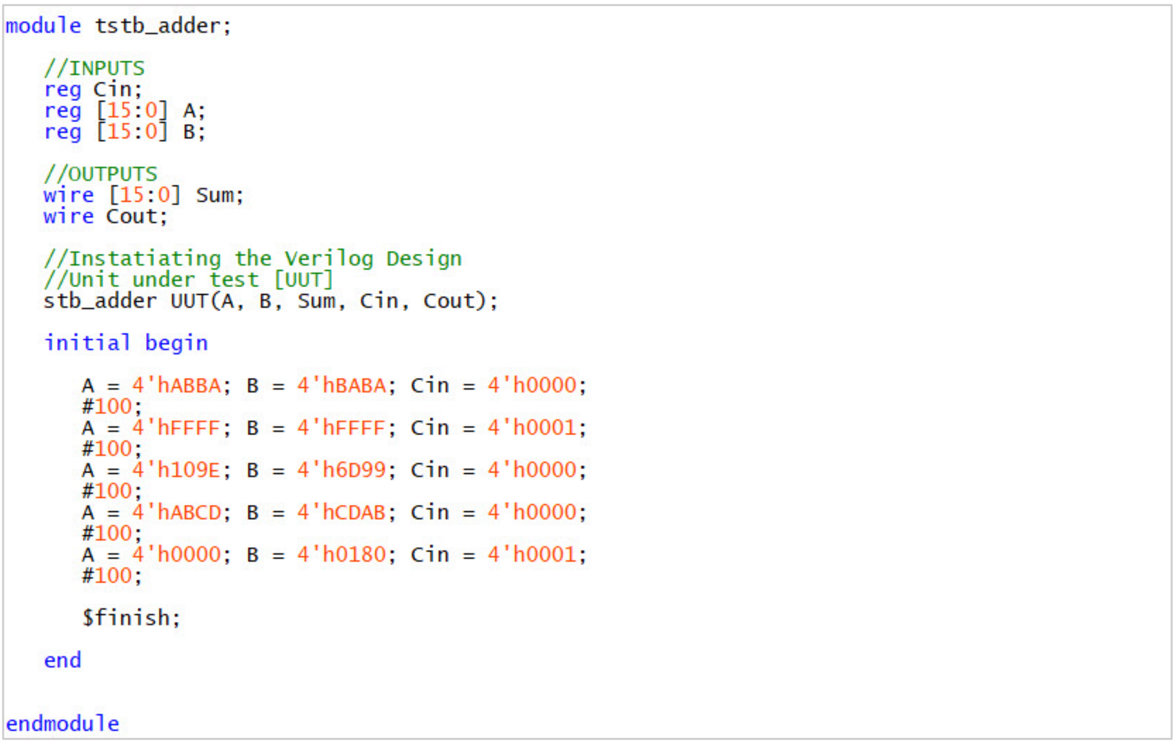
Figure 2: Breakpoint reached for 8-bit binary counter simulation

B. Design and Simulate a 16-bit Full-Adder



```
module stb_adder (A, B, Sum, Cin, Cout);

   input Cin;
   input [15:0] A, B;

   output reg [15:0] Sum;
   output reg Cout;

   always @(A or B or Cin)
      begin
         {Cout,Sum} = A + B;
      end

endmodule
```

Figure 3: 16-bit full adder Verilog code



```
module tstb_adder;

   //INPUTS
   reg Cin;
   reg [15:0] A;
   reg [15:0] B;

   //OUTPUTS
   wire [15:0] Sum;
   wire Cout;

   //Instatiating the Verilog Design
   //Unit under test [UUT]
   stb_adder UUT(A, B, Sum, Cin, Cout);

   initial begin

      A = 4'hABBA; B = 4'hBABA; Cin = 4'h0000;
      #100;
      A = 4'hFFFF; B = 4'hFFFF; Cin = 4'h0001;
      #100;
      A = 4'h109E; B = 4'h6D99; Cin = 4'h0000;
      #100;
      A = 4'hABCD; B = 4'hCDAB; Cin = 4'h0000;
      #100;
      A = 4'h0000; B = 4'h0180; Cin = 4'h0001;
      #100;

      $finish;

   end


endmodule
```

Figure 4: 16-bit full adder testbench code

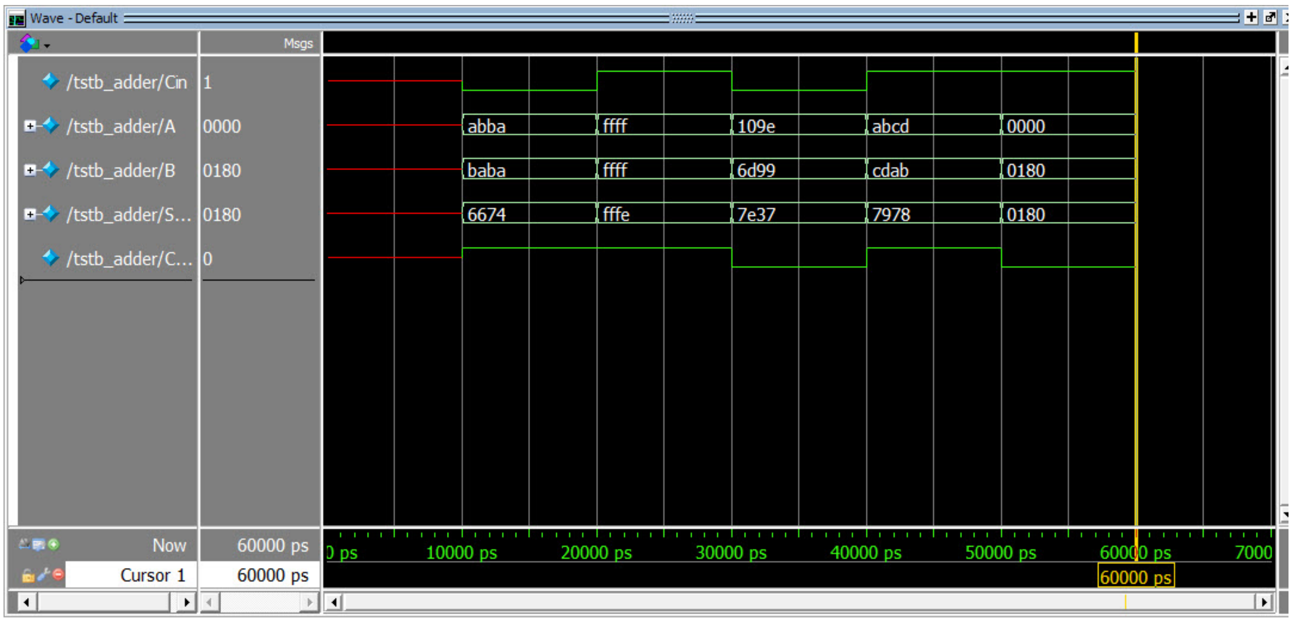


Figure 5: ModelSim waveform simulation for 16-bit full adder

Table 1: 16-bit full adder table of inputs and outputs

[15:0] A	ABBA	FFFF	109E	ABCD	0000
[15:0] B	BABA	FFFF	6D99	CDAB	0180
Cin	0	1	0	1	1
[15:0] Sum	6674	FFFE	7E37	7978	0180
Cout	1	1	0	1	0

3.0 Discussion

A testbench file is used for verification of a digital hardware design. It is used to simulate and analyse a design without need for any physical hardware.

Complicated circuits can be analysed easily with the testbench file on ModelSim unlike the waveform functional simulation in Quartus Prime which would be very tedious and error prone. Any unexpected results can be easily spotted.

The use of testbench allows better detection of errors in the Verilog code before being implemented on hardware. In the industry, this will prevent producing faulty hardware.

4.0 Conclusion

A. Basic Simulation

Successfully compiled and simulated.

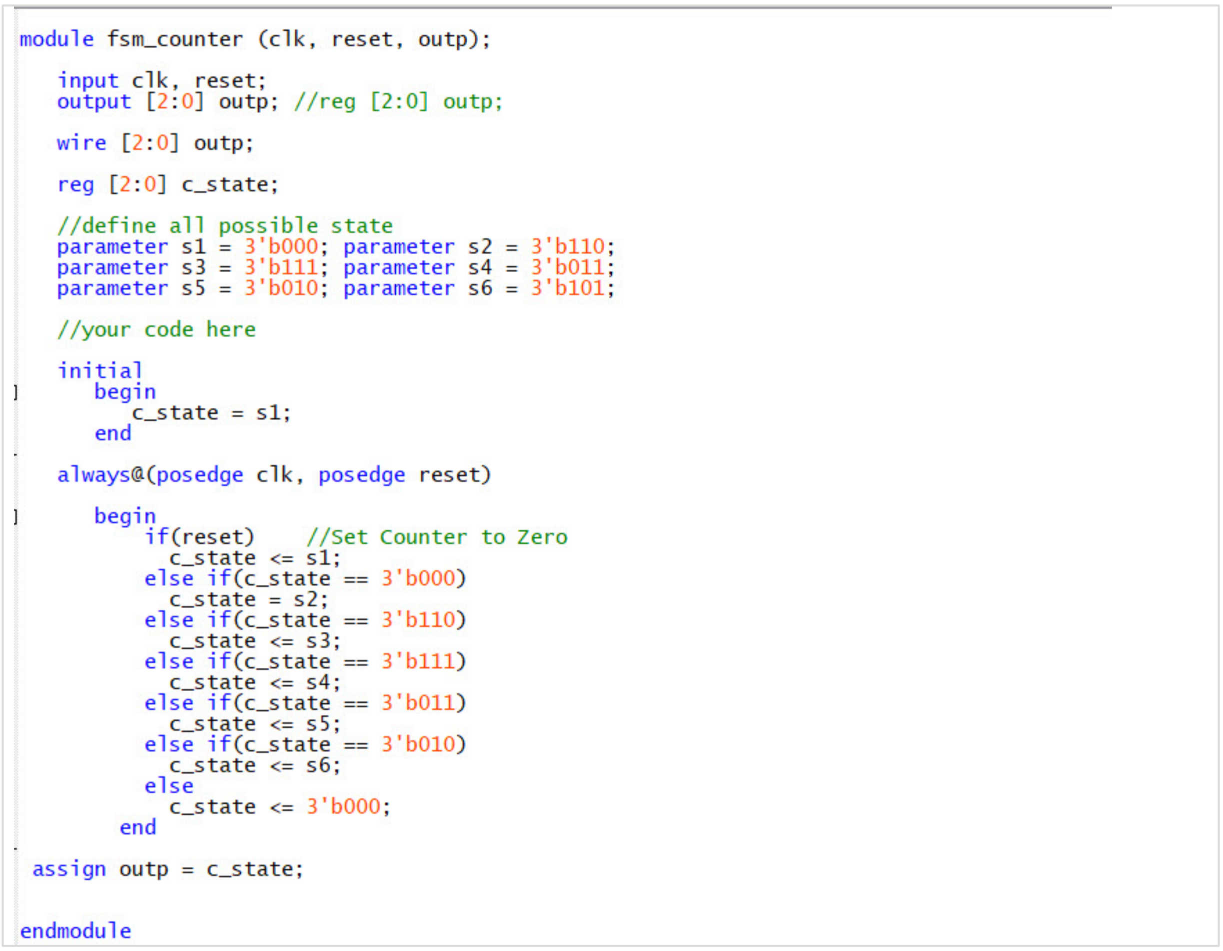
B. Design and Simulate a 16-bit Full-Adder

Successfully compiled and simulated.

CONTINUATION FROM WEEK 8

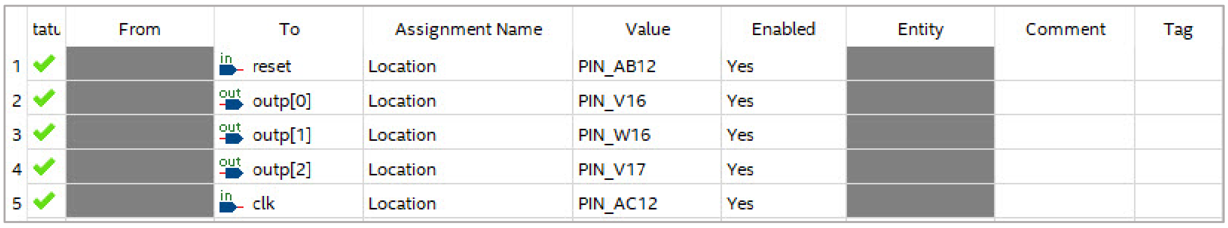
LAB 6: MULTIPLEXERS AND STATE MACHINE

The finite state machine [Part B] was redone and demonstrated. With the previous code, of which switches were used as the input, an erroneous output was obtained. Push buttons were used instead and the correct output was obtained and demonstrated successfully.



```
module fsm_counter (clk, reset, outp);

   input clk, reset;
   output [2:0] outp; //reg [2:0] outp;

   wire [2:0] outp;

   reg [2:0] c_state;

   //define all possible state
   parameter s1 = 3'b000; parameter s2 = 3'b110;
   parameter s3 = 3'b111; parameter s4 = 3'b011;
   parameter s5 = 3'b010; parameter s6 = 3'b101;

   //your code here

   initial
      begin
         c_state = s1;
      end

   always@(posedge clk, posedge reset)

      begin
          if(reset)     //Set Counter to Zero
            c_state <= s1;
          else if(c_state == 3'b000)
            c_state = s2;
          else if(c_state == 3'b110)
            c_state <= s3;
          else if(c_state == 3'b111)
            c_state <= s4;
          else if(c_state == 3'b011)
            c_state <= s5;
          else if(c_state == 3'b010)
            c_state <= s6;
          else
            c_state <= 3'b000;
        end

 assign outp = c_state;


endmodule
```

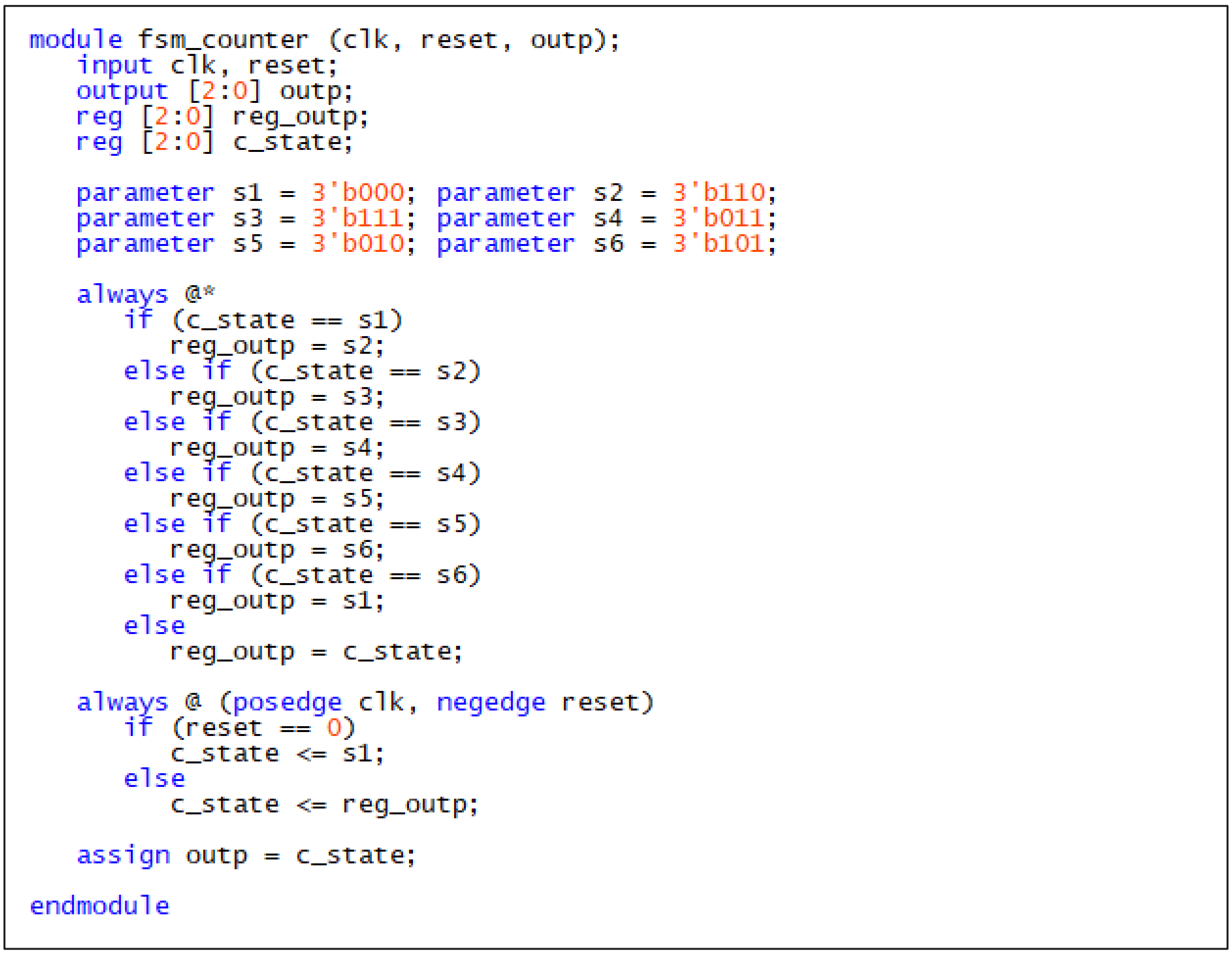
Figure 6: Finite State Machine Counter Code [Previous]



	tatu	From	To	Assignment Name	Value	Enabled	Entity	Comment	Tag
1	✓		in reset	Location	PIN_AB12	Yes			
2	✓		out outp[0]	Location	PIN_V16	Yes			
3	✓		out outp[1]	Location	PIN_W16	Yes			
4	✓		out outp[2]	Location	PIN_V17	Yes			
5	✓		in clk	Location	PIN_AC12	Yes			

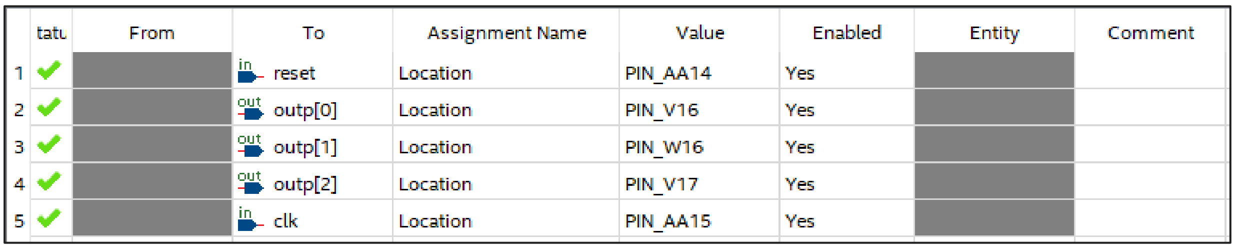
Figure 7: Finite State Machine Counter Pin Assignment [Previous]

LOG BOOK [Week 11]
Name: TABINA KAMAL
Matric No.: 208651
Assigned Date: 06 January 2021



```
module fsm_counter (clk, reset, outp);
   input clk, reset;
   output [2:0] outp;
   reg [2:0] reg_outp;
   reg [2:0] c_state;

   parameter s1 = 3'b000; parameter s2 = 3'b110;
   parameter s3 = 3'b111; parameter s4 = 3'b011;
   parameter s5 = 3'b010; parameter s6 = 3'b101;

   always @*
      if (c_state == s1)
         reg_outp = s2;
      else if (c_state == s2)
         reg_outp = s3;
      else if (c_state == s3)
         reg_outp = s4;
      else if (c_state == s4)
         reg_outp = s5;
      else if (c_state == s5)
         reg_outp = s6;
      else if (c_state == s6)
         reg_outp = s1;
      else
         reg_outp = c_state;

   always @ (posedge clk, negedge reset)
      if (reset == 0)
         c_state <= s1;
      else
         c_state <= reg_outp;

   assign outp = c_state;

endmodule
```

Figure 8: Finite State Machine Counter Code [UPDATED]



	tatu	From	To	Assignment Name	Value	Enabled	Entity	Comment
1	✓		reset	Location	PIN_AA14	Yes		
2	✓		outp[0]	Location	PIN_V16	Yes		
3	✓		outp[1]	Location	PIN_W16	Yes		
4	✓		outp[2]	Location	PIN_V17	Yes		
5	✓		clk	Location	PIN_AA15	Yes		

Figure 9: Finite State Machine Counter Pin Assignment [UPDATED]