COE 202, Term 142

Digital Logic Design

Assignment# 4 Solution

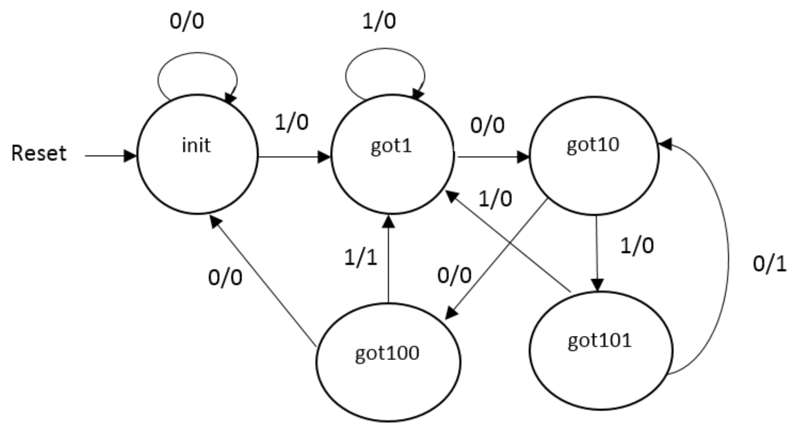
Due date: Tuesday, May 12

Q.1. It is required to design a synchronous sequential circuit that receives a serial input **X** and produces a serial output **Z** that is set to 1 when the circuits detects either the sequence 1010 or the sequence 1001. Assume that the detection of sequences is overlapping. Assume the existence of an asynchronous reset input to reset the machine to a reset state. The following is an example of an input/output stream:

Example:

Input	**X**	1 0 1 0 1 0 0 1 0 1 0 0 0
Output	**Z**	0 0 0 1 0 1 0 1 0 0 1 0 0

(i) Derive the state diagram of the circuit assuming a **Mealy** model.



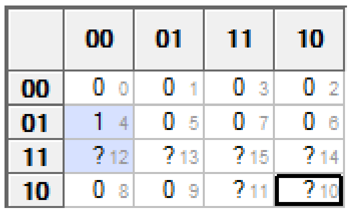
(ii) Implement your design using D flip flops with minimal number of flip flops and combinational logic.

We will use the following state assignments:
init: 000, got1: 001, got10: 010, got101: 011, got100: 100

<u>State Transition Table</u>:

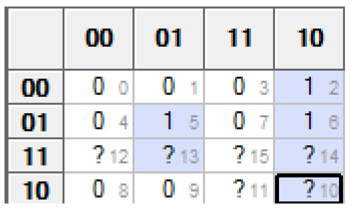
Current State			Input	Next State			Output
F2	F1	F0	X	$F2^+$	$F1^+$	$F0^+$	Z
0	0	0	0	0	0	0	0
0	0	0	1	0	0	1	0
0	0	1	0	0	1	0	0
0	0	1	1	0	0	1	0
0	1	0	0	1	0	0	0
0	1	0	1	0	1	1	0
0	1	1	0	0	1	0	1
0	1	1	1	0	0	1	0
1	0	0	0	0	0	0	0
1	0	0	1	0	0	1	1
1	0	1	0	x	x	x	x
1	0	1	1	x	x	x	x
1	1	0	0	x	x	x	x
1	1	0	1	x	x	x	x
1	1	1	0	x	x	x	x
1	1	1	1	x	x	x	x

<u>K-Map Minimization</u>:



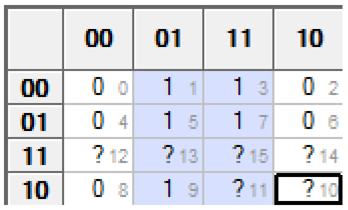
	00	01	11	10
00	0 0	0 1	0 3	0 2
01	1 4	0 5	0 7	0 6
11	? 12	? 13	? 15	? 14
10	0 8	0 9	? 11	? 10

$F2^+$ = F1 F0' X'



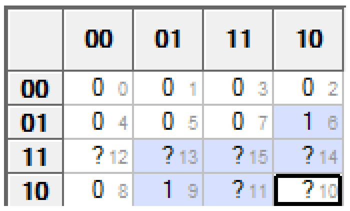
	00	01	11	10
00	0 0	0 1	0 3	1 2
01	0 4	1 5	0 7	1 6
11	? 12	? 13	? 15	? 14
10	0 8	0 9	? 11	? 10

$F1^+$ = F1 F0' X + F0 X'



	00	01	11	10
00	0 0	1 1	1 3	0 2
01	0 4	1 5	1 7	0 6
11	? 12	? 13	? 15	? 14
10	0 8	1 9	? 11	? 10

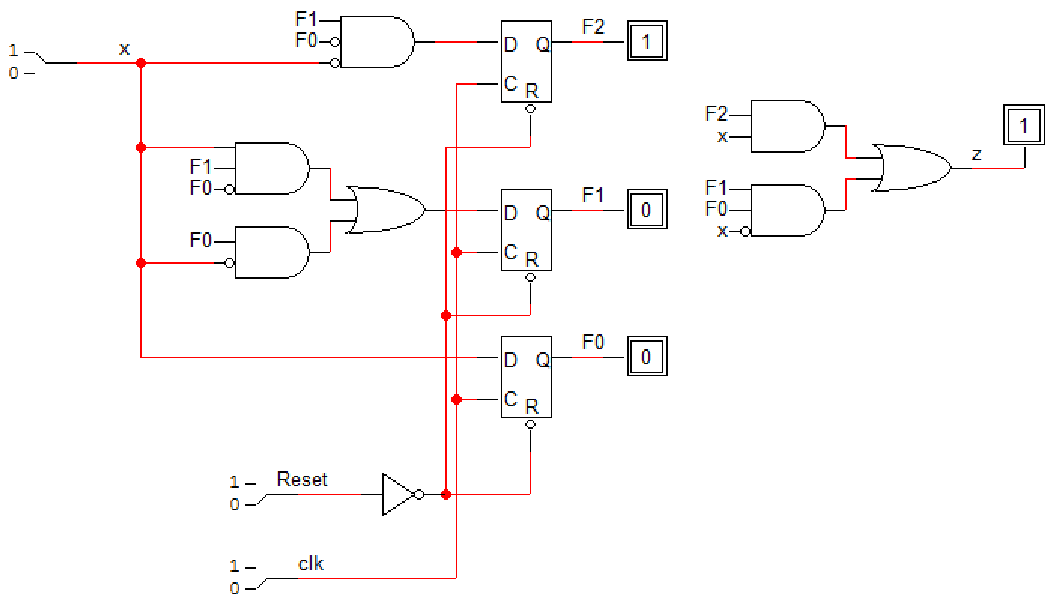
$F0^+ = X$



	00	01	11	10
00	0 0	0 1	0 3	0 2
01	0 4	0 5	0 7	1 6
11	? 12	? 13	? 15	? 14
10	0 8	1 9	? 11	? 10

$Z = F2\ X + F1\ F0\ X'$

The resulting circuit is shown below:



(iii) Write a structural Verilog model that models your implemented sequential circuit by modeling the D Flip-Flops and instantiating them and modeling the combinational part using either assign statement or gate primitives.

```
module Ass4_Structural (output  z, input x, clk, Reset );

dff2 M0 (F0, F0b, x, 1'b0, Reset, clk);
dff2 M1 (F1, F1b, DF1, 1'b0, Reset, clk);
dff2 M2 (F2, F2b, DF2, 1'b0, Reset, clk);

and (DF2, F1, F0b, !x);
and (w1, F1, F0b, x);
and (w2, F0, !x);
or  (DF1, w1, w2);
and (w3, F2, x);
and  (w4, F1, F0, !x);
or  (z, w3, w4);

endmodule
```

(iv) Write a test bench that tests your structural Verilog model in (iii) using the given input sequence. Verify that your circuit produces the correct output by including the generated waveform from simulations.

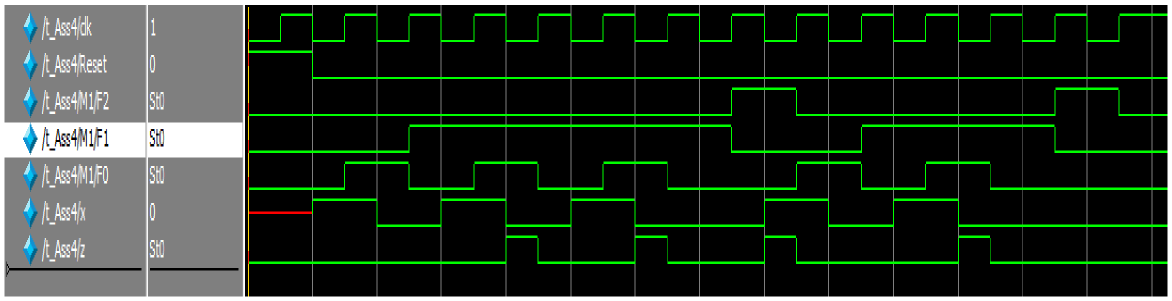
The test bench used to test the structural model is as follows:

```
module t_Ass4();

  wire   z;
  reg     x, clk, Reset;
  Ass4_Structural M1 (z, x, clk, Reset );

initial begin
    Reset=1; clk=0;  // input sequence x = 1 0 1 0 1 0 0 1 0 1 0 0 0
    #10  Reset=1; clk=1;
    #10  x=1; Reset=0; clk=0;
    #10  x=1; clk=1;
    #10  x=0; clk=0;
    #10  x=0; clk=1;
    #10  x=1; clk=0;
    #10  x=1; clk=1;
    #10  x=0; clk=0;
    #10  x=0; clk=1;
    #10  x=1; clk=0;
    #10  x=1; clk=1;
    #10  x=0; clk=0;
    #10  x=0; clk=1;
    #10  x=0; clk=0;
    #10  x=0; clk=1;
    #10  x=1; clk=0;
    #10  x=1; clk=1;
    #10  x=0; clk=0;
    #10  x=0; clk=1;
    #10  x=1; clk=0;
    #10  x=1; clk=1;
    #10  x=0; clk=0;
    #10  x=0; clk=1;
    #10  x=0; clk=0;
    #10  x=0; clk=1;
    #10  x=0; clk=0;
    #10  x=0; clk=1;
  end
endmodule
```

The resulting waveform is shown below which confirms correct behavior of the structural model:



(v) Write a behavioral Verilog model that models your state diagram in (i).

```
module Ass4_Behavioral (output reg z, input x, clk, Reset );

parameter init = 3'b000, got1=3'b001, got10=3'b010, got101=3'b011,
got100=3'b100;

reg [2:0] state, next_state;
always @(posedge clk)
  if (Reset) state <= init; else state <= next_state;

always @(state, x) begin
z = 0;
case (state)
      init:   if (x)  next_state=got1; else next_state=init;
      got1:   if (x)  next_state=got1; else next_state=got10;
      got10:  if (x)  next_state=got101; else next_state=got100;
      got101: if (x)  next_state=got1; else begin z=1; next_state=got10; end
      got100: if (x)  begin z=1; next_state=got1; end else next_state=init;
      default: begin z=1'bx;  next_state=3'bxxx; end  // making other conditions
don't care conditions
   endcase
end
endmodule
```

(vi) Use the test bench developed in (iv) to test the correctness of your behavioral model developed in (v). Verify that your behavioral model produces the correct output by including the generated waveform from simulations.

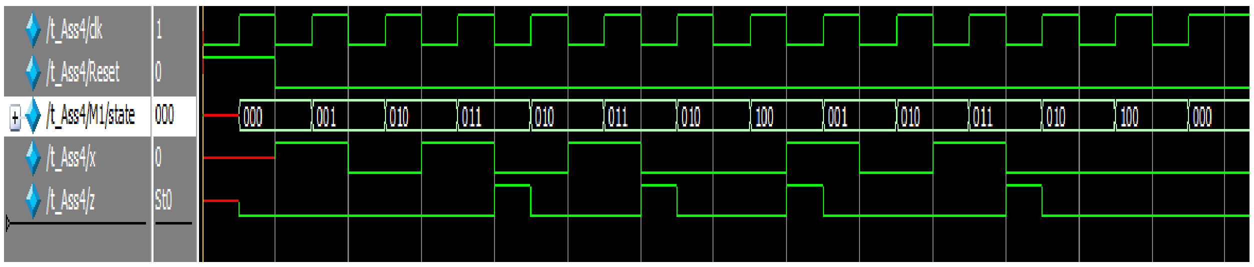
The test bench has been modified to test the behavioral model as follows:

```
module t_Ass4();
```

```
  wire   z;
  reg     x, clk, Reset;
  Ass4_Behavioral M1 (z, x, clk, Reset );

initial begin
    Reset=1; clk=0;  // input sequence x = 1 0 1 0 1 0 0 1 0 1 0 0 0
    #10  Reset=1; clk=1;
    #10  x=1; Reset=0; clk=0;
    #10  x=1; clk=1;
    #10  x=0; clk=0;
    #10  x=0; clk=1;
    #10  x=1; clk=0;
    #10  x=1; clk=1;
    #10  x=0; clk=0;
    #10  x=0; clk=1;
    #10  x=1; clk=0;
    #10  x=1; clk=1;
    #10  x=0; clk=0;
    #10  x=0; clk=1;
    #10  x=0; clk=0;
    #10  x=0; clk=1;
    #10  x=1; clk=0;
    #10  x=1; clk=1;
    #10  x=0; clk=0;
    #10  x=0; clk=1;
    #10  x=1; clk=0;
    #10  x=1; clk=1;
    #10  x=0; clk=0;
    #10  x=0; clk=1;
    #10  x=0; clk=0;
    #10  x=0; clk=1;
    #10  x=0; clk=0;
    #10  x=0; clk=1;
  end
endmodule
```

The resulting waveform is shown below which confirms correct behavior of the behavioral model:


/t_Ass4/clk
/t_Ass4/Reset
/t_Ass4/M1/state
/t_Ass4/x
/t_Ass4/z
St0