COE 202, Term 151

Digital Logic Design

Assignment# 4 Solution

Due date: Sunday, Dec. 13

Q.1. It is required to design a synchronous sequential circuit that receives a serial input **X** applied from least significant bit to most significant bit and produces a serial output **Z=2*X-1**. Assume the existence of an asynchronous reset input to reset the machine to a reset state. The following are examples of input/output streams:

Examples:

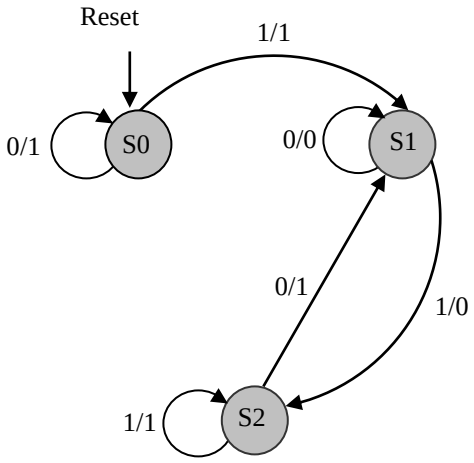
⟹

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

⟹

Input	**X**	0 0 1 0 0
Output	**Z**	1 1 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 need two D-FFs and we will assume the use of the state variables $\mathbf{Y_1}$, and $\mathbf{Y_0}$. We will assume the following state encoding: S0=00, S1=01, S2=10.

PS $(Y_1\ Y_0)^t$	NS $(Y1\ Y_0)^{t+1}$		Z	
	X = 0	X = 1	X = 0	X = 1
0 0	0 0	0 1	1	1
0 1	0 1	1 0	0	0
1 0	0 1	1 0	1	1

	00	01	11	10
0	0 0	1 1	? 3	1 2
1	1 4	0 5	? 7	0 6

D0 = X' Y1 + X' Y0 + X Y1' Y0' = X' (Y1 + Y0) + X Y1' Y0' = X ⊕ (Y1 + Y0)

	00	01	11	10
0	0 0	0 1	? 3	0 2
1	0 4	1 5	? 7	1 6

D1 = X Y1 + X Y0 = X (Y1 + Y0)

	00	01	11	10
0	1 0	0 1	? 3	1 2
1	1 4	0 5	? 7	1 6

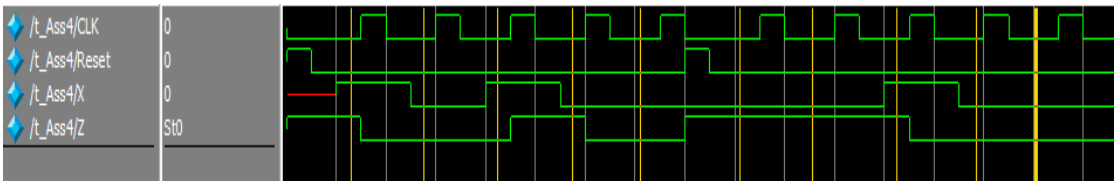
Z = Y0'

(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 Ass4Struct(output Z, input X, Reset, CLK);
dff2 F0 (Y0, Y0b, DY0, 1'b0, Reset, CLK);
dff2 F1 (Y1, Y1b, DY1, 1'b0, Reset, CLK);
or  (w1, Y1, Y0);
xor  (DY0, X, w1);
and  (DY1, X, w1);
assign Z=Y0b;
endmodule
```

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

```
module t_Ass4();
  wire  Z;
  reg   X, Reset, CLK;

Ass4Struct M1 (Z, X, Reset, CLK);

initial begin
    Reset=1; CLK=0; // Applying First Sequence
    #10 Reset=0;
    #10 X=1;
    #10 CLK=1;
    #10 CLK=0;
    #10 X=0;
    #10 CLK=1;
    #10 CLK=0;
    #10 X=1;
    #10 CLK=1;
    #10 CLK=0;
    #10 X=0;
    #10 CLK=1;
    #10 CLK=0;
    #10 X=0;
    #10 CLK=1;
    #10 CLK=0;
     Reset=1; CLK=0; // Applying Second Sequence
    #10 Reset=0;
    #10 X=0;
    #10 CLK=1;
    #10 CLK=0;
    #10 X=0;
    #10 CLK=1;
    #10 CLK=0;
    #10 X=1;
    #10 CLK=1;
    #10 CLK=0;
    #10 X=0;
    #10 CLK=1;
    #10 CLK=0;
    #10 X=0;
    #10 CLK=1;
    #10 CLK=0;
  end
endmodule
```



Looking at the waveform, one can see that the correct outputs are obtained for both applied input sequences as indicated by the shown cursors.

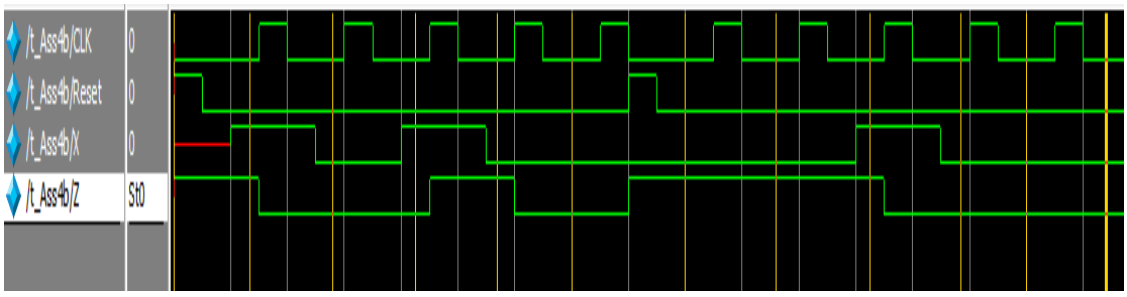
(v) Write a behavioral Verilog model that models your state diagram in (i).

```
module Ass4Behav(output reg Z, input X, Reset,CLK);

parameter S0 = 2'b00, S1=2'b01, S2=2'b10;
reg [1:0] state, next_state;

always @(posedge CLK, posedge Reset)
  if (Reset) state <= S0; else state <= next_state;

always @(state, X) begin
case (state)
     S0:  begin Z=1;  if (X)  next_state=S1; else next_state=S0; end
     S1:  begin Z=0;  if (X)  next_state=S2; else next_state=S1; end
     S2:  begin Z=1;  if (X)  next_state=S2; else next_state=S1; end
     default: begin  Z='bx; next_state='bx; end
   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.

```
module t_Ass4b();
  wire   Z;
  reg    X, Reset, CLK;
  Ass4Behav M1 (Z, X, Reset, CLK);


initial begin
    Reset=1; CLK=0; // Applying First Sequence
    #10 Reset=0;
    #10 X=1;
    #10 CLK=1;
```

```
    #10 CLK=0;
    #10 X=0;
    #10 CLK=1;
    #10 CLK=0;
    #10 X=1;
    #10 CLK=1;
    #10 CLK=0;
    #10 X=0;
    #10 CLK=1;
    #10 CLK=0;
    #10 X=0;
    #10 CLK=1;
    #10 CLK=0;
     Reset=1; CLK=0; // Applying Second Sequence
    #10 Reset=0;
    #10 X=0;
    #10 CLK=1;
    #10 CLK=0;
    #10 X=0;
    #10 CLK=1;
    #10 CLK=0;
    #10 X=1;
    #10 CLK=1;
    #10 CLK=0;
    #10 X=0;
    #10 CLK=1;
    #10 CLK=0;
    #10 X=0;
    #10 CLK=1;
    #10 CLK=0;
  end
endmodule
```



We can see that we are also getting the correct output sequences for the two input sequences based on the behavioral model.