COE 202, Term 201

 Digital Logic Design

Assignment# 4 Solution

Due date: Thursday, Dec. 10, 2020

It is required to design a synchronous sequential circuit that receives a serial input X that produces 1 when the input sequence is either {1010} (i.e., 1 followed by 0 followed by 1 followed by 0) or {1001} (i.e., 1 followed by 0 followed by 0 followed by 1) assuming overlapping sequences. The output Z is also a bit stream that produces a 1 only after detecting any of the two sequences. Use an asynchronous reset input to reset the sequential circuit to its initial state.

Example:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Clock cycle | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| X | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| Z | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |

**a)** (10 points) Draw a Mealy state diagram of the sequential circuit.



Reset

**b)** (10 points) Implement your design using a minimal number of D-type flip flops and combinational logic. Specify the K-maps and write the minimal next state and output equations.

We will use the state assignment: S0 = 000, S1 = 001, S2 = 010, S3 = 011, S4 = 100

|  |  |  |  |
| --- | --- | --- | --- |
| **Current State** | **Input** | **Next State** | **Output** |
| A  | B  | C  | X  | A+  | B+  | C+  | 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  | 0  | 1  | 1  | 0  |
| 0  | 1  | 0  | 1  | 1  | 0  | 0  | 0  |
| 0  | 1  | 1  | 0  |  0  | 0  | 0  | 0  |
| 0  | 1  | 1  | 1  | 0  | 0  | 1  | 1  |
| 1  | 0  | 0  | 0  | 0  | 1  | 0  | 1  |
| 1  | 0  | 0  | 1  | 0  | 0  | 1  | 0  |
| 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  |

The K-Maps are as follows:

A+ = BC’X



B+ = B’CX’+ BC’X’+ AX’

 C+ = BC’X’ + B’X + CX

 Z = BCX + AX’

**c)** (10 points) 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 dff (output reg q, output q\_bar, input data, set, rst, clk);

assign q\_bar = !q;

always @(posedge clk, posedge set, posedge rst)

 if (rst == 1'b1) q <= 0;

 else if (set == 1'b1) q <=1; else q <= data;

endmodule

module SeqCrt1(output Z, input X, Reset, CLK);

wire [2:0] stateS;

assign stateS = {A, B, C};

dff F0 (A, Ab, DA, 1'b0, Reset, CLK);

dff F1 (B, Bb, DB, 1'b0, Reset, CLK);

dff F2 (C, Cb, DC, 1'b0, Reset, CLK);

assign DA = B&~C&X;

assign DB = ~B&C&~X|B&~C&~X|A&~X;

assign DC = B&~C&~X|~B&X|C&X;

assign Z = B&C&X|A&~X;

endmodule

**d)** (10 points) Write a behavioral Verilog description that models your state diagram in part (a).

module SeqCrt2 (output reg Z, input X, Reset, clk);

parameter S0=3'b000, S1=3'b001, S2=3'b010, S3=3'b011, S4=3'b100;

reg [2:0] state, next\_state;

always @(posedge clk, posedge Reset)

 if (Reset) state <= S0;else state <= next\_state;

always @(state, X) begin

 Z = 0;

 case (state)

 S0: if (X) next\_state = S1; else next\_state = S0;

 S1: if (X) next\_state = S1; else next\_state = S2;

 S2: if (X) next\_state = S4; else next\_state = S3;

 S3: if (X) begin Z = 1 ;next\_state = S1; end else next\_state = S0;

 S4: if (X) next\_state = S1; else begin Z = 1; next\_state = S2; end

endcase

end

endmodule

**e)** (10 points) Write a test bench that tests BOTH the structural Verilog model of part (c) and the behavioral Verilog model of part (d) using the example input sequence shown above. Call the outputs Z1 and Z2 for the models in (c) and (d), respectively. Start by resetting all flip-flops and then apply the input sequence of X. Verify that your circuit produces the correct output by including the generated waveform from simulation and comparing to the given example above.

module Test\_SeqCrt();

 wire Z1, Z2;

 reg X, rst, clk;

 SeqCrt1 M1 (Z1, X, rst, clk); SeqCrt2 M2 (Z2, X, rst, clk);

initial begin

rst = 0 ; clk = 0 ; #2 ; rst = 1; #3; rst = 0; forever #5 clk = ~ clk ; end

initial begin

 $dumpfile("dump.vcd");

 $dumpvars(0, Test\_SeqCrt);

 X=0;

@(negedge clk) X=1;

@(negedge clk) X=0;

@(negedge clk) X=1;

@(negedge clk) X=0;

@(negedge clk) X=0;

@(negedge clk) X=0;

@(negedge clk) X=1;

@(negedge clk) X=0;

@(negedge clk) X=0;

@(negedge clk) X=1;

@(negedge clk) X=0;

@(negedge clk) X=0;

@(negedge clk) X=1;

@(negedge clk) X=0;

@(negedge clk) X=1;

@(negedge clk) X=0;

@(negedge clk) X=1;

@(negedge clk) X=0;

#10 $finish;

end

endmodule

The simulation waveform is shown below which shows that the correct urput sequences are obtained as desired:



f) Submit a report (Word or PDF document) that should contain:

1. The state diagram of your design (part a).
2. The state table, K-maps, and logic equations of your sequential circuit (part b).
3. A copy of the Verilog modules and test benches of parts (c), (d), and (e).
4. The timing diagrams (waveforms) taken directly as snapshots from the simulator for parts (e) and analysis of output.

The assignment can be solved individually, or in groups of two students. Submit a soft copy of the report on Blackboard.