University of Washington – Computer Science & Engineering
Spring 2021 Instructor: Clarice Larson 2021-05-18

CSE 369 QUIZ 2

Name: _Molly Model____________________

UWNetID: _model___________________________

Please do not turn the page until 11:30.

Instructions

- This quiz contains 4 pages, including this cover page. You may use the backs of the pages for scratch work.
- Please clearly indicate (box, circle) your final answer.
- The quiz is open book and open notes.
- Please silence and put away all cell phones and other mobile or noise-making devices.
- You have 25 minutes to complete this quiz.

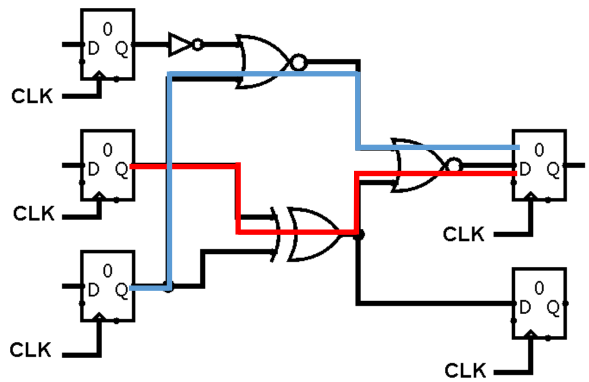
Advice

- Read questions carefully before starting. Read *all* questions first and start where you feel the most confident to maximize the use of your time.
- There may be partial credit for incomplete answers; please show your work.
- Relax. You are here to learn.

Question	Points	Score
(1) SL & Timing	6	6
(2) FSM Implementation	10	10
(3) FSM Design	10	10
Total:	**26**	**26**

Question 1: Sequential Logic & Timing [6 pts]

Consider the following circuit with t_{XOR} = **30 ns** (10^{-9} s), t_{NOR} = **10 ns**, t_{NOT} = **5 ns**, t_{setup} = **5 ns**, and t_{C2Q} = **30 ns**.



(A) Calculate the **minimum clock period** that will allow the circuit to function correctly. Make sure to *include units*. [3 pts]

$t_{\text{period}} \geq$ **75 ns**

The critical path is shown above in red.
We need $t_{\text{C2Q}} + t_{\text{XOR}} + t_{\text{NOR}} \leq t_{\text{period}} - t_{\text{setup}}$.
Then $t_{\text{period}} \geq 30 + 30 + 10 + 5 = 75$ ns.

[2 pt] Longest path indicated
[1.5 pt] 2nd longest path indicated
[1 pt] 3rd longest path indicated
[1 pt] Computation includes setup time

(B) Calculate the **maximum hold time** that will allow the circuit to function correctly. Make sure to *include units*. [3 pts]

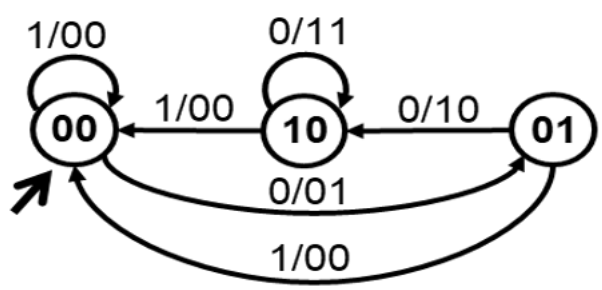
$t_{\text{hold}} \leq$ **50 ns**

The shortest path to a register input is shown above in blue.
We need $t_{\text{C2Q}} + t_{\text{NOR}} + t_{\text{NOR}} \geq t_{\text{hold}}$.
Then $t_{\text{hold}} \leq 30 + 10 + 10 = 50$ ns.

[2 pt] Shortest path indicated
[1.5 pt] 2nd shortest path indicated
[1 pt] 3rd shortest path indicated
[1 pt] Computation includes setup time

Question 2: Finite State Machine Implementation [10 pts]

(A) Fill in the provided truth table based on the FSM shown. [2 pts]

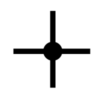


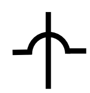
PS_1	PS_0	In	NS_1	NS_0	Out_1	Out_0
0	0	0	0	1	0	1
0	0	1	0	0	0	0
0	1	0	1	0	1	0
0	1	1	0	0	0	0
1	0	0	1	0	1	1
1	0	1	0	0	0	0
1	1	0	X	X	X	X
1	1	1	X	X	X	X

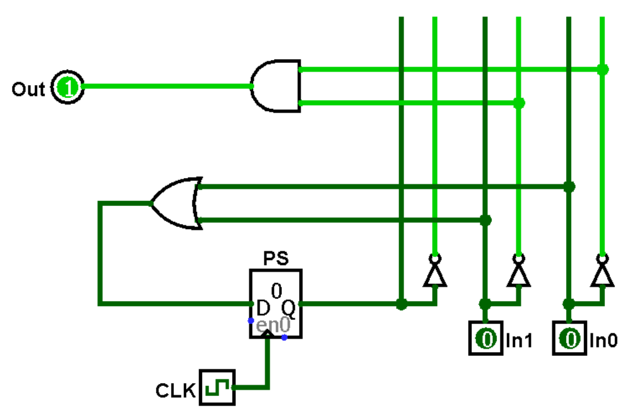
[0.25 pt each] Correct term

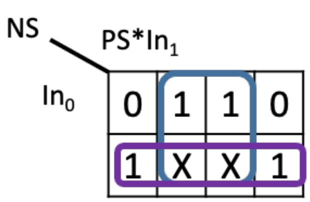
(B) Complete the circuit diagram below using *minimal logic* based on the truth table shown below. You are welcome to use 2- and 3-input logic gates. [8 pts]

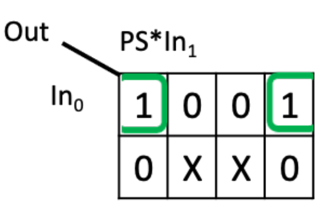
PS	In_1	In_0	NS	Out
0	0	0	0	1
0	0	1	1	0
0	1	0	1	0
0	1	1	X	X
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	X	X

Wire connection: 

Wire crossing: 







$NS = In_1 + In_0$

$Out = \overline{In_1} * \overline{In_0}$

For both NS and Out:

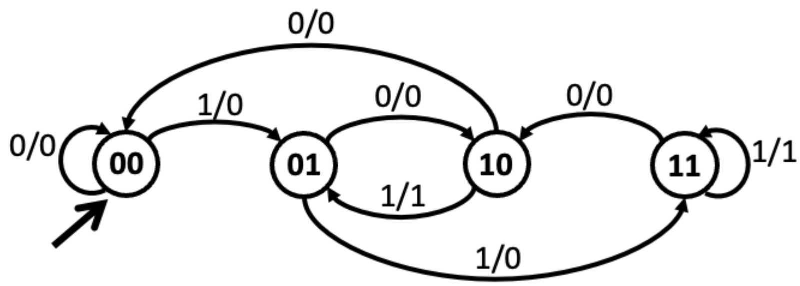
[1 pt] Truth table transcribed to K-Map correctly

[2 pt] Correct simplification

[1 pt] Correct circuit diagram

Question 3: Finite State Machine Design [10 pts]

For this problem, consider the FSM below:



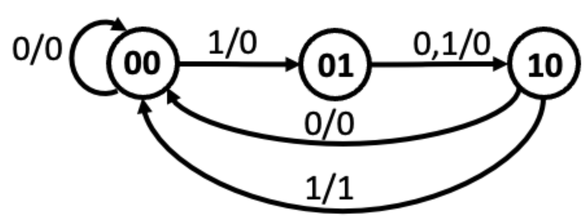
(A) What two 3-input sequences does this FSM "recognize" (*i.e.* when it outputs a `1`)? [2 pts]

101	111

(B) Complete the testbench `initial` block to *thoroughly* test the state diagram. Even though they may be unnecessary, please fill in all blanks. You are welcome to fill out the Verilog comments to help you keep track of state, but these will not be graded. [5 pts]

```
initial begin
                        In <= 1;        // state: 00
   @(posedge clk);      In <= 0;        // state: _01_
   @(posedge clk);      In <= 1;        // state: _10_
   @(posedge clk);      In <= __1__;    // state: _01_
   @(posedge clk);      In <= __1__;    // state: _11_
   @(posedge clk);      In <= __0__;    // state: _11_
   @(posedge clk);      In <= __0__;    // state: _10_
   @(posedge clk);      In <= __0__;    // state: _00_
   @(posedge clk);                      // state: _00__
   $stop();
end
```

(C) The FSM below attempts to recognize the same sequences. Does it recognize the same sequences? If it does, explain why. If it doesn't, list all of the issues with it. [3 pts]



[+1 pt] Does not recognize overlapping 111
[+1 pt] Does not recognize overlapping 101
[+1 pt] Does not have a reset signal