University of Washington – Computer Science & Engineering
Spring 2026 Instructor: Naomi Alterman 2026-05-19

CSE 369 QUIZ 2

Name: Sestina Solutiono

Student ID Number: _____

Please do not turn the page until 2:20.

Instructions

- This quiz contains **8** 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 closed book and closed notes.
- Please silence and put away all cell phones and other mobile or noise-making devices.
- Remove all hats, headphones, smart glasses, watches, and other digital wearables.
- You have 30 (+5) minutes to complete this quiz.

Advice

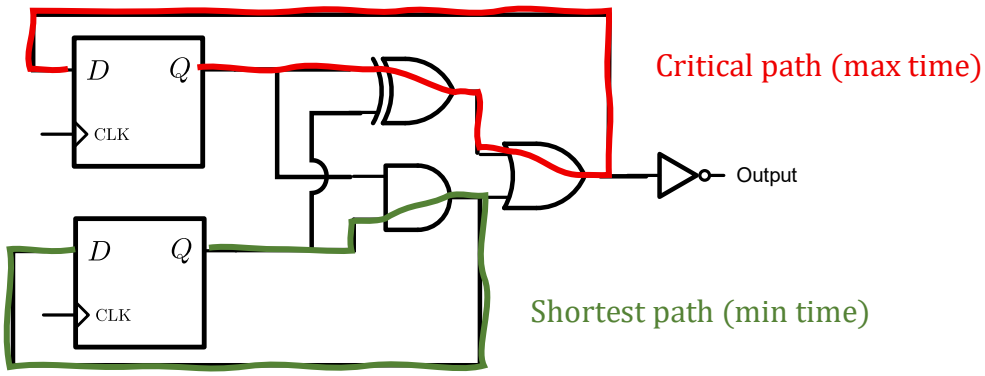
- Read questions carefully before starting. Read *all* questions first and start where you feel the most confident to maximize the use of your time.
- There may be partial credit for incomplete answers; please show your work.
- Relax. If you've been practicing, you got this. If you haven't, you'll learn something now.

Question	Points	Score
(1) SL & Timing	14	
(2) FSM Implementation	12	
(3) FSM Design	18	
Total:	**44**	

(This page left intentionally blank)

Question 1: Sequential Logic & Timing [14 pts]

Consider the following circuit diagram with t_{setup} = **9 ns**, t_{C2Q} = **3 ns**, t_{NOT} = **3 ns**, t_{AND} = t_{OR} = **6 ns** and t_{XOR} = **9 ns**. Assume there are no timing requirements for the Output port. Assume all CLK inputs are connected to the same clock source.



(A) Calculate the **minimum clock period** that will allow the circuit to function correctly. [8 pts]

27 ns

$$t_{max} \leq t_{period} - t_{setup}$$
$$t_{C2Q} + t_{XOR} + t_{OR} \leq t_{period} - t_{setup}$$
$$3 + 9 + 6 \leq t_{period} - 9$$
$$27 \leq t_{period}$$

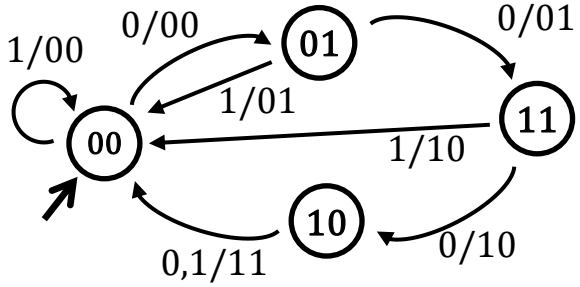
(B) Calculate the **maximum hold time** (t_{hold}) that will allow the circuit to function correctly. [6 pts]

9 ns

$$t_{min} \geq t_{hold}$$
$$t_{C2Q} + t_{AND} \geq t_{hold}$$
$$3 + 6 \geq t_{hold}$$

Question 2: Finite State Machine Implementation [12 pts]

(A) Fill in the blanks of the provided truth table based on the FSM shown. [6 pts]



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

(B) Complete the circuit diagram below using *minimal logic* based on the truth table shown below. **Use only 2-input logic gates.** [6 pts]

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

Wire connection:

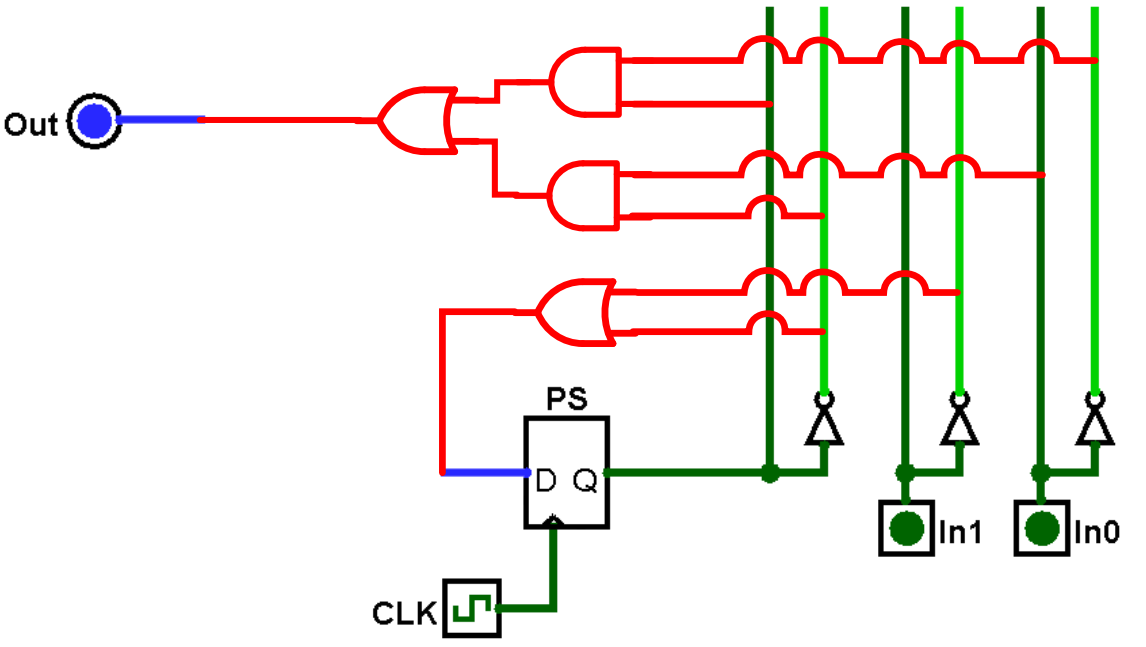
Wire crossing:

NS

PS\In1,In0	00	01	11	10
0	1	1	1	1
1	0	0	1	1

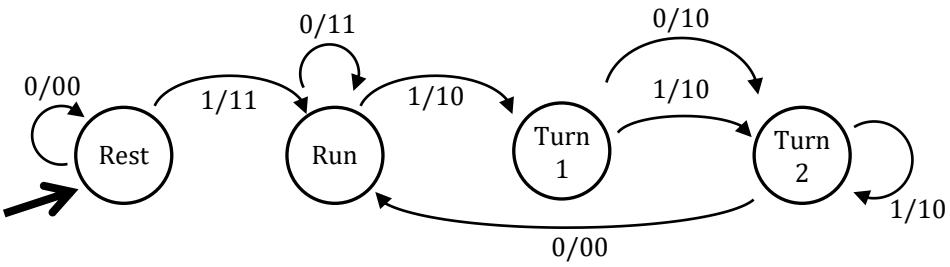
Out

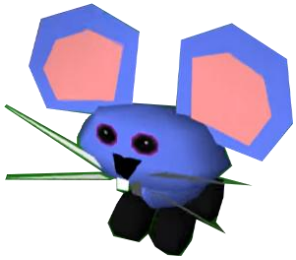
PS\In1,In0	00	01	11	10
0	0	1	1	0
1	1	0	0	1



Question 3: Finite State Machine Design and Testing [18 pts]

We're building a little robot mouse toy for Danni the Cat to play with. Its behavior is controlled by an FSM. The input bit "HIT" represents when it's being hit by her paw. The two output bits {"WHEEL_L", "WHEEL_R"} control the two wheels the mouse rides around on.





(A) Complete the test bench initial block to *thoroughly* test the FSM by filling in the blanks. You may not need all blanks (just leave them empty). You may fill out the comments to track the state, but these won't be graded. [8 pts]

```
initial begin
                  reset = 1; In = 0;
      @(posedge clk);   reset = 0;  // present-state: rest
      @(posedge clk);   In = 1;     // present-state: rest
      @(posedge clk);   In = 0;     // present-state: run
      @(posedge clk);   In = 1;     // present-state: run
      @(posedge clk);   In = 0;     // present-state: turn 1
      @(posedge clk);   In = 0;     // present-state: turn 2
      @(posedge clk);   In = 1;     // present-state: run
      @(posedge clk);   In = 1;     // present-state: turn 1
      @(posedge clk);   In = 1;     // present-state: turn 2
      @(posedge clk);   In = ____;  // present-state: turn 2
      @(posedge clk);   In = ____;  // present-state: ____
      @(posedge clk);   In = ____;  // present-state: ____
      @(posedge clk);   In = ____;  // present-state: ____
      @(posedge clk);   In = ____;  // present-state: ____
      @(posedge clk);   In = ____;  // present-state: ____
      @(posedge clk);   In = ____;  // present-state: ____
      $stop();
end
```

(B) What is the minimum number of flip flops we need to store the state of this FSM? [1 pt]

2

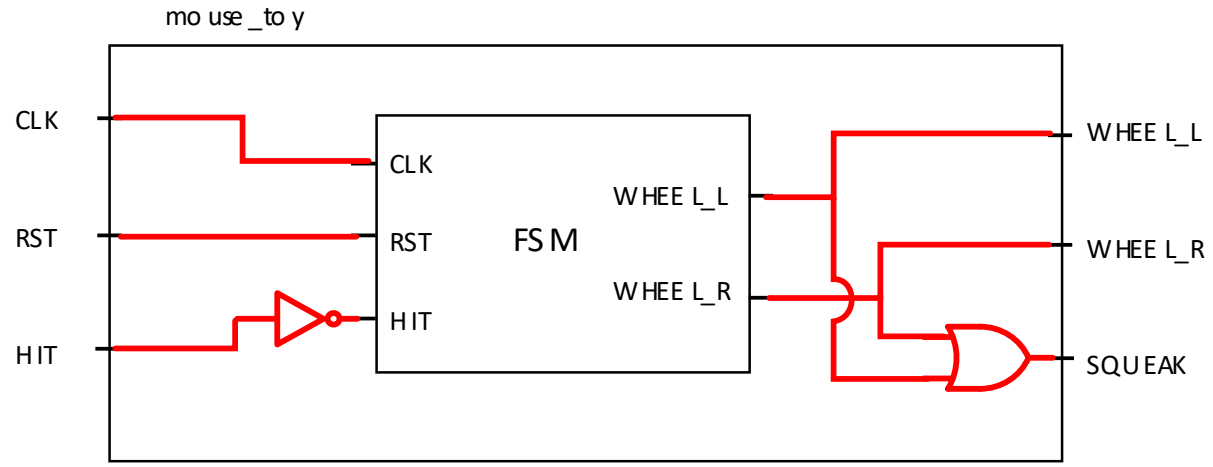
(there are four states, and we need 2 bits to encode four unique integers as state codes)

(C) Which state cannot be returned to once we've transitioned out of it? What are the implications this has for testing it? [3 pt]

"Rest"

We need to either test the behavior of all self-transitions (arrows from that state right back to that state) before we leave it, or reset the FSM to get back to "Rest" for each of those self-transition behavior we're testing. Both are acceptable answers.

(D) Complete the block diagram below to show how our FSM connects to the other toy components. The external "hit" sensor is **active low** and the wheel motors are **active high**. There is also a speaker that squeaks when powered (also **active high**). The mouse should squeak whenever it is in motion. [6 pt]



(Use this page for scratch work)

(Use this page for scratch work)