University of Washington – Computer Science & Engineering
Spring 2023 Instructor: Justin Hsia 2023-05-30

CSE 369 QUIZ 3

Name: _Molly_Model_____________________

Student ID Number: _1234567________

Please do not turn the page until 1:40.

Instructions

- This quiz contains 4 pages, including this cover page.
- Show scratch work for partial credit, but put your final answers in the boxes and blanks provided.
- 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, and watches.
- You have 60 (+10) 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) Counters	12	12
(2) Routing Elements	10	10
(3) Shift Registers	9	9
Total:	**31**	**31**

Question 1: Counters [12 pts]

Implement the 3-bit "odd" counter using a *minimal number of 2-input logic gates*. It goes through the state sequence: **001 → 011 → 101 → 111** → 001 → ⋯

(A) Complete the truth table. [3 pts]

PS_2	PS_1	PS_0	NS_2	NS_1	NS_0
0	0	0	X	X	X
0	0	1	0	1	1
0	1	0	X	X	X
0	1	1	1	0	1
1	0	0	X	X	X
1	0	1	1	1	1
1	1	0	X	X	X
1	1	1	0	0	1

(B) Solve for the minimal logic. [6 pts]
(K-maps are optional,
but will allow for more partial credit.)

$PS_1 PS_0$

NS_2	00	01	11	10
PS_2 0	X	0	1	X
1	X	1	0	X

$NS_2 = PS_2\overline{PS_1} + \overline{PS_2}PS_1$

$= PS_2 \oplus PS_1$

NS_1	00	01	11	10
PS_2 0	X	1	0	X
1	X	1	0	X

$NS_1 = \overline{PS_1}$

NS_0	00	01	11	10
PS_2 0	X	1	1	X
1	X	1	1	X

$NS_0 = 1$

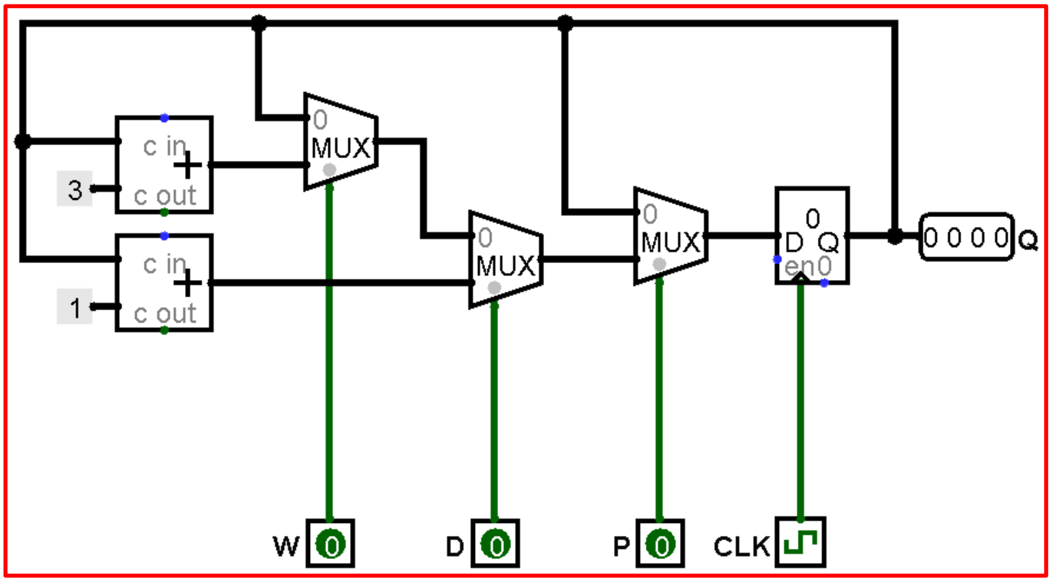
(C) *Briefly* describe how could we simplify the hardware circuit even further. Hint: draw out the current minimal logic circuit and look at what hardware might be unnecessary. [3 pts]

Because we are only cycling through odd numbers, PS_0 is always 1, which does not need to be stored. We could remove one DFF (one bit of state) and instead concatenate a 1 as the least significant bit to the remaining two bits of state.

Question 2: Routing Elements [10 pts]

We are creating a sequential circuit with 1-bit inputs `P` (played), `W` (win), and `D` (draw/tie) and n-bit output `Q` to accumulate the standings for a soccer/futbol team. When a team plays a game (P), they accumulate 3 points for a win ($W\overline{D}$), 1 point for a draw (D) and 0 points for a loss ($\overline{W}\overline{D}$); otherwise ($\overline{P}$), the point total remains the same.

(A) Draw out the circuit below. You can freely use registers, constants, 2:1 MUXes, and adders. Make sure you label the corresponding selector bits for ports of routing elements. [6 pts]



(B) Now assume that we instantiate our circuit for a team that starts with `Q` = `0` points. Based on the SystemVerilog testbench below, what will the team's final record and points total be? [4 pts]

```
initial begin
   integer i;
   initial begin
      for (i = 0; i < 8; i++) begin
         {W, D, P} = i; @(posedge clk);
      end
      @(posedge clk);  $stop();
end
```

Wins: 1	Draws: 3	Losses: 1
		Points: 6

This testbench runs through all combinations 3'b000 to 3'b111 in order. Every other is not played ($\overline{P}$) and can be ignored. Of the games played (001, 011, 101, 111), these translate to Loss, Draw, Win, Draw. There is one last clock cycle after the for-loop that remains at the input combination 111 → Draw. Points = 3*Wins + Draws.

Question 3: Shift Registers [9 pts]

In Lab 7, we created a 9-bit linear feedback shift register (LFSR) to generate "randomized" opponent behavior. Let's explore this idea of randomness a bit further by comparing the "optimal" 9-bit and 10-bit LFSRs, as given by the chart from Lab 7.

(A) Circle one: Which LFSR has the longer maximal state sequence: **9-bit / 10-bit.** [1 pt]

(B) The goal in Lab 7 was to generate a 9-bit random number (0–511). Briefly describe how you could get a 9-bit random number from the 10-bit LFSR. How much hardware is involved in this process? [4 pts]

9-bit number: Can grab any 9 of the 10 bits in any well-defined order. Most simply, either take the upper 9 bits or the lower 9 bits.
Hardware: Just wires to route/reorder the bits.

(C) If you had to decide between using either the "optimal" 9-bit LFSR or the "optimal" 10-bit LFSR to produce a 9-bit random number sequence, which would you choose and why? [4 pt]

This is a somewhat open-ended question; the explanation matters much more than the choice.

Circle: 9-bit / 10-bit

Explain choice: Many possible explanations, including:

- 9-bit requires less hardware (one less DFF).
- 9-bit sequence doesn't produce duplicates during its cycle, while the 10-bit sequence will repeat each value twice during its (roughly twice as long) cycle – this could actually be used to argue for either one being more "random" than the other.
- 9-bit is more intuitive and uses more natural connections.