University of Washington – Computer Science & Engineering

Winter 2017 Instructor: Justin Hsia 2017-03-13

CSE 369 QUIZ 3

Name: _Perry_Perfect_____

UWNetID: _1234567_____

Please do not turn the page until 10:30.

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 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 35 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) Shift Registers	11	11
(3) Routing Elements	9	9
Total:	**32**	**32**

Question 1: Counters [12 pts]

Implement a counter that goes through the following state sequence: **000** → **111** → **110** → **101** → **001** → 000 → ... using a *minimal number of 2-input logic gates.*

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

PS_2PS_1 / PS_0

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

$NS_2 = \overline{PS_0} + PS_1$

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

$NS_1 = PS_0 \text{ xnor } PS_1$

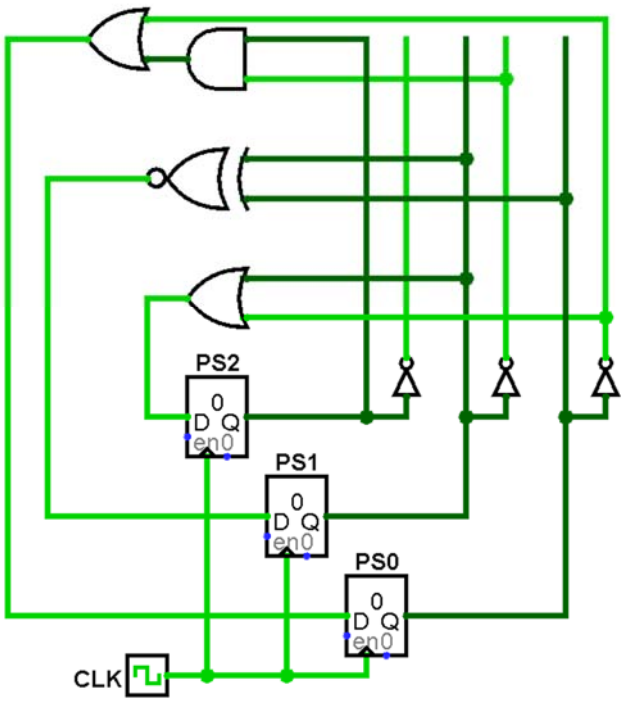
most credit given for $NS_1 = PS_0PS_1 + \overline{PS_0}\,\overline{PS_2}$

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

$NS_0 = \overline{PS_0} + PS_2\overline{PS_1}$

Wire connection:

Wire crossing:



Question 2: Shift Registers [11 pts]

We have a 4-bit linear feedback shift register (LFSR) that goes through the following state sequence: 0000 → 0001 → 0010 → 0101 → 1011 → ⋯

(A) Circle one: This LFSR is shifting bits to the **LEFT** / **RIGHT**. [1 pt]

(B) The bit that is shifted in is a function of two of the LFSR bits. We number the bits starting from 0 increasing from right to left (like standard binary). What is the name of the gate being used and which two bits are its inputs? [6 pt]

Gate:	**XNOR**
Bits:	__**2**__ and __**0**__

From the first transition, we see that we are shifting left (shifting in on the right) and that gate(0,0) = 1. The second transition tells us that gate(0,1) = 0 and that one of the bits is bit 0 (far right). All of the named two-input gates are associative, so gate(1,0) = gate(0,1) = 0. The third transition narrows the options for the 2nd tap to bit 3 or bit 2. The fourth transition indicates that gate(1,1) = 1 and the 2nd tap must be bit 2. Looking at the truth table, the gate is actually XNOR.

(C) Complete the Verilog code below to implement this LFSR. Feel free to use "**state[i] ? state[j]**" to indicate the correct answer to part B. [4 pt]

```
module LFSR (state, shift, reset, clk);
  output reg [3:0] state;
  input      shift, reset, clk;

  always @(posedge clk) begin

    if ( _reset_ )

      state <= _4'b0000_____________________________;

    else if ( _shift_ )

      state <= _{ state[2:0], state[i] ? state[j] };
  end

endmodule
```

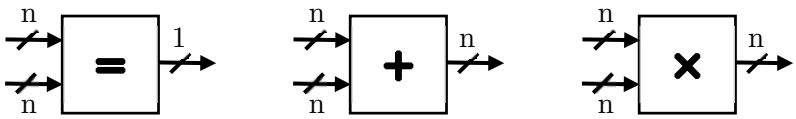
Question 3: Routing Elements [9 pts]

Implement a circuit that computes the **factorial function `n! = n*(n-1)!`**. Note that it will take `n` clock cycles to compute `n!` and we will let it run infinitely (*no stop condition*).

Note 1: Both registers (after a `Reset`) start with value 0. Make sure that your circuit doesn't get stuck at the value 0. Hint: what's the title of this problem?

Note 2: Make sure that your `n` and `n!` bus values line up properly (other than 0! and 0).

Assume you can freely use gates and routing elements discussed in class plus the constants **`0`** and **`1`** and the following logic blocks:



Other working alternatives exist:

- Instead of the MUX, use an adder with the output of the comparator (though in reality we would need a zero-extender to match bit widths).