University of Washington – Computer Science & Engineering
Spring 2020 Instructor: Clarice Larson 2020-06-02

CSE 369 QUIZ 3

Name: ______________________________

UWNetID: ______________________________

Please do not turn the page until 11:30.

Instructions

- This quiz contains 4 pages, including this cover page.
- Show scratch work for partial credit, but put a box around final answers.
- 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 40 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	
(2) Shift Registers	9	
(3) Polynomials	11	
Total:	**32**	

Question 1: Counters [12 pts]

Implement a counter that goes through the following state sequence: **000** → **010** → **011** → **101** → 000 → ... using a *minimal number of 2-input logic gates*.

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

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

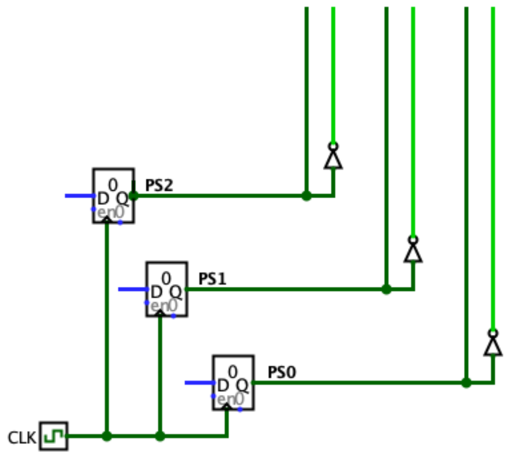
(B) Complete the K-maps below and find the *minimum sum-of-products solutions*. [6 pts]

NS_2	00	01	11	10
0				
1				

NS_1	00	01	11	10
0				
1				

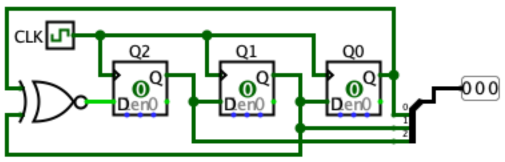
NS_0	00	01	11	10
0				
1				

(C) Draw your minimal logic circuit. [3 pts]



Question 2: Shift Registers [9 pts]

We are using a 3-bit LFSR as a pseudo-random number generator, but we only have a 2-input XNOR gate available. Assume we start in state 000.



(A) Draw out the full state transition diagram (i.e. include ALL states) for this LFSR below: [4 pts]

(B) What is/are the sink state(s) of this LFSR? [1 pt]

Sinks(s):

(C) Complete the Verilog implementation below. [4 pts]

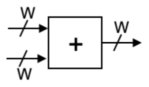
```
module LFSR (Q, enable, reset, clk);
   input  logic enable, reset, clk;
   output logic [2:0] Q;

   always_ff @(posedge clk)
   if (reset)
      Q <= ______________;
   else if (enable)
      Q <= { ______________, ______________, ______________ };

endmodule
```

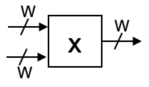
Question 3: Polynomials [11 pts]

A third degree polynomial or a cubic function is a function of the form $F_3 = a_0 + a_1x + a_2x^2 + a_3x^3$.
Hint: Horner's method rewrites a polynomial as:

$$F_n = a_0 + x(a_1 + x(a_2 + x(a_3 + \cdot\cdot\cdot + x(a_{n-1} + xa_n)\cdot\cdot\cdot)))$$

The coefficients and x are n bits wide. Assume $t_{\text{ADD}} = 50$ ps, $t_{\text{MULT}} = 100$ ps, $t_{\text{C2Q}} = 30$ ps. The following logic blocks will be needed (w is the width of the bus):



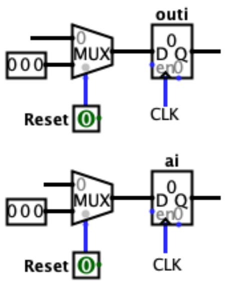


(A) Implement a *minimal* combinational logic circuit using only adders and multipliers that computes a cubic polynomial of n bit numbers. [6 pts]

(B) What is the delay through your combinational logic circuit? [1pt]

_____________ ps

(C) Implement a *minimal* sequential logic circuit that computes a cubic polynomial. Assume that an n bit coefficient is available every clock cycle and that your registers start at 0. You can specify the order that the coefficients arrive. Again, you may only use adders and multipliers in addition to the 2 registers shown. Note: the 2 registers have a reset line to set them to 0. [3 pts]



(D) What is the delay through the critical path of your sequential logic circuit? [1pt]

_____________ ps