University of Washington – Computer Science & Engineering

Spring 2017 Instructor: Justin Hsia 2017-05-30

CSE 369 QUIZ 3

Name: _Perry_Perfect_____

UWNetID: _perfect_____

Please do not turn the page until 10:40.

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 35 minutes to complete this quiz.

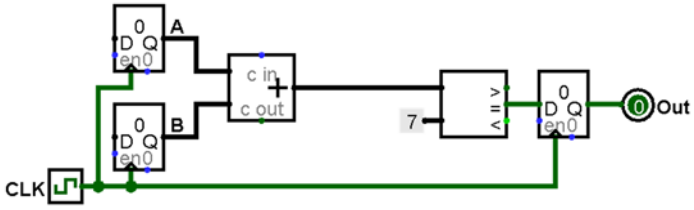
Advice

- 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) Timing Revisited	10	10
(2) Building Blocks	12	12
(3) Shift Registers	10	10
Total:	**32**	**32**

Question 1: Timing Revisited [10 pts]

Consider the following circuit diagram with: t_{setup} = **40 ps**, t_{hold} = **10 ps**, t_{C2Q} = **60 ps**, t_{ADD} = **80 ps**, and t_{COMP} = **100 ps**. Fill in your answers in the boxes below, ***including units***.



(A) What is the minimum clock period that will ensure proper behavior? [2 pt]

$t_{C2Q} + t_{ADD} + t_{COMP} \leq t_{period} - t_{setup}$

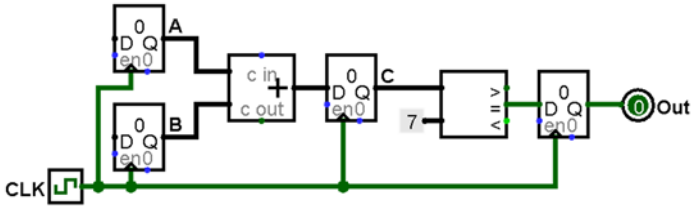
280 ps

(B) For a particular set of inputs A_i and B_i, how long does it take to compute the associated output Out_i? Measure from the moment A and B update to the moment Out updates. You may use the variable t_{period} (answer to part A) in your answer. [2 pt]

A, B update t_{C2Q} after one clock trigger, Out updates t_{C2Q} after the next clock trigger

t_{period} = 280 ps

Now we add a register between the adder and comparator:



(C) What is the new minimum clock period that will ensure proper behavior? [2 pt]

$t_{C2Q} + t_{COMP} \leq t_{period} - t_{setup}$

200 ps

(D) For a particular set of inputs A_i and B_i, how long does it now take to compute the associated output Out_i? Measure from the moment A and B update to the moment Out updates. You may use the variable t_{period} (answer to part C) in your answer. [2 pt]

A, B update t_{C2Q} after one clock trigger, C updates t_{C2Q} after the next clock trigger, Out updates t_{C2Q} after the next clock trigger.

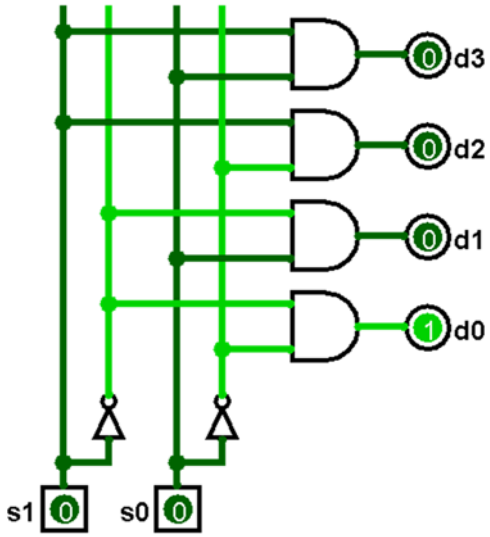
$2t_{period}$ = 400 ps

(E) Does adding this extra register help or hurt? *Explain briefly.* [2 pt]

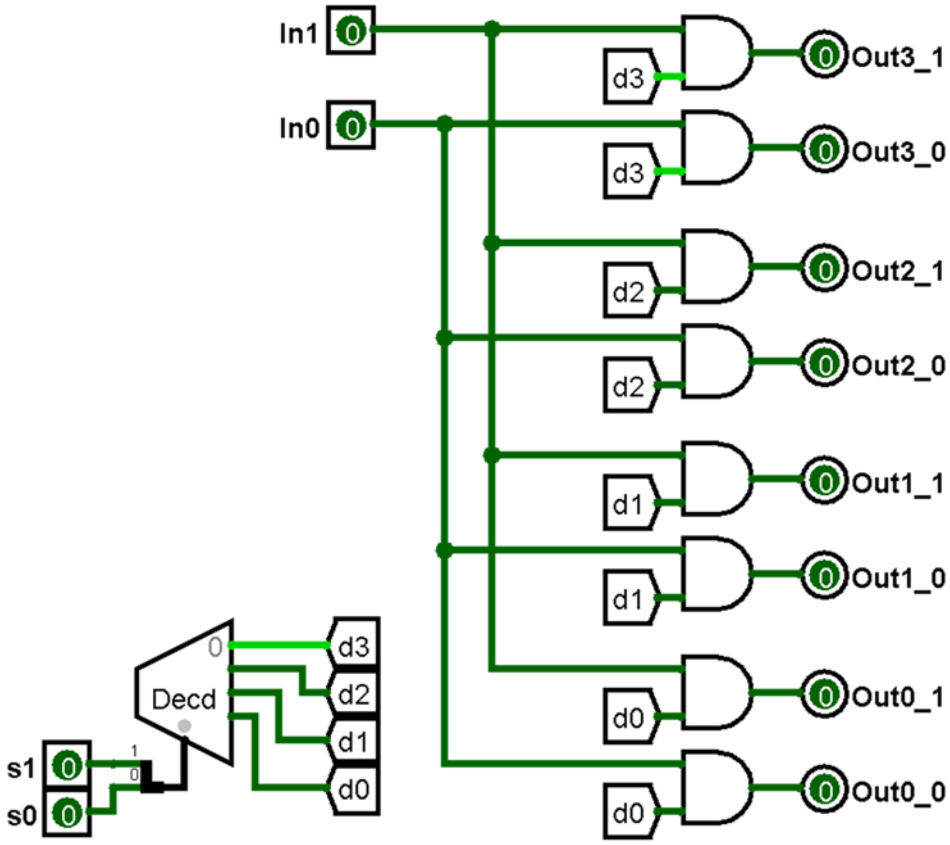
Help: Faster computation rate (shorter clock period)
Hurt: Longer individual computation time

Question 2: Building Blocks [12 pts]

(A) Implement a 2:4 binary decoder below using only NOT, AND, and OR gates. The 2 input bits (s_1 and s_0) set the corresponding output bit (one of d_0 through d_3) high. [4 pt]



(B) Implement a 2-bit, 1-to-4 DEMUX below using only NOT, AND, and OR gates. This passes the 2 input bits onto 1 of 4 sets of outputs (OutN_x). Assume you have a working 2:4 binary decoder and *write in the signals* d_0, d_1, d_2, *and* d_3 *where needed.* [8 pt]



Question 3: Shift Registers [10 pts]

We have a 4-bit linear feedback shift register (LFSR) that goes through the following state sequence: 0000 → 1000 → 1100 → 0110 → 0011 → 0001 → 1000 → ⋯

(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? **Hint:** all named gates are associative (*i.e.* F(0,1) = F(1,0)). [6 pt]

1st transition tells us we are shifting right and that F(0,0) = 1. 2nd transition tells us that either F(0,1) = 1 with one tap being bit 3 or neither tap is bit 3. 3rd transition tells us that either NAND of bits 3 & 2 or F(0,1) = 0 with one tap being bit 2. 4th transition eliminates NAND possibility, leaving us with either NOR of bits 2 & 1 or NOR/XNOR of bits 2 & 0. No extra info from 5th transition. 6th transition confirms NOR of bits 2 & 1.

Gate:	**NOR**
Bits:	__**2**__ and __**1**__

(C) The Verilog code below is supposed to implement a 4-bit parallel-in, serial-out (PISO) register. The output is the highest state bit. It will shift in the lowest bit of the input bus. Find errors in the code and rewrite the offending lines in the boxes. [3 pt]

```
module PISO (out, in, load, shift, clk);
  output       out;
  input  [3:0] in;
  input        load, shift, clk;
  wire   [3:0] state;

  always_ff @(posedge clk) begin
    if ( load )
      state  = in;
    else if ( shift )
      state <= {in[3], state[2:0]};
  end

  assign out = state[3];

endmodule
```

1) `reg   [3:0] state;`
2) `state <= in;`
3) `state <= {state[2:0], in[0]};`