EECS 270 *Midterm Exam*

Spring 2013

Name: ______________________________ unique name: ___________

Sign the honor code:

I have neither given nor received aid on this exam nor observed anyone else doing so.

____

Scores:

Problem #	Points
1	/12
2	/12
3	/10
4	/8
5	/8
6	/16
7	/13
8	/9
9	/11
Total	**/100**

NOTES:

1. Open book and Open notes
2. There are 12 pages total. Count them to be sure you have them all.
3. No electronic devices of any kind are allowed on this exam.
4. This exam is fairly long: *don't spend too much time on any one problem*.
5. You have about 120 minutes for the exam.
6. Some questions may be more difficult than others. You may want to skip around.
7. **Be sure to show work and explain what you've done when asked to do so.** Even if work isn't requested it is a good idea to provide your work as it will help with partial credit.

1) Fill in each blank or circle the best answer.**[12 points, -2 per wrong or blank answer, min 0]**

a) The 6-bit 2's complement number representation of -7 is 11001.

b) 11001, when treated as a 5-bit signed-magnitude number, has a decimal representation of −9.

c) The range of representation for a 6-bit 2's complement number is from 2^5-1 to -2^5.

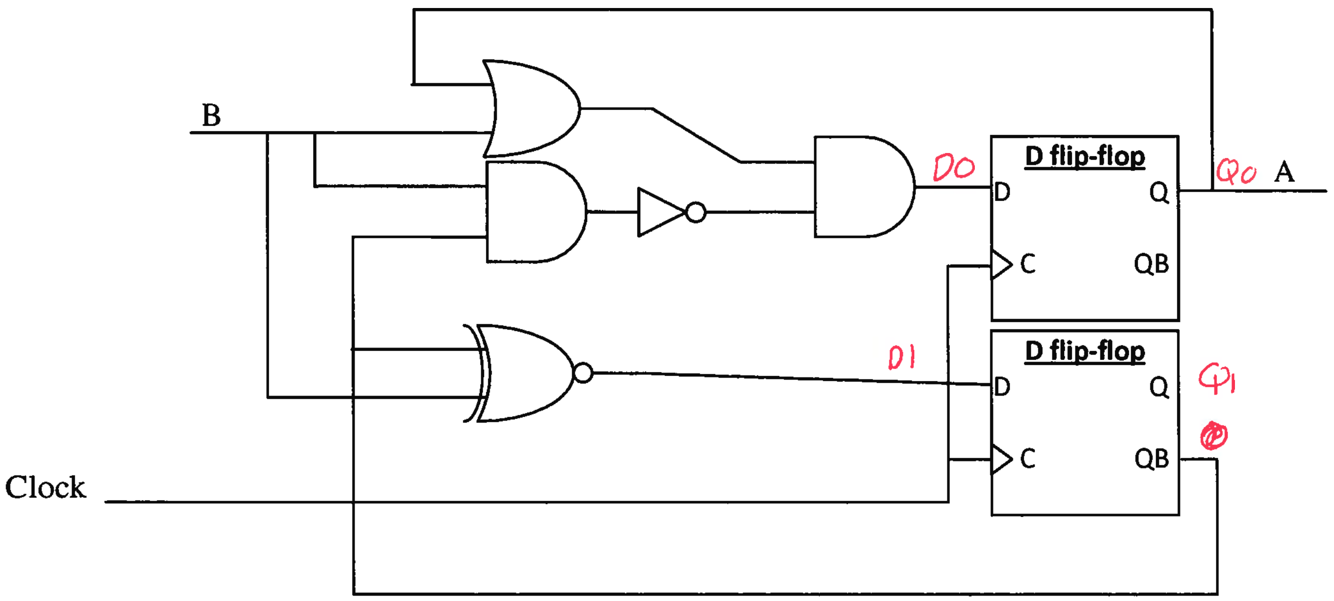
d) Consider a memory device that has 2048 2^{11} addresses each 32 2^5 bits in size. If this was made out of a square memory (equal number of rows and columns in the memory device) the row decoder would have 8 inputs while the column MUX would have 3 selection bits.

e) A <u>sum-of-products</u> representation of !(A*!B) is $\bar{A}+B$

f) If you were to represent A+B+C+!D using canonical sum-of-products, there would be 15 minterms.

g) If you were to represent A ⊕ B ⊕ C in canonical product-of-sums form, there would be ~~8~~ 4 maxterms.

h) A 4-bit 8 to 1 MUX would require 3 select lines.



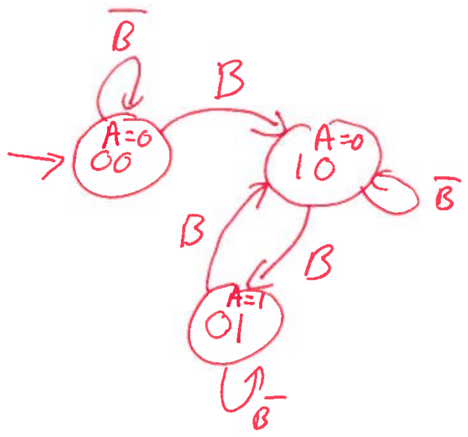
2) Draw the state transition diagram which is implemented by the above circuit. You should assume the initial state is when both flip-flops are "0". You should only include states that can be reached from the initial state. *Clearly show your work.* **[12 points]**

$D_0 = (Q_0 + B) \cdot \overline{B \bullet \overline{Q}_1} = (Q_0 + B) \cdot (\overline{B} + Q_1)$

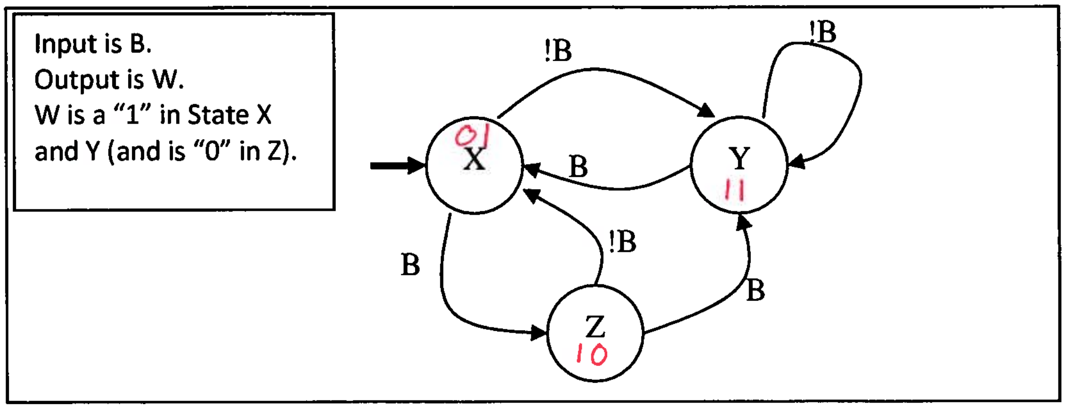
$D_1 = (\overline{B \oplus \overline{Q}_1}) = B \oplus Q_1$

$A = Q_0$

Q_1	Q_0	B	D_1	D_0
0	0	0	0	0
0	0	1	1	0
0	1	0	0	1
0	1	1	1	0
1	0	0	1	0
1	0	1	0	1
1	1	0	1	1
1	1	1	0	1



3) Design a state machine which implements the following state transition diagram. Assign state bits S[1:0] as 01 for state X, 11 for state Y, and 10 for state Z. You are to assume that you will never reach the state S[1:0]=00, so you don't care what happens in that case. You must show your work to get <u>any</u> credit! *<u>You only need to compute the next state and output logic, you don't need to draw the gates or flip-flops!</u>* Place your answer where shown, ***<u>all answers must be in sum-of-products form</u>***. **[10 points]**


Input is B.
Output is W.
W is a "1" in State X and Y (and is "0" in Z).

S1	S0	B	D1	D0
0	0	0	X	X
0	0	1	X	X
0	1	0	1	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	1
1	1	0	1	1
1	1	1	0	1

(Be sure all are in sum-of-products form!)

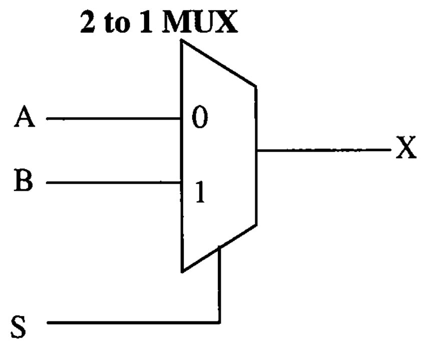
NS1= ~~$\overline{S1} + S0B$~~ $\overline{S1} + S_0\overline{B} + \overline{S_0}B$

NS0= $S1 + \overline{S0} + \overline{B}$

W= $S0$

4) Implement a 1-bit 2 to 1 MUX using only 2-input NOR gates. For full credit, use 4 or fewer NOR gates. Use the inputs and outputs A, B, S and X as shown below. As always, you can freely use Vcc and ground. **[8 points, half credit for using more than 4 gates]**

$X = A \cdot \overline{S} + B \cdot S$

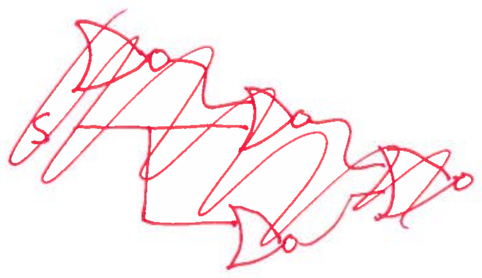


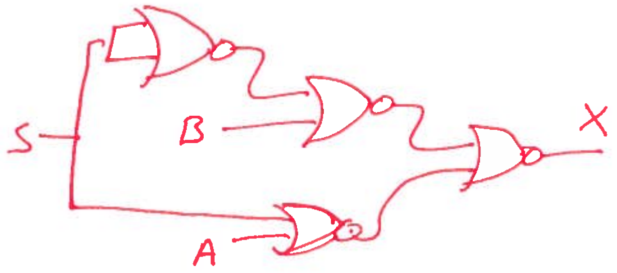
A B S	X
0 0 0	0
0 0 1	0
0 1 0	0
0 1 1	1
1 0 0	1
1 0 1	0
1 1 0	1
1 1 1	1

$(A+S)$

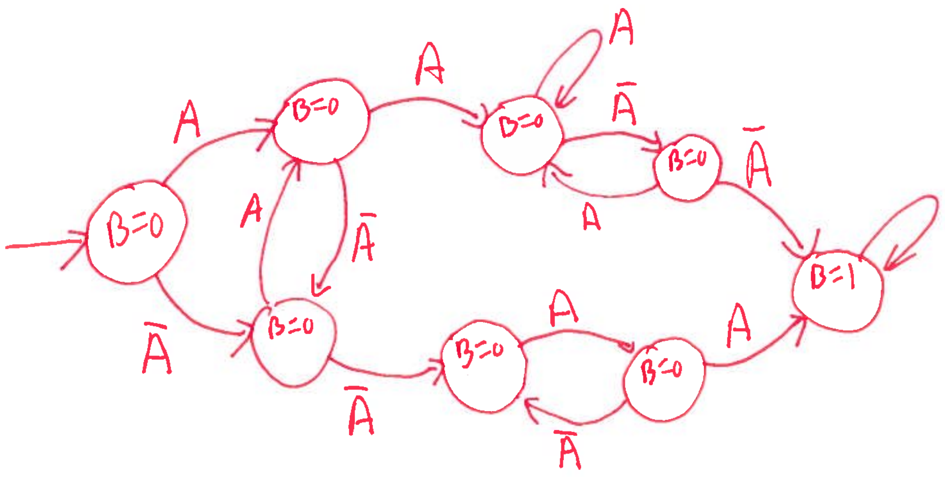
$(B+\overline{S})$

$$(A+S)\cdot(B+\overline{S}) = \overline{(\overline{A+S}) + (\overline{B+\overline{S}})}$$



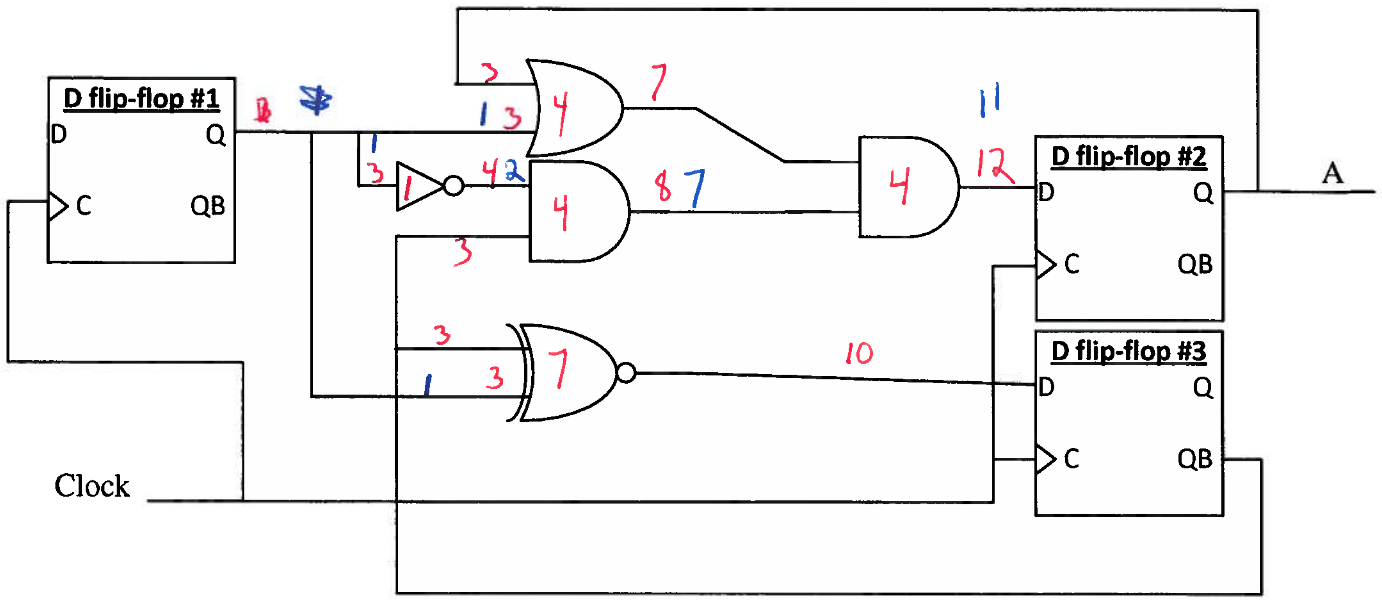


5) Design a state transition diagram that has one input (A) and one output (B). The output should go high and stay high only when A has been high for two cycles in a row ***and*** low for two cycles in a row in any order. So inputs of 0011 and 1100 would both cause B to go (and stay) high, as would 00101011 (for example). You are to use as few states as possible.
[8 points, 3 are for a minimal answer]



	Min	Max
OR/AND	2ns	4ns
NOT	1ns	1ns
XNOR	2ns	7ns

DFF:		Min	Max
	Clock to Q	2ns	3ns
	Set-up time	4 ns	
	Hold time	??? ns	



6) Answer the following questions **[16 points]**

a) In order for this circuit to work correctly, what range of values that would be acceptable for the hold time *requirement* of the D flip-flops? Assume the only options range from 1ns to 10ns. Clearly show your work. [5]

Smallest 1ns **Largest** 4ns

~~2~~

Fastest path is Dff1 to XNOR

Clock to Q + min XNOR delay = 2ns + 2ns.

anything smaller also works!

b) What is the lowest clock period that could be safely used on this circuit? Clearly show your work. [5]

Max path for Comb logic is NOT + AND + OR = 9ns.

$\underbrace{\text{Clock to Q}^{\,3ns} + 9ns}_{12ns}$ + 4ns (setup) = $\boxed{16ns}$

Continued on next page!

c) Say that flip-flop #1's clock edge *always* happens 2ns before the clock-edge that goes to flip-flops #2 and #3. Would that change your answer to part a, part b, both or neither? Indicate which answers would change and what they would change to. **[6]**

a) For part a the ~~fastest path~~ input to DFF3 can change 2ns after DFF3's rising edge. Answer becomes 1ns to 2ns

b) For part b DFF #1 will have its Q change no later than 1ns after DFF2+3 have their rising edge. New math done in blue.

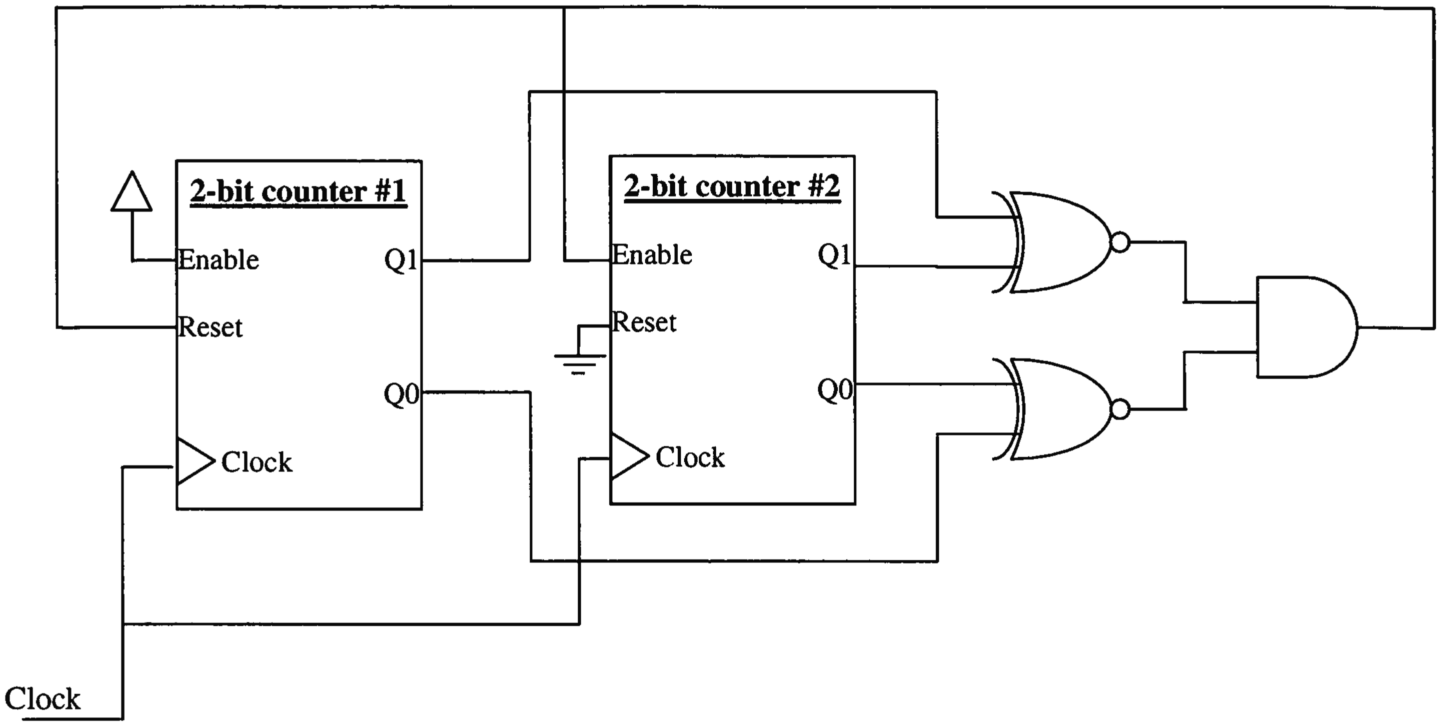
Gets 15ns

7) Verilog **[13 points]**

a) Complete the following module which is to implement a 2-bit modulo counter. The counter should count in order (0, 1, 2, 3, 0, 1, etc.) if reset is low and enable is high on each rising edge. If reset is high, the output should go to 0 on the next rising edge. If reset is low and enable is low, the output should be held. **[6]**

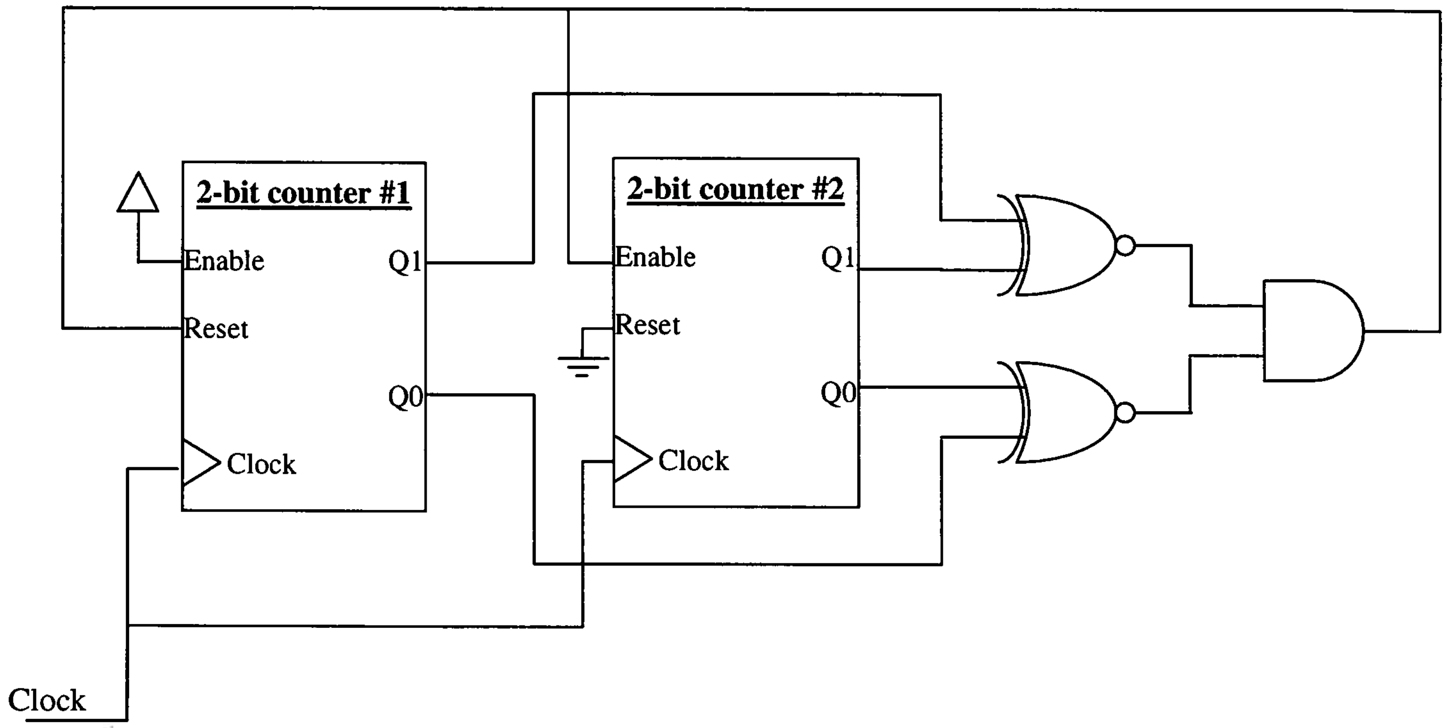
```
module counter_2(clock, reset, enable, count);
  input clock, reset, enable;
  output [1:0] count;

  reg [1:0] s, ns;

  //next state
  always @*
     if ~reset & enable)
          ns = s + 1;
     else
          ns = s;

  //state register
  always @(posedge clock)
     S <= ns;

  assign count = s;
endmodule
```

b) Using the counter you designed on the previous page, complete a Verilog module which implements the following circuit. The output should be the Q[1:0] of counter #2. [7]



```
module fun_counter(clock,Q)
  input         clock;
  output  [1:0] Q;

  wire equal;
  wire [1:0] c1, c2;
  wire zero, high;

  assign zero = 0;
  assign high = 1;
  assign equal = c1[1] ~^ c2[1] & c1[0] ~^ c2[0];
  counter_2 cn1(clock, equal, high, c1);
  counter_2 cn2(clock, low , equal , c2);
  assign Q = c2;
  endmodule
```

8) Consider the circuit diagram below where both counters are modulo counters and the resets are synchronous. Assuming both counters are initially zero, indicate the pattern that will appear out of counter #2. Write your answer using decimal numbers. So, for example, a possible (but wrong) answer would be "0, 1, 2, 1, 1, 3" which indicates that the output would be the pattern 012113012113012113 (etc.) repeating forever. Explain your answer. **[9 points]**



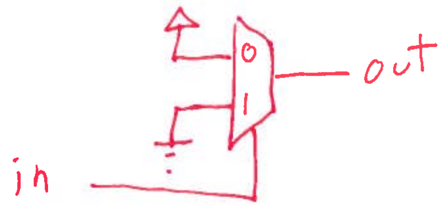
0, 1, 1, 2, 2, 2, 3, 3, 3, 3.

Every time counter 1 reaches counter 2's ~~[illegible]~~ value, counter 1 is reset and counter 2 increments.

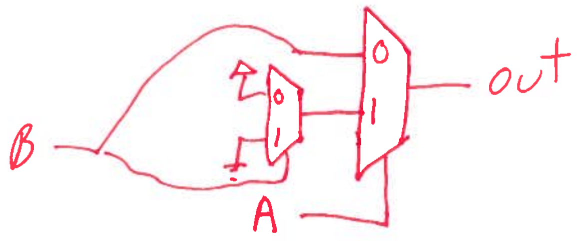
Pattern from counter #2: 0 1 1 2 2 2 3 3 3 3

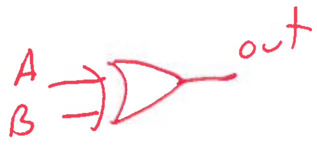
9) Using MUXes **[11 points]**

a) Design a NOT gate using only a 1-bit 2 to 1 MUX. **[2]**



b) Design a 2-input XOR gate using only a 1-bit 2 to 1 MUX. **[3]**





c) Design a D flip-flop using *only* 1-bit 2 to 1 MUXes. For full credit, you should use as few MUXes as possible. *No credit will be given to any answer that uses more than 4 MUXes.* **[6 points, 3 for being minimal]**