EECS 270 *Final Exam* Answer Key

Winter 2017

Name: _____ unique name: _____

Sign the honor code:

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

_____

NOTES:

1. ***The exam is open books and open notes. You may* not *use any device capable of communication (cell phones, calculators with wireless, etc.)***
2. You have about 120 minutes for the exam total.
3. Some questions may be harder than others. Manage your time wisely.
4. **Do NOT write anything you want graded on the back of any page.** If you need extra space for a problem, write it on the last (blank) page. If you do so:
   - On the page of the problem itself, make it clear your answer is on the last page.
   - Make it clear on the last page what problem(s) you are putting there.

1) Fill in each blank or circle the best answer. **[13 points, -2 per wrong or blank answer, min 0]**

a) The 5-bit 2's complement representation for -7 is _____11001____.

b) Find the minimal sum-of-products for A*B*C+A*B*C' _____A*B_____.

c) ***B*C'*D' / B*D / C*D' / B*C / A*C*D*** is an implicant of A*B'+A'*D. (error, no answer)

d) Write 22.1_4 as a binary number: _____1010.01_____

e) That A*(B+C) = (B+C)*A is an example of the ***associative / commutative / distributive / combining*** rule of logic.

f) If you were to represent (A+B)*(B+C) using canonical sum-of-products, there would be

_____5_____ minterms.

g) A 4-bit 16 to 1 MUX would require ***1 / 2 / 3 / 4 / 6 / 8*** select lines.

h) If you wanted a code which allowed the receiver to correct any two-bit error, that could would need to having a Hamming Distance of at least _____5_____

i) One significant difference between UART and SPI is that:
SPI has a shared clock and UART does not
SPI has a "clear to send" mechanism and UART does not
SPI transactions are of a fixed length for and UART transactions are not
SPI uses MUXes to share lines, while UART uses HiZ

2) Consider the following error correction scheme. Bits A, B, C, D and E are data while bits W, X, Y and Z are error correction where:

W=P(A,B,C)
X=P(B,C,D)
Y=P(A,C,E)

Assume the function P(…) is the parity of the bits. If we wish to do one bit of error correction, which bits should Z be the even one's parity of? Find the smallest set of bits that will work. If there is more than one possible solution that meets those requirements, list them all. **[5 points]**

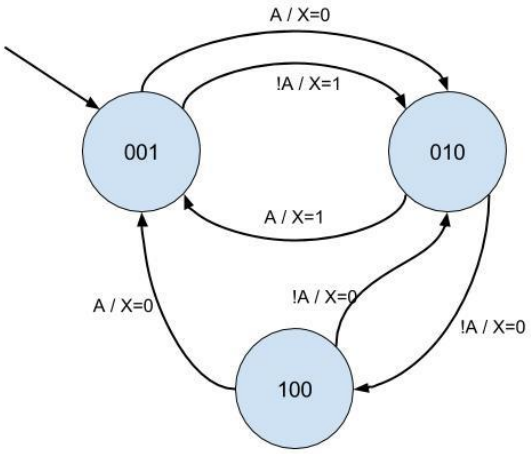
P(D, E)

3) Write a Verilog module called **Sat8bit.** It is to implement an eight-bit *saturating* up counter with enable and reset.
   - Inputs are **clock**, **reset** and **enable**
   - Output is the counter's value, **Q[7:0]**.

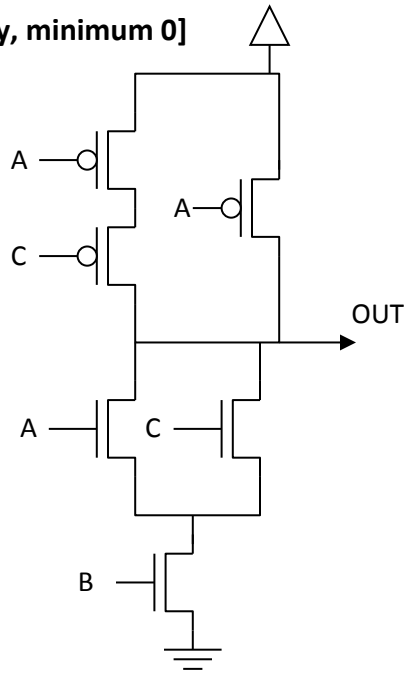
   The counter should set its value to 0 if **reset** is asserted on the positive edge of **clock**. If reset is a 0 and enable is a 1 it is to increment its value on the rising edge the **clock.** Otherwise the register should hold its value. You will be graded for correctness, syntax, and efficiency of your design and code. **[9 points]**

```
module Sat8bit(clock, reset, enable, Q);
  input clock;
  input reset;
  input enable;

  output reg [7:0] Q;

  always @(posedge clock) begin
    if (reset) begin
      Q <= 8'h0;
    end
    else if (enable && (Q!=8'hff)) begin
      Q <= Q + 1;
    end
  end
 endmodule
```

4) Draw the state transition diagram which corresponds to the following logic equations. Assume the initial state is 001 and that the states are one-hot encoded. Assume S[2:0]=001 is the initial state. **[6 points]**
   a) NS2=S1*!A
   b) NS1=S2*!A+S0
   c) NS0=S1*A+S2*A
   d) X=S1*A + S0*!A



5) Transistor to truth table **[6 points, -1 per wrong or blank entry, minimum 0]**

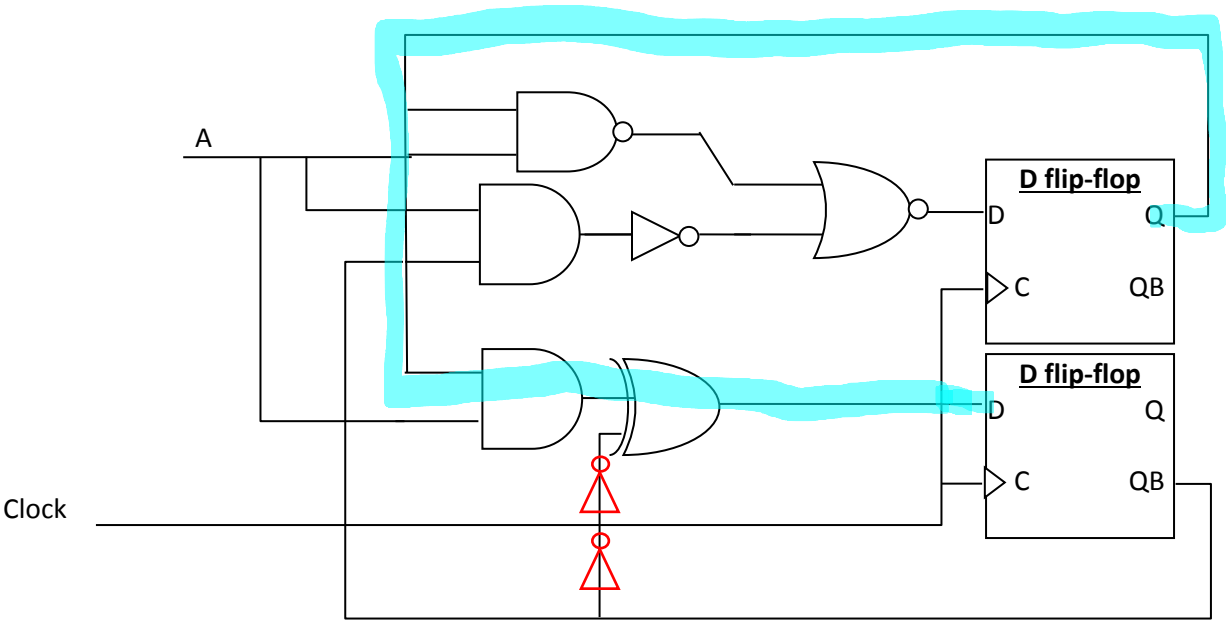


A	B	C	OUT
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	Smoke
1	0	0	Hi-Z
1	0	1	Hi-Z
1	1	0	0
1	1	1	0

Fill in the above truth table with either “1”, “0”, “Hi-Z” or “Smoke” (the last if OUT is connected to both Vcc and Ground).

	Min	Max
OR/AND	3ns	5ns
NOR/NAND	2ns	4ns
NOT	1ns	2ns
XOR	3ns	9ns

DFF:		**Min**	**Max**
	Clock to Q	2ns	3ns
	Set-up time	7 ns	
	Hold time	6 ns	



6) Assuming the input A ***always*** arrives 2ns after the rising edge of the clock, answer the following questions. **[12 points]**

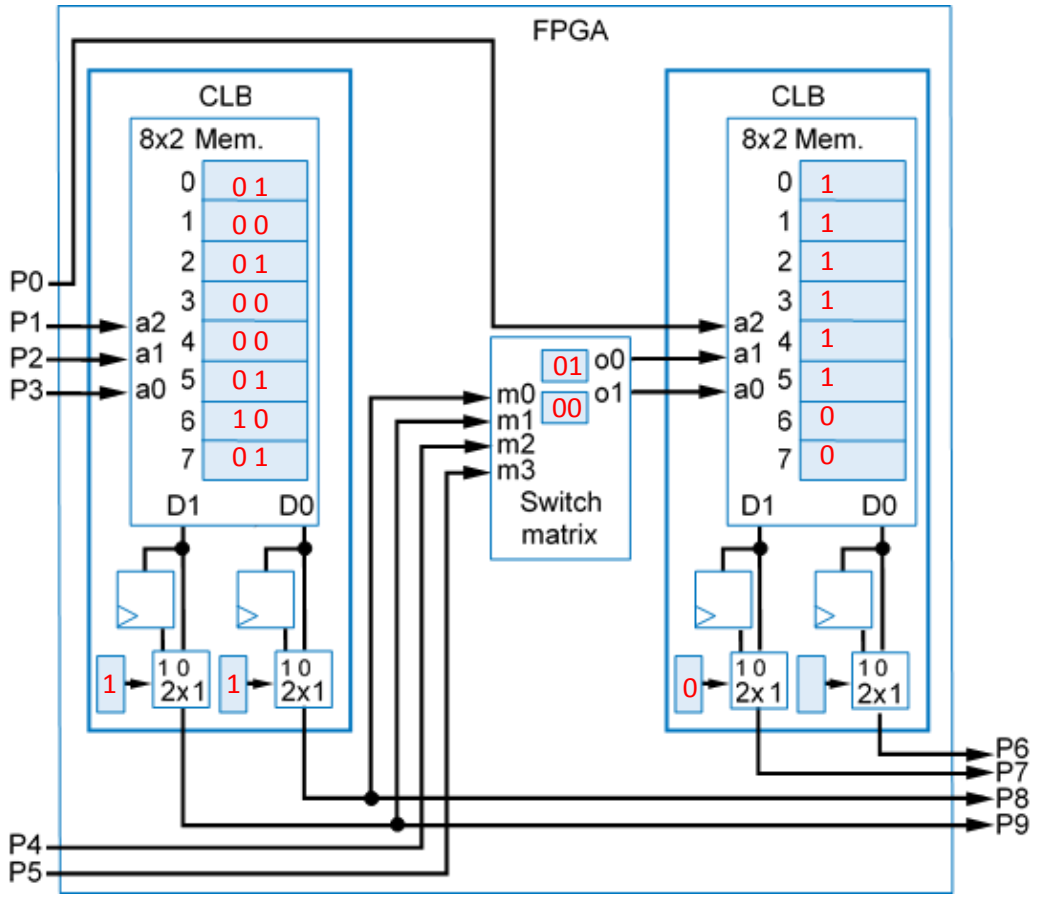
a) Add inverter pairs as needed to insure that there are no hold time violations. Add as few pairs as possible. *If there is more than one way to add as few as possible, add them in a way that minimizes the period this can be clocked at.* **[6]**

b) What is the lowest clock period that could be safely used to clock this circuit (including changes you made above)? Clearly show your work. **[6]**

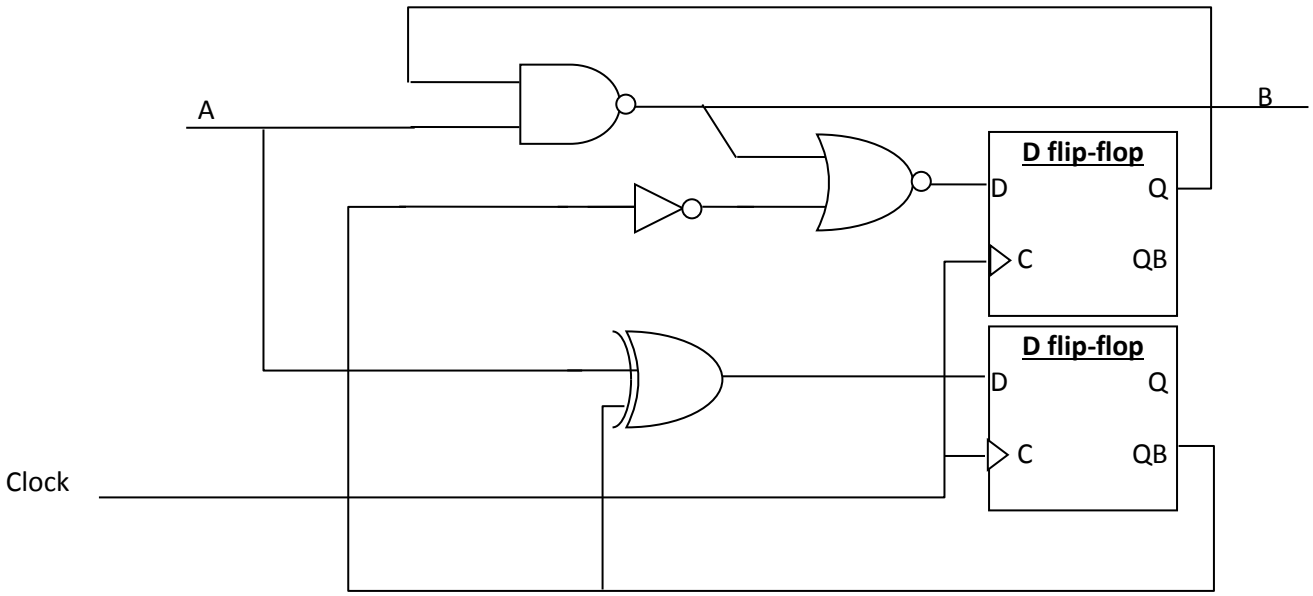
Clk-to-Q + CLD + Setup =

3 + 14 + 7 = 24ns

P0	P1	P2	P3	P4	P5	P6	P7	P8	P9
A	A	X	Y				B	Y	X

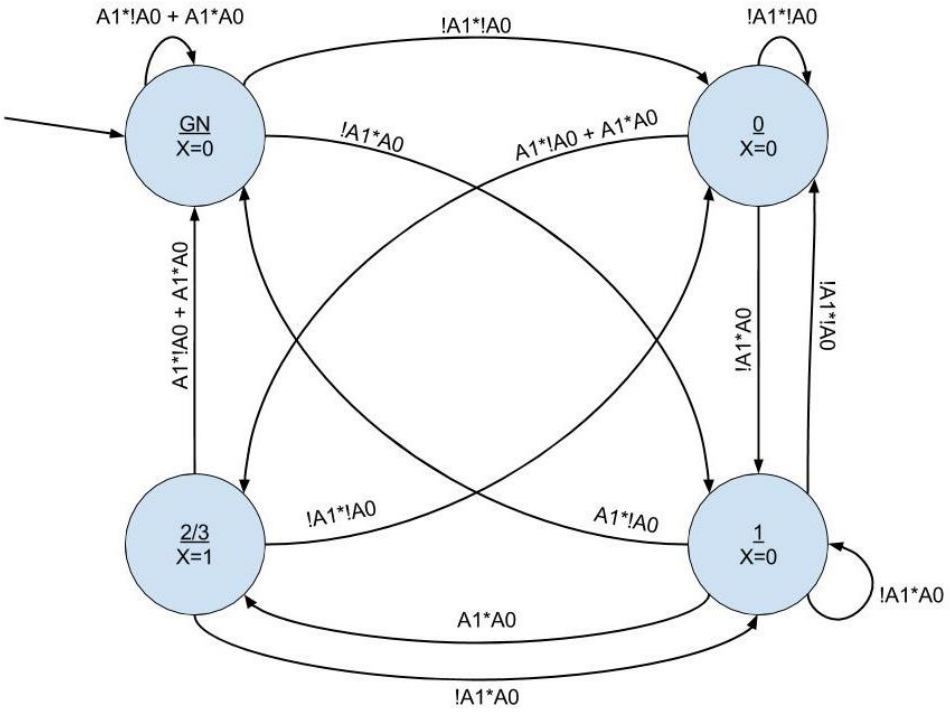


7) Fill in blank boxes as needed to implement the following circuit. Leave boxes blank if their values don't matter. Use X, Y, and Z as temporary values to connect outputs back into inputs. **[12 points]**



8) Consider a finite state machine which has one two-bit input A[1:0] and one output X. X should go high every time A (treated as an unsigned number) increases by 2 or more (so from A=1 to A=3 would be such a change). **[14 points]**

a) Provide a Moore-type state transition diagram for this problem. Use as few states as possible. *Your transitions should be logic equations in terms of each bit of A*. **[10, the most you can get is 6 points if you use more states than needed]**



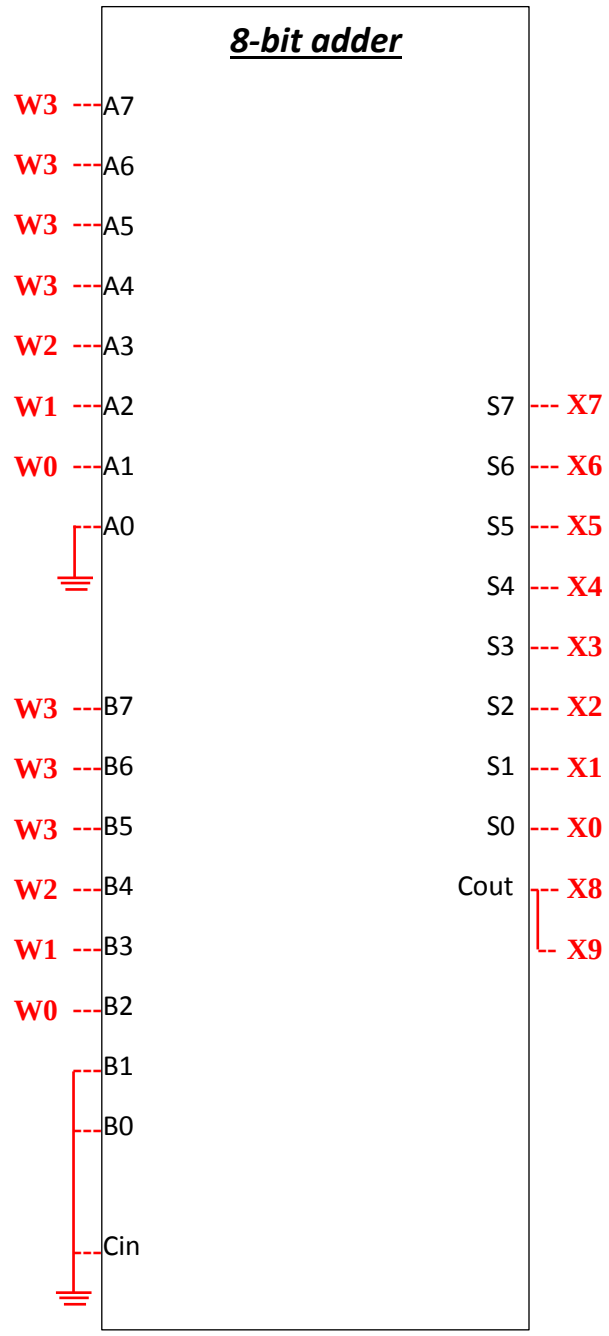
b) How many states would you need to solve the above problem if you used a Mealy machine? Briefly justify your answer. **[4]**

3 States, with a mealy machine you could combine states "GN" and "2/3" since the "GN" state is essentially a "2/3" state but with output X=0.

9) Design a circuit which has as an input W[3:0] and output X[9:0] where both are treated as 2's complement numbers. X should be equal to 6 times W's value. You may only use:
   - One 8-bit adder (with carry in and carry out)
   - Up to three NOT gates (you may use bubbles to represent these)
   - Power and ground.

   As always, you may not need all of those devices. **[10 points]**

W*6 = W*(4+2) = W*2 + W*4 = X



10) You have been tasked with designing a circuit which generates the SCLK and SS# for SPI where 8-bits are transferred per transaction (as shown below). In particular your design is to have 2 inputs:

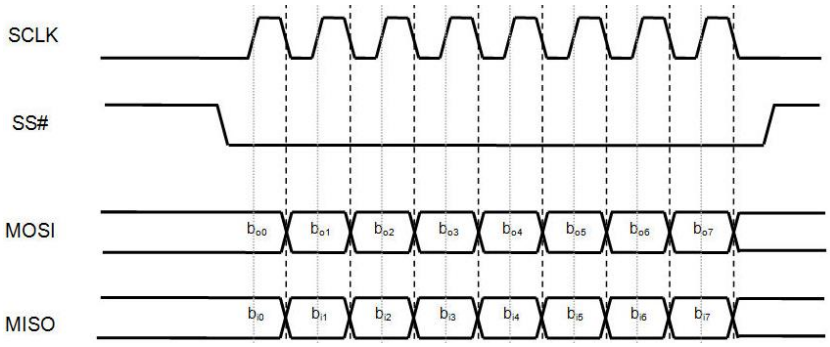
- **CLK**: a 2MHz clock
- **GO**: A signal that indicates you are to start generating SCLK and SS# for the transaction. It will never be "1" for any two clock ticks within 1ms of each other.

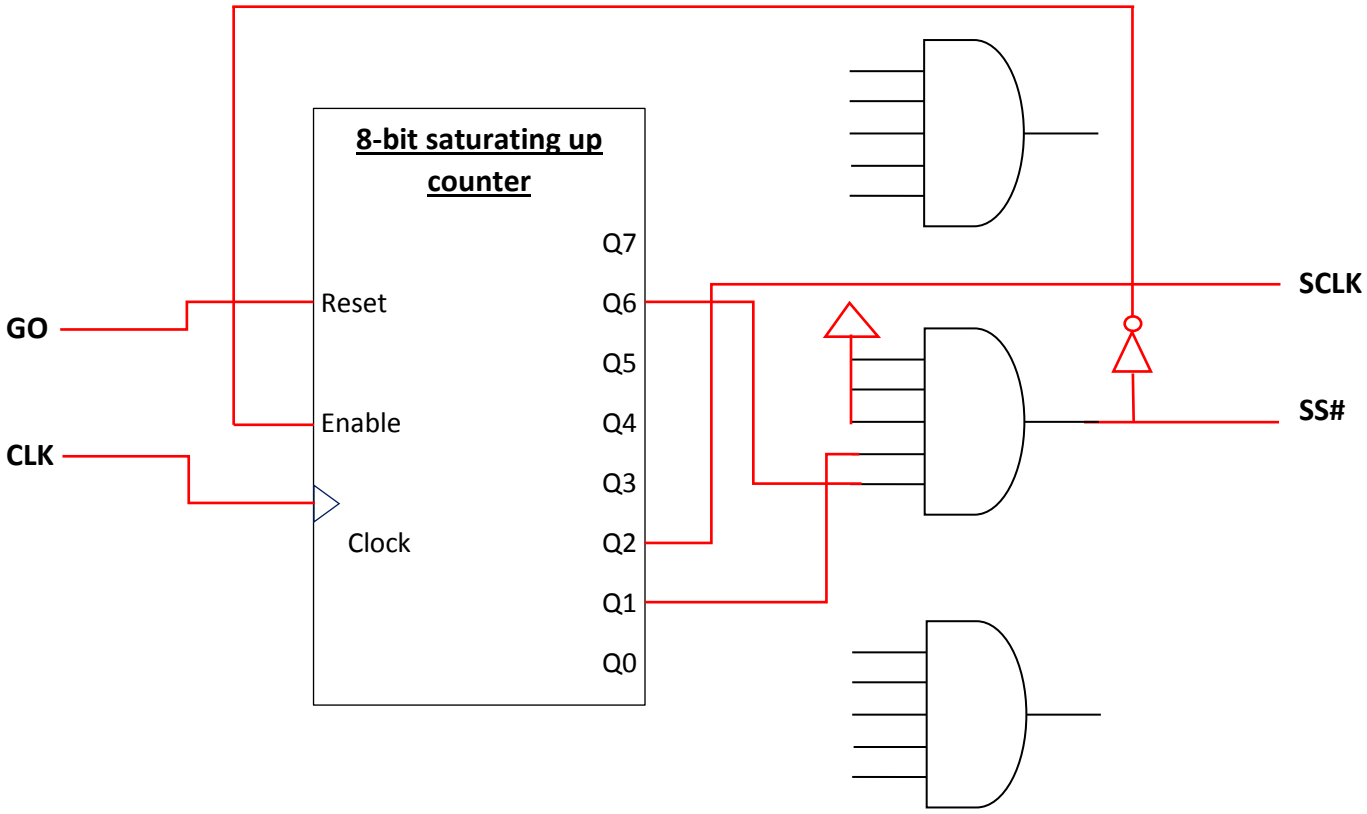
Your design has two outputs: **SCLK** and **SS#**. SCLK is to have a period of 4µs, and there should be a gap of at least 2µs between any change in SCLK and any change in SS#. The transaction must be finished before the next time GO could possibly be asserted high. You need not worry about keeping SCLK and SS# glitch free (you can assume they are being run through a flip-flop before being sent off chip).

As often seems to be the case, you have fairly limited resources (and you may not need all of them).

- One saturating 8-bit up counter with enable and reset (identical to problem 3).
- Three 5-input AND gates
- Five NOT gates (not drawn below, you may use bubbles if you wish)

You may freely use power and ground.
[13 points]





This page intentionally left blank. See coversheet for explanation.