EECS 270 *Midterm 2 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. ***This part of 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.**[11 points, -2 per wrong or blank answer, min 0]**

   a) Using standard static CMOS logic, a 3-input NAND gate requires ***2 / 3 / 4 / 6 / 8 / 12*** transistors. A 2-input AND gate requires ***2 / 3 / 4 / 6 / 8 / 12*** transistors.

   b) !(A+B+!C), when expanded into canonical sum-of-products form, has _____1_____ minterms.

   c) You are treating the 8-bit numbers A[7:0] and B[7:0] as unsigned numbers. If you set B[4:0]=A[7:3] and B[7:5]=3'b000, B is now equal to A
   plus / minus / times / modulo / divided by 2 / 3 / 4 / 8 / 16 / 32

   d) When building a 512 by 2 memory out of a square memory of minimum size, the row decoder will have ____5____ bits of input while the column MUX will have _____4_____ bits needed for its select input.

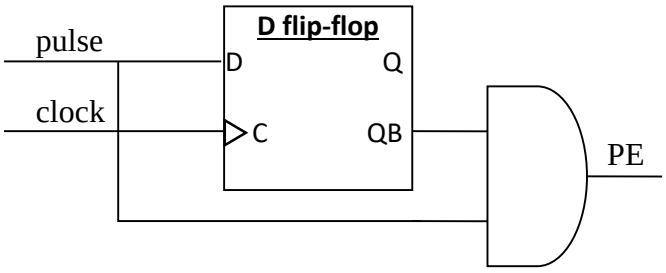
   e) The primary difference between a Mealy machine and a Moore machine is that in a Mealy machine the output depends on the current state and the _____inputs_____.

   f) Flash memory is a type of ***dynamic / static / non-volatile / DC / Garrick*** memory.

   g) A 6-bit 2's complement number can represent values from **___-32___** to **____31____.**

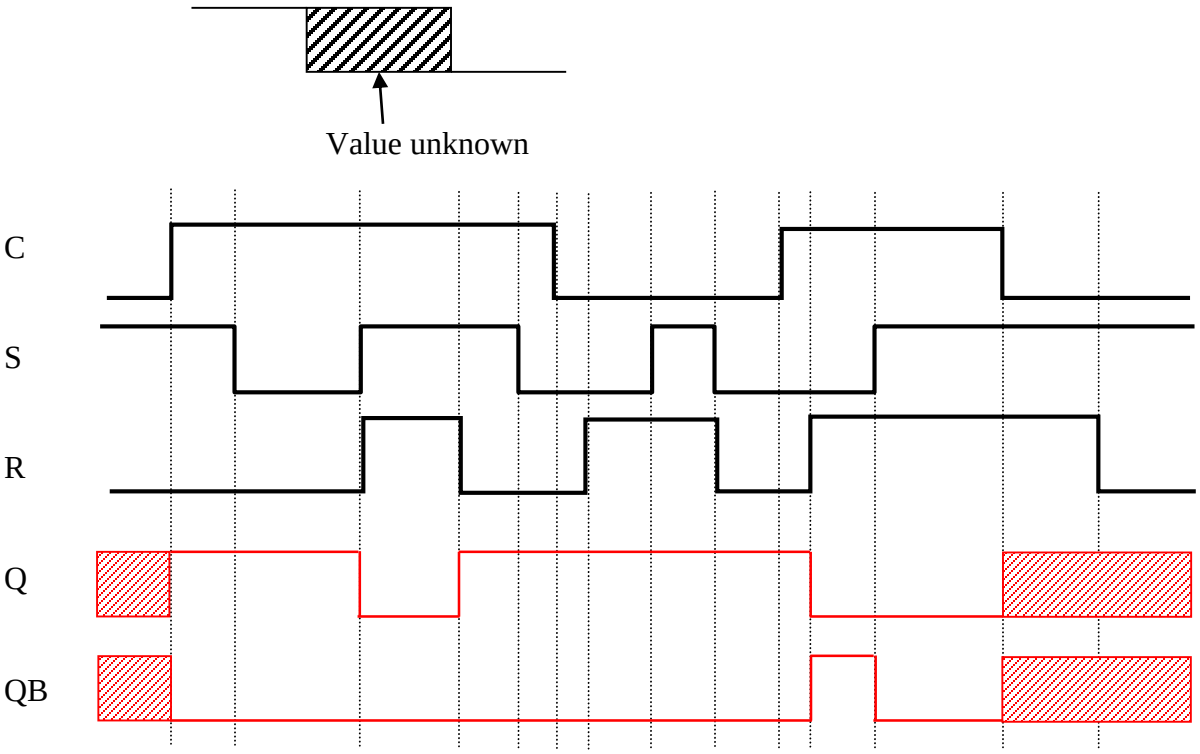
2. Complete the following Verilog module to implement the circuit shown. You may not make any changes to the supplied code. You are to follow our standard coding guidelines. **[4 points]**

```
module PED (pulse, clock, PE);
  input pulse, clock;
  output PE;
  wire PE;

  reg state;

  assign PE = ~state & pulse;

  always @(posedge clock)
  begin
     state <= pulse;
  end


endmodule
```



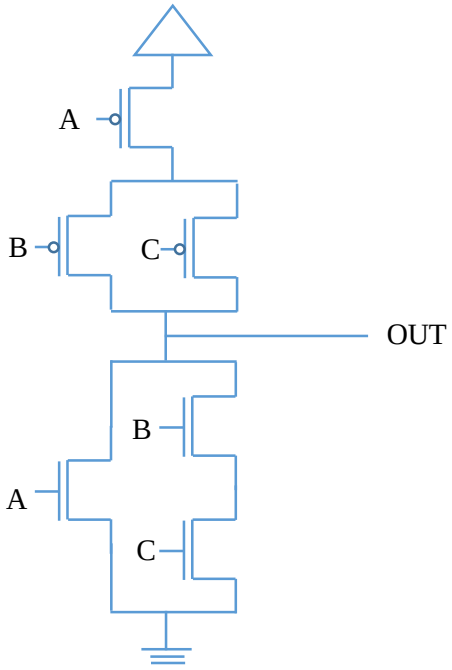
3. Complete the following timing diagram for an SR-latch with enable. You may assume that the time scale is such that the gate delay is extremely small and your answer should not reflect those delays. Changes shown to be simultaneous are exactly simultaneous. **[5 points]**

   If the value is unknown (or oscillating) at some point, clearly indicate that with hashes (like this)



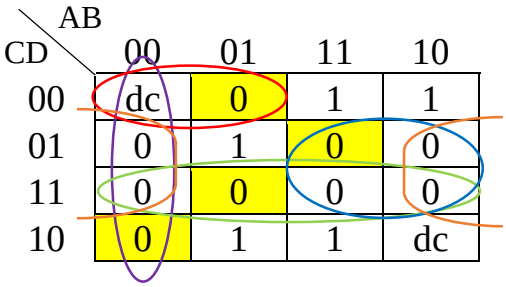
4. Using NMOS and PMOS transistors, and following the static CMOS design rules described in class, design a CMOS device which implements the function !A*(!B+!C). Your design should use as few transistors as possible. **[7 points,** *4 for a correct solution that uses 10 or fewer transistors, the 3 additional points are for a correct solution that uses as few transistors as possible***]**

!A*(!B+!C)
!A*!(B*C)
!(A+(B*C))



5. Find the minimal ***product-of-sums*** for the function : $\Sigma_{(A,B,C,D)}=(5,6,8,12,14)+d(0,10)$ using a K-map. Clearly show your work. **[7 points]**

(A + C + D) (!C + !D) (!A + !D) (A + B)

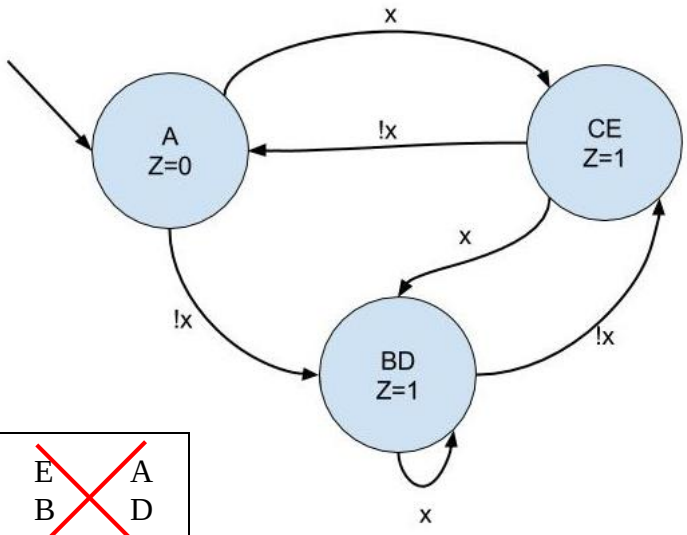


6. The following table describes a state machine.

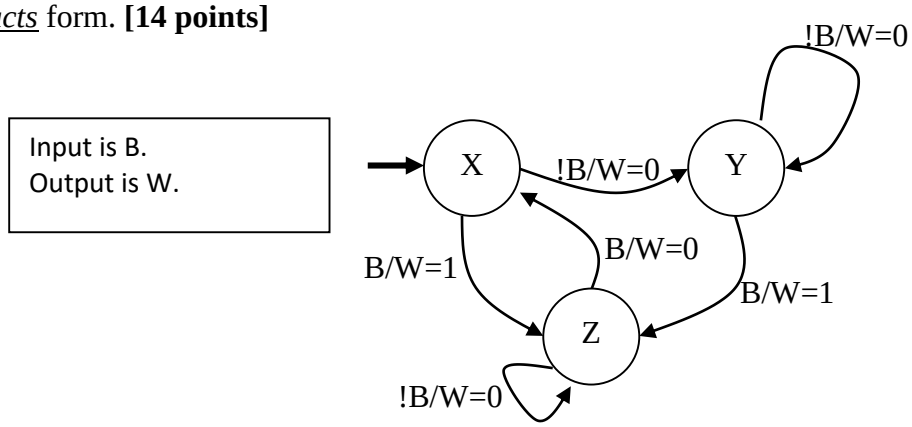
Present state	Next state x=0	Next state x=1	Output Z
A	B	C	0
B	C	D	1
C	A	D	1
D	E	B	1
E	A	B	1

Minimize the number of states in this machine and *draw* the state transition diagram which describes this minimized machine. Show your work. A is the initial state. **[10 points]**

A	~~C D / A B~~	~~A D / A B~~	~~E B / A B~~	~~E B / A D~~
B	~~A D / C D~~	~~E B / C D~~		
C	~~A D / A B~~	~~A D / C D~~		
D	~~E B / A B~~			
	E	**A**	**B**	**C**



7. Design a state machine which implements the following state transition diagram. Assign state bits **S[1:0] as 00 for state X, 01 for state Y, and 11 for state Z**. You are to assume that you will never reach the state S[1:0]=10, so you don't care what happens in that case. You must show your work to get any credit! *You only need to compute the next state and output logic, you don't need to draw the gates or flip-flops!* Place your answer where shown, all answers must be in ***minimal*** *sum-of-products* form. **[14 points]**



S1	S0	B	NS1	NS0	W
0	0	0	0	1	0
0	0	1	1	1	1
0	1	0	0	1	0
0	1	1	1	1	1
1	0	0	dc	dc	dc
1	0	1	dc	dc	dc
1	1	0	1	1	0
1	1	1	0	0	0

NS1

B \ S1S0	00	01	11	10
0	0	0	1	dc
1	1	1	0	dc

NS0

B \ S1S0	00	01	11	10
0	1	1	1	dc
1	1	1	0	dc

W

B \ S1S0	00	01	11	10
0	0	0	0	dc
1	1	1	0	dc

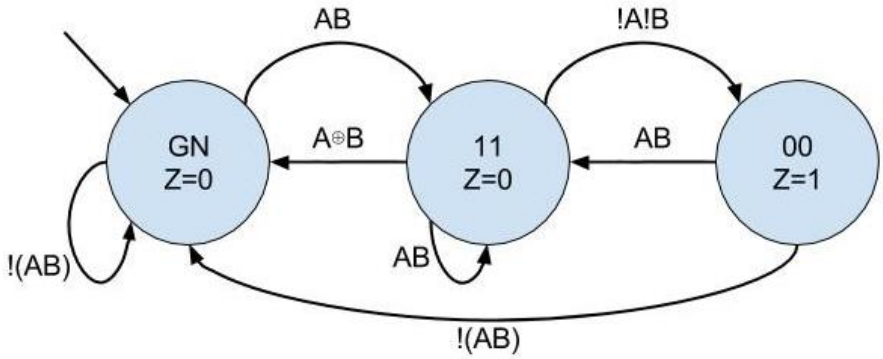
(Be sure all are in minimal sum-of-products form!)

NS1= _____!S1*B + S1*!B_____

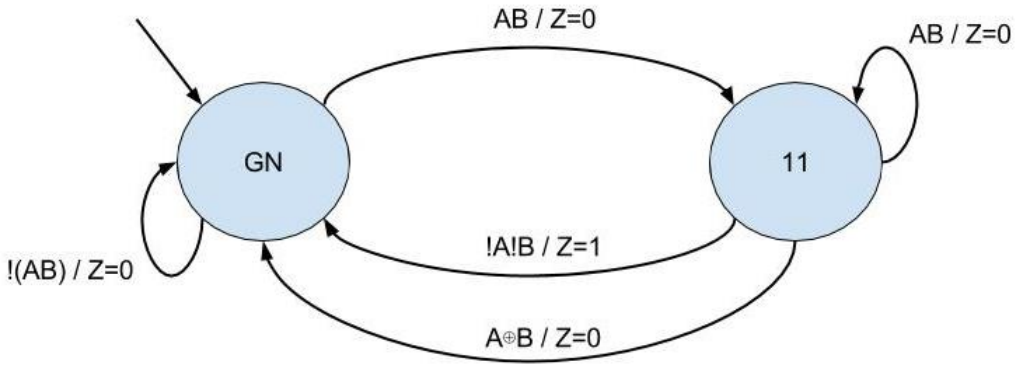
NS0= _____!S1 + !B_____

W= _____!S1*B_____

8. Design a Moore-type state transistion diagram for a state machine with 2 inputs (A and B) and 1 output (Z). Z should be high if and only if A and B were both "1" in the previous last cycle and are both "0" this cycle. Your machine should use as few states as possible. **[5 points]**

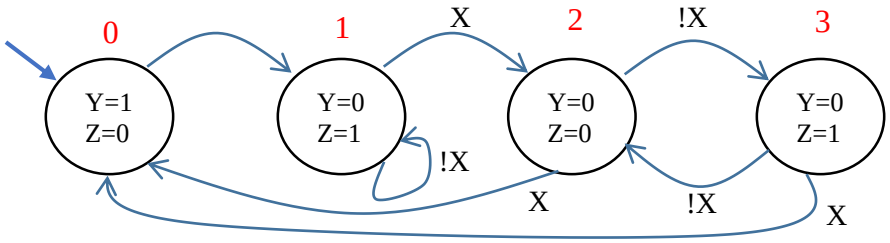


9. As the problem above, but design a Mealy-type state transition diagram with as few states as possible. **[8 points]**



10. You have been tasked with designing a state machine that implements the state-transition diagram found below. Sadly, the only sequential logic part you have is an old chip that implements a 2-bit saturating up/down counter with enable and reset identical to the device you were to create in lab 5. You need to use that chip to implement that state machine. As a reminder, the inputs and outputs to the counter are as follows:
    - **Taken** *(output)*: "1" if the counter is a 2 or 3, otherwise a "0".
    - **Strong** *(output):* "1" if the counter is a 0 or 3, otherwise a "0".
    - **Reset** *(input):* If this input is a high on the rising edge of clock, the counter will go to 0.
    - **Enable** *(input)*: If this input is low and reset is low the counter will hold its value.
    - **Up/down** *(input):* If enable is high and reset is low this input will cause the counter of count up by one if the value is a "1", otherwise it will count down by 1. *Recall this is a saturating counter*.

    Notice that the state machine itself has one input (X) and two outputs (Y and Z). Below you are asked to design the needed combinational logic to drive the three inputs to the counter as well as the logic needed to generate the two state machine outputs (Y and Z).



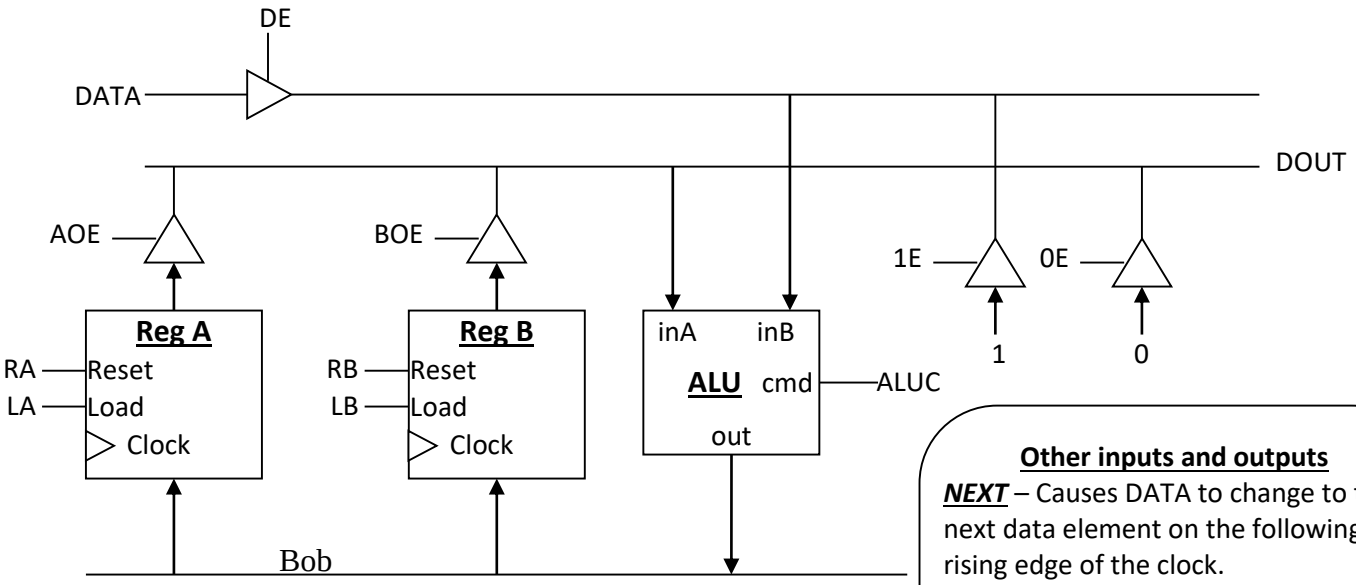
Count	T	S	X	NCount	R	E	U
00	0	1	0	01	0	1	1
00	0	1	1	01	0	1	1
01	0	0	0	01	0	0	dc
01	0	0	1	10	0	1	1
10	1	0	0	11	0	1	1
10	1	0	1	00	1	dc	dc
11	1	1	0	10	0	1	0
11	1	1	1	00	1	dc	dc

The above states are labeled according to their counter value, however, as in lab, we have access to only T and S and thus have to use those to know our current state

State	T	S	Y	Z
00	0	1	1	0
01	0	0	0	1
10	1	0	0	0
11	1	1	0	1

Write logic equations for the following. Keep your solutions as simple as possible. **[14 points]**

- Reset: _____T*X_____
- Enable: _____T + S + X_____
- Up/down: _____!T + !S _____
- Y: _____!T*S_____
- Z: _____!(T⊕S)_____



Other inputs and outputs

NEXT – Causes DATA to change to the next data element on the following rising edge of the clock.

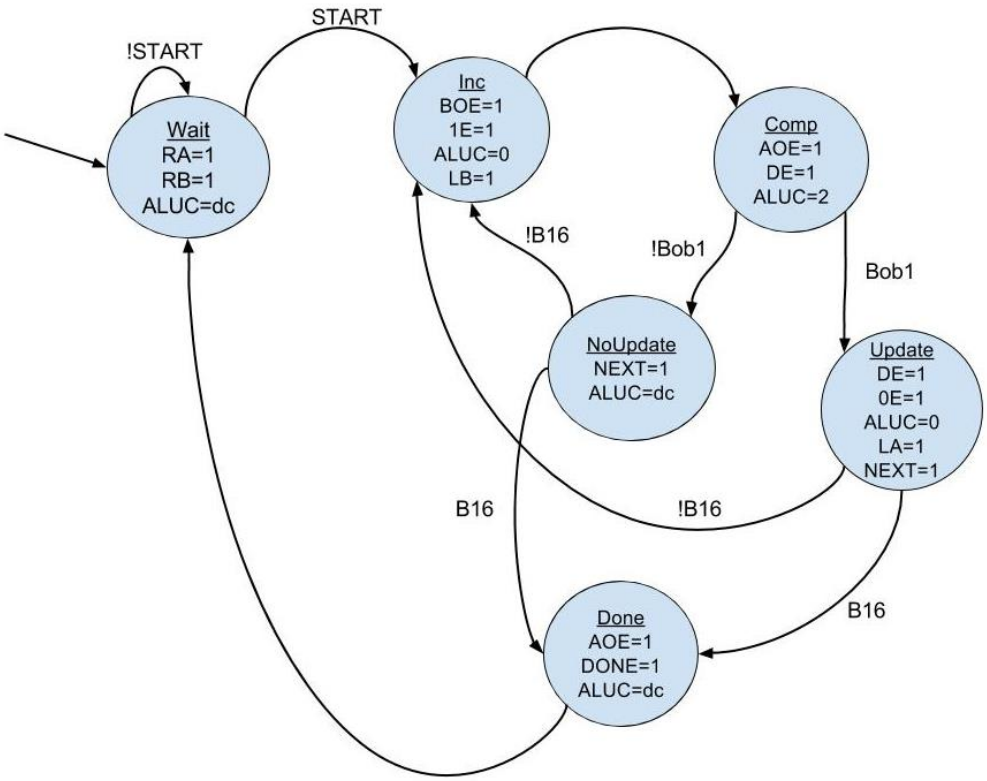
DONE – Indicates the result is available on DOUT. Only needs to be asserted for one clock period.

START – You are to start finding the average.

Bob1 – True if Bob==1

B16 – True if RegB==16.

11. You are to design a Moore-type state transition diagram which causes the above datapath to find the maximum of 16 unsigned numbers. When the START signal goes high, you are to begin. Once done, the result should be placed on DOUT and the DONE signal should be set high. START will not go high again until after DONE is asserted. Any value not specified in a given state will be assumed to be zero. **[15 points]**



ALU

cmd	Operation
0	out=inA+inB
1	out=inA-inB
2	out=inA<inB
3	out=inA>inB

Note X>Y means the output is a 1 if X>Y otherwise it is a 0.

This page intentionally blank. See cover sheet for directions.