Short Answer/Fill in the blank

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

   a. The 5-bit 2's complement number representation of -4 is 11100.

   b. 10001, when treated as a 5-bit signed-magnitude number, has a decimal representation of -1.

   c. Individual DRAM cells are typically ***larger/smaller*** than SRAM cells and in general DRAM cells are ***faster/slower*** than SRAM cells.

   d. Say you want to perform "conditional negation." That is, you have an input x and an input y, and if x is 0 the output should be y, but if x is 1 the output should be !y. If you were going to implement this as a single gate, you'd use a(n) XOR gate.

   e. Consider a memory device that has 256 addresses each 16 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 6 inputs while the column MUX would have 2 selection bits.

   f. The canonical product-of-sums representation of !(A+B) is $(A+\bar{B})(\bar{A}+B)(\bar{A}+\bar{B})$

   g. Say a tri-state device was outputting a value of "HiZ" and that value was driven to the input of a two-input AND gate where the other input was a "1". The AND gate's output would be ***"0" / "1" / "HiZ" / unknown***.

2. Using the *rules of logic*, convert !(A+!C)*(A*C) into a *minimal* sum-of-products form. Provide the name of the rule used for each step. **[5 points]**

$\overline{(A+\overline{C})}AC$

$= \overline{A}\,\overline{\overline{C}}AC$ DeMorgan

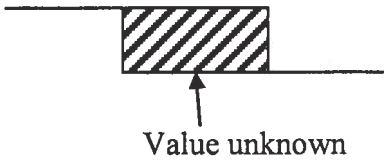
$= \overline{A}CAC$ Involution

$= \overline{A}AC$ Idempotent

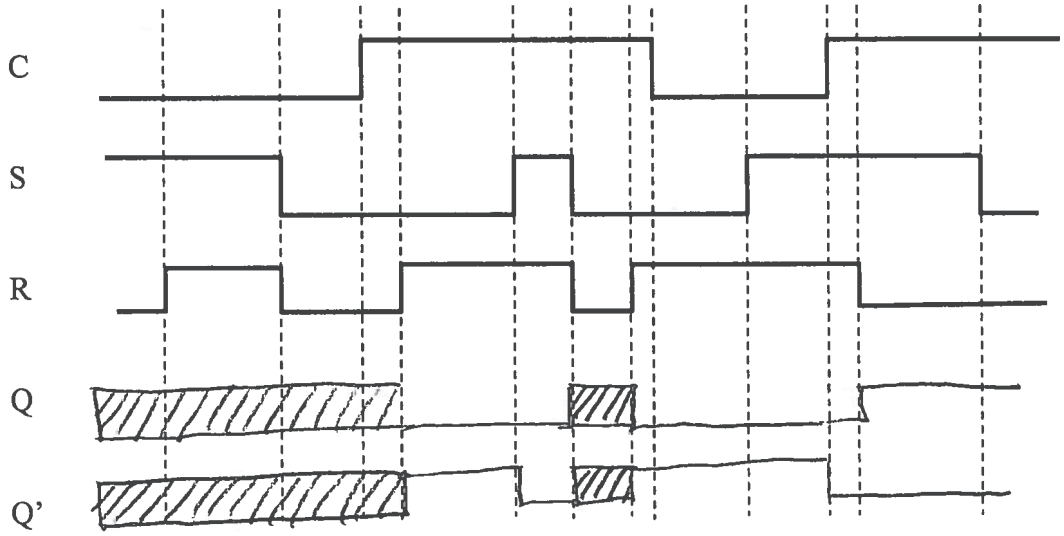
$= 0 \cdot C$ Complement

$= 0$ Null element

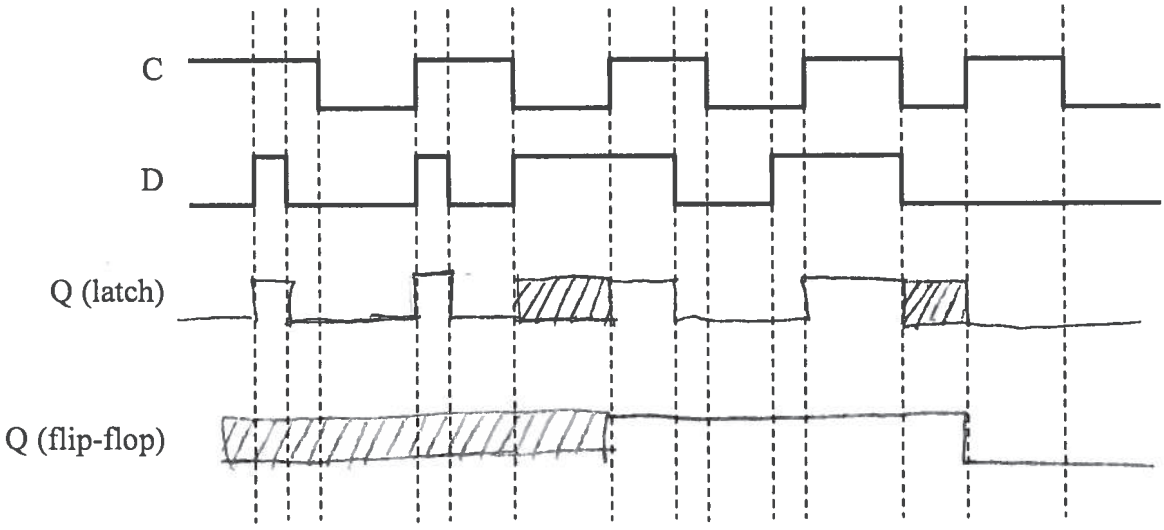
3. Complete the following timing diagrams. If the value is unknown (or oscillating) at some point, clearly indicate that with hashes (as shown). Each Q value will be graded as either right or wrong (no partial credit on a given Q value).



a) Complete the timing diagram below for an SR latch with enable. **[6 points]**



b) Complete the timing diagram below for both a D latch and a D flip-flop. **[6 points]**

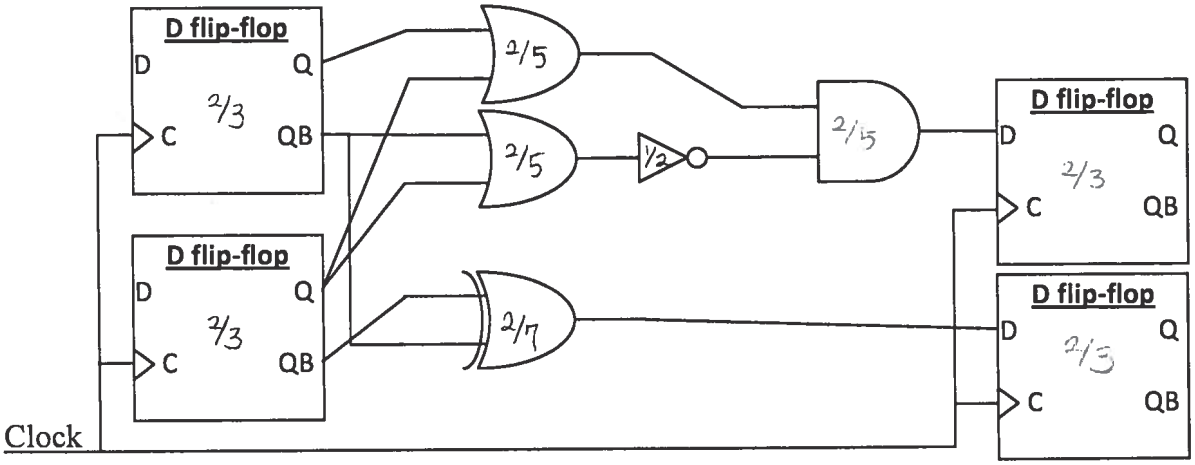


4. Say you have the following values associated with the process you are using (notice the hold and set-up times are not specified). Assume you intend to clock this circuit at 50MHz.

50MHz ↔ 20ns

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

		Min	Max
OR/AND		2ns	5ns
NOT		1ns	2ns
XOR		2ns	7ns



a. Assuming you want a reliable circuit, what is the highest value the flip-flop could have for a hold time? Show your work. **[4 points]**

Clk to Q + CL Delay ≥ hold time

2 + 2 ≥ hold time

4 ≥ hold time

4 ns

b. Assuming you want a reliable circuit, what is the highest value the flip-flop could have for a setup time? Show your work. **[4 points]**

Clk to Q + Clk Delay + Setup time = 20

3 + 12 + setup time = 20

15 + setup time = 20

5 ns

(Problem continued on the next page)

c. As you know, no two events happen at the same time. As such, no two flip-flops will see the clock's rise time at exactly the same time. This is called "clock skew". Let's assume that we know that the clocks going to each device could be off from each other by up to 1ns. Put differently, all we know is that each flip-flop will see a rising edge of the clock within 1ns of when any other flip-flop will. Assume we don't know anything about the order the flip-flops will see the clock edge.

Clk to Q
min max
2 ± 1 3 ± 1

i. Redo part "a" under these assumptions. *Briefly explain* your work. **[3 points]**

$T_{hold} = T_{hold} - T_{skew}$

$= 3\,ns$

Clk to Q + skew + CL Delay ≥ hold
2. −1 2.
3 ≥ hold

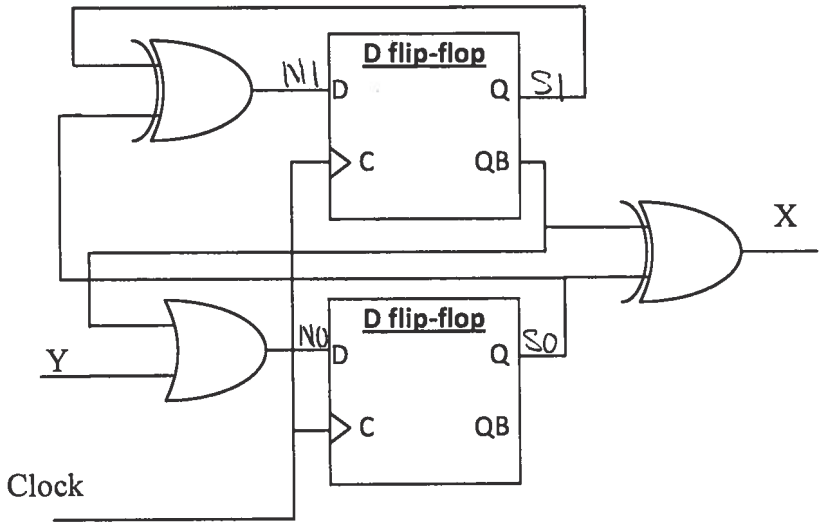
ii. Redo part "b" under these assumptions. *Briefly explain* your work. **[3 points]**

Clk to Q + skew + CL delay + setup = 20
3 2.

16 + setup = 20

4 ns

5. Draw the state-transition diagram that describes the following state-machine. Assume the initial state is when the flip-flops both have a value of 0. Show all state (even if a given state is unreachable). *Show your work.*
[10 points]



State Encoding

A 0 0

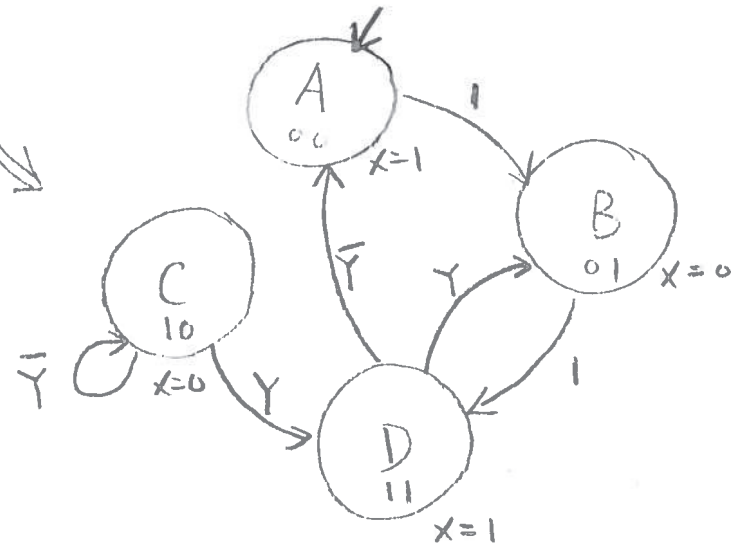
B 0 1

C 1 0

D 1 1

	S1	S0	Y	N1	N0	X
A	0	0	0	0	1	1
	0	0	1	0	1	1
B	0	1	0	1	1	0
	0	1	1	1	1	0
C	1	0	0	1	0	0
	1	0	1	1	1	0
D	1	1	0	0	0	1
	1	1	1	0	1	1

$$\begin{cases} N1 = S1 \oplus S0 \\ N0 = \bar{S1} + Y \\ X = \bar{S1} \oplus S0 \end{cases}$$



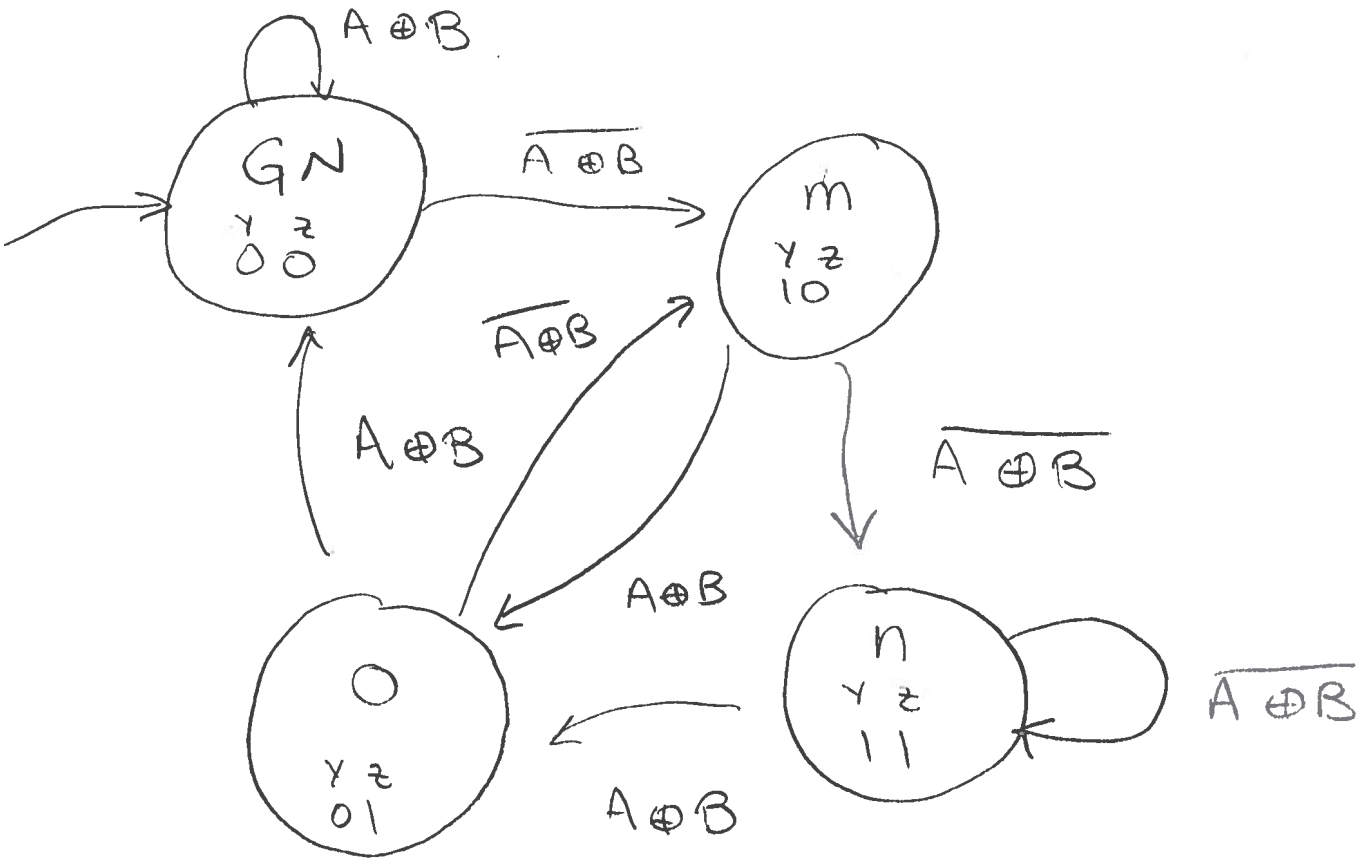
6. Design a state-transition diagram for a state machine with two inputs, A and B, and two outputs, "Y" and "Z". Y should be a "1" if A and B were the same (equal) in the previous cycle. Z should be a "1" if A and B were the same two cycles ago (the cycle previous to the previous cycle). For example, the following input:

   `A:010010001`
   `B:001011001`

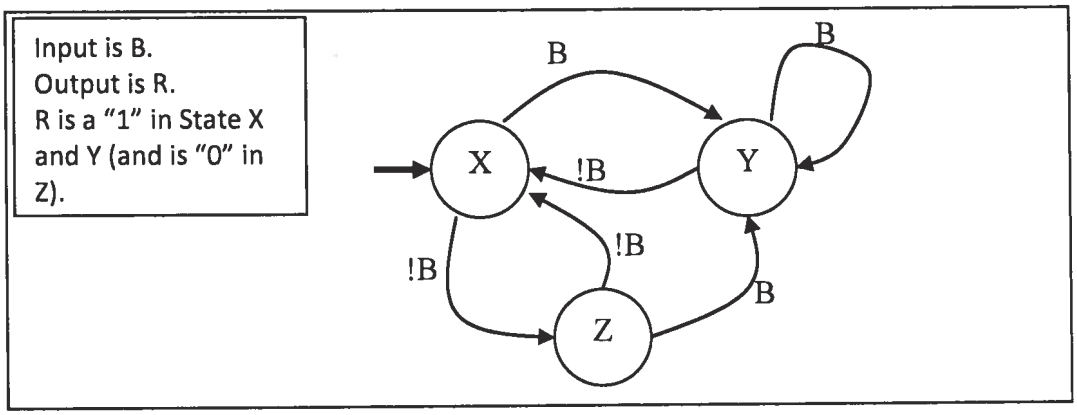
   Would generate this output:

   `Y:010011011`
   `Z:001001101`

   Your answer must have no more than 7 states. [**12 points**; 9 points for a correct answer; 3 for a correct *and* minimal-state answer]



7. Design a state machine which implements the following state transition diagram. Assign state bits S[1:0] as 11 for state X, 01 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 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. **[10 points]**



State Encoding

X : 11

Y : 01

Z : 10

	S1	S0	B	N1	N0	R
⇒	0	0	0			
	0	0	1			
Y	0	1	0	1	1	1
	0	1	1	0	1	1
Z	1	0	0	1	1	0
	1	0	1	0	1	0
X	1	1	0	1	0	1
	1	1	1	0	1	1

$N1 = \overline{B}$

$N0 = \overline{S1} + \overline{S0} + B$ ⇐

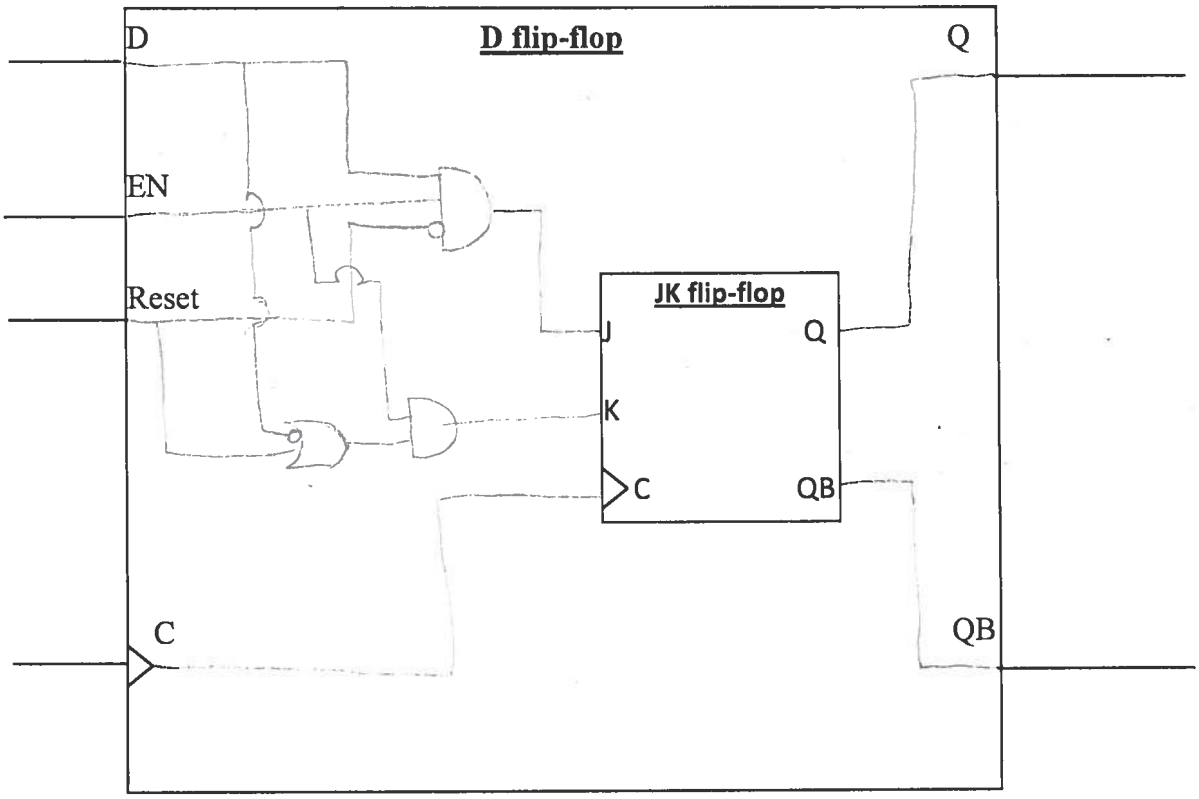
$R = S0$

NS1= $\overline{B}$

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

R= $S0$

8. Build a D flip-flop with enable and reset using only a JK flip-flop (without enable or reset) and standard gates. **[10 points]**



D	EN	Reset	Q	J	K
0	0	0	last Q	0	0
0	0	1	last Q	0	0
0	1	0	0	0	1
0	1	1	0	0	1
1	0	0	last Q	0	0
1	0	1	last Q	0	0
1	1	0	1	1	0
1	1	1	0	0	1

$J = D \cdot EN \cdot \overline{Reset}$

$K = \overline{D}EN + DEN\overline{Reset}$

$= EN(\overline{D} + D\overline{Reset})$

$= EN(\overline{D} + \overline{Reset})$

9. Using only the devices listed below, design a circuit which takes two *4-bit signed-magnitude numbers* (X[3:0] and Y[3:0]) and outputs the smaller of the two, named OUT[3:0].

   In your design you may use the following devices (as well as freely using "0" and "1" as a inputs as desired)
   - AND, OR, and XOR gates (any number of inputs)
   - Inverters
   - 2 to 1 MUXes
   - 4-bit *unsigned* comparator
   - 8 to 4 MUXes
   - 16 to 4 priority encoder

   You must clearly label any device you use (other than gates and inverters) and your design should be clear enough that someone else could understand how everything was to be connected. Your grade will be based in part upon the efficiency of your design. **[15]**