Systems I
Fall 2014
Mid-term exam — Solutions

This is a closed-book, closed-note exam. Answer all of the questions **clearly, completely, and concisely**. You have 50 minutes, so be sure to use your time wisely. All work should be written in your blue book.

1. Question: (xxx points) When we devised how a basic memory element would work, we came up with the following truth table:

Q	R	S	Q
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	x
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	x

We later used the *cross-coupled* NOR *gates* to implement this function and devise our S-R latch. **Using Boolean algebra, prove that the two are equivalent. Note:** If you have difficulty devising an algebraic proof, you may prove the equivalence using any other legitimate method, but for a partial loss of credit.[1]

Answer: **First**, based on the structure of the cross-coupled NOR gates, we can express formulae for Q and $\bar{Q}$:

$$Q = \overline{R + \bar{Q}}$$
$$\bar{Q} = \overline{S + Q}$$

Substituting for $\bar{Q}$ into the first of these equations:

$$Q = \overline{R + (\overline{S + Q})}$$

Second, from the truth table above, we can begin with its DNF:

$$Q = \bar{Q}\bar{R}S + Q\bar{R}\bar{S} + Q\bar{R}S$$

[1]Better something than nothing!

The following transformations show how the latter can be algebraically transformed into the former:[2]

$Q =$	Boolean algebraic rule/law
$\bar{Q}\bar{R}S + Q\bar{R}\bar{S} + Q\bar{R}S$	DNF
$\bar{R}(\bar{Q}S + Q\bar{S} + QS)$	distributivity
$\bar{R}(\bar{Q}S + QS + Q\bar{S})$	commutativity
$\bar{R}(\bar{Q}S + QS + QS + Q\bar{S})$	idempotence
$\bar{R}(S[\bar{Q} + Q] + Q[S + \bar{S}])$	distributivity
$\bar{R}(S + Q)$	complementation
$\bar{R}(\overline{\overline{S + Q}})$	double negation
$\overline{R + (\overline{S + Q})}$	DeMorgan's
$\overline{R + \bar{Q}}$	substitution of $\bar{Q}$

DISCUSSION: First, *xxx points*? Whoops. Just a failure to complete the edits there. Each question has equal value.
MORE TBA.[3]

2. QUESTION: (xxx points) Answer the following questions with short answers (no more than 3 or 4 sentences):

   (a) What does it mean for the NAND logic function to be *universal*?

   (b) In a given circuit, what is its *critical path*?

   (c) What does it mean for a memory element to be *level triggered* vs. *edge triggered*? Which do we use in our counter-based circuits, and why?

   ANSWERS:

   (a) A set of *universal operators* are ones that can be used to implement/express **any** possible logic function. Thus, for NAND to be a universal operator implies that expressions composed using only that operator may be used to any and all logic functions.

   (b) In any circuit, there is some *longest path* from some input to some output. It is *longest* in the sense that it traverses the largest number of gates, making that output require the greatest time (in that circuit) before its output settles (ceases *floating*) to its final, correct value.[4]

   (c) Memory elements are controlled by a *clock* input, C that determines **when** the

[2]Although we devised some of our own names for the Boolean algebraic properties in class, which are acceptable for your tests, I use here standard Boolean algebraic law names.

[3]*TBA = To Be Added.*

[4]If different gates have different *delays*, then the critical path is the one from an input to an output for which the sum of the gate delays is maximal.

element adopts the data input value as its output. A *level triggered* memory element (e.g., a *D-latch*) is *open* (accepting new values) while $C = 1$. An *edge triggered* memory element (e.g., a *D-flip-flop*) is *open* at C transitions from one value to another (e.g., from 0 to 1).

Discussion: **TBA.**

3. (xxx points) QUESTION: We want a *3-bit, bi-directional counter.* That is, we want a counter circuit that counts either forwards (incrementing) or backwards (decrementing) through the unsigned 3-bit integers from 0 to 7. The circuit has two inputs:

   (a) A **clock button** to control the advancement of the counter.

   (b) A **direction switch**, S, that determines in which direction the counter runs. If $S = 0$, then the counter should count upwards (via incrementation); if $S = 1$, then the counter should count backwards (via decrementation).

   Devise the logic and draw the circuit diagram of this bi-directional counter. Keep in mind that when the direction switch is changed, the output should **not** change immediately. Only a cycling of the clock input should cause the output to update.

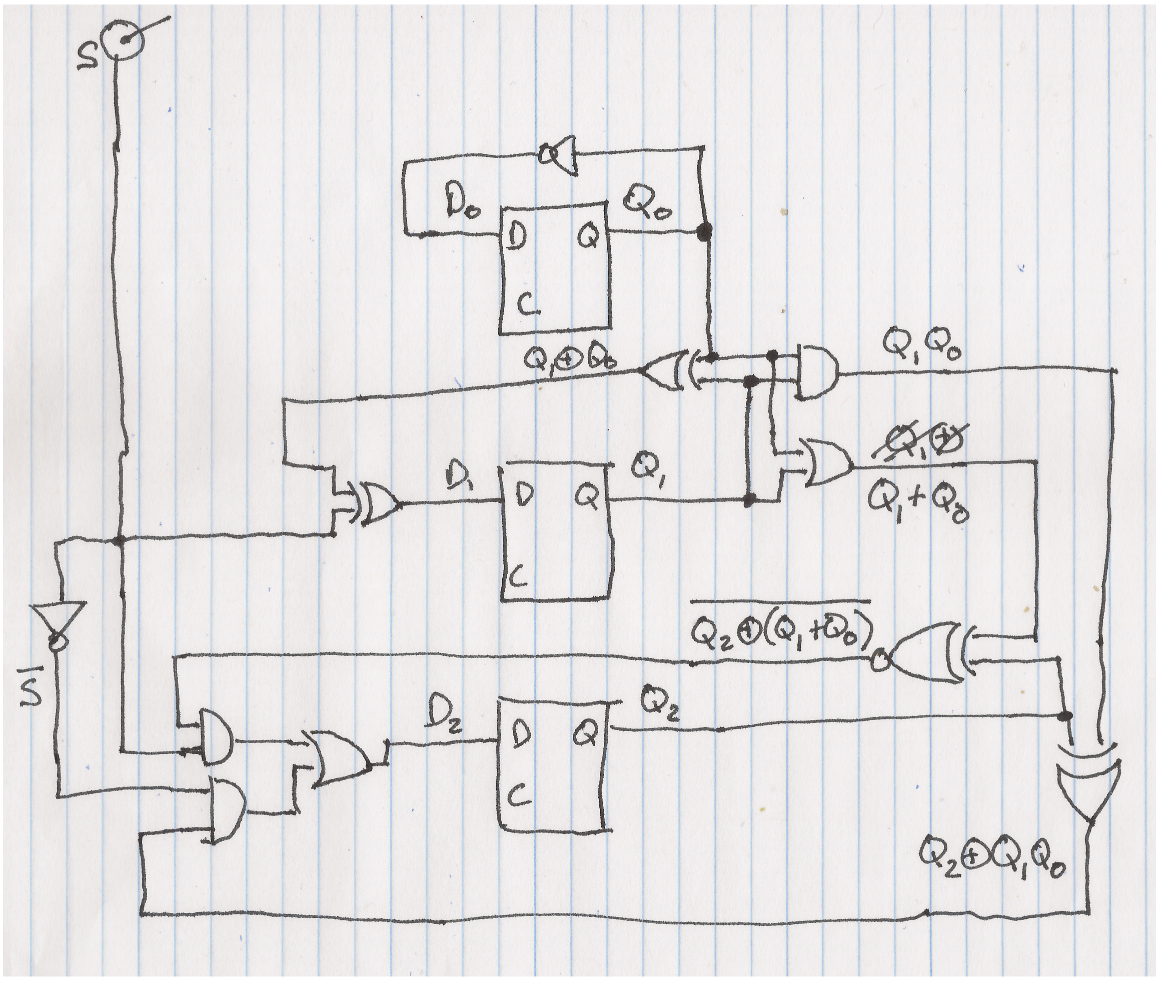
   ANSWER: The directional input S implies the following truth table, representing the sequential logic of counting forward (when $S = 0$) or backward (when $S = 1$):

S	Q_2	Q_1	Q_0	D_2	D_1	D_0
0	0	0	0	0	0	1
0	0	0	1	0	1	0
0	0	1	0	0	1	1
0	0	1	1	1	0	0
0	1	0	0	1	0	1
0	1	0	1	1	1	0
0	1	1	0	1	1	1
0	1	1	1	0	0	0
1	0	0	0	1	1	1
1	0	0	1	0	0	0
1	0	1	0	0	0	1
1	0	1	1	0	1	0
1	1	0	0	0	1	1
1	1	0	1	1	0	0
1	1	1	0	1	0	1
1	1	1	1	1	1	0

   This table yields the following funtions:

   - $D_0 = \bar{Q_0}$
   - $D_1 = S \oplus (Q_1 \oplus Q_0)$
   - $D_2 = \bar{S}(Q_2 \oplus Q_1Q_0) + S(\overline{Q_2 \oplus (Q_1 + Q_0)})$

   The following sequential logic circuit is an implementation of the above functions:



DISCUSSION: **TBA.**

4. (xxx points) QUESTION: Consider our multiplier circuit from Lab 4. It handles *unsigned integers* correctly, but what if we want it **also to handle signed, two's-complement integers**? It doesn't handle them properly as it's currently designed, so:

   Describe and sketch how you would change or augment our multiplier to handle two signed integers as inputs, and then produce the correct signed output. If you draw the multiplier, you need not draw its every detail, but only those that are altered or affected by your changes.

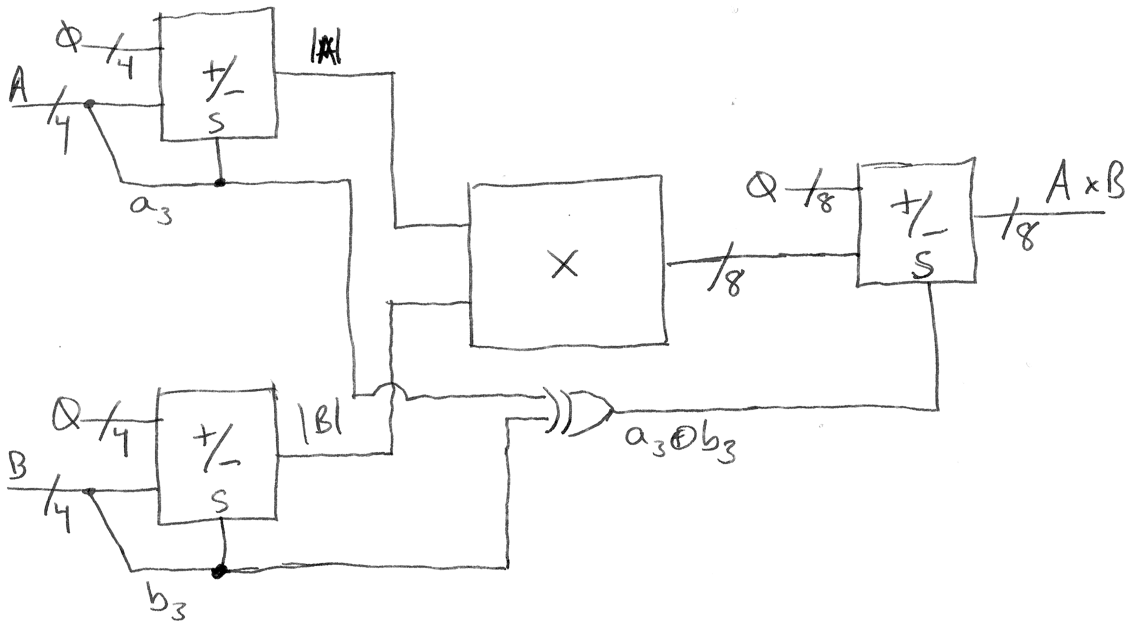
   ANSWER: The simplest approach is to multiply only the non-negative magnitudes of the values, and then to negate the product if the signs of the original inputs demand it. Assume two 4-bit inputs $A = (a_3a_2a_1a_0)$ and $B = (b_3b_2b_1b_0)$. Our multiplier should:

   (a) Take the *absolute value* of A and B.

   (b) Pass $|A|$ and $|B|$ as inputs to the multiplier.

(c) If the signs of A and B differ, *negate the product* output by the multiplier.

In the following circuit, a_3 and b_3 indicate the signs of A and B respectively. Therefore, these bits determine whether to negate each input in calculating the absolute values, and $a_3 \oplus b_3$ determines whether to negate the product. Negation is performed by passing the value to an adder/subtracter circuit. This circuit's operational control is specific by the input S, where $S = 0$ triggers addition, while $S = 1$ triggers subtraction.



DISCUSSION: **TBA.**