# CS 16 Fall 2008 — Mid-term

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 budget your time. All work should be written in your blue book.

1. (10 points) Use a Karnaugh map to simplify the boolean function described by the truth table below. Draw your rectangles clearly and express your result as a boolean algebraic equation—**do not draw a circuit.**

| A | B | C | D | Y |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 0 |

2. (15 points) Recall the basic rules for *two's complement addition overflow:* If the two inputs have the same sign and the output has a different sign from those two inputs, then overflow has occurred.
   Prove that when the carry-in and carry-out of the most significant bit of a ripple-carry addition differ, that also indicates overflow. That is, show the equivalence of these two methods of overflow detection.

3. (25 points) The multiplier that we built for lab-4 worked only on positive integers; it can produce incorrect results on negative numbers (using two's complement).
   Given that multiplier (4-bit multiplier and multiplicand, yielding an 8-bit product), how can it be modified or augmented to correctly multiply any combination of positive and negative inputs? **Draw a circuit** to show how this improved multiplier would be structured.
   NOTE: You may assume high-level components. For example, if you need an adder, just draw a box and label it with an addition-sign. Just be sure it is clear what each component does. If you must invent new components that we've not used, make clear by description or diagram how it would be constructed.

4. (25 points) Consider the following sequence of 1-bit values:

   $$0, 1, 1, 1, 0$$

   **Construct a clocked circuit** that repeatedly emits this pattern of 1-bit output values every 5 clock cycles.
   NOTE: You may express any combinational functions either as explicit gates or by use of a ROM. If you use a ROM, it must be clear which lines provide the input address, which lines carry the output data, and what the complete contents of the ROM are.

5. (25 points) Consider the following sequence of 1-bit values:

   $$0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, \ldots$$

   That is, between 0's values, there is first zero 1's, and then one 1, and then two 1's, and then three 1's, and so on. This sequence ends with a 0 is followed by 255 1's, at which point it should repeat.
   **Construct a clocked circuit** that repeatedly emits this pattern of 1-bit output values every 32,896 clock cycles.
   NOTE: First, obviously, you cannot just enumerate this pattern—you must devise a new approach. Second, again assume that you have high-level components at your disposal. Third, you again may express any combinational functions using either gates or a ROM, using the same rules for a ROM as above.