Due: Friday, 17 April 2009

Submission: Turn in a printed or neatly written copy of your work at the beginning of class.

1. Recall that when using a two's complement representation, it is possible to get a result from an addition that is incorrect due to overflow. Here is an example from a previous homework that overflows:

$$\begin{array}{r} 01111 \\ +\,00001 \\ \hline \end{array}$$

   Why is this overflow? Because the both summands are positive, yet the result in the fifth column (for a five-bit addition) is a 1, indicating a negative answer. Of course, that shouldn't happen. Here is another example from a previous homework that does *not* exhibit overflow:

$$\begin{array}{r} 11110 \\ +\,10101 \\ \hline \end{array}$$

   Why is this not overflow? First, recall that we ignore the last carry bit. Then we note that the result in the fifth column is a 1, which indicates that the sum is negative. This is what we expect, since the summands are both negative. In summary, the addition exhibits overflow only when both summands have the same sign, but the sign of the sum differs from that of the summands.

   In this problem, you will design a circuit for detecting overflow. It should work for an adder of arbitrary "width" (i.e., the number of bits in the two numbers being added).

   (a) The output of the circuit is whether or not there is overflow. What should the inputs be?

   (b) Write the truth table for detecting overflow, specifying all values of the inputs and the corresponding output in each case.

   (c) Write a Boolean logic statement—e.g., $[(\text{NOT}\, x_1)\, \text{AND}\, x_2]$—expressing the truth table.

   (d) Draw (or implement in LogiSim, if you choose) the circuit represented by your logic statement. Be sure to label your inputs clearly.

2. Print and submit your implementation the following circuits from the lab on Friday, 4/10 entitled "Digital Circuits."

   (a) `XOR.circ`

   (b) `half-adder.circ`

   (c) `flip-flop.circ`