**California Institute of Technology**
**Department of Computer Science**
**Computer Architecture**

CS184a, Winter 2003 — Assignment 1: Logic — Monday, January 6

---

**Due:** Monday, January 13, 9:00AM

You may do sections (A and B) or (B and C). C is primarily intended as a more challenging (interesting) alternative for students who have already had considerable experience with digital logic.

You may use hierarchial schematics. Where appropriate quantities are 4b, unsigned numbers. Use of a schematic drawing program for circuits is encouraged.

# A: Basic Logic

1. Implement $A > B$ out of 2-input NAND gates.

2. Show the logic (in basic gates and registers) for a simple vending machine.

   | | |
   |---|---|
   | Inputs: | n, d, and q, (nickle, dime, quarter) |
   | Output: | v (vend), nc (nickle change) |
   | Function: | Collect $\geq$30 cents, then vend and give change in nickles. |

   Don't worry about running nout of nickles to provide as change.

   Include a diagram of your state-transition graph in your writeup.

3. Using your comparison function from A.1, show logic for a spatial sorting function to sort 4, 4b inputs into ascending order.

# B: Properities of Boolean Functions

1. Consider all two-input functions. For each identify if universal; you may tie the inputs of a function to a constant 0 or 1.

2. Counting each gate as unit size, give a bound on the size difference between an optimal implementation of an arbitrary $n$-input function when the implementation may use an optimal mixture of the full set of 2-input functions from B.2 as gates compared to an implementation which uses only 2-input nand gates.

# C: Advanced Logic Problems

1. Consider n-input functions. Using only two-input NAND gates, give a bound on the number of functions that can be implemented with depth $l$. Compare your result to B.1. (Your bound should be non-trivial, but does not need to be tight.)

2. How does your answer to C.1 change if you can use any 2-input function? 3-input function?

3. Firing Squad – Design the logic for an FSmodule.

   - FSmodules can be assembled into a 1d array of arbitrary length.
   - Each FSmodule is connected exclusively to his left and right neighbors.
   - The leftmost FSmodule will get a start input.
   - FSmodules may have configuration input bits which distinguish the leftmost and rightmost modules from the rest (*i.e.* a module will be leftmost, rightmost, or a chained element).
   - All FSmodules are clocked together.
   - Data can travel from one FSmodule to his adjacent neighbor in one cycle.
   - You can have a constant number of wires between adjacent FSmodules (independent of the length of the 1d array).
   - The state in an FSmodule is finite and independent of the length of the 1d array.
   - In response to an input pulse on the leftmost module, the array of FSmodules should all, simultaneously flash an output light.
   - The number of cycles between the input pulse and the synchronized firing of the FSmodules' lights is not restricted.

   Show your state-transition graph and gate logic. Describe the operation of your solution.