CS121 Problem Set 1 (Parallel architectures)

1. Consider the problem of adding a list of n numbers. Assume that one person can add two numbers in time t_c. How long will one person take to add n numbers?

   Now assume eight people are available for adding these n numbers and that the list has already been divided into eight equal parts. Each person can add two numbers in time t_c. Furthermore, a person can pass on the result of an addition (in the form of a single number) to the person sitting next to him or her in time t_w. How long will it take to add n numbers if

   i) All eight people sat in a circle.
   ii) The eight people are sitting in two rows of four people each.

   If you are free to seat the eight people in any configuration, how would you seat them as to minimize the time taken to add the list of numbers? Is there a best configuration independent of t_c and t_w?

2. Compare the execution of the following code to calculate the greatest common divisor of x and y as performed on a SIMD versus an MIMD architecture.

```
gcd(x,y) {
while (x != y) {
      if (x > y) x = x-y;
      else y = y-x;
      }
}
```

   Suppose two arrays $x = (x_0, ..., x_{n-1})$ and $y = (y_0, ..., y_{n-1})$ are each distributed across n processors with x_i and y_i on processor i. We wish to find $z = (z_0, ..., z_{n-1})$, where $z_i = \text{gcd}(x_i, y_i)$. Assuming that a compare or a subtract instruction each takes one time step, how many time steps would it take to calculate the result when $x = (52,24)$ and $y = (12, 64)$ using

   i) A two processor SIMD architecture.
   ii) A two processor MIMID architecture.

   In each case, write a trace for the two processors, indicating clearly which instruction is executed at which time step.

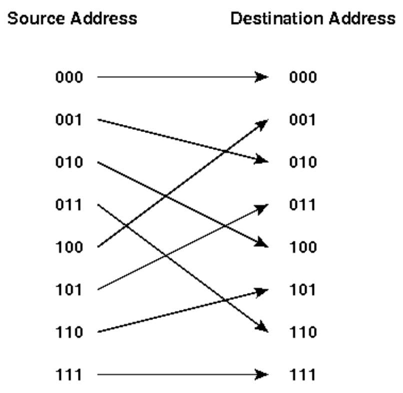
3. The *distance* between nodes u and v in a static network is the length of the shortest path from u to v. Suppose the distance between u and v in a hypercube is h. Show the following.

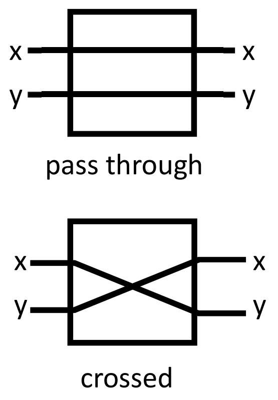
   a) There are h! different paths of length h from u to v, where we allow some edges to appear in more than one path.

b) If we have a set of paths in which no edge appears in more than one path, then the set contains at most *h* paths.

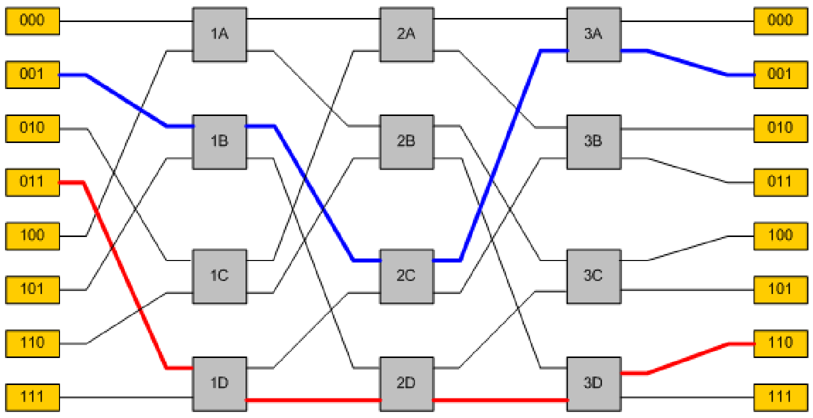
4) A *perfect shuffle* for a set of 2^n processors consists of connecting each processor *i* to another processor *j* whose ID in binary is a left rotation of *i*'s ID in binary. A perfect shuffle on 8 processors is shown below. For example, the left rotation of 100 is 001 (the 1 bit wraps around to the right side), so there is an edge between 100 and 001.

A *switch* consists of two inputs *x* and *y* and two outputs. If the inputs are *passed through* the switch, then the outputs are also *x* and *y*. Otherwise, the inputs are *crossed*, and the outputs are *y* and *x*. A switch is shown below.





An *Omega network* for a set of 2^n processors is an indirect network consisting of *n* layers of perfect shuffles of the processors. Switches in the network can be set to either pass through or crossed in order to perform routing. For example, in the Omega network below, switches 1B, 2C and 3A are set to pass through to route from processor 001 to 001, while switch 2D is set to pass through and switches 1D and 3D are set to crossed to route from 011 to 110.



Source: http://www.xatlantis.ch/education/graphics/bus_design2.png

For this problem, devise an algorithm to route a message between two processors in an Omega network. Explain why your algorithm is correct.