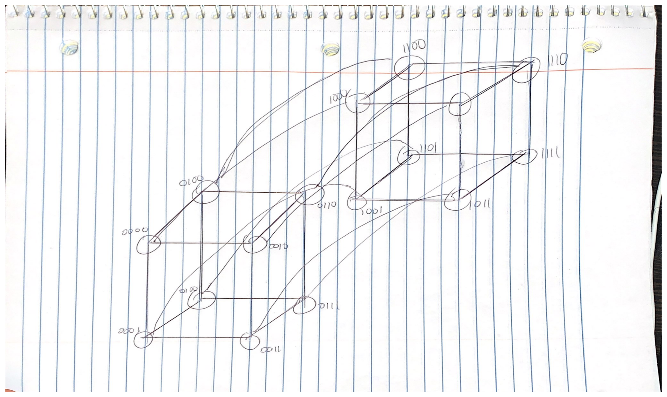
Ryan H. Johnston, ide709

Test 1 Sample Questions Solutions

1. *Architectures*

   (a) *SIMD* is *Single Instruction Multiple Data*, it runs instructions sequentially and writes to multiple data sources. *MIMD* is *Multiple Instruction Multiple Data*, it runs multiple instructions in parallel to one another while writing to mulutiple data sources. The Fox servers are not a *SIMD* machines, but an *MIMD* machines.



   (b) i.

   ii. The self-routing schema will be represented as an algorithm:

```
src: The binary label of the source node (an n-bit binary string)
dest: The binary label of the destination node (an n-bit binary string)
message: The message to be routed


function HypercubeSelfRouting(src, dest, message):
    current_node <- src

    while current_node != dest do
        For i from 1 to n do
            if current_node[i] != dest[i] then
                Toggle the i-th bit of current_node (flip from 0 to 1 or 1 to 0
                Forward message to the neighbor node with label current_node
                break (Exit the loop once the first differing bit is processed)
    end while
```

(c) The solutions are listed below:
- Diameter = O(log n)
- Max Degree = O(n)
- Number of Edges = O(n log n)
- Bisection Width = O(n)

2. *Algorithms and Analysis*

(a)
  i. Divide A[0.. n-1] into p equal chunks.
  ii. Each processor i computes the XOR of its assigned chunk.
  iii. Perform a parallel prefix operation (specifically for XOR) to combine the results of the processors. The final XOR value of the entire array will be the result of the last processor after the prefix operation.

(b)
- $T_p = O(\frac{n}{p} + \log p)$
- $W_p = O(n)$
- $S_p = \frac{T_1}{T_p} = \frac{O(n)}{O(\frac{n}{p}+logp)}$
- $p \cdot T_p = p \cdot (\frac{n}{p} + \log p) = O(n + p\log p)$

(c) For the algorithm to be cost-optimal, the cost should be O(n). Given $p \leq O(\frac{n}{logn})$, our cost is: $O(n+p\log p) = O\left(n + \frac{n}{\log n} \times \log\left(\frac{n}{\log n}\right)\right) = O(n)$

(d) Using *Exclusive Read Exclusive Write* (*EREW*). Multiple read accesses to a memory location are allowed, but multiple write accesses to a memory location are serialized.

3. *Programming*

(a)
```
#include <cstdio>
#include <cstdlib>

int main() {

    int i = 0;
    int a[4] = {0,1,2,3};
    int b[4] = {4,5,6,7};
    int scalar_product = 0;

    // execute the iteration of both vectors in
       parallel
    #pragma omp parallel for reduction(+:
        scalar_product)
    for (i = 0; i < 4; ++i) {
        scalar_product += a[i]*b[i]; // multipliy
            bits a[i]*b[i] and add it to the
            scalar_product
    }

    // print out the scalar product
    fprintf(stdout, "Final␣scalar_product␣is␣%d\n",
        scalar_product);

    return 0;
}
```

(b) **No solution**