Name _____

Interconnection Networks
EE 7725
Final Examination

14 December 1995, 10:00–12:00 C. S. T.

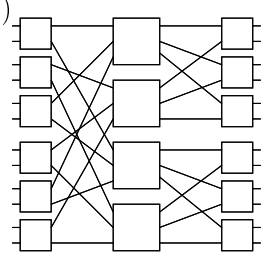
Modified.

Problem 1	_____	(25 pts)
Problem 2	_____	(30 pts)
Problem 3	_____	(20 pts)
Problem 4	_____	(25 pts)
Exam Total	_____	(100 pts)

Good Luck!

Problem 1: A *generalized butterfly network* is a generalization of a butterfly network in the same way a generalized omega network is a generalization of the omega network. A $(m_0, m_1, \ldots, m_{n-1})$-GB network is a generalized butterfly using $m_0 \times m_0$ cells in the first stage, $m_1 \times m_1$ cells in the second stage, and so on.

(a) Draw a $(2, 3, 2)$-GB network. (10 pts)



(b) Find the sequence of cells used to route request $(232, 741)$ in a $(7, 3, 8, 5)$-GB network. The cell numbers may be specified as mixed-radix numbers. Be sure to specify the radix for each number. The table of decimal to mixed radix conversions may be helpful. (15 pts)

		Mixed-Radix		Decimal	
Stage	Radix	Link	Cell	Link	Cell
0	3 8 5 7	0 6 3 6	0 6 3	33	237
1	7 8 5 3	6 6 3 0	6 6 3	273	819
2	7 3 5 8	6 0 3 4	6 0 3	93	748
3	7 3 8 5	6 0 4 1	6 0 4	148	741

Radix	232	741
MSD,...,LSD	MSD,...,LSD	MSD,...,LSD
7, 3, 8, 5	1, 2, 6, 2	6, 0, 4, 1
7, 3, 5, 8	1, 2, 4, 0	6, 0, 2, 5
7, 8, 3, 5	1, 7, 1, 2	6, 1, 1, 1
7, 8, 5, 3	1, 7, 2, 1	6, 1, 2, 0
7, 5, 3, 8	1, 4, 2, 0	6, 0, 2, 5
7, 5, 8, 3	1, 4, 5, 1	6, 0, 7, 0
3, 7, 8, 5	0, 5, 6, 2	2, 4, 4, 1
3, 7, 5, 8	0, 5, 4, 0	2, 4, 2, 5
3, 8, 7, 5	0, 6, 4, 2	2, 5, 1, 1
3, 8, 5, 7	0, 6, 3, 1	2, 5, 0, 6
3, 5, 7, 8	0, 4, 1, 0	2, 3, 1, 5
3, 5, 8, 7	0, 4, 1, 1	2, 3, 1, 6
8, 7, 3, 5	2, 1, 1, 2	7, 0, 1, 1
8, 7, 5, 3	2, 1, 2, 1	7, 0, 2, 0
8, 3, 7, 5	2, 0, 4, 2	7, 0, 1, 1
8, 3, 5, 7	2, 0, 3, 1	7, 0, 0, 6
8, 5, 7, 3	2, 1, 0, 1	7, 0, 2, 0
8, 5, 3, 7	2, 1, 0, 1	7, 0, 0, 6
5, 7, 3, 8	1, 2, 2, 0	4, 2, 2, 5
5, 7, 8, 3	1, 2, 5, 1	4, 2, 7, 0
5, 3, 7, 8	1, 1, 1, 0	4, 1, 1, 5
5, 3, 8, 7	1, 1, 1, 1	4, 1, 1, 6
5, 8, 7, 3	1, 3, 0, 1	4, 3, 2, 0
5, 8, 3, 7	1, 3, 0, 1	4, 3, 0, 6

Problem 2: A *ranker* is a device with N 1-bit inputs and N $\lceil \log_2 N \rceil$-bit outputs. As usual, let the inputs and outputs be numbered 0 to $N-1$. A sequence of 0's and 1's is placed at the inputs. The ranker puts x at output i iff there are exactly x 1's placed at inputs 0 to $i-1$. Output 0 is always has a 0. (Output 0 isn't really necessary. It's only there to make the problem description slightly simpler.) For example if sequence 0,1,1,0,1 were at the inputs, then sequence 0,0,1,2,2 would be at the outputs.

(a) Explain how a ranker and a banyan could be used to build a device that sorts 1-bit keys along with a data item associated with each key. Specify which banyan to use, any non-standard features of the cells, and any other hardware needed. *Hint: The solution is based on one of the permutation families.* (12 pts)

Use a ranker and an inverse omega network. Associate a ranker input and output with each inverse omega network input. Enter the key into the ranker and use the rank as a destination for the inverse omega. Also include the key in the message. The network cells will route a message using its rank if its key is 1; if its key is zero the message will be sent to any available cell output. Counting only requests with a key of one the inverse omega network is offered a packing connection assignment which it can realize. This properly routes all 1 keys to consecutive outputs starting at zero and all 0 keys to the other outputs.

(b) Can one conclude from the zero-one theorem that this network sorts all sequences? (8 pts)

No, the zero-one theorem applies to comparison networks; because of the ranker this is not a comparison network.

(c) Explain how to construct a self-routing permutation network using these (or any other) 1-bit sorters. (10 pts)

Use a 2^n-input 1-bit sorter to sort requests by the most-significant digit of their destinations. At the outputs of the sorter requests will be divided into two 2^{n-1}-request groups (based on the destination MSD). Use two 2^{n-1}-input 1-bit sorters to sort the next-most-significant digit (one for each group). Add 1-bit sorters for the remaining digit positions. (The cost of this self-routing network is about the same as Batcher's sorting network.)

Problem 3: A permutation is a *complete J-partition* if it is obtained by recursively J-partitioning an omega network into two parts until partitions consist of single cells. Let $\mathcal{C}$ denote the set of all complete J-partitions of a 2^n-input omega network. Show that $\Lambda \subseteq \mathcal{C}$ and $\Upsilon \subseteq \mathcal{C}$, that is, the lower- and upper-triangular permutations are a subset of the set of all complete J-partitions. (20 pts)

$\Upsilon \subseteq \mathcal{C}$ will be proven by defining $\mathcal{C}_U$, a subset of $\mathcal{C}$ and then showing that $\mathcal{C}_U = \Upsilon$. A permutation in $\mathcal{C}_U$ is obtained by partitioning an omega network using $J = \{n-1\}$ and repeating the partitioning on the two parts until they have two inputs.

Let J_x specify how the network(s) at level x are partitioned. Partition the original network, level 0, into two parts using stage 0, $J_0 = \{n-1\}$. Partition each of these (level 1) partitions into two parts using stage 0 (in the partition's numbering), $J_1 = \{n-2\}$. At level $0 \leq x < n-1$ partition using $J_x = \{n-x-1\}$. Call the "X" permutation for level 0 X_0. Call the X permutations for the two level 1 partitions $X_{1,0}$ and $X_{1,1}$. Call the X permutations for the 2^x level x partitions $X_{x,i}$ for $0 \leq i < 2^x$.

In all but the last stage, the destination of a request passing through a cell is determined by the permutation X for the partition of that cell. In stage 0 all cells belong to the same partition and so all are set to the same state, using X_0. Half of the stage-1 cells belong to one partition and are set using $X_{1,0}$, the other stage-1 cells are set using $X_{1,1}$.

The partition that a stage-x cell belongs to is determined by the x least-significant digits of the cell number and the X permutations on the path to the inputs. The partition and stage number uniquely determine the state of the cell, $X_{x,i}$.

For upper-triangular permutations the state of a cell y in stage x is $f_x(y_{(x-1)} \cdots y_{(0)})$. Equivalently, the state in terms of request (a, α) is $f_x(\alpha_{(n-1)} \cdots \alpha_{(n-x)})$. Comparing the two it can be seen that $X_{x,i}$ is the same as $f_x(i)$, and so $\Upsilon = \mathcal{C}_U \subseteq \mathcal{C}$.

A similar proof can be given for lower triangular permutations using $J = \{0\}$.

Problem 4: Answer each problem below.

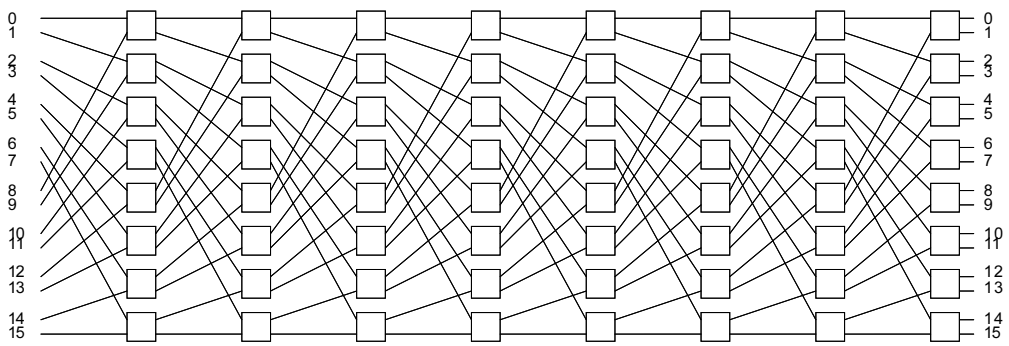
(a) As was shown in class, the cells in the input and output stages of a $(3, (2^\mu, k, 2^\mu), (k, 2^\mu, k), \mathrm{T}, \mathrm{T})$ Clos network need not realize all permutations. If these cells were implemented using crossbars, and each cell had to be identical, how many crosspoints would be saved? Explain. (5 pts)

Zero. Even though such cells need not realize all permutations, there must still be a path between every input and output. Each such path would require a crosspoint, for a total of $(2^\mu)^2$ crosspoints.

(b) As we all know, the cells in a *d-dilated* network have $2d$ inputs and $2d$ outputs. A naïve implementation would use $(2d)^2$ crosspoints. The alert engineer might realize that a significant number of crosspoints can be eliminated if each cell input in the upper set connects to exactly one of the upper set of outputs (its counterpart, perhaps). Complete the idea, draw a diagram of the cell, and compute the number of crosspoints. The less-expensive cell should be able to handle any conflict that the naïve cell could.(5 pts)

In the modified crossbar each upper set input link connects to all lower set output links and each lower set input link connects to all upper set output links. As stated in the problem, each upper set input link connects to one upper set output link and each lower set input link connects to one lower set output link. Such a crossbar has $2d(d+1)$ crosspoints.

(c) One futile approach to proving that an omega-omega network ($2n$ shuffle stages) can realize all permutations is to show that it is topologically equivalent to a Beneš network (ignoring one of the stages in the omega-omega network). Prove that this approach is futile (that the omega-omega is not topologically equivalent to a Beneš network when ignoring a stage). The 16-input omega-omega network below can be used to help explain the proof. *Hint 1: A mapping function is not necessary. Hint 2: The proof can be quickly explained. Hint 3: The proof must consider the center stages of the network.* (5 pts)



Consider a cell in stage $0 \leq x < n$ of a Beneš network. This cell can reach 2^{n-x+1} cells in stage $2n-2-x$. In an omega omega network a stage $0 \leq x < n$ cell can reach $\min\{2^{n-1}, 2^{2n-1-2x}\}$ cells in stage $2n-2-x$. Since the number of reachable cells is different the two networks are not topologically equivalent.

This paragraph is part of the next two problems. In the network using feedback to limit the size of saturation trees, a damping method was employed to prevent a saturation tree from reforming at outputs which undergo a hot-to-cold transition.

(d) Explain how the damping method works. (5 pts)

A number M is computed based on the amount of time an output is hot. Packets held at input i are released when $i \equiv \phi \pmod{M}$, where ϕ is an integer clock value.

(e) Suppose a similar feedback method were used in k-ary n-cubes, but without the damping mechanism. Would the size (number of requests) in a re-formed saturation tree be a function of k (for a fixed N)? Explain. (5 pts)

Yes. In an omega network (and most MINs) all inputs are at the same distance from every output, so if messages are simultaneously released they will simultaneously converge on the output or cells leading to the output. In k-ary n-cubes, the distance between input/output pairs varies. When $k = N$ ($n = 1$) the saturation tree will be smallest because messages from nodes furthest from the hot node will arrive after many other messages were received. When $k = 2$ messages will arrive over a smaller time interval and so will find a larger tree.

Have a good vacation.