Routing and Processor Allocation on a Hypercycle-based Multiprocessor

N.J. Dimopoulos, S. Radhakrishnan, D. Radvan
Department of Electrical and Computer Engineering
University of Victoria
Victoria, B.C.
CANADA

Abstract[†]

In this work, we present a brief overview of the Hypercycles which is a class of multidimensional graphs. These graphs can be used to structure Concurrent Multiprocessor Systems so as to best match the requirements of a given application.

We present the backtrack-to-the-origin-and-retry routing as applied to Hypercycles, as well as its performance evaluation based on simulations.

Also, we present the a First Fit processor allocation strategy for Hypercycle-based Multiprocessors, and prove that such a strategy is statically optimal.

Finally, we discuss the implementation status of a router component for the backtrack-to-the-origin-and-retry routing.

1. Introduction

Message passing concurrent computers such as the Hypercube[6, 11], Cosmic Cube[9], MAX[7, 8], consist of several processing nodes that interact via messages exchanged over communication channels linking these nodes into one functional entity.

There are many ways of interconnecting the computational nodes, the Hypercube, Cosmic Cube, and the Connection Machine[12] having adopted a regular interconnection pattern corresponding to a binary n-dimensional cube, while MAX adopts a less structured, yet unspecified topology.

Hypercycles[4,5,8] can be considered as products of "basic" graphs that use a rich set of component "basic" graphs ranging in complexity from simple rings to the fully connected ones. Because the "basic" graphs are defined, we are able to provide analytical expressions for routing. Our goals are

(a) To provide computer interconnection networks that match the node requirements of a given embedded system.

(b) To increase throughput of a given network by providing routing expressions that can be computed analytically (and hence are candidates for VLSI implementation) and which provide a number of alternate paths from a source to a destination.

The Hypercycles, being regular graphs, retain the advantages of easy routing and regularity. Yet, since we are dealing with a class, rather than isolated graphs, we have the flexibility of adopting any particular graph (from the class) that closely matches the requirements of a given application.

Given a hypercycle-based concurrent computer system, one needs to allocate resources (i.e. processors) to incoming jobs. In this work we are proposing an efficient first-fit processor allocation strategy, and prove that such a strategy is statically optimal.

This work is divided into these parts. Section 2.0 introduces the Mixed Radix System, Section 3.0 introduces the Hypercycles. Sections 4.0 and 5.0 present the backtrack-to-the-origin-and-retry routng while Section 6.0 introduces the First Fit processor allocation strategy, and discusses simulation results of the First Fit allocation strategy for various Hypercycles. Finally Section 7.0 gives the status of the implementation of a backtrack-to-the-origin-and-retry routng component.

2. Mixed Radix Number System

The mixed radix representation [2], is a positional number representation, and it is a generalization of the the standard b-base

[†] This work was supported the Natural Sciences and Engineering Research Council of Canada under grant #OGP0001337, by the Institute for Robotics and Intelligent Systems under the National Networks of Centers of Excellence Program, and by the Canadian Microelectronics Corporation.

representation, in that it allows each position to follow its own base independently of the other.

Given a decimal number M factored into r factors as $M = m_1 \times m_2 \times m_3 \times \cdots \times m_r$ then any number $0 \le X \le M-1$ can be represented as $(X)_{m_1 m_2 \ldots m_r} = x_1 x_2 \ldots x_r \mid_{m_1 m_2 \ldots m_r}$ where $0 \le x_i \le (m_i - 1)$; $i = 1,2,\ldots,r$ and the x_i's are chosen so that $X = \sum_{i=1}^{r} x_i w_i$

where $w_i = \dfrac{M}{m_1 m_2 \cdots m_i}$

3. Hypercycles.

An undirected graph $\mathfrak{G}$ is defined as: $\mathfrak{G} = (\mathfrak{N}, \mathfrak{E})$, where $\mathfrak{N}$ is the set of nodes $\mathfrak{N} = \{ \alpha_i \; ; \; i = 1,2,\ldots N\}$, and $\mathfrak{E}$ the set of edges

$$\mathfrak{E} = \left\{ e_{ij_i} = \left(\alpha_i , \beta_{j_i} \right) j_i = 1,2,\ldots,d_i \; ; \; i = 1,2,\ldots,\mathrm{N} \right\}$$

with $\alpha_i , \beta_{j_i} \in \mathfrak{N}$ and d_i the degree of node α_i.

An r- dimensional Hypercycle, is the following regular undirected graph:

Hypercycles, have degrees [5] $d = \sum_{i=1}^{r} f(m_i ,\rho$

where

$$f(m_i ,\rho_i) = \begin{cases} 2\rho_i & \text{if} \quad 2\rho_i < m_i \\ m_i - 1 & \text{if} \quad 2\rho_i = m_i \end{cases}$$

and diameter k

$$k = \sum_{i=1}^{r} \left\lceil \frac{\lfloor m_i / 2 \rfloor}{\rho_i} \right\rceil$$

The n-cube is a Hypercycle, wi $M = 2 \times 2 \times \cdots \times 2 = 2^n$ and $\rho = 1,1,1,\ldots,1$.

3.1. Routing

Hypercycles, have routing properties that a similar to those of the n-cube. Given nodes $(\alpha)_{m_1 m_2 \ldots m_i \ldots m_r} = \alpha_1 \alpha_2 \ldots \alpha_i \ldots \alpha_r$ and $(\alpha^*)_{m_1 m_2 \ldots m_i \ldots m_r} = \alpha_1 \alpha_2 \ldots \xi \ldots \alpha_r$, a wal from node α to node α^*, can be constructed follows:
$\alpha_1 \alpha_2 \ldots \alpha_i \ldots \alpha_r$, $\alpha_1 \alpha_2 \ldots \xi_1 \ldots \alpha_r$, $\alpha_1 \alpha_2 \ldots \xi_2 \ldots \alpha_r$, $\alpha_1 \alpha_2 \ldots \xi \ldots \alpha_r$. such that[1]

$$\boxed{\begin{array}{l} \xi_{j_i+1} = \begin{cases} \left(\xi_{j_i} + \rho_i\right) \bmod m_i & \text{if} \quad \left[\left(\xi - \xi_{j_i}\right) \bmod m_i = \left|\xi_{j_i}, \xi\right|\right] > \rho_i & \text{(a} \\ \left(\xi_{j_i} + \left|\xi_{j_i}, \xi\right| \bmod \rho_i\right) \bmod m_i & \text{if} \quad \left[\left(\xi - \xi_{j_i}\right) \bmod m_i = \left|\xi_{j_i}, \xi\right|\right] > \rho_i \text{ and } \left|\xi_{j_i}, \xi\right| \bmod \rho_i \neq 0 & \text{(b} \\ \left(\xi_{j_i} - \rho_i\right) \bmod m_i & \text{if} \quad \left[\left(\xi_{j_i} - \xi\right) \bmod m_i = \left|\xi_{j_i}, \xi\right|\right] > \rho_i & \text{(c} \\ \left(\xi_{j_i} - \left|\xi_{j_i}, \xi\right| \bmod \rho_i\right) \bmod m_i & \text{if} \quad \left[\left(\xi_{j_i} - \xi\right) \bmod m_i = \left|\xi_{j_i}, \xi\right|\right] > \rho_i \text{ and } \left|\xi_{j_i}, \xi\right| \bmod \rho_i \neq 0 & \text{(d} \\ \xi & \text{if} \quad \left|\xi_{j_i}, \xi\right| \le \rho_i & \text{(e} \end{cases} \\ \xi_0 = \alpha_i \qquad\qquad \xi_{max} = \xi \end{array}}$$

Eqn. 3.1

$\mathfrak{G}^{\rho}_{\boldsymbol{m}} = \left\{ \mathfrak{N}^{\rho}_{\boldsymbol{m}} , \mathfrak{E}^{\rho}_{\boldsymbol{m}} \right\}$ where $\boldsymbol{m} = m_1, m_2, m_3, \ldots, m_r$ a mixed radix, $\rho = \rho_1, \rho_2, \ldots, \rho_r$; $\rho_i \le m_i / 2$ the connectivity vector, determining the connectivity in each dimension which ranges from a cycle ($\rho_i = 1$) to fully connected ($\rho_i = \lfloor m_i / 2 \rfloor$), and $\mathfrak{N}^{\rho}_{\boldsymbol{m}} = \{0, 1, 2, \ldots, M-1\}$. Given $\alpha, \beta \in \mathfrak{N}^{\rho}_{\boldsymbol{m}}$ then $(\alpha, \beta) \in \mathfrak{E}^{\rho}_{\boldsymbol{m}}$ if and only if there exists
$1 \le j \le r$ such that $\beta_j = (\alpha_j \pm \xi_j) \bmod m_j$ with
$1 \le \xi_j \le \rho_j$ and $\alpha_i = \beta_i$; $i \ne j$

Equation 3.1.1 defines all the minimu length paths from a source to a destination ir single dimension. Parts (a), and (c) constitut greedy strategy where the maximum st towards the destination is taken. Parts (b) a (d) form alternate paths by allowing the st described in part (e) to be taken earlier. Obse that there is only one step of length smaller th the maximum, and when it is taken it guaranteed that the remaining steps will maximal. This is because

$$\left| \left(\xi_{j_i} \pm \left|\xi_{j_i}, \xi\right| \bmod \rho_i\right) \bmod m_i , \xi \right| \bmod \rho_i = 0$$

Given an origin $(\alpha)_{m_1 m_2 \ldots m_r} = \alpha_1 \alpha_2 \ldots \alpha_r$ a a destination $(\beta)_{m_1 m_2 \ldots m_r} = \beta_1 \beta_2 \ldots \beta_r$ th

[1] We define $|a, b| = \mathbf{min}\{(a-b)\mathbf{mod} m_i , (b-a)\mathbf{mod} m_i\}$

distinct walks of minimum length that connect them are constructed according by sequentially modifying the source address, each time substituting a source digit by an intermediate walk digit determined according to equation 3.1.3, until the destination is formed. The following walk connects **source** to **destination.** the destination is reached. The following walk connects **source** to **destination.**

source = $\alpha_1 \alpha_2 \alpha_3 \ldots \alpha_r$; $\alpha_1 \xi_1 \alpha_3 \ldots \alpha_r$; $\alpha_1 \xi_1 \psi_1 \ldots \alpha_r$;
$\alpha_1 \xi_2 \psi_1 \ldots \alpha_r$; $\alpha_1 \xi_2 \psi_2 \ldots \alpha_r$; ...; $\alpha_1 \xi_2 \beta_3 \ldots \alpha_r$;...;
$\beta_1 \beta_2 \beta_3 \ldots \beta_r$ = **destination**

Figure 1a., gives an example of two distinct walks of equal length that connect a source to a destination, for a Hypercycle.

4. Deadlock Avoidance in Routing.

In section 3.1, we have given a method that establishes at least one path from a source to a destination node. In this part, we are concerned with optimally choosing one of the paths. Routing must be efficient and deadlock free. Deadlock occurs when resources (in this case node to node communication segments) are allocated so that the completion of a partial path requires a segment already allocated to a different partial path which in turn waits for a segment in the first partial path. It is obvious that no messages can propagate over the deadlocked paths, and the only remedy is to break the already established and deadlocked partial paths and try again.

Deadlock may occur easily in cases where the segments that form the paths are chosen at random. Certain routing algorithms (e.g. e-cube routing) prevent deadlocks by ordering the resources (channels) to be allocated. Thus a lower order resource cannot be committed if a needed higher order resource cannot be obtained. The disadvantage of this approach in an interconnection network is that it limits the number of paths connecting a source to a destination to exactly one, even though several alternate free paths may exist at a particular moment. We are proposing to adopt a strategy where deadlocks are avoided by requiring a blocked partial path to backtrack to its origin and retry.

5. Backtrack-to-the-origin-and-retry routing

For Hypercycle-based interconnection networks, because of the existence of cycles in each dimension, the use of an e-cube type routing that prevents deadlocks, is impractical. We are proposing instead a deadlock avoiding routing strategy. According to our backtrack-to-the-origin-and-retry routing we identify, at each node, all nodes that can be used for the continuation of the path. For all such identified nodes, we also identify the corresponding ports that can be used in order to continue the path. Since several paths may be forming in parallel, some of these ports may already be allocated to some other path. After excluding all the allocated ports, we select one of the remaining free ports at random. The subsequent link in the path is established is then established through the selected port, and the procedure repeats itself until the destination is reached, or no free ports could be found. If no free ports are to be found at an intermediate node in the path, then a break is returned to the origin (through the already established partial path to the blocking node), the partial path is dissolved, and a new attempt for the creation of the required path is initiated. This routing strategy avoids deadlocks through backtracking, and also guarantees that the formed path will be of a minimum length, since each subsequent link is selected according to equation (3.1.1). The backtrack-to-the-origin-and-retry routing is a type of two-phase locking [10], where as resources we consider the various links necessary for the completion of the source-to-destination circuit.

We have used Extend™† to construct a simulator capable of simulating any Hypercycle based network. For this simulator, we implemented both the backtrack-to-the-origin-and-retry as well as the e-cube routing strategies. The e-cube routing can only be used for binary cube networks. For each node, we assumed a Poisson message generator which generates packets with uniform distribution of destinations. Each packet carries the destination address which is used for routing. Links are assigned priorities, so that collisions can be resolved. We assumed a packet transmission time (over an established source to destination path) of 100 simulation-clock ticks. We use the simulator to obtain the throughput and delay characteristics of several networks for both e-cube and backtrack-to-the-origin-and-retry in terms to the offered load. Both the offered load and the throughput were normalized in terms to the maximum capacity of each network taken to be proportional to the number of links in the corresponding graph. The average delay was expressed in actual time units necessary to establish a source-to-destination circuit. Simulation results were reported in [4] and some typical behavior is depicted in figs. 2 and 3.

† Extend is a trademark of Imagine That inc.

The performance of the backtrack-to-the-origin-and-retry for both binary cubes and hypercycles of similar sizes, is superior to that of the e-cube as it can be seen in fig. 2. This is attributed to the fact that the backtrack-to-the-origin-and-retry can use alternative paths to the destination instead of the single path allotted by the e-cube routing. The additional advantage of the backtrack-to-the-origin and-retry is its inherent fault tolerance. Indeed, if one of links in the network failed, it could be marked as permanently busy, and packets would be routed around it. This obviously is not the case for the e-cube routing.

Under heavy loads, e-cube routing has slightly higher throughput rates for a given load than backtrack-to-the-origin. Backtrack routing does not effectively route packets of longer distances because the path is dissolved as soon as a blocked route is encountered, which has a high probability of occurrence under a heavy load. An adaptive algorithm which alternates between backtrack-to-the-origin and e-cube routing under heavy loads may solve this instability.

For graphs of higher degrees, fig 3 , shows the effect of alternate paths on system performance. The 7 node Hypercycle has only one path choice per source/destination connection (because it is a fully connected graph). This effectively reduces the system to e-cube style routing only. The 15-node ($\mathcal{G}_{53}^{21}$) and 16-node ($\mathcal{G}_{44}^{22}$) graphs offer two alternate routes between source and destination and therefore provide higher system throughput. Generally, system throughput and delay are functions of both average distance and the average number of alternate paths between any two nodes.

6. Processor Allocation

In this section, we shall investigate the problem of allocating processors in a hypercycle-based concurrent computer to incoming jobs. For our purposes, a job is considered as a collection of tasks which can execute concurrently. A job therefore requires a portion of the processors in order to execute efficiently. We consider a sequence of incoming jobs, each requesting a minimum number of nodes. We shall use a first fit strategy, where the first unallocated fragment that can accommodate the request is assigned to the incoming job. We shall prove that this strategy is statically optimal, and it requires minimal searching. Our strategy is a generalization of a similar one developed by Chen and Shin [3] for Hypercube Multiprocessors.

Definition 1. A subgraph is closed under the hypercycle routing as defined above, iff all the intermediate nodes required to complete a path between a source and a destination node are nodes of the subgraph.

Definition 2. Given two nodes $n_p = b_n b_{n-1} \dots b_0$ and $n_q = \tilde{b}_n \tilde{b}_{n-1} \cdots \tilde{b}_0$ we say that n_p lies before n_q and denote $n_p \prec n_q$ iff there exists an index $0 \le \rho \le n$ such that $b_\rho < \tilde{b}_\rho$ and $b_l = \tilde{b}_l;\ \ l = n, n-1, \cdots, \rho+1$.

The above defined ordering relation arranges the set of nodes in a unique sequence.

Definition 3. A region $\Re$ is defined as a set of consecutive nodes. i.e. $\Re(n_\alpha, n_\beta) \equiv [n_\alpha, n_\beta] = \{n \mid n_\alpha \prec n \prec n_\beta\} \cup \{n_\alpha, n_\beta\}$

Definition 4. Given two regions h_p and h_q we say that h_p lies before h_q, and denote $h_p \prec h_q$, iff for every node $b_n b_{n-1} \dots b_0 = n_p \in h_p$ and $\tilde{b}_n \tilde{b}_{n-1} \cdots \tilde{b}_0 = n_q \in h_q$ there exists an index $0 \le \rho \le n$ such that $b_\rho < \tilde{b}_\rho$ and $b_l = \tilde{b}_l;\ \ l = n, n-1, \cdots, \rho+1$

Definition 5. A fragment is a region whose nodes have identical most significant digits. Nodes wit identical most significant digits belong to the same fragment. We denote a fragment as $\mathcal{F}(b_n b_{n-1} \dots b_{n-k+1}) \equiv b_n b_{n-1} \dots b_{n-k+1}{}^{*n-k} = \{n \mid n = b_n b_{n-1} \dots b_{n-k+1} \beta_{n-k} \dots \beta_0,\ \beta_j \in \{0,1,\dots m_j\};\ j = n-k, n-k-1, \dots, 0\}$

Theorem 1. Fragments are closed under the hypercycle routing.

Proof If α and β are two nodes in a fragment $\mathcal{F}(b_n b_{n-1} \dots b_{n-k+1})$, then $\alpha = b_n b_{n-1} \dots b_{n-k+1} \alpha_{n-k} \dots \alpha_0$ and $\beta = b_n b_{n-1} \dots b_{n-k+1} \beta_{n-k} \dots \beta_0$. According to the routing equation (0), any intermediate ξ node on the path from the source α to the destination β will be of the form $\xi = b_n b_{n-1} \dots b_{n-k+1} \xi_{n-k} \dots \xi_0 \in \mathcal{F}(b_n b_{n-1} \dots b_{n-k+1})$. ■

Because of the ordering relation $\prec$ presented in Definition 2 above, one can arrange the nodes of any Hypercycle in a linear fashion and number them from 0 to M -1. The First Fit allocation strategy utilizes a list of allocation bits numbered from 0 to M -1 and corresponding to the nodes in the Hypercycle. If a processor has been allocated to a job, its corresponding allocation bit is set to 1, otherwise it is cleared to 0.

The First Fit allocation strategy searches the list until it discovers a region of unallocated processors corresponding to a fragment with a size equal to that of the request. If such a region is found, the corresponding allocation bits are set to 1, and the strategy tries to accommodate a subsequent request. The allocation bits are cleared as soon as the job assigned to the corresponding nodes is terminated.

In detail, the First Fit Allocation Strategy can be described as follows.

First Fit Allocation Strategy

1. Let $|f_j| = m_{l_j} m_{l_j - 1} \cdots m_1 m_0$ be the size of the smallest fragment that will fit the jth request I_j.
2. Find the least integer m such that all the allocation bits in the region $[m|f_j|, (m+1)|f_j| - 1]$ are zeroes, and set all the allocation bits in this region to 1.
3. Allocate nodes with addresses in the region to request I_j.

Processor Relinquishment

1. Reset the allocation bits of the released region to zero.

Definition 6. A fragment $b_n b_{n-1} \ldots b_{n-k+1} *^{n-k}$ is said to be a hole if and only if this region is available, but at least one of the[2] $b_n b_{n-1} \ldots [b'_{n-k+1}] *^{n-k}$ is not.

Lemma 1 Let $\{h_i(j)\}_{i=1}^{u}$ be the hole sequence resulting from the allocation of fragments to the request sequence $\{I_r\}_{r=1}^{j}$. If $h_i(j) = b_n b_{n-1} \ldots b_{n-k+1} *^{n-k}$ for some i, then $b_{n-k+1} \neq 0$.

Proof. Suppose that $b_{n-k+1} = 0$. Then the region $b_n b_{n-1} \ldots 0 *^{n-k}$ is a hole, which means that at least one of the fragments $b_n b_{n-1} \ldots [0'] *^{n-k}$ is unavailable. But this contradicts the first fit search of the allocation strategy. ■

[2] $[b'_{n-k+1}]$ denotes any of $\{0, 1, \ldots, m_{n-k+1}\} - b_{n-k+1}$, where the - (minus sign) denotes the set theoretic subtraction.

Let $\{h_i(j)\}_{i=1}^{u}$ denote the sequence of holes resulting from the allocation of fragments to the sequence of requests $\{I_r\}_{r=1}^{j}$. Arrange these holes in a lexicographical order so that if $p < q$ then $h_p(j) \prec h_q(j)$.

Lemma 2. Given a sequence of holes $\{h_i(j)\}_{i=1}^{u}$ resulting from the sequence of requests $\{I_r\}_{r=1}^{j}$, then

(a) $|h_i(j)| \leq |h_{i+1}(j)|$; $i < u$

(b) If there exists k such that $|h_k(j)| < |h_{k+1}(j)|$ then $\sum_{p=1}^{k} |h_p(j)| < |h_k(j)|$.

Proof. (a) Suppose that there exists i such that $h_1 = |h_i(j)| > |h_{i+1}(j)| = h_2$. This means that hole $h_{i+1}(j) = b_n b_{n-1} \cdots b_{n-k-1} *^{n-k}$ is located after hole $h_i(j) = \tilde{b}_n \tilde{b}_{n-1} \cdots \tilde{b}_{n-k-\sigma} *^{n-k-\sigma-1}$. Since the fragment $b_n b_{n-1} \cdots b_{n-k-1} *^{n-k}$ is a hole, then the fragment $b_n b_{n-1} \cdots 0 *^{n-k}$ of size h_2 is occupied. But this is contrary to the first-fit allocation strategy, since the fragment $\tilde{b}_n \tilde{b}_{n-1} \cdots \tilde{b}_{n-k-\sigma} 0 \cdots 0 *^{n-k}$ of size h_2 lying before $b_n b_{n-1} \cdots 0 *^{n-k}$ is available (as part of hole $h_i(j)$).

(b) Given a hole $h = b_n b_{n-1} \cdots b_{n-k-1} *^{n-k}$, it has size $|h| = m_{n-k} \times m_{n-k-1} \times \cdots \times m_0$ and there are at most[3] $m_{n-k+1} - 1$ other holes of the same size. Thus if

[3] There cannot be any holes of size $|h|$ before or after the region of hole h. Observe that since $h = b_n b_{n-1} \cdots b_{n-k-1} *^{n-k}$, the fragment $f = b_n b_{n-1} \cdots 0 *^{n-k}$ is allocated (from lemma 1). If there were a such hole before the current region, then the fragment f should not have been allocated since its allocation would have contradicted the first-fit allocation strategy. Similarly, if there is a hole of size $|h|$ in a region after the current region, then the fragment $f' = b'_n b'_{n-1} \cdots 0 *^{n-k}$ should be allocated. But this is also contrary to the first-fit strategy since the hole h is before the fragment f'.

$h_{i+1}(j)=b_n b_{n-1}\cdots b_{n-k-1} *^{n-k}$ is a hole, then there exist at most
$(m_0 - 1)$ holes of size 1
$(m_1 - 1)$ holes of size m_0
.
.
.
$(m_{n-k} - 1)$ holes of size $m_{n-k-1}\times\cdots\times m_0$ while the size of $h_{i+1}(j)$ is $|h_{i+1}(j)|=m_{n-k}\times m_{n-k-1}\times\cdots\times m_0$.
Thus

$$\sum_{\rho=1}^{n-k}|h_\rho(j)|=\sum_{\rho=1}^{n-k}(m_\rho-1)\prod_{\lambda=-1}^{\rho-1}m_\lambda=m_{n-k}\times m_{n-k-1}\times\cdots\times m_0-1<$$
$$< m_{n-k}\times m_{n-k-1}\times\cdots\times m_0=|h_{i+1}(j)|$$

■

Definition 7. Given a sequence of requests $\{I_r\}_{r=1}^{j}$ and if each request is limited to a fragment, then an allocation strategy is called statically optimum if it can accommodate any sequence of requests $\{I_r\}_{r=1}^{j}$ such that

$$\sum_{r=1}^{j}|I_r|<m_n\times m_{n-1}\times\cdots\times m_0=|G_m^\rho|.$$

Theorem 2. The first-fit allocation strategy is statically optimal.

Proof Let $\{I_r\}_{r=1}^{j}$ a sequence of requests such that

$\sum_{r=1}^{j}|I_r|\le m_n\times m_{n-1}\times\cdots\times m_0=|G_m^\rho|$. Also assume that the sequence of requests $\{I_r\}_{r=1}^{j-1}$ can be accommodated by the first-fit strategy, and denote by $\{h_i(j-1)\}_{i=1}^{u}$ the sequence of holes resulted from this allocation. We shall prove that $\{I_r\}_{r=1}^{j}$ can also be accommodated, i.e. $|h_u(j-1)|\ge|I_j|$.

i. Assume that $|h_u(j-1)|>|h_{u-1}(j-1)|$ then, by lemma 2,

$$\sum_{\rho=1}^{u-1}|h_\rho(j-1)|<|h_u(j-1)| \qquad (1)$$

Also

$$\sum_{\lambda=1}^{u}|h_\lambda(j-1)|+\sum_{r=1}^{j-1}|I_r|=|G_m^\rho|\Rightarrow$$
$$|h_u(j-1)|+\sum_{\lambda=1}^{u-1}|h_\lambda(j-1)|+\sum_{r=1}^{j-1}|I_r|=|G_m^\rho|\Rightarrow$$
$$|h_u(j-1)|+\sum_{\lambda=1}^{u-1}|h_\lambda(j-1)|=|G_m^\rho|-\sum_{r=1}^{j-1}|I_r|\ge|I_j| \qquad (2)$$

because of the assumption $\sum_{r=1}^{j}|I_r|\le|G_m^\rho|$.
Combining (1) and (2), one obtains

$$2|h_u(j-1)|>|I_j| \qquad (3)$$

Since both $h_u(j-1)$ and I_j are fragments, and since there are no fragments available such that $2|h_u(j-1)|>|I_j|>|h_u(j-1)|$, then $|I_j|\le|h_u(j-1)|$.

ii. Assume that $|h_u(j-1)|=|h_{u-1}(j-1)|$. Denote by ρ the number of holes in this region. From the definition of a hole and footnote 2, $\rho<m_{n-n_u}$[4]. Then, in the sequence of holes $\{h_i(j-1)\}_{i=1}^{u}$ produced by the first $(j-1)$ requests, there are ρ holes of equal size followed by zero or more smaller holes. Thus

$$|h_u(j-1)|=|h_{u-1}(j-1)|=\cdots=|h_{u-(\rho-1)}(j-1)|>$$
$$>|h_{u-(\rho-1)-1}(j-1)|\ge\cdots\ge|h_1(j-1)| \qquad (4)$$

Then by lemma 2, we have

$$\sum_{\lambda=1}^{u-(\rho-1)-1}|h_\lambda(j-1)|<|h_{u-(\rho-1)}(j-1)| \qquad (5)$$

Also, because the sum of all the holes and allocated fragments sums up to the number of nodes in the Hypercycle, we have

[4]We assume that

$$h_u(j-1)=m_n m_{n-1}\cdots m_{n-n_u} *^{n-n_u-1}$$

$$\sum_{\lambda=1}^{u}|h_\lambda(j-1)|+\sum_{r=1}^{j-1}|I_r|=|G_m^\rho|\Rightarrow$$

$$\sum_{\lambda=u-(\rho-1)}^{u}|h_\lambda(j-1)|+\sum_{\lambda=1}^{u-(\rho-1)-1}|h_\lambda(j-1)|+\sum_{r=1}^{j-1}|I_r|=|G_m^\rho|\Rightarrow$$

$$\rho|h_u(j-1)|+\sum_{\lambda=1}^{u-(\rho-1)-1}|h_\lambda(j-1)|=|G_m^\rho|-\sum_{r=1}^{j-1}|I_r|>|I_j| \qquad (6)$$

because of the assumption $\sum_{r=1}^{j}|I_r|<|G_m^\rho|$.

Combining now (5) and (6), we obtain

$$(\rho+1)|h_u(j-1)|>|I_j| \qquad (7)$$

Since both $h_u(j\text{-}1)$ and I_j are fragments, and since there are no fragments available such that

$m_{n-n_u}|h_u(j-1)|>|I_j|>|h_u(j-1)|$, then $|I_j|\le|h_u(j-1)|$. ■

6.1. First Fit Allocation Strategy Simulation

We have constructed a simulator capable of simulating the behavior of the First Fit Allocation Strategy for various Networks. The simulator allocates jobs that fit in fragments of the graph under simulation. We have assumed an uniform distribution of fragment sizes. Job run times are uniformly distributed between 1 and a maximum that is user declared. Job arrival is Poison distributed. TABLE I. presents the measured Average Delay and Node Utilization[5] for several hypercycles For these simulations we used job run-times uniformly distributed in the interval [1,7].. These measurements correspond to job streams just before saturation.

It is interesting to compare the hypercycles with **m** =2 3 4 4 and **m** =4 4 3 2. Although these are equivalent networks, the hypercycle with **m** =4 4 3 2 has lower delays. This is due to the existence of a larger number of small fragments in this network as compared to the one with **m** =2 3 4 4 . Also, the same observation holds for the network with **m** = 4 4 3 3 2

7. Discussion

The backtrack-to-the-origin-and-retry routing, as discussed above, is currently being implemented in hardware. Figure 4 gives a block diagram of an r-dimensional Hypercycle Routing Engine, while Figure 5 presents the implementation of a version of such a routing engine incorporating four Next Port Generators (and hence capable of handling four-dimensional graphs) and 16 ports, using Northern Telecom's 1.2μ double poly, double metal process. Early simulations of the component yield propagation delays of less than 50ns per port allocation. The component is currently under fabrication by the Canadian Microelectronics Corporation.

As it can be seen in Figure 5. we are implementing our routing engine as a system having four modules. The destination address is used in the Next-Port Generator to generate all possible ports that can be used in forming the path to the required destination address. Subsequently, the Port Validator masks out the ports which are currently used by other paths. Finally, The Port Selector, selects at random one of the validated ports which is then used to continue the circuit towards the required destination. It is worth noting that the system is programmable, in the sense that it needs the parameters m, ρ as defined earlier and which define the structure of the network, as well as ξ which the address of the current node.

[5]Node Utilization is defined as the average number of nodes that are allocated to jobs during the simulation interval.

REFERENCES

1. Berge C., *Principles of Combinatorics* Academic Press, New York and London, 1971.
2. Bhuyan, L. N., and D. P. Agrawal, "Design and Performance of Generalized Interconnection Networks" *IEEE Trans. Comput.* Vol. C-32, No. 12, pp. 1081-1090, Dec. 1983.
3. Chen, M. S., K. G. Shin "Processor Allocation in an N-Cube Multiprocessor Using Gray Codes" *IEEE Trans. Comput.*, Vol. C-36, No. 12, pp. 1396-1407 (Dec. 1987).
4. Dimopoulos, N. J., D. Radvan, K. F. Li "Performance Evaluation of the Backtrack to the Origin and Retry Routing for Hypercycle-based Interconnection Networks" *Proceedings of the Tenth International Conference on Distributed Computer Systems*, pp. 278-284, Paris, France (May 1990).
5. Dimopoulos, N.J., R.D. Rasmussen, G.S. Bolotin, B.F. Lewis, R. M. Manning "Hypercycles, Interconnection Networks with simple Routing Strategies" *Proceedings of the Canadian Conference on Electrical and Computer Engineering*,

pp. 577-580, Vancouver, B.C., Canada, Nov. 1988.

6. Peterson, J.C., J. O. Tuazon, D. Lieberman, M. Pniel "The MARK III Hypercube - Ensemble Concurrent Computer" *Proceedings of the 1985 International Conference on Parallel Processing* pp. 71-73, Aug. 20-23 1985.

7. Rasmussen, R. D., G. S. Bolotin, N. J. Dimopoulos, B. F. Lewis, and R. M. Manning "Advanced General Purpose Multicomputer for Space Applications" *Proceedings of the 1987 International Conference on Parallel Processing.* pp. 54-57. (Aug. 1987)

8. Rasmussen, R. D., N. J. Dimopoulos, G. S. Bolotin, B. F. Lewis, and R. M. Manning "MAX: Advanced General Purpose Real-Time Multicomputer for Space Applications" *Proceedings of the IEEE Real Time Systems Symposium* pp. 70-78, San Jose, CA.,(Dec. 1987).

9. Seitz, C. L. "The cosmic cube" *CACM* vol. 28, pp.22-33, Jan. 1985

10. Tanenbaum, A. .S., *Operating Systems: Design and Implementation* Prentice Hall, 1987.

11. Tuazon, J. O., J. C. Peterson, M. Pniel, and D. Lieberman "Caltech/JPL MARK II Hypercube Concurrent Processor"*Proceedings of the 1985 International Conference on Parallel Processing* pp. 666-673, Aug. 20-23 1985.

12. Waltz, D. L. "Applications of the Connection Machine" *IEEE Computer* January 1987, pp.85-97

TABLE I

GRAPH	Average Job Interarrival Time	Node Utilization	Average Delay
m=2 2 2 2 2	3.0	.678	20.11
m=4 4 3 2	3.0	.586	21.7
m=2 3 4 4	4.0	.667	24.7
m=4 4 3 3 2	2.0	.535	15.4

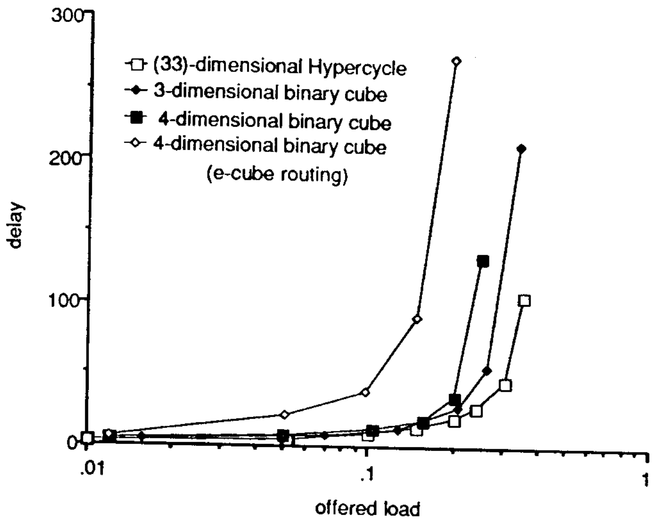


Figure 2. Delay vs. offered load for the 4-cube using backtrack-to-the-origin and e-cube routing. The offered load is normalized to the capacity of the interconnection network. The delay is normalized to the data transmission time.

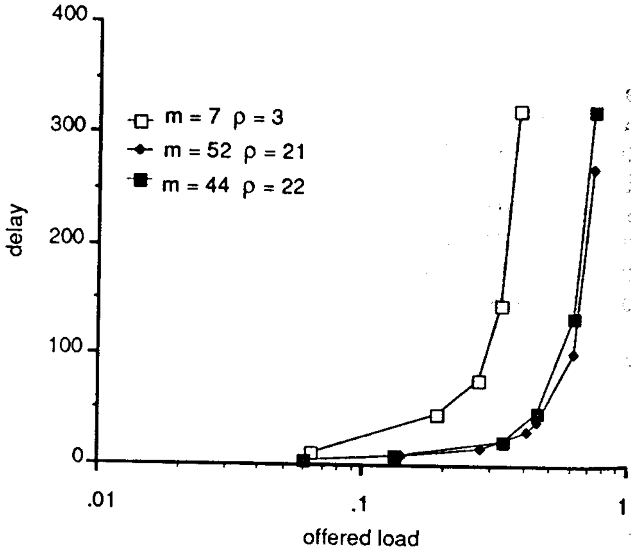
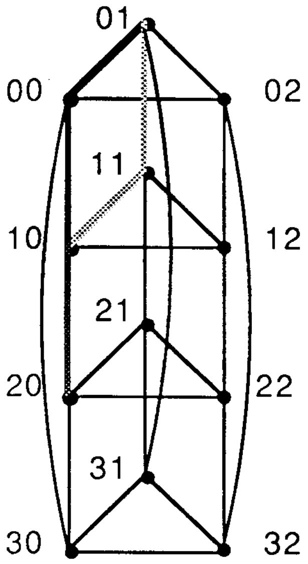
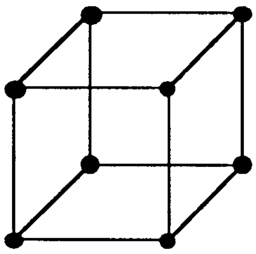


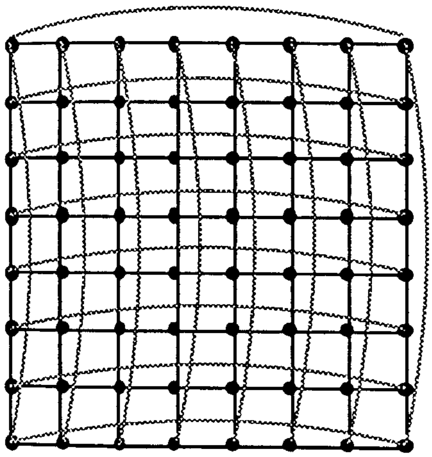
Figure 3. Delay vs. offered load for several graphs of degree six. The offered load and delay are normalized to the capacity of the interconnection network.



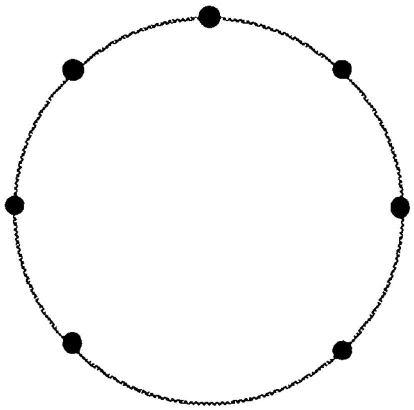
a. Hypercycle $\mathfrak{G}_{4\,3}^{1\,1}$



b. Binary 3-cube $\mathfrak{G}_{222}^{111}$



c. Illiac IV $\mathfrak{G}_{8\,8}^{1\,1}$



d. Ring $\mathfrak{G}_{7}^{1}$

Figure 1. Examples of Hypercycles.