Deadlock-preventing routing in hypercycles

Routage empêchant les impasses lors d'hypercycles

N.J. Dimopoulos *and* R. Sivakumar,

Department of Electrical and Computer Engineering, University of Victoria, P.O. Box 3055, Victoria, B.C. V8W 3P6

Hypercycles make up a class of multidimensional graphs obtained by allowing each dimension to incorporate more than two elements and a cyclic interconnection. Hypercycles offer simple routing and the ability, given a fixed degree, to choose among a number of graphs of varying size. These graphs can be used in the design of interconnection networks for distributed systems tailored specifically to the topology of a particular application. We present and prove a deadlock-preventing routing strategy for a subset of hypercycles, and a VLSI hypercycle router component which implements the deadlock-preventing routing.

Les hypercycles forment une classe de graphes multidimensionnels qui sont obtenus en permettant à chaque dimension d'incorporer une connexion cyclique et plusieurs éléments. Les hypercycles facilitent le routage et offrent la possibilité, pour un degré donné, de choisir parmi des graphes de différentes tailles. Ces graphes peuvent être employés pour le design de réseaux d'interconnexions de systèmes distribués, réseaux qui sont spécialement adaptés à la topologie d'une application donnée. Nous présentons cette stratégie anti-impasse dans le cas de sous-ensembles d'hypercycles et en faisons la démonstration pour un élément de routage d'hypercycles en VLSI.

I. Introduction

Message-passing concurrent computers such as the Hypercube [1], Cosmic Cube [2], MAX [3]-[4], iPSC [5] and J-Machine [6] 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, iPSC and Cosmic Cube, for example, have adopted a regular interconnection pattern corresponding to a binary n-dimensional cube, while MAX adopts an unstructured topology.

Hypercycles [4]-[7] are products [8] of circulants. The circulants used range in complexity from simple rings to fully connected graphs. Hypercycles are generalizations of many popular interconnection networks. Binary n-cubes, k-ary n-cubes [9], generalized hypercubes, rings, toruses, etc., are examples of hypercycles.

Many properties and algorithms that have been introduced for hypercubes, toruses, k-ary n-cubes, etc., can be used directly, or can be extended to the entire class of hypercycles [10]. This makes it possible to choose a topology that best suits the system requirements of a specific class of applications.

A number of different routing policies have been introduced for a variety of interconnection networks. Thus, in Hypercubes, e-cube routing [11] prevents deadlocks by ordering the resources (i.e., channels or virtual channels) comprising a path, thereby guaranteeing that no circular dependencies exist for any paths formed. Similar approaches have been devised for toruses and k-ary n-cubes [11]-[12], where virtual channels are introduced in order to break any circular dependencies. A different approach to deadlock avoidance has been introduced with the Hyperswitch [13] and the various backtracking strategies for hypercycles [7]. There, deadlocks are avoided by forcing a forming path to backtrack and try again. These strategies appear to have excellent throughput because they tend to utilize more available paths leading to the destination, but they suffer from thrashing at high offered loads. Routing complexity has been extensively studied, and lower bounds have been established for a variety of routing disciplines and graphs by many authors. Excellent overviews can be found in [14] and [15].

In this work, we present a deadlock-preventing routing algorithm for certain hypercycles. This routing does not make use of virtual channels and thus it is well suited for circuit-switching environments.

This work is divided into the following parts. Section II introduces hypercycles and discusses their properties. Section III presents a deadlock-preventing routing strategy for a class of hypercycles. Section IV discusses the router component which has been designed and fabricated, and finally we conclude with Section V.

II. Introduction to hypercycles

A. Mixed-radix number system

The mixed-radix representation [16] is a positional number representation, and it is a generalization of the standard b-base representation in that it allows each position to follow its own base independently of the other.

Given a number, M, factored as $M = m_n \times m_{n-1} \times \ldots \times m_1$, any number $0 \le X \le M - 1$ can be represented as $(X)_{m_n m_{n-1} \ldots m_1} = x_n x_{n-1} \ldots x_1 \,|_{m_n m_{n-1} \ldots m_1}$, where $0 \le x_i \le (m_i - 1)$; $i = 1, 2, \ldots, n$, and the x_i's are chosen so that $X = \sum_{i=1}^{n} x_i w_i$ and $w_i = \dfrac{M}{m_n m_{n-1} \ldots m_i}$.

B. Hypercycles

An n-dimensional hypercycle is a regular undirected graph: $\mathcal{G}_m^{\rho} = \left\{\mathcal{N}_m^{\rho}, \mathcal{E}_m^{\rho}\right\}$, where $\mathcal{N}_m^{\rho}$ is the set of nodes, $\mathcal{E}_m^{\rho}$ is the set of edges, $\boldsymbol{m} = m_n, m_{n-1}, \ldots, m_1$ (a mixed radix), $\rho = \rho_n, \rho_{n-1}, \ldots, \rho_1$; $\rho_i \le m_i/2$ (the connectivity vector), determining the connectivity in each dimension, ranging from a ring ($\rho_i = 1$) to fully connected ($\rho_i = \lfloor m_i / 2 \rfloor$), and $\mathcal{N}_m^{\rho} = \{0, 1, 2, \ldots, M-1\}$. Given $\alpha, \beta \in \mathcal{N}_m^{\rho}$, we have $(\alpha, \beta) \in \mathcal{E}_m^{\rho}$ if and only if there exists $1 \le j \le n$ such that $\beta_j = (\alpha_j \pm \xi_j) \bmod m_j$ with $1 \le \xi_j \le \rho_j$ and $\alpha_i = \beta_i; i \ne j$.

The degree,

$$d = \sum_{i=1}^{n} f(m_i, \rho_i), \text{ where } f(m_i, \rho_i) = \begin{cases} 2\rho_i & \text{if } 2\rho_i < m_i \\ m_i - 1 & \text{if } 2\rho_i = m_i, \end{cases}$$

and diameter, $k = \sum_{i=1}^{n} \left\lfloor \frac{\lfloor m_i / 2 \rfloor}{\rho_i} \right\rfloor$, of the hypercycle have been derived in [7].

The n-cube is a hypercycle with $M = 2 \times 2 \times \ldots \times 2 = 2^n$ and $\rho = 1, 1, 1, \ldots, 1$.

C. Routing

Hypercycles have routing properties similar to those of the n-cube. Given nodes $(\alpha)_{m_n m_{n-1} \ldots m_1} = \alpha_n \alpha_{n-1} \ldots \alpha_i \ldots \alpha_1$ and $(\alpha^*)_{m_n m_{n-1} \ldots m_1} = \alpha_n \alpha_{n-1} \ldots \xi \ldots \alpha_1$, a walk from α to α^* is formed as follows: $\alpha_n \alpha_{n-1} \ldots \alpha_i \ldots \alpha_1, \alpha_n \alpha_{n-1} \ldots \xi_1 \ldots \alpha_1, \alpha_n \alpha_{n-1} \ldots \xi_2 \ldots \alpha_1, \ldots, \alpha_n \alpha_{n-1} \ldots \xi \ldots \alpha_1$, such that

$$\xi_{j_i+1} = \begin{cases} (\xi_{j_i} + \rho_i) \bmod m_i & \text{if } (\xi - \xi_{j_i}) \bmod m_i = |\xi_{j_i}, \xi| > \rho_i \quad \text{(a)} \\ (\xi_{j_i} + |\xi_{j_i}, \xi| \bmod \rho_i) \bmod m_i & \text{if } (\xi - \xi_{j_i}) \bmod m_i = |\xi_{j_i}, \xi| > \rho_i \\ & \quad \text{and } |\xi_{j_i}, \xi| \bmod \rho_i \neq 0 \quad \text{(b)} \\ (\xi_{j_i} - \rho_i) \bmod m_i & \text{if } (\xi_{j_i} - \xi) \bmod m_i = |\xi_{j_i}, \xi| > \rho_i \quad \text{(c)} \\ (\xi_{j_i} - |\xi_{j_i}, \xi| \bmod \rho_i) \bmod m_i & \text{if } (\xi_{j_i} - \xi) \bmod m_i = |\xi_{j_i}, \xi| > \rho_i \\ & \quad \text{and } |\xi_{j_i}, \xi| \bmod \rho_i \neq 0 \quad \text{(d)} \\ \xi & \text{if } |\xi_{j_i}, \xi| \leq \rho_i \quad \text{(e)} \end{cases} \tag{1.1}$$

$$\xi_o = \alpha_i, \tag{1.2}$$

$$\xi_{\max} = \xi. \tag{1.3}$$

Equations (1.1)-(1.3) define all the minimum-length paths from a source to a destination in a single dimension. Parts (a) and (c) constitute a greedy strategy where the maximum step towards the destination is taken. Parts (b) and (d) form alternate paths by allowing the step described in part (e) to be taken earlier. Observe that there is only one step of length smaller than the maximum, and when it is taken it is guaranteed that the remaining steps will be maximal.

Given an origin, $(\alpha)_{m_n m_{n-1} \ldots m_1} = \alpha_n \alpha_{n-1} \ldots \alpha_i \ldots \alpha_1$, and a destination, $(\beta)_{m_n m_{n-1} \ldots m_1} = \beta_n \beta_{n-1} \ldots \beta_i \ldots \beta_1$, we construct distinct paths of minimum length, connecting them by sequentially modifying the source address, each time substituting an intermediate walk digit determined according to (1.1)-(1.3) for a source digit, until the destination is formed. The following walk connects source to destination:

$$\textbf{source} = \alpha_n \alpha_{n-1} \ldots \alpha_3 \alpha_2 \alpha_1, \alpha_n \alpha_{n-1} \ldots \alpha_3 \xi_1 \alpha_1, \alpha_n \alpha_{n-1} \ldots \psi_1 \xi_1 \alpha_1$$

$$\alpha_n \alpha_{n-1} \ldots \psi_2 \xi_1 \alpha_1, \alpha_n \alpha_{n-1} \ldots \beta_3 \xi_1 \alpha_1, \ldots \beta_n \beta_{n-1} \ldots \beta_3 \beta_2 \beta_1 = \textbf{destination}.$$

If only the greedy strategy is followed, the result is a total of $l = \binom{q}{q_n, q_{n-1}, \ldots q_1} = \frac{q!}{q_n! q_{n-1}! \ldots q_1!}$ connecting paths of minimum length, with q_i being the distance along dimension i. The total distance between source and destination is given as $\text{dis}(a, b) = q = \sum_{i=1}^{n} q_i$. We call such paths *greedy paths*.

As an example, in Fig. 1(a), both paths 01 11 10 20 and 01 00 10 20 have the same minimum length and connect source 01 to destination 20.

III. Deadlock-preventing routing in hypercycles

In Section II.C above, we presented a routing function that establishes at least one path of minimum length from a source to a destination node. In this section, we are concerned with choosing one of these paths. Routing must be efficient and deadlock-free. Deadlocks must be prevented, avoided, or detected and broken. Deadlocks occur when resources (in this case node-to-node communication channels) 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 prevent deadlocks by ordering the resources (communication channels) to be allocated. Thus a lower-order resource cannot be committed if a needed higher-order resource cannot be obtained. It has been proven that the *e*-cube routing [11] prevents deadlocks in the case of the hypercube.

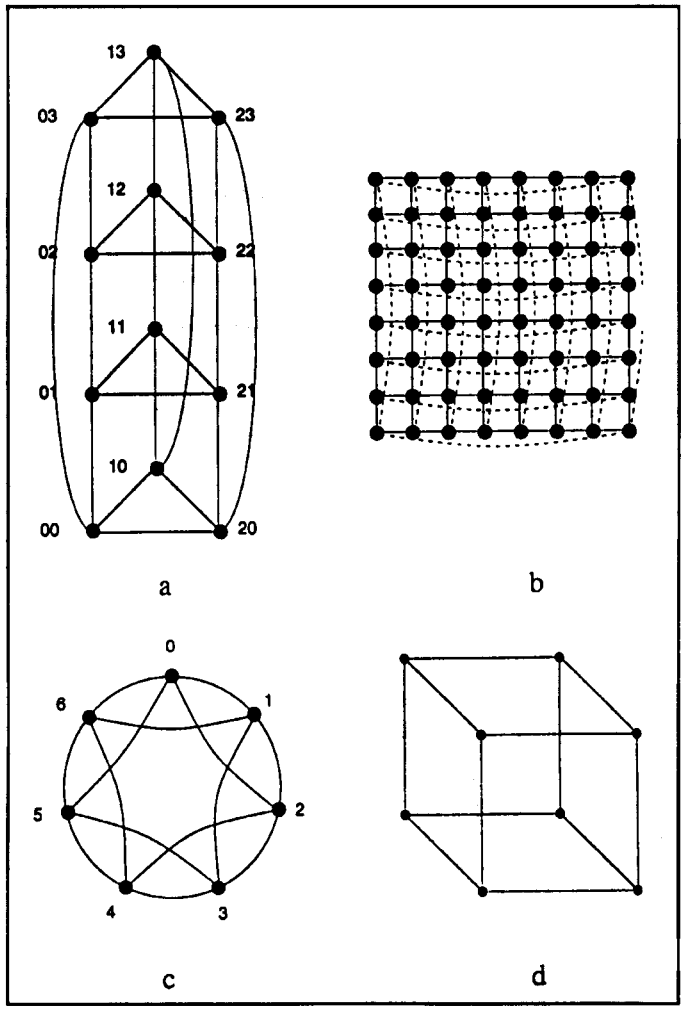

Figure 1: *Examples of hypercycles: a) hypercycle* $\mathcal{G}_{34}^{11}$; *b) ILIAC IV* $\mathcal{G}_{88}^{11}$; *c) circulant* $\mathcal{G}_{7}^{2}$; *d) hypercube* $\mathcal{G}_{222}^{111}$.

In this section, we introduce a deadlock-preventing routing for certain hypercycles and prove that it is indeed deadlock-preventing. To do this, we utilize the notion of a dependence graph, which describes interdependencies between the various communication circuits as they are formed. The proof technique consists of proving that such a dependence graph is acyclic under all possible communication patterns allowed by the routing chosen.

Definition 1: Given a graph, $\mathcal{G}$, on which a circuit-switching routing is used, we denote by P_i the partially completed path between a source, S_i, and a destination, D_i; by I_i the last node of the partial path P_i; and by L_i the set of all legal outgoing links from node I_i which can be used in order to complete the partial path to the destination D_i.

Observe that if $L_i = \emptyset$, then $I_i = D_i$, i.e., the path is complete, while if $P_i = \emptyset$, then $I_i = S_i$, and the path has not yet commenced.

Definition 2: With reference to a graph, $\mathcal{G}$, on which circuit-switching routing is used, we define the corresponding dependence graph, $\mathcal{P} = \{\mathcal{N}, \mathcal{E}\}$, where $\mathcal{N}$ is the set of partially completed paths, and $(P_i, P_j) \in \mathcal{E}$ *iff* $\exists\, l \in L_i$, such that $l \in P_j$.

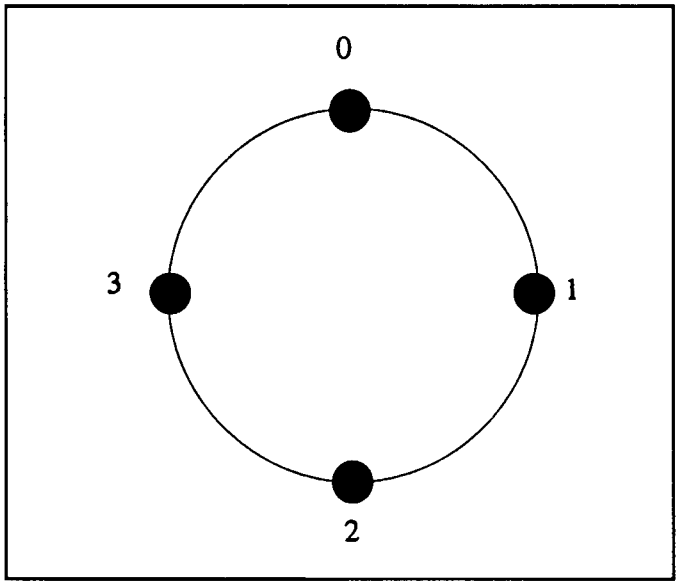

Figure 2: *Four-node hypercycle* $\mathcal{G}^1_4$.

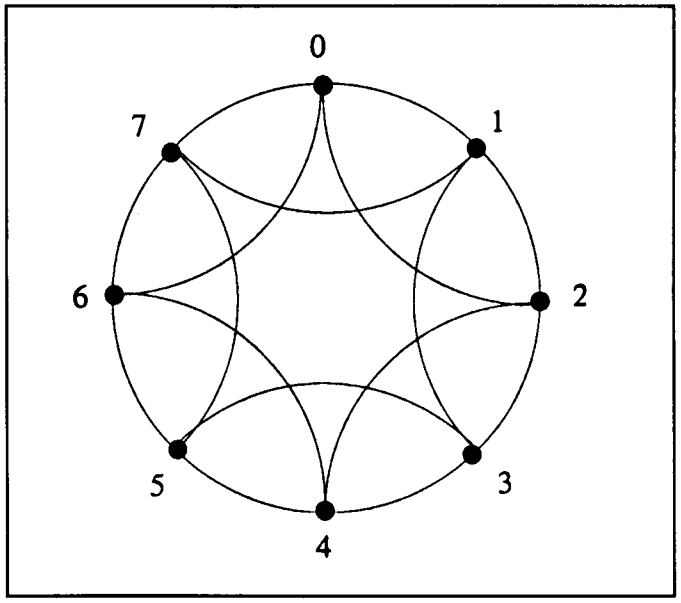

Figure 3: *Eight-node hypercycle* $\mathcal{G}^2_8$.

Definition 3: We say that a set of source-destination pairs, (S_i, D_i); $i = 1, 2, ..., \mu$, is deadlocked if $\forall l \in L_i\ \exists j$, such that $l \in P_j$; $i = 1, 2, ..., \mu$, where L_i and P_i are the sets of outgoing links and already formed partial paths respectively.

In other words, a set of source-destination pairs is deadlocked if all the links leading out of the formed partial paths are already contained in the said partial paths.

Lemma 1: If a set of source-destination pairs, (S_i, D_i); $i = 1, 2, ..., \mu$, is deadlocked, then the corresponding dependence graph contains at least one cycle.

Proof: If there are no cycles in the dependence graph, then it possesses at least one terminal node, P. If P is a terminal node, there does not exist a succeeding partial path connected to P, i.e., $\forall l \in L\ \exists j$, such that $l \in P_j$. But this contradicts Definition 3. Q.E.D.

In order to achieve deadlock-preventing routing, we introduce asymmetry in the way each node routes messages. Take as an example the four-node hypercycle, $\mathcal{G}^1_4$, as depicted in Fig. 2. Node 2 can be reached from node 0 by travelling either clockwise or counter-clockwise. Similarly, node 3 can be reached from node 1 by traversing the hypercycle in two directions. If the directionality is chosen at random, or identically for all nodes, deadlocks can occur. For example, suppose that at the same time, paths 012 and 210 are in the process of being formed, and assume that the first has already acquired segment 01, while the latter has acquired segment 21. Both paths meet at node 1 and cyclically wait for each other *ad infinitum*. If, on the other hand, the directionality alternates, deadlocks are prevented, as we shall prove.

In a similar fashion, one can impose asymmetry in the routing for larger graphs such as the $\mathcal{G}^2_8$, as depicted in Fig. 3, where if nodes 0, 1, 4 and 5 route on a clockwise orientation and nodes 2, 3, 6 and 7 on a counter-clockwise orientation, deadlocks do not occur. We shall name this type of asymmetric routing the odd/even preference routing.

Definition 4: In one-dimensional hypercycles, $\mathcal{G}_m{}^\rho$, where a destination can be reached through two alternate routes (i.e., when $(\alpha - \beta)$ **mod** $m = (\beta - \alpha)$ **mod** $m = |\alpha, \beta|$, where α is the source and β is the destination), we define the odd/even preference routing so that in the greedy mode, routes originating at source nodes with an even ρ-quotient† proceed in the clockwise (counter-clockwise) direction, while

† Given integers α and ρ, we define as the ρ-quotient of α the quantity $\lfloor \alpha / \rho \rfloor$.

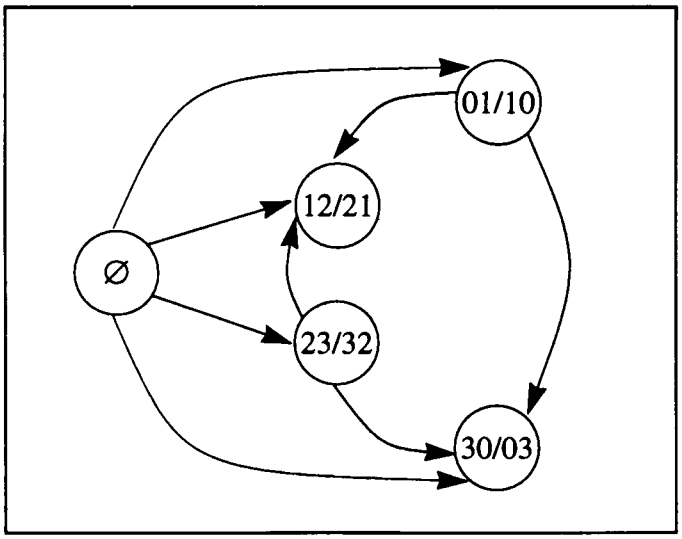

Figure 4: *Dependence graph for* $\mathcal{G}^1_4$.

routes originating at nodes with an odd ρ-quotient proceed in the opposite direction. If α* denotes the next intermediate node, then

$$\alpha^* = \begin{cases} (\alpha + \rho) \bmod m \text{ if } \lfloor \alpha / \rho \rfloor \text{ is even} \\ (\alpha + \rho) \bmod m \text{ if } \lfloor \alpha / \rho \rfloor \text{ is odd.} \end{cases}$$

Example 1: To illustrate the definitions stated above, we present the dependence graph for the odd/even routing in $\mathcal{G}^1{}_4$ in Fig. 4. To make the representation of the dependence graph legible, we have omitted the nodes corresponding to paths 012, 103, 230 and 321. These nodes are terminal nodes and as such do not contribute to any possible loops. As can be verified, this dependence graph is devoid of loops, and thus the odd/even routing in $\mathcal{G}^1{}_4$ is deadlock-preventing. If arbitrary routing were permitted, then edges such as the one depicted by the dashed line in the diagram would be permitted, giving rise to loops and therefore possible deadlocks during routing.

Theorem 1: In $\mathcal{G}^1{}_4$ the odd/even preference routing is deadlock-preventing.

Proof: By construction. Example 1 constructed the dependence graph for $\mathcal{G}^1{}_4$ and odd/even preference routing. A full proof can be found in [17]. Q.E.D.

In the subsequent treatment, we make use of the terms *maximum-length* and *link length*. One-dimensional hypercycles can be visualized as rings with chords. Thus, any node will have links connecting it to its two immediate neighbours lying to its left and right (i.e., to nodes $(\alpha \pm 1) \bmod m$), but also to more distant nodes through the chords (i.e., to nodes $(\alpha \pm x) \bmod m$, $x \leq \rho$). Links therefore can be differentiated by whether or not they correspond to the chords, as well as by the length of the chord.

Definition 5: The length of a link connecting nodes α and $(\alpha \pm x) \bmod m$ in a one-dimensional hypercycle, $\mathcal{G}_m{}^\rho$, is defined to be x. If $x = \rho$, then such a link is called a *maximum-length* link.

Theorem 2: One-dimensional hypercycles, $\mathcal{G}_m{}^\rho$, of diameter two and $\lfloor m / 2 \rfloor < 2\rho$ have greedy routing as outlined in section II.C, which is deadlock-preventing.

Proof: Since $\lfloor m / 2 \rfloor < 2\rho$, any two-link path from a source to a destination consists of a maximum-length link of length ρ followed by a link of length less than ρ. Thus, in the dependence graph, all the partially completed paths, P_i, such that $P_i \neq \emptyset$, and $L_i \neq \emptyset$, consist of maximum links, while all the requested links in the sets L_i are not maximum links. Thus $\forall l_j \in L_j \nexists \sigma$, such that $l_j \in P_\sigma$. Therefore, a cycle cannot exist in such a dependence graph, and the routing is deadlock-preventing. Q.E.D.

Theorem 3: Fully connected graphs are deadlock-preventing.

Proof: Since the graph is fully connected (diameter 1), all partial paths between any source-destination pair have lengths of at most one. Therefore, the corresponding dependence graph is devoid of cycles since, if (P_i, P_j) is an edge in the dependence graph, $P_i = \emptyset$, and thus, according to Definition 2, there cannot be another edge of the form (P_x, P_i). Q.E.D.

Proofs of Theorems 4, 5 and 6 can be found in the Appendix.

Theorem 4 establishes a class of one-dimensional hypercycles (circulants) of diameter 2, for which deadlock-preventing routing can be established. These hypercycles can be distinguished by the fact that source-destination pairs at distance two can be reached either through a maximum length link followed by a non-maximum-length link, or, if two maximum-length links are required for the path, then there exist two alternate paths which connect the source to the destination. The fact that the paths are composed of a maximum followed by a non-maximum-length link in conjunction with the existence of two alternate paths for the cases where two maximum-length links are required, is used to guarantee that cycles do not exist in the dependence graph.

Theorem 4: One-dimensional hypercycles, $\mathcal{G}_m{}^\rho$, with $m = 4\rho$, have a deadlock-preventing odd/even preference routing.

We are now ready to define a deadlock-preventing routing for multidimensional product graphs. In the generalized *e*-cube routing, a link cannot be reserved unless all the necessary links at higher dimensions have been allocated to the path forming. Specifically, consider a start node, $S = \alpha_n\alpha_{n-1}...\alpha_j...\alpha_i...\alpha_1$, which is connected to node $D_1 = \alpha_n\alpha_{n-1}...\delta_j...\alpha_i...\alpha_1$ through link l_1 and to node $D_2 = \alpha_n\alpha_{n-1}...\alpha_j...\delta_i...\alpha_1$ through link l_2. Then link l_1 is considered to be a higher-dimension link as compared to link l_2, and this is denoted by $l_2 < l_1$. For example, in $\mathcal{G}^{11}{}_{34}$ (see Fig. 1), link (00, 01) is of a lower dimension as compared to link (00, 20), while links (00, 01) and (00, 02) are of the same dimension.

Theorem 5: Generalized *e*-cube routing is deadlock-preventing on a graph that is the product of fully connected graphs.

Examples of graphs mentioned in Theorem 5 are the Hypercube and the Generalized Hypercubes [16].

Theorem 6: Generalized *e*-cube routing is deadlock-preventing on a graph that is the product of graphs, each of which possesses deadlock-preventing routing.

IV. Router implementation

We have designed, implemented and tested a hypercycle router component that implements the deadlock-preventing routing discussed in Section III. The architecture of an *n*-dimensional router is given in Fig. 5. The major functional blocks of the router are:

1. *n* modules of *e*-cube decoder, one for each of the *n* dimensions;
2. *n* modules of next port generators (NPGs);
3. port selector and validator.

The *e*-cube decoders and next port generators establish, in each dimension, whether the odd-even preference routing is required as per Definition 4 and Theorems 1 and 4. The next port generator implements the greedy and odd-even preference routing. Specifically, it calculates, given the destination, the address of the current node and the topology as specified by *m* and ρ, the port that may be used to continue the path along the dimension controlled by the next port generator in question. Notice that for each dimension *i*, there are $2\rho_i$ edges connecting to neighbouring nodes in the graph. Each of these edges represents a bidirectional communications channel. A number of ports on each router are used to deliver the messages to the communication media used to realize the interconnect. Depending on the topology, subsets of these ports are allocated to each dimension, and the next port generator calculates the appropriate port to be used [18]. The set of next port generators in the router will therefore yield one or more ports over which the path may be extended towards the destination. The port selector and validator module is responsible for choosing one of the ports suggested by the next port generators to continue the path. According to the generalized *e*-cube routing as discussed above, the path must be extended along the port corresponding to the highest dimension. If this port is free (i.e., it is not being used by any other path), it is selected by the port selector; otherwise, a *No_Ports_Available* signal is generated, indicating that the path is blocked and a wait is required until the port is freed. Specifically, the port selector and validator module performs the following functions:

1. checks whether the programmed hypercycle configuration is admissible for deadlock-preventing routing;
2. validates the computed ports by comparison with available free ports;
3. selects the highest-dimensioned port if it is free; otherwise, generates a *No_Ports_Available* signal;
4. signals when the next computed port address is ready;
5. informs whether the destination has been reached.

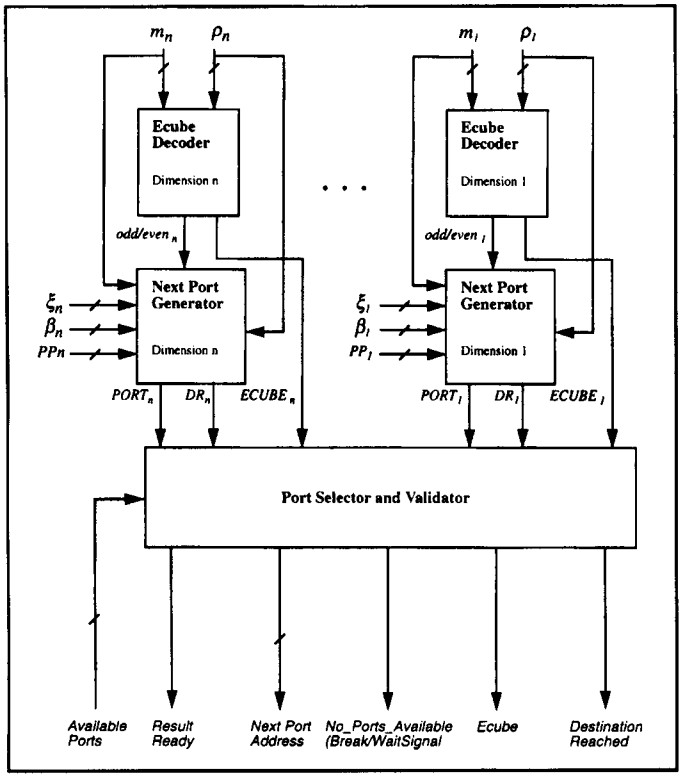

Figure 5: *Block diagram of the deadlock-preventing router.*

A VLSI chip implementing the discussed deadlock-preventing routing strategy for hypercycles has been designed and fabricated using 1.2 μm NTE CMOS4S technology (Figs. 6-7). The micrograph of the chip is given in Fig. 5. The designed router is programmable and can be configured for any hypercycle interconnection network with up to 15 nodes per dimension, and up to four dimensions. There is a maximum of 16 ports. Examples of large deadlock-preventing hypercycles implementable with this router include the $\mathcal{G}_{8888}^{2222}$ with 4096 nodes, the $\mathcal{G}_{12\ 12\ 4\ 4}^{3\ 3\ 1\ 1}$ with 2304 nodes, the $\mathcal{G}_{11\ 12\ 3\ 4}^{3\ 3\ 1\ 1}$ with 1584 nodes, etc. The chip has 51 pins housed in a 68-pin PGA and it incorporates 19 624 transistors.

Each decision cycle of the hypercycle router is composed of a *Load* followed by an *Execute* phase. During the *Load* phase, the router is loaded with the Destination address and the list of available ports. The *Execute* phase computes the next port address and outputs it, provided that the port is free. A *No_Ports_Available* signal is generated if the required port is not free, and a *Destination_Reached* signal is generated when the present node is indeed the intended message destination.

The *Load* phase requires two clock cycles (one each to load the available ports and the destination), while the *Execute* phase requires four clock cycles for a total of six clock cycles per decision.

The chip was characterized by a worst-case propagation delay of 78 ns. The propagation delay is defined as the time between the onset of the load phase and the time at which the port number appears at the

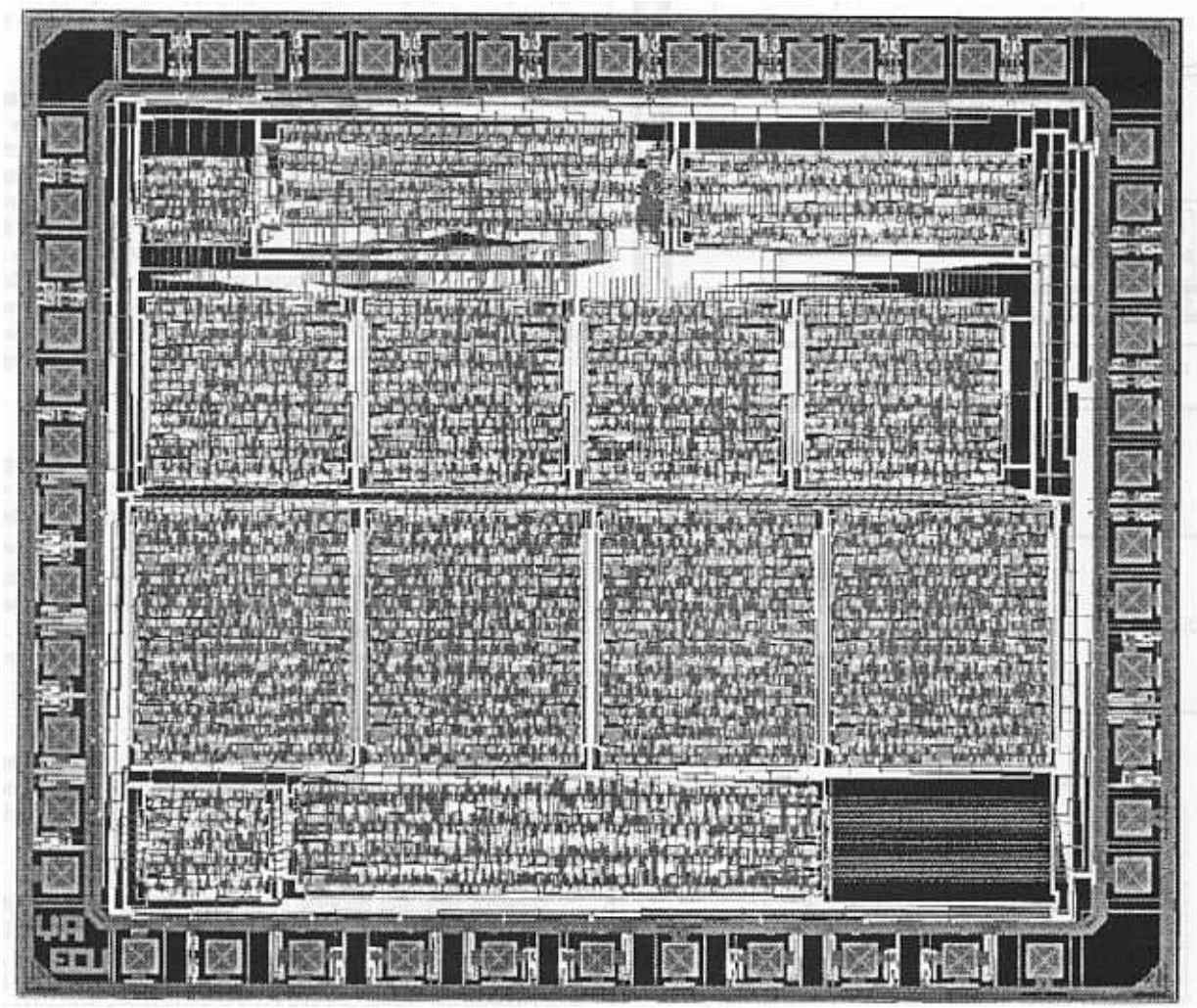

Figure 6: *Layout of the deadlock-preventing router in 1.2 μ NTCMOS4S technology.*

output. Because of the large propagation delay incurred at the output pads, the load phase can be overlapped with the output phase of the previous decision cycle, with a commensurate decrease in the latency to 62 ns.

The deadlock-preventing router discussed in this section will form part of a routing engine capable of being configured for a number of hypercycles and a variety of routing policies. The structure of such a routing engine is depicted in Fig. 8. It comprises a controller, a crossbar, an interface and a number of routers. The controller intercepts messages arriving at the routing engine either from the local node or the neighbours. It extracts the destination from the header of the message and passes it to one of the routers, which will compute the port through which the message is to be forwarded. Then the controller configures the switch to effect the connection prescribed by the router. Observe that the existence of a number of different routers (i.e., backtracking [21], deadlock preventing, etc.) allows the use of the most appropriate routing policy based on the traffic characteristics.

V. Conclusions and discussion

In this work, we presented the hypercycles, a class of multidimensional graphs which are essentially generalizations of several well-known graphs including the n-cube, toruses, k-ary n-cubes, rings, etc.

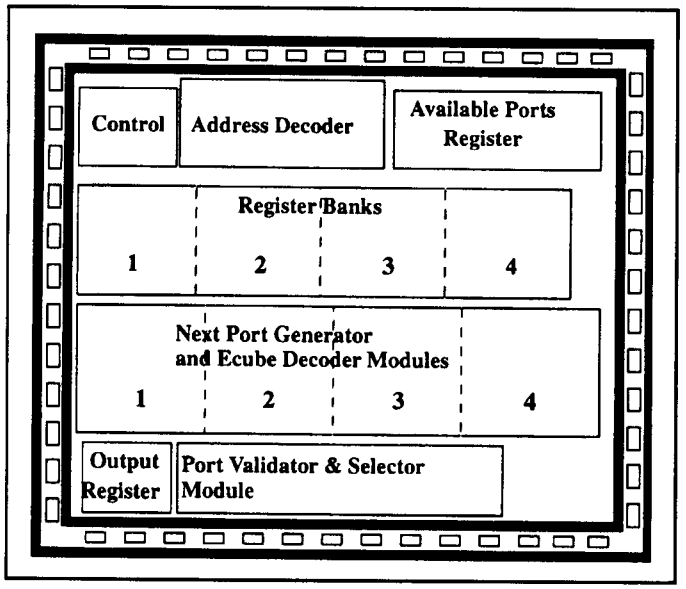

Figure 7: *The floor plan of the router depicted in Fig. 6.*

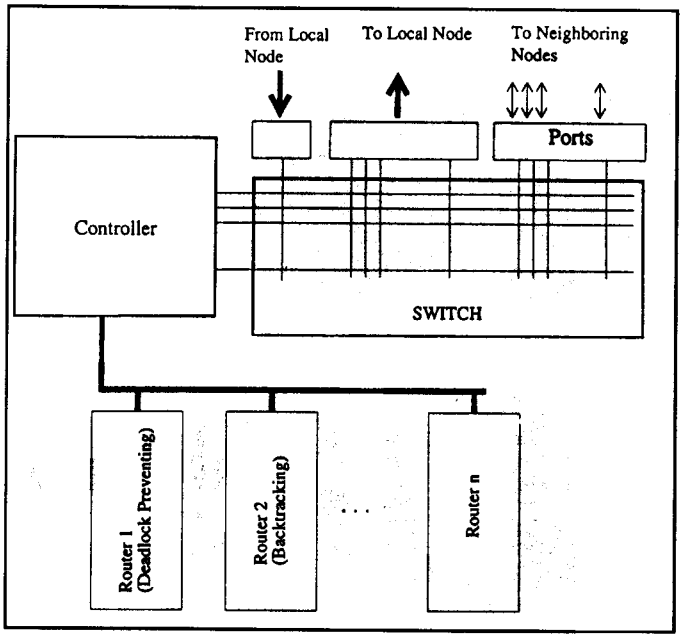

Figure 8: *The structure of the routing engine.*

Although these graphs are not the densest possible, they are attractive because of their simple routing. Since the node addresses are represented in a mixed radix as a sequence of n-digits, each one of these digits is processed independently and in parallel with the remaining digits. Thus the hardware involved in the routing can be made fast (because of the parallelism) and simple (since each module need only handle arithmetic **mod** m_i, as compared to arithmetic **mod** $m_1m_2...m_r$, needed when all the address digits are necessary, as is the case with such networks as the chordal rings [19] or the cube connected cycles [20]).

We have established a deadlock-preventing routing strategy for a subclass of the hypercycles and presented a deadlock-preventing router, which has been fabricated. We are currently developing a programmable routing engine for hypercycles which will incorporate a variety of routing strategies and be configurable for a large class of hypercycle topologies.

Appendix

A. Proof of Theorem 4

Since $m = 4\rho$, one can number the nodes of this hypercycle as $\{0, 1, ..., \rho - 1, \rho, \rho + 1, ..., 2\rho - 1, 2\rho, 2\rho + 1, ..., 4\rho - 1\}$. Now partition these nodes into ρ groups of four nodes each, as follows: $g_a = \{k\rho + a; k = 0, 1, 2, 3\}$ $a = 0, 1, 2, ..., \rho - 1$. Observe that $\{k\rho + a + \rho\}$ **mod** $4\rho = \{(k + 1)\rho + a\}$ **mod** $4\rho \in g_a$. Thus, every node in each group g_a can be reached from any other node in the same group, with a path that consists entirely of nodes in g_a. Therefore, for routing purposes, $\mathcal{G}_m{}^{\rho}$ can be partioned in ρ groups, each of which is closed under the hypercycle routing, and each can be mapped onto $\mathcal{G}^{1}{}_{4}$ $(k\rho + a \leftrightarrow k)$, for which it has been proven (Theorem 1) that the odd/even routing is deadlock-preventing.

For routes which originate at a node in one of the groups and terminate at a node in another group, observe the following. Any arbitrary node in the graph belongs to a specific group, g_a, or is directly accessible from a node in this group. Indeed, a node n can be written as $n = k\rho + x = (k\rho + x)$ **mod** $m = (k\rho + \alpha + (x - \alpha))$ **mod** m; $0 \le \alpha \le \rho - 1$. Since $0 \le x \le \rho - 1$, therefore $-(\rho - 1) \le x - \alpha \le \rho - 1$, and thus node n is directly accessible from node $k\rho + a \in g_a$. Also, since $m = 4\rho$, the diameter of $\mathcal{G}_m{}^{\rho}$ is 2. Thus, routes which connect nodes belonging to two different groups, g_a and $g_{a'}$, comprise either one link or two links: one of maximum length ρ (within the group containing the node of origin), followed by a link of length less than ρ. Thus the argumentation used in proving Theorem 2 applies as well. Q.E.D.

B. Proof of Theorem 5

Assume that there is a cycle present in the dependence graph. Name the sequence of partial paths which form the cycle as P_i; $i = 0, 1, 2, ..., k$, such that $P_i, L_i \neq \emptyset$. Since this sequence of partial paths forms a cycle, therefore $\forall l\, \exists l_i \in L_i$, such that $l_i \in P_{(i+1) \bmod k}$. Observe now that because of the generalized e-cube routing, a link of a higher dimension cannot be allocated unless all the lower-dimensioned links required have been allocated. Thus $l_i < l_{(i+1) \bmod k}$; $i = 0, 1, ..., k$. But this is a contradiction because of the transitivity and strict ordering of the relation $<$ .Q.E.D.

C. Proof of Theorem 6

In a similar manner to that employed in Theorem 5, assume that there is a cycle in the corresponding dependence graph. Then, there will be a sequence of partial paths, P_i, $i = 0, 1, 2, ..., k$, such that P_i, $L_i \neq \emptyset$ and $\forall l\, \exists l_i \in L_i$, such that $l_i \in P_{(i+1) \bmod k}$.

Observe now that because of the generalized e-cube routing, one cannot allocate a link to a partial path unless all the required links of a lower or equal dimension have been allocated. Thus, $l_i \le l_{(i+1) \bmod k}$; $i = 0, 1, ..., k$. This implies that all the requested links must lie at the same dimension. Thus, one can form a cycle in the dependence graph, consisting of portions of the partial paths relevant to this dimension.

But this is not possible, since we assumed that each of the component graphs in the product graph possesses a deadlock-preventing routing. Q.E.D.

Acknowledgements

This work has been supported by 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 Centres of Excellence Program, and by the Canadian Microelectronics Corporation.

References

[1] J.C. Peterson, J.O. Tuazon, D. Lieberman and M. Pniel, "The MARK III hypercube-ensemble concurrent computer," in *Proc. 1985 Int. Conf. on Parallel Processing*, 1985, pp. 71-73.

[2] C.L. Seitz, "The cosmic cube," CACM, vol. 28, no. 1, Jan. 1985, pp. 22-33.

[3] R.D. Rasmussen, G.S. Bolotin, N.J. Dimopoulos, B.F. Lewis and R.M. Manning, "Advanced general purpose multicomputer for space applications," in *Proc. 1987 Int. Conf. on Parallel Processing*, 1987, pp. 54-57.

[4] R.D. Rasmussen, N.J. Dimopoulos, G.S. Bolotin, B.F. Lewis and R.M. Manning, "MAX: Advanced general purpose real-time multicomputer for space applications," in *Proc. IEEE Real Time Syst. Symp.*, San Jose, Calif., 1987, pp. 70-78.

[5] *iPSC User's Guide*, Intel Corp., Portland, Ore., No. 17455-3, 1985.

[6] W.J. Dally, J.A. Stuart Fiske, J.S. Keen, R.A. Lethin, M.D. Noakes, P.R. Nuth, R.E. Davison and G. Fyler, "The message-driven processor: A multicomputer processing node with efficient mechanisms," *IEEE Micro*, vol. 12, no. 2, April 1992, pp. 23-40.

[7] N.J. Dimopoulos, D. Radvan and K.F. Li, "Performance evaluation of the backtrack to the origin and retry routing for hypercycle based interconnection networks," *Proc. Tenth Int. Conf. on Distributed Syst.*, Paris, 1990, pp. 278-284.

[8] G. Sabidussi, "Graph multiplication," *Math Z.*, vol. 72, 1960, pp. 446-457.

[9] W.J. Dally, "Performance analysis of k-ary n-cube interconnection networks," *IEEE Trans. Comput.*, vol. C-39, no. 6, June 1990, pp. 775-784.

[10] N.J. Dimopoulos, S. Radhakrishnan and D. Radvan, "Routing and processor allocation on a hypercycle-based multiprocessor," *Proc. 1991 Int. Conf. on Supercomputing*, Cologne, Germany, June 1991, pp. 105-114.

[11] W.J. Dally and C.L. Seitz, "Deadlock-free message routing in multiprocessor inter connection networks," *IEEE Trans. Comput.*, vol. C-36, no. 5, May 1987, pp. 547-553.

[12] D.H. Linder and J.C. Harden, "An adaptive and fault tolerant wormhole outing strategy for k-ary n-cubes," *IEEE Trans. Comput.*, vol. C-40, no. 1, Jan. 1991, pp. 2-12.

[13] E. Chow, H. Madan and J. Peterson, "A real-time adaptive message routing network for the hypercube computer," in *Proc. Real-Time Systems Symp.*, San Jose, Calif., 1987, pp. 88-96.

[14] D.P. Bertsekas and J.N. Tsitsiklis, *Parallel and Distributed Computation*, Englewood Cliffs, N.J.: Prentice Hall, 1989.

[15] L.G. Valiant, "General purpose parallel architectures," in *Handbook of Theoretical Computer Science*, J. van Leeuwen, Amsterdam: Elsevier Science Publishers B.V., 1990, pp. 945-971.

[16] L.N. Bhuyan and D.P. Agrawal, "Design and performance of generalized interconnection networks," *IEEE Trans. Comput.*, vol. C-32, no. 12, Dec. 1983, pp. 1081-1090.

[17] N.J. Dimopoulos and R. Sivakumar, "Deadlock preventing routing in hypercycles," in *Proc. Int. Workshop on Principles of Parallel Proc. (OPOPAC)*, I. Lavallé and Y. Paker, Eds., Paris: Hermes, 1993, pp. 47-61.

[18] D. Radvan, "Performance evaluation and router design for backtrack-to-the-origin-and-retry routing in hypercycle-based interconnection networks," M.A.Sc. thesis, Dept. of Electrical and Computer Engineering, University of Victoria, Victoria, B.C., 1990.

[19] M. Imase, T. Soneoka and K. Okada, "Connectivity of regular directed graphs with small diameters," *IEEE Trans. Comput.*, vol. C-34, no. 3, Mar. 1985, pp. 267-273.

[20] G.E. Carlsson, J.E. Cruthirds, H.B. Sexton and C.G. Wright, "Interconnection networks based on a generalization of cube-connected cycles," *IEEE Trans. Comput.*, vol. C-34, no. 8, Aug. 1985, pp. 769-772.

[21] R. Sivakumar, N.J. Dimopoulos, V. Dimakopoulos, M. Chowdhury and D. Radvan, "Implementation of the routing engine for hypercycle based interconnection networks," in *Proc. 1991 Can. Conf. on Very Large Scale Integration*, Kingston, Ont., 1991. pp. 6.4.1-6.4.7.