DEADLOCK-PREVENTING ROUTING IN HYPERCYCLES

N. J. Dimopoulos & R. Sivakumar
Dept of Electrical & Computer Engineering
University of Victoria
Victoria, B.C, Canada - V8W 3P6
Email: nikitas | rsiva@ece.uvic.ca

Abstract
Hypercycles are a class of multidimensional graphs which are products of circulant graphs obtained by allowing each dimension to incorporate more than two elements and a cyclic interconnection. Hypercycles, offer simple routing, incremental expansion, variable diameter, enhanced fault-tolerance, and the ability, given a fixed degree, to choose among a number of alternative size graphs. These graphs can be used in the design of interconnection networks for distributed systems tailored specifically to the topology of a particular application. In this work, we present a deadlock-preventing routing strategy for a subset of hypercycles and a 1.2 μm CMOS VLSI Hypercycle router component which implements the deadlock preventing routing.

1. Introduction

Message passing concurrent computers such as the Caltech's Cosmic Cube [1], MAX [2,3] and Intel's iPSC [4] are examples of hypercubes that consist of several processing nodes that interact via messages exchanged over communication channels linking these nodes into one functional entity. The binary *n*-cube or hypercube [5,6] that has been proven to be efficient with significant speedups in many real time applications such as computational aerodynamics, quantum physics and image processing. However, embedded real time applications[2,3] tend to exhibit varied structures that do not necessarily map optimally onto the hypercubes. Observe that the binary *n*-cube expands as a power of 2 and the number of communication links at each node is given by $\log_2 n$. This constitutes a significant increase in resource allocation especially in space, mass and power limited environments such as in spacecrafts.

1.1 Hypercycles

Recently, a new class of multidimensional graphs called *hypercycles* [7,8] have been introduced. These graphs are products of circulant graphs [9] and offer simple routing, variable connectivity, enhanced fault-tolerance and range in complexity from simple rings to fully connected graphs. They are generalizations of several popular interconnection networks such as rings, toruses, binary n-cubes, k-ary n-cubes[10] and generalized hypercubes[11].

The mixed radix representation [12] is used in defining a Hypercycle. An n- dimensional *Hypercycle*, is a regular undirected graph given by $\mathcal{G}^{\rho}_{m} = \{\mathcal{N}^{\rho}_{m}, \mathcal{E}^{\rho}_{m}\}$ where $\mathcal{N}^{\rho}_{m}$ is the set of nodes, $\mathcal{E}^{\rho}_{m}$ is the set of edges, $m = m_n, m_{n-1}, \cdots, m_1$ a mixed radix, $\rho = \rho_n, \rho_{n-1}, \cdots, \rho_1$; $1 \le \rho_i \le \lfloor m_i/2 \rfloor$, the connectivity vector, and $\mathcal{N}^{\rho}_{m} = \{0,1,2,\ldots,M-1\}$. Given $\alpha, \beta \in \mathcal{N}^{\rho}_{m}$ then $(\alpha, \beta) \in \mathcal{E}^{\rho}_{m}$ 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 $\alpha_i = \beta_i; i \ne j$. Some examples of Hypercycles are shown in Fig. 1. Many properties and algorithms used for example in routing and processor allocation can be extended to the entire class of hypercycles making it possible to choose a topology that best suits the system requirements of a specific class of applications.

2. Deadlock-Free routing

Deadlocks occur 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. Thus routing algorithms should be designed such that deadlocks must be prevented, avoided or detected and broken. A number of different routing policies have been introduced for a variety of interconnection networks. Thus in hypercubes e-cube routing [13] prevents deadlocks by ordering the resources (i.e. channels or virtual channels) comprising a path thus guaranteeing that no circular dependencies exist for any paths formed. Similar approaches have been devised for toruses and k-ary n-cubes [13-15] where virtual channels are introduced in order to break any circular dependencies. A different approach of deadlock avoidance has been introduced with the Hyperswitch [5] and the various backtracking strategies for hypercycles [13]. There, deadlocks are avoided by forcing a partially formed 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. These routing schemes do not make use of virtual channels and thus they are well suited for circuit switching environments.

2.1 Deadlock-Prevention in Hypercycles

Deadlocks may occur easily in cases where the segments that form the paths are chosen at random. Certain routing algorithms like e-cube 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. It has been proven that the e-cube routing [13,14] prevents deadlocks in the case of the hypercube. We now introduce a deadlock-preventing routing strategy called the *prioritized dimension routing* for a certain class of Hypercycles of diameter 1 and 2 and prove that it indeed is deadlock-preventing.

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 l_i the last node of the partial path P_i, and by L_i the set of all legal outgoing links from node l_i which can be used in order to complete the partial path to the destination D_i.

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

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 $l \in P_j$.

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 $l \in P_j; i = 1, 2, \ldots, \mu$.

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. $\forall l \in L \; \neg\exists j$ such that $l \in P$. 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. Consider the example of a 4-node hypercycle $\mathcal{G}_4^1$ as depicted in Fig. 1a. Node 2 can be reached from node 0 travelling either clockwise or counter clockwise. Similarly, node 3 can be reached from node 1 traversing the hypercycle in two directions. If the directionality is chosen at random, or identically for all nodes, deadlock can occur. If on the other hand, the directionality alternates, as we shall prove, deadlocks are prevented.

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

Definition 4. The ρ–quotient of a one dimensional hypercycle $\mathcal{G}_m^\rho$ is defined as $\left\lfloor \frac{q}{\rho} \right\rfloor$ where $0 \le q \le m-1$.

2.2 Odd-even preference routing

In one dimensional hypercycles $\mathcal{G}_m^{\rho}$ where a destination can be reached through two alternate routes i.e., $(\alpha-\beta) \bmod m = (\beta-\alpha) \bmod m = |\alpha, \beta|$ where α is the source and β is the destination, we define the odd/even preference routing so as in the greedy mode, routes originating at source nodes with an even $\rho-$ quotient proceed in the clockwise (counterclockwise) direction, while routes originating at nodes with an odd $\rho-$quotient, proceed in the opposite direction. If α^* denotes the next in termediate node, then

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

As an illustration, we present the dependence graph shown in Fig. 2 for the odd/even routing in $\mathcal{G}_4^1$. 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 it can be verified, this dependence graph is devoid of loops and thus the odd/even routing in $\mathcal{G}_4^1$ is deadlock free. If arbitrary routing was 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. It can be shown easily that the odd/even preference routing is deadlock preventing in $\mathcal{G}_4^1$ and the detailed proof is given in [16].

Theorem 1. One-dimensional hypercycles $\mathcal{G}_m^{\rho}$ of diameter two and $\lfloor m/2 \rfloor < 2\rho$ have greedy routing [16] is deadlock preventing.

Proof Since $\lfloor m/2 \rfloor < 2\rho$ it means that any two-link path from a source to a destination, consists of a maximum 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 $\forall l_j \in L_j \neg\exists\sigma$ such that $l_j \in P_{\sigma}$. Therefore, a cycle cannot exist in such a dependency graph, and the routing is deadlock preventing.

Q.E.D.

Theorem 2. One dimensional Hypercycles $\mathcal{G}_m^{\rho}$ with $m = 4\rho$ have a deadlock preventing odd/even preference routing.

Proof Since $m = 4\rho$, one can number the nodes of this hypercycle as $\{0, 1, \ldots, \rho-1, \rho, \rho+1, \ldots, 2\rho-1, 2\rho, 2\rho+1, \ldots, 4\rho-1\}$. Partition now these nodes into ρ groups of four nodes each as follows.

$$g_a = \{k\rho + a; k = 0, 1, 2, 3\} \; a = 0, 1, 2, \ldots, \rho-1$$

Observe

$$\{k\rho + a + \rho\} \bmod 4\rho = \{(k+1)\rho + a\} \bmod 4\rho \in g_a$$

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 partitioned in ρ groups, each of which is closed under the hypercycle routing, and each can be mapped onto $\mathcal{G}_4^1$ ($k\rho + a \leftrightarrow k$), for which, it has been proven in Theorem.1 that the odd/even routing is deadlock preventing.

Q.E.D.

Theorem 3. Fully connected graphs are deadlock free.

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.

Theorem 4. One dimensional Hypercycles $\mathcal{G}_m^{\rho}$ with $m = 4\rho - k$ where $k = 1, 2, \ldots, 2\rho-2$ are deadlock-free under greedy routing.

Proof. Since $\lfloor m/2 \rfloor = \lfloor (4\rho - k)/2 \rfloor < 2\rho$ for $k = 1, 2, \ldots, 2\rho-2$, the diameter of $\mathcal{G}_m^{\rho}$ is 2 and Theorem 1 applies.

Theorem 5. One dimensional Hypercycles $\mathcal{G}_m^{\rho}$ with $m = 4\rho - k$ where $k = 2\rho - 1$ or 2ρ are deadlock-free under greedy routing.

Proof. When $k = 2\rho - 1$ or 2ρ, the diameter is 1 and Theorem 3 applies.

Q.E.D.

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.

Proof. 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

$\forall i \;\; \exists l_i \in L_i$ 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 at a lower or equal dimension have been allocated. Thus, $l_i \prec l_{(i+1) \bmod k}; i = 0, 1, \ldots, k$ 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

2.3 Algorithm

The generalized ecube prioritized dimension routing proceeds as follows: Routing from a source to destination node is accomplished in order of decreasing dimension. Given an n-dimensional hypercycle, the following priority levels are assigned by definition. Dimension n has the highest priority while dimension 1 has the least priority. When a destination address is presented, the algorithm checks whether the destination has been reached in dimension i, $(i=n \ldots 1)$ by comparing the source(current) and destination address bits. If it has not, then the next shortest path address in dimension i based on the greedy strategy is generated based on the odd-even preference scheme using the greedy strategy. The computed port address is then validated based on the available ports and this constitutes the next port address to forward the message; otherwise a break signal is generated. On the other hand, if the destination has indeed been reached in the i^{th} dimension, routing proceeds in a similar fashion in the remaining j dimensions (of lower priority) for $j = i-1 \ldots 1$ till the source and destination address bits are equalized.

3. Architecture

Fig. 3 gives a block diagram of an n-dimensional Hypercycle router. The major functional blocks of the router comprises of the following:

1. Four modules of Ecube decoder, one for each of the 4 dimensions.
2. Four modules of Next Port Generators (NPG).
3. Port Selector and Validator.

The Decoders and Next Port Generators establish whether the deadlock preventing routing can be used on the Hypercycle configured. The next port generator implements the greedy and odd-even preference routing. The Port Selector and Validator selects the highest dimension if it is free, otherwise a *No_Ports_Available* is generated which will cause the routing to stop and wait until the required port is freed.

The router is programmable in that the Hypercycle Network on which the routing is performed is described through its mixed radix m and connectivity ρ vectors. These vectors together with the node address, available ports and destination address are stored in registers.

4. Implementation

The proposed deadlock-free router chip has been designed and fabricated in 1.2 µm NTE CMOS technology. The micrograph of the chip is illustrated in Fig. 4. The designed router can be configured for a 16 port, 15 node per dimension, 4-dimensional Hypercycle network with a maximum of 50,625 nodes. The chip has 51 pins housed in a 68-pin PGA.Each decision cycle of the Hypercycle router is composed of two phases, namely

1. *Load phase*
2. *Execute phase*

In the *Load phase*, the router is initialized with data needed for its operation during system configuration. In this phase, the controller loads the router with information such as the connectivity, current address, destination address, population, offset and available ports. Note that the above data is needed only once during configuration.

In the *Execute phase*, the data loaded into the router is processed and the resulting next port address (if an available one is found) together with the destination reached or wait signal is generated. The router operates on a synchronous mode with the clock, providing a result every 4 clock cycles after the data has been loaded into the memory bank. It takes one clock cycle to load each register. The chip is designed to operate at a maximum frequency of 125 Mhz with 6 clock cycles per routing decision. It incorporates approximately 20000 transistors and the measured throughput using the IMS XL-60 test bed is 16 million routing decisions a second. The router is fully testable with a fault coverage of 97 % for single stuck-at faults. The CAD tools used in the design process comprise of Cadence, SILOS, VERILOG and Logic-III (UVIC). This router is part of a programmable routing engine for Hypercycles which will incorporate in addition to deadlock preventing routing deadlock avoiding routing and be capable of adopting the most suitable routing.

5. Conclusions and Discussions

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.

Although these graphs are not the densest possible, they are attractive, because of their variable connectivity, simple routing, incremental expandability and ease of implementation makes them attractive for designing interconnection networks for concurrent computers. 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 modm_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 [17], or the cube connected cycles [14].

We have established a deadlock preventing routing strategy for a subclass of the hypercycles and demonstrated a practical and viable silicon implementation of a router that prevents deadlocks. We are currently developing a programmable routing engine for hypercycles which will incorporate a variety of routing strategies and be configured for a large class of hypercycle topologies.

Acknowledgement

This work has been supported by the Natural Sciences and Engineering Research Council Canada, under grants #OGP0041188 and #STR013422, by the Institute for Robotics and Intelligent Systems under the National Networks of Centers of Excellence Program, and by the Canadian Microelectronics Corporation

6. Bibliography

1. C. L. Seitz, "The cosmic cube", CACM, vol. 28, pp. 22 - 33, Jan 1989
2. 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" *Proceedings of the IEEE Real Time Systems Symposium* pp. 70-78, San Jose, CA., Dec. 1987.
3. R. D. Rasmussen, 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, 1987.
4. iPSC User's Guide, No. 17455-3, Intel Corp., Portland, Ore., 1985.
5. 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, 1985.
6. E. Chow, H. Madan, J. Peterson "A Real-Time Adaptive Message Routing Network for the Hypercube Computer" *Proceedings of the Real-Time Systems Symposium*, pp. 88-96, San Jose CA., 1987.
7. N. J. Dimopoulos, 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 Systems*, Paris, pp. 278-284, 1990.
8. R. Sivakumar, N. J. Dimopoulos, V. Dimakopoulos, M. Chowdhury, D. Radvan "Implementation of the Routing Engine for Hypercycle Based Interconnection Networks" *Proceedings of the 1991 Canadian Conference on Very Large Scale Integration* pp. 6.4.1-6.4.7, Kingston, 1991.
9. G. Sabidussi "Graph Multiplication" *Math. Zeitschr.* Vol. 72, pp. 446-457, 1960.
10. W. J. Dally "Performance Analysis of k-ary n-cube Interconnection Networks" *IEEE Trans. Comput.*, Vol. 39, no. 6, pp. 775-784, June 1990.
11. L. N. Bhuyan and D. P. Agrawal, " Generalized Hypercubes and Hyperbus Structures for a Computer Network" *IEEE Trans. Comput.* Vol. C-33, No. 4, pp.323-333, April 1984.
12. L. N. Bhuyan and D. P. Agrawal, "Design and Performance of Generalized Interconnection Networks" *IEEE Trans. Comput.* Vol. C-32, pp. 1081-1090, Dec. 1983.
13. W. J. Dally, J. A. Stuart Fiske, J. S. Keen, R. A. Lethin, M. D. Noakes, P. R. Nuth, R. E. Davison, G. Fyler "The Message-Driven Processor: A Multicomputer Processing Node with Efficient Mechanisms" *IEEE Micro* pp. 23-40, April 1992.
14. W. J. Dally and C. L. Seitz "Deadlock-Free Message Routing in Multiprocessor Interconnection Networks" *IEEE Trans. Comput.* Vol. C-36, pp. 547-553, May 1987.
15. D. H. Linder and J. C. Harden "An Adaptive and Fault Tolerant Wormhole routing Strategy for k-ary n-cubes" *IEEE trans. Comput.* vol. 40, No. 1, pp. 2-12, Jan. 1991.
16. N. J. Dimopoulos, M. Chowdhury, V. Dimakopoulos, and R. Sivakumar, "Routing and Broadcasting in Hypercycles, Deadlock Free and Backtracking Strategies", *Proceedings of PARLE 92*, Paris, July 1992
17. M. Imase, T. Soneoka, and K. Okada, "Connectivity of Regular Directed Graphs with Small Diameters" *IEEE Trans. Comput.*, Vol. C-34, pp. 267-273, Mar. 1985.

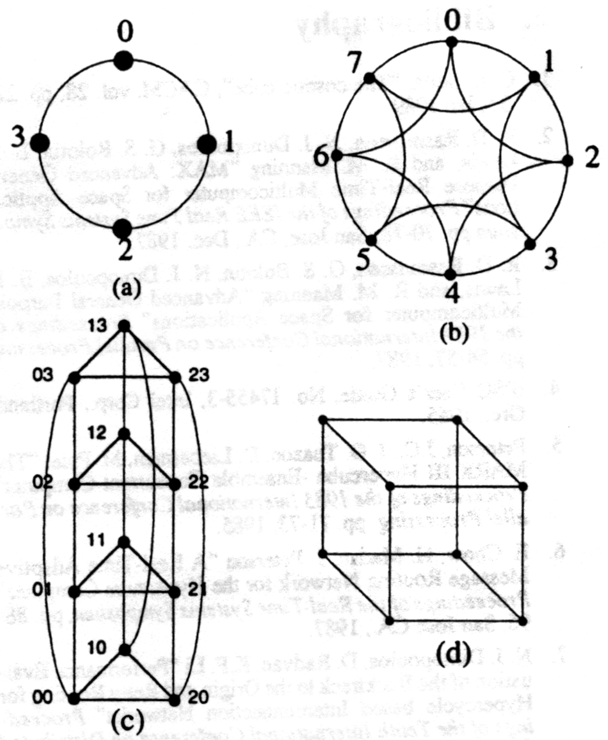

Fig. 1 Examples of Hypercycle Graphs

a. Ring G_4^1 *b. Circulant* G_8^2 *c. Hypercycle* G_{43}^{11}

d. Hypercube G_{222}^{111}

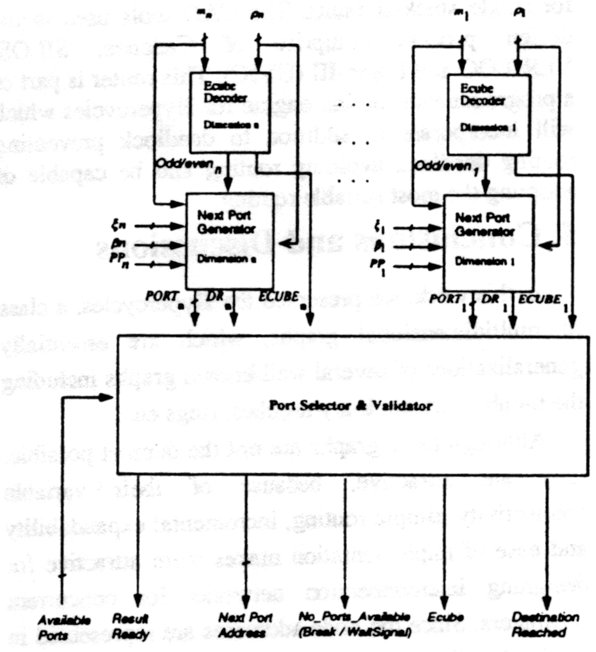

Fig. 3. Block diagram of n-dimensional ecube prioritized dimension router

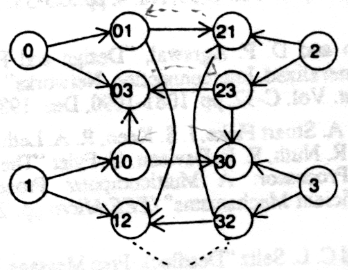

Fig. 2. Dependency Graph for Hypercycle G_4^1

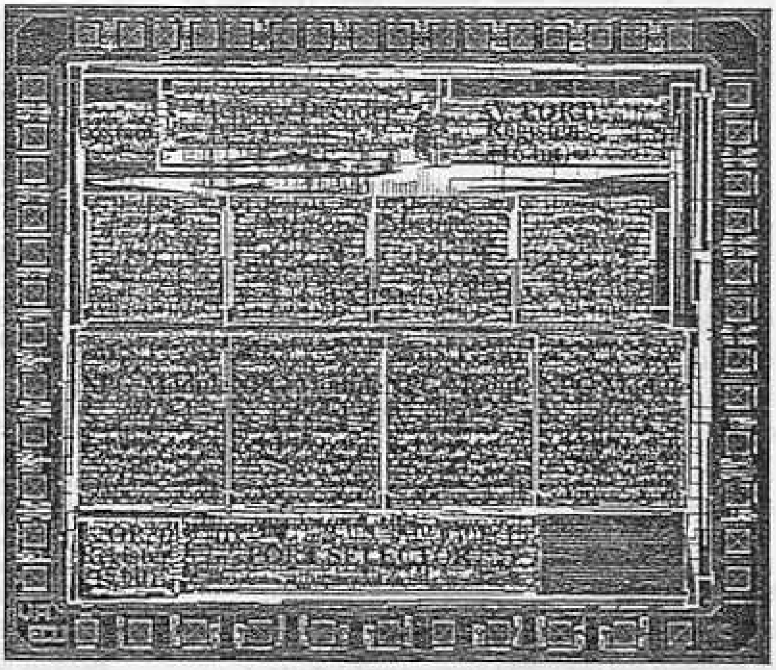

Fig. 4. Layout of the 4-D deadlock-free router in 1.2 μm NTE CMOS technology