IMPLEMENTATION OF THE ROUTING ENGINE FOR HYPERCYCLE BASED INTERCONNECTION NETWORKS

R. Sivakumar, N. J. Dimopoulos, V. Dimakopoulos,
M. Chowdhury and Don Radvan
Electrical and Computer Engineering Department
University of Victoria
Victoria, B.C. Canada

ABSTRACT†

In this work, we present the design and implementation of a Routing Engine for Hypercycle-based interconnection Networks. Hypercycles, is a class of multidimensional graphs, which are generalizations of the n-cube. These graphs are obtained by allowing each dimension to incorporate more than two elements and a cyclic interconnection strategy. Hypercycles, offer simple routing, and the ability, given a fixed degree, to chose 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. For routing, we have adopted a circuit switching back-track-to-the-origin-and-retry strategy, whereupon paths that block at intermediate nodes are abandoned, and a new attempt is made. Intermediate nodes are chosen at random at each point from among the ones that form the shortest paths from a source to a destination. Simulation results have established the performance for a variety of configurations. Hypercycles, because of the richness of their connectivity, exhibit superior performance as compared to Hypercubes both in terms of throughput and in terms of matching the size/topology requirements of an application.

INTRODUCTION

Message passing concurrent computers such as the Hypercube [5, 11], Cosmic Cube [8], MAX[6, 7], 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, the Connection Machine [12] etc. having adopted a regular interconnection pattern corresponding to a binary n-dimensional cube, while MAX adopts a less structured, yet unspecified topology.

† This work has been supported by the Natural Sciences and Engineering Research Council Canada, under grant #OGP0001337 and by the Institute for Robotics and Intelligent Systems (IRIS)

The advantages of having a regularly structured interconnection are many-fold, and they have been proven time and again in their being incorporated in many recent designs [2,5,6,7,8,11,12]. In these structures, easy deadlock-free routing [3] can be accomplished by locally computing each successive intermediate node -for a path that originates at a source node and terminates at a destination node- as a function of the current position and the desired destination. Many regular problems (such as the ones found in image processing, physics etc.) have been mapped on such regular structures, and run on the corresponding machines exhibiting significant speedups.

In contrast, embedded real-time applications [6, 7], tend to exhibit variable structures that do not necessarily map optimally to an n-cube. In addition, since the size of a binary n-cube is given as 2^n, it means that a particular configuration cannot be expanded but in predefined quantum steps. This constitutes a significant increase in resource allocation, especially in power-mass limited environments.

Hypercycles [7,4] can be considered as products of "basic" graphs that allow a rich set of component "basic" graphs ranging in complexity from simple rings to the fully connected ones.

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.

MIXED RADIX NUMBER SYSTEM

The mixed radix representation [1], is a positional number representation, and it is a generalization of the the standard b-base 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 $(X)_{m_1 m_2 \ldots m_r} = x_1 x_2 \ldots x_r |_{m_1 m_2 \ldots m_r}$ 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

$$w_i = \frac{M}{m_1 m_2 \cdots m_i}$$

HYPERCYCLES.

An r- dimensional Hypercycle, is the following regular undirected graph: $G_m^{\rho} = \left\{ N_m^{\rho}, E_m^{\rho} \right\}$ where N_m^{ρ} ($N_m^{\rho} = \{0,1,2,\ldots,M-1\}$) is the set of nodes, E_m^{ρ} the set of edges, $m = m_1, m_2, m_3, \ldots, m_r$ 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 ring ($\rho_i = 1$) to fully connected ($\rho_i = \lfloor m_i / 2 \rfloor$).

Given $\alpha, \beta \in N_m^{\rho}$ then $(\alpha, \beta) \in E_m^{\rho}$ 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 $\alpha_i = \beta_i$; $i \ne j$

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

ROUTING

Hypercycles, have routing properties that are 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 walk, from node α to node α^*, can be constructed as 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[§]

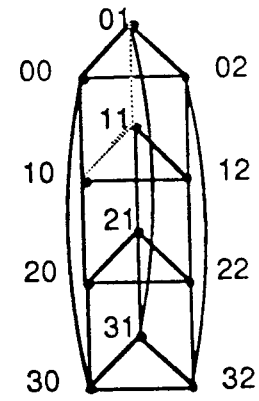
$$\xi_{j_i+1} = \begin{cases} \left(\xi_{j_i} + \rho_i\right) \bmod m_i & \text{if } \left[\left(\xi - \xi_{j_i}\right) \bmod m_i = \left|\xi_{j_i}, \xi\right|\right] > \rho_i & \text{(a)} \\ \left(\xi_{j_i} + \left|\xi_{j_i}, \xi\right| \bmod \rho_i\right) \bmod m_i & \text{if } \left[\left(\xi - \xi_{j_i}\right) \bmod m_i = \left|\xi_{j_i}, \xi\right|\right] > \rho_i \text{ and } \left|\xi_{j_i}, \xi\right| \bmod \rho_i \ne 0 & \text{(b)} \\ \left(\xi_{j_i} - \rho_i\right) \bmod m_i & \text{if } \left[\left(\xi_{j_i} - \xi\right) \bmod m_i = \left|\xi_{j_i}, \xi\right|\right] > \rho_i & \text{(c)} \\ \left(\xi_{j_i} - \left|\xi_{j_i}, \xi\right| \bmod \rho_i\right) \bmod m_i & \text{if } \left[\left(\xi_{j_i} - \xi\right) \bmod m_i = \left|\xi_{j_i}, \xi\right|\right] > \rho_i \text{ and } \left|\xi_{j_i}, \xi\right| \bmod \rho_i \ne 0 & \text{(d)} \\ \xi & \text{if } \left|\xi_{j_i}, \xi\right| \le \rho_i & \text{(e)} \end{cases}$$

$$\xi_0 = \alpha_i \qquad \xi_{max} = \xi$$

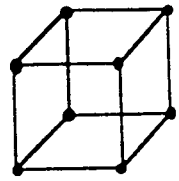
Eqn. 1.

Equation 1. defines 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

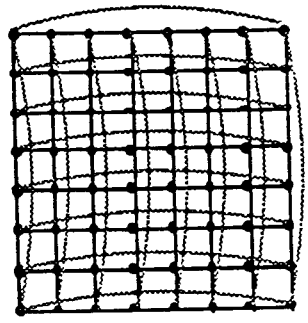
§ We define $|a, b| = \min\{(a-b) \bmod m_i, (b-a) \bmod m_i\}$



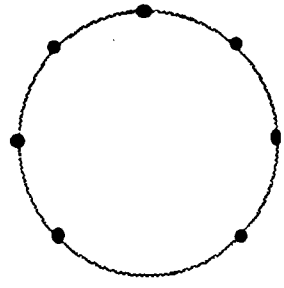
a. Hypercycle $G_{4\,3}^{1\,1}$



b. Binary 3-cube G_{222}^{111}



c. Illiac IV G_{88}^{11}



d Ring G_7^1

Figure 1. Examples of Hypercycles

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. 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 $(\alpha)_{m_1 m_2 \ldots m_r} = \alpha_1 \alpha_2 \ldots \alpha_r$ and a destination $(\beta)_{m_1 m_2 \ldots m_r} = \beta_1 \beta_2 \ldots \beta_r$ then 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 1, until the destination is formed. The following walk connects **source** to **destination.**

source $= \alpha_1 \alpha_2 \alpha_3 \ldots \alpha_r;\ \alpha_1 \xi_1 \alpha_3 \ldots \alpha_r;\ \alpha_1 \xi_1 \psi_1 \ldots \alpha_r;$

$\alpha_1 \xi_2 \psi_1 \ldots \alpha_r;\ \alpha_1 \xi_2 \psi_2 \ldots \alpha_r;\ \ldots;\ \alpha_1 \xi_2 \beta_3 \ldots \alpha_r;\ldots;$

$\beta_1 \beta_2 \beta_3 \ldots \beta_r$ **= destination**

If only the greedy strategy is followed, the so formed paths are called *greedy paths.*

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

DEADLOCK AVOIDANCE IN ROUTING.

Deadlocks may occur easily in cases where the segments that form the circuit connecting a source to a destination are chosen at random. Certain routing algorithms (e.g. virtual channels, e-cube routing [3]) 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. For Hypercycles, we have adopted a routing strategy where deadlocks are avoided by requiring a blocked partial path to backtrack to its origin and retry.

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 impossible. We have adopted 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 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. 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. Simulation results depicting the average delay in forming a circuit and transmitting a message for various hypercycles ,are depicted in fig. 2.

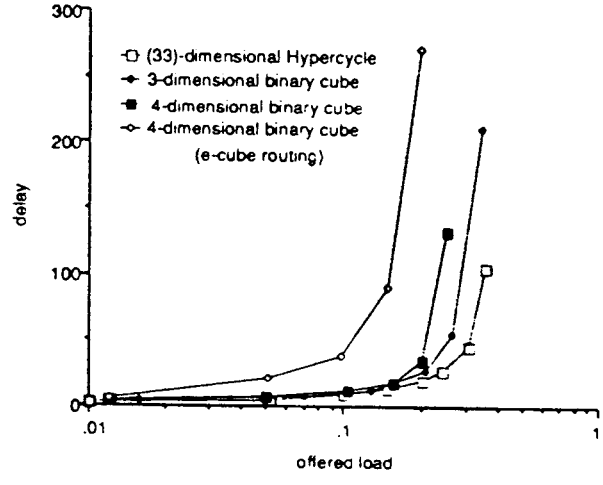


Figure 2. Delay vs. offered load for the 4-cube using backtrack-to-the-origin and e-cube routing. The offered load is normalized to the capacity of the network. The delay is normalized to the message transmission time.

The performance of the backtrack-to-the-origin-and-retry for both binary cubes and hypercycles of similar sizes, is clearly superior to that of the e-cube. 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. Generally, system throughput and delay are functions of both average distance and the average number of alternate paths between any two nodes.

HYPERCYCLE ROUTING ENGINE

The functionality of the backtrack-to-the-origin-and-retry as discussed above, includes routing in the

† Extend is a trademark of Imagine That inc.

forward direction as well as backtracking, and message request management. These functions can be partitioned into two components. The first component (the router) implements the forward routing complete with the random selection of one of the available channels and the detection of a blocked route. The second component (the controller) implements the remaining functionality which includes backoff policies, and message management.

A general structure of the envisioned system is given in Fig. 3.

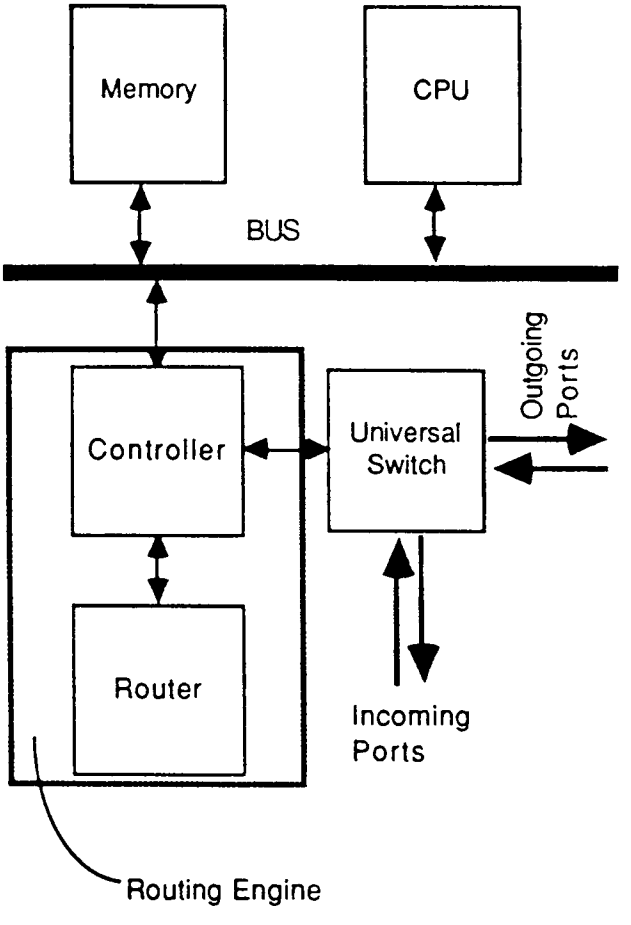


Figure 3. The structure of a hypercycle system's node.

As it can be seen, messages arrive at the node through the incoming ports or are generated by the node itself. During routing, only the header of a message participates in the establishment of the route, while the body of the message is queued at the originating node and awaits the completion of a route to its destination before it is transmitted. In the ensuing discussion, we are using the term message to mean its header for the route establishment and the complete message during transmission. Messages are intercepted by the controller, which queues them. The controller passes the destination address found in the message to the router, which finds a valid port for the continuation of the message's route. If the router cannot find an available port, the message is considered blocked, the controller decides on a backoff according to the established policy and proceeds with the next message. If a valid port is obtained, then the controller configures the switch so that the port through which the message arrived is connected to the selected outgoing port, and thus the message is forwarded to the subsequent node of its route.

In this work, we are presenting our implementation of the router component.

Fig. 4 gives a block diagram of an r-dimensional Hypercycle router. As it can be seen, we are implementing our router as a system having four modules, namely the Destination Address Register, the Next Port Generators, the Port Validator and the Port Selector comprising a Valid Port Counter, the Random Number Generator and an r out of k Selector...

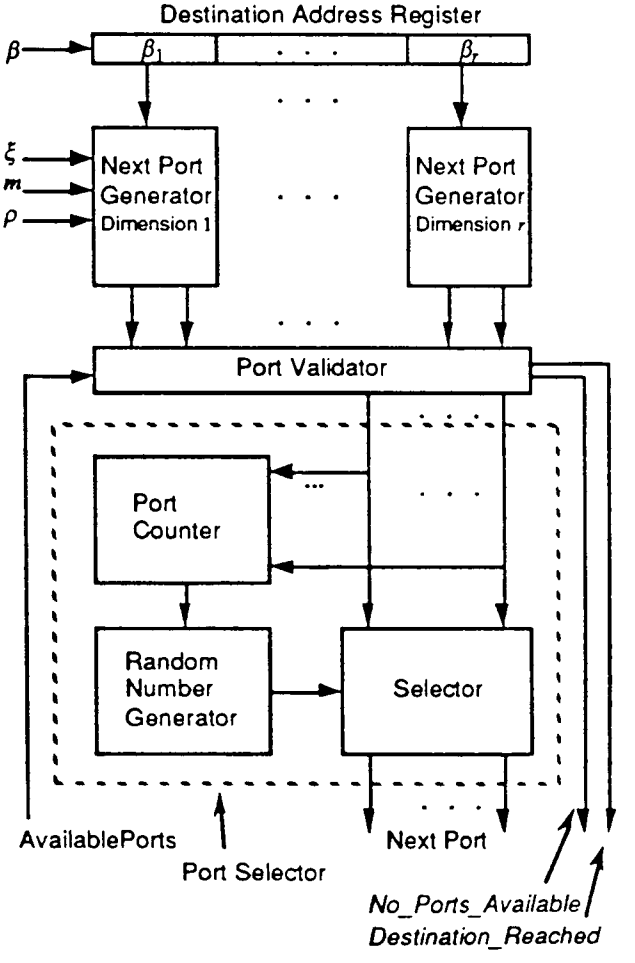


Figure 4. Block diagram of the Hypercycle routing engine. Routing in each dimension is done in parallel.

The destination address is used by the Next-Port Generator to generate all possible ports that could be

used in continuing the path to the required destination. 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.

For the random number generator, we use a mixed congruential random number generator of the form $r_{n+1} = (r_n + c) \bmod 2^{16} - 1$, with c being an additive parameter, to generate 16-bit random numbers. For any hypercycle, the number of the ports available for the continuation of any given route varies depending on the traffic and the route itself. Therefore, in order to select at random an appropriate port, we first obtain the number k of valid ports which can be used for the extension of the route, and then the order of the port to be selected is determined as $r_n \bmod k + 1$ with r_n being a 16-bit random number generated by the mixed congruential random number generator as discussed above. We have implemented a **mod**k extractor for 16-bit numbers (n = 16) and k = 2 through 10 The area-time complexity of of the extractor is calculated to be $O(\rho_0 \log \rho_0)$ with $\rho_0 = \lceil n/k \rceil$. The selector implements a binary search method with an area complexity of $O(n (\log n)^2)$ and time complexity of $O((\log n)^2)$ [9].

IMPLEMENTATION

We have completed the implementation [9] of a 16 port 4-dimensional router in NTE's 1.2μ CMOS4S. Our design has been fabricated by CMC. Statistics of the fabricated component are given in TABLE I.

The structure of the component including the internal registers is presented in fig. 5. while its pin configuration is presented in fig. 6. The mask of the fabricated component is presented in fig. 7.

The router includes eight 16-bit registers used to hold the configuration parameters as well as the destination address and the currently available ports. The configuration parameters include the address of the node, the mixed radix vector $m = m_1, m_2, m_3, \ldots, m_r$, the connectivity vector $\rho = \rho_1, \rho_2, \ldots, \rho_r$, the seed for the random number generator, the additive term for the random number generator, the port population per dimension, and the port offsets per dimension. A 16-bit bidirectional bus (I/O<0-15>) is used in order to load or read the registers. The Valid Port's address is to be found on the port bus (PORT<0-5>).

The component operates either in a load or execute mode. In the load mode, the registers can be read or written, while in the execute mode, a valid port is found and is reported on the port bus.. The configuration (CONFIG) mode (MODE) and select (A B C) signals are used to control the operation of the component.

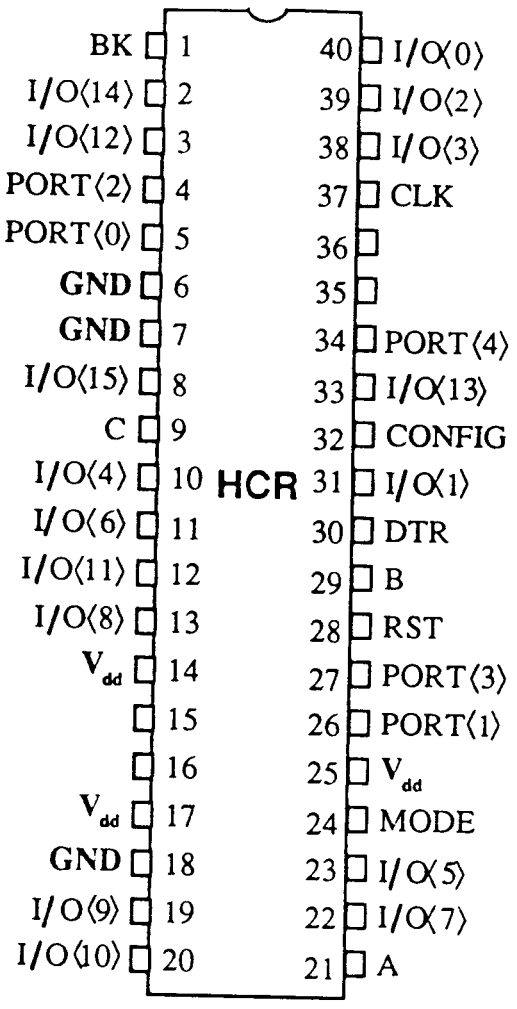


Figure 6. The pin arrangement for the Hypercycle Router component.

It is worth noting that the router has been designed to be programmable so that it can be used to perform routing for a variety of hypercycle configurations.

TABLE I

Technology	1.2 μ NTE CMOS4S
Area of the chip	5263.6 × 5263.6 μ^2
Area of the core	4581.9 × 4581.9 μ^2
Number of Pins	36
Number of Standard Cells	5951
Number of Transistors	25842
Propagation Delay	50 ns
Area × Time Complexity	1.049e09 ns·μ^2

TESTING

We have partially tested the fabricated components. Typical AC characteristics obtained through these tests,

are presented in TABLE II. These agree with our simulations which predicted maximum propagation delays of the order of 50ns.

CONCLUSIONS

In this work, we presented our implementation of a router to be used in the routing engines of Hypercycle-based concurrent systems. Such systems are ideal as embedded systems because of their ability to be configured to match the requirements of the application.

Our design was fabricated in the 1.2 μ CMOS4S process, and exhibits excellent performance, achieving in excess of 20 million routing decisions per second.

We are currently in the process of designing the accompanying controller, and we are proceeding with further studies of the performance of hypercycles with respect to alternative routing policies.

TABLE II.

Chip No.	Set-Up Time	Hold Time	Load Cycle	Execute Cycle
1	3.9 ns	52.0 ns	5.9 ns	38 ns
4	4.1 ns	2.0 ns	6.1 ns	40 ns

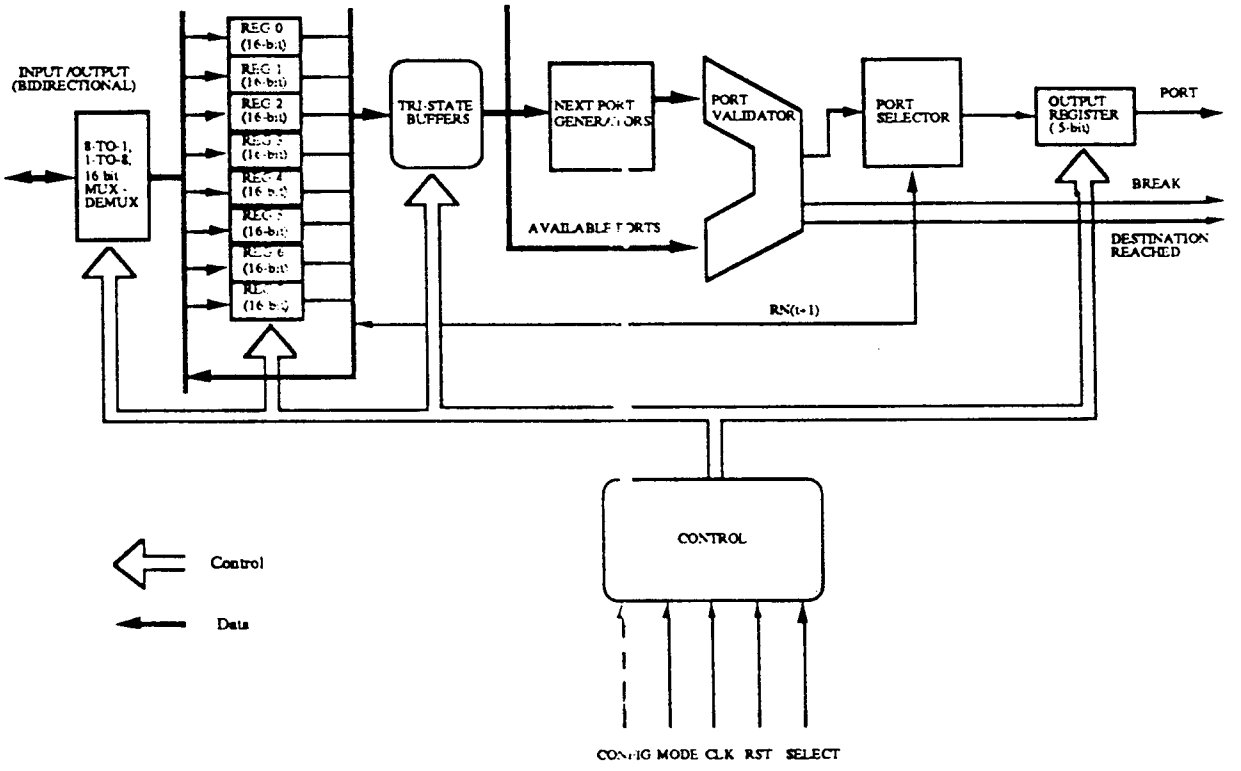


Figure 5. The structure of the Hypercycle Router.

REFERENCES

1. 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.
2. 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., (Dec. 1987)
3. Dally, W.J., and C. L. Seitz "Deadlock-Free Message Routing in Multiprocessor Interconnection Networks" *IEEE Trans. Comput.* Vol. C-36, No. 5, pp. 547-553, May 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 Systems*, Paris, France, pp. 278-284 (June 1990).
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, Aug. 20-23 1985.

6. 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)
7. 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).
8. Seitz, C. L. "The cosmic cube" *CACM* vol. 28, pp.22-33, Jan. 1985.
9. Sivakumar R. *VLSI Implementation of a Router for the Backtrack-to-the-Origin-and-Retry Routing Scheme of the Hypercycle Based Interconnection Networks* M.A.Sc. thesis, University of Victoria, 1991.
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* Jan. 1987, pp.85-97

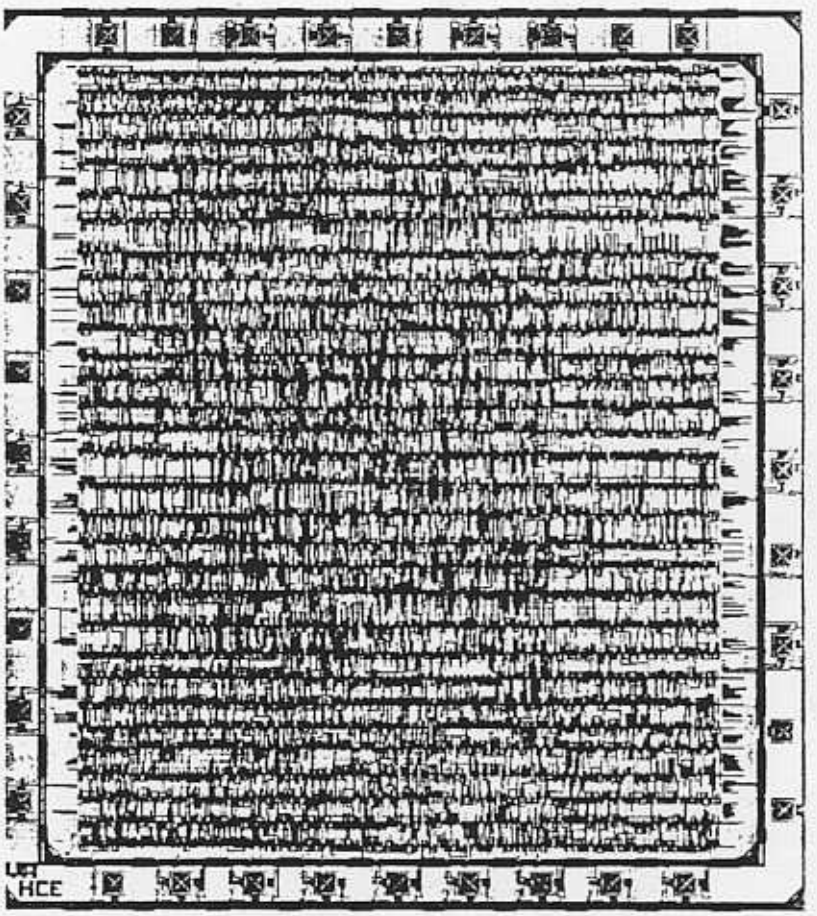


Figure 7. The implementation of a 16-port 4-dimensional Hypercycle Routing Engine in NTE's 1.2 μ CMOS4S process.