Canadian Conference on Electrical and Computer Engineering
Congrès canadien en génie électrique et informatique
Ottawa, Ontario, Canada, September 3 - 6 septembre 1990

ROUTER IMPLEMENTATION FOR HYPERCYCLE-BASED INTERCONNECTION NETWORKS[¶]

Don Radvan, Nikitas J. Dimopoulos , R. Sivakumar,
Electrical and Computer Engineering Department
University of Victoria
Victoria, B.C. Canada

ABSTRACT

In this work, we present details of the implementation of a Hypercycle-based router with a backtrack-to-the-origin-and-retry routing.

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.

In a back-track-to-the-origin-and-retry routing, 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.

The router can be used in constructing concurrent computer systems with varying topologies.

1.0 Introduction

Message passing concurrent computers such as the Hypercube[5], 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, and the Connection Machine[9] having adopted a regular interconnection pattern corresponding to a binary n-dimensional cube, while MAX adopts a less structured, yet unspecified topology.

In regularly structured interconnection networks easy deadlock-free routing [1] 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.

Hypercycles can be considered as products of "basic" graphs. Hypercycles are designed so as:

(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 maximum number of alternate paths from a source to a destination. The existence of alternate paths guarantees that a message will not be blocked waiting for its single route to be freed, but it would in turn search for the availability of alternate paths. This strategy also provides for fault protection, since a faulty path can be marked permanently busy, and thus messages can be routed around it..

In previous work [2,3], we have defined the class of Hypercycle networks and studied the performance of a backtrack-to-the-origin-and-retry routing that uses random link selection at intermediate points in constructing the source-to-destination circuit. The random selection of intermediate links prevents deadlocks and at the same time it provides increased throughput and fault tolerance.

2.0 Hypercycles.

An r- dimensional Hypercycle, is the following regular undirected graph: $G_{\mathbf{m}}^{\rho} \left\{ \mathcal{N}_{\mathbf{m}}^{\rho}, \mathcal{E}_{\mathbf{m}}^{\rho} \right\}$ where $\mathbf{m} = m_1, m_2, m_3, \ldots, m_r$ 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 $\mathcal{N}_{\mathbf{m}}^{\rho}$ $= \{0,1,2,\ldots,M-1\}$. Given $\alpha, \beta \in \mathcal{N}_{\mathbf{m}}^{\rho}$ then $(\alpha, \beta) \in \mathcal{E}_{\mathbf{m}}^{\rho}$ 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 $\alpha_i = \beta_i$; $i \ne j$

2.1 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[§]

$$\xi_{j_i+1} = \begin{cases} (\xi_{j_i} + \rho_i) \bmod m_i & \text{if } \left[(\xi - \xi_{j_i}) \bmod m_i \equiv \left| \xi_{j_i}, \xi \right| \right] > \rho_i \\ (\xi_{j_i} - \rho_i) \bmod m_i & \text{if } \left[(\xi_{j_i} - \xi) \bmod m_i \equiv \left| \xi_{j_i}, \xi \right| \right] > \rho_i \\ \xi & \text{if } \left| \xi_{j_i}, \xi \right| \le \rho_i \end{cases}$$

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

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**

2.2 Backtrack-to-the-origin-and-retry

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.

[¶] This research has been supported by the Natural Sciences and Engineering Research Council Canada under grant #OGP0041188

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

In an r-dimensional Hypercycle, the Next Port Generator is replicated r times.

3.2 Port Validator:

The possible ports are fed from the Next Port Generator to the Port Validator. It then matches these ports with those ports that are not currently in use. If there are any ports offered which are still available, these are forwarded. If there are no offered ports which are still available, then this unit generates a break-connection signal to be passed down the partially completed path.

3.3 Port Selector:

If the Port Validator has found available ports for routing, these are passed to the Port Selector. The Port Selector must accomplish the following tasks: count the number of valid ports, generate a random number between one and this count, select the port corresponding to the random number.

Other subsystems, primarily residing in the communication switch, would detect break signals and connect and disconnect communication links under the control of the router itself.

4.0 Design of NPG

The Next Port Generator (NPG) implements the routing equations as presented in section 2.1 and provides a physical port number to which the message can be routed. Since there is one NPG for each dimension in the system, routing for each dimension is done in parallel. Figure 4. shows a block diagram of a single NPG. The operation of the subsystem is as follows.

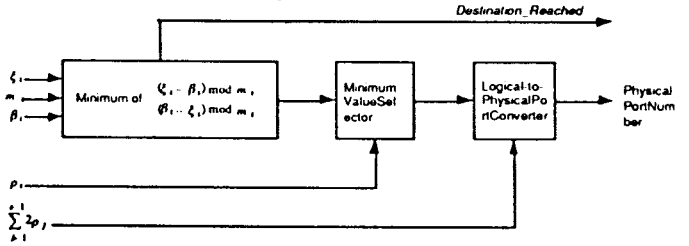

Figure 4. The block Diagram of the Next Port Generator.

Given a local address ξ_i, a dimensional population m_i, and a destination address β_i, the minimum of $(\xi_i - \beta_i) \bmod m_i$ and $(\beta_i - \xi_i) \bmod m_i$ as required by the routing in section 2.1, is calculated and forwarded. This essentially decides whether it is shorter to move clockwise or counterclockwise about the cycle and what the length of such a move would be. Note that it can be determined at this point if the destination address within this dimension has been reached since

$(\xi_i - \beta_i) \bmod m_i = 0$ if $\xi_i = \beta_i$

The next step is to decide which is smaller, the connectivity ρ_i, which decides the maximum possible leap, clockwise or counterclockwise, or the destination itself. This must be done since a step size greater than ρ_i cannot be taken. The result of this choice is the optimum logical port in this dimension (assuming a greedy strategy, as is the one used here, is employed). This logical port is then forwarded to a circuit which converts it to a physical port, based on the number of ports which have preceded this dimension. It is assumed here that the lowest port numbers are used to route for dimension 1, the next lowest possible set for dimension 2, etc.. For example, in a $G_5^{2\,1}{}_3$ Hypercycle, the first $2\rho_1 = 4$ ports are used for dimension 1 and the remaining $2\rho_2 = 2$ ports are used for dimension 2.

The above operations can be simplified noting that $\lambda \bmod n = n - |\lambda|$ for $\lambda < 0$ and $|\lambda| < n$ and $\lambda \bmod n = \lambda$ for $0 \le \lambda < n$. Therefore, it is sufficient to find the minimum of $|\beta_i - \xi_i|$ and $m_i - |\beta_i - \xi_i|$ instead of the minimum of $(\xi_i - \beta_i) \bmod m_i$ and $(\beta_i - \xi_i) \bmod m_i$

The design of the next port generator is given in Figure 5. This design implements a single dimensional next port generator with the defining parameters m_l, ρ_l, , β_l and ξ_l having the following maximum values respectively 15, 7 15 and 15.

To capture the schematic, the CASE™ Technology Stellar Design Environment[4] was used. This design package permits the hierarchical design of circuits using industry standard components.

Figure 5 shows the implementation of the block diagram of Figure 4. All of the primitive components are assumed to be of the 74LS family. The operation of this circuit is as follows.

Firstly, ξ_l is subtracted from β_l. This is done using the sign-bit subtractor component indicated by SUB1. The result of this subtraction is a positive number from 0 to 14 (since the maximum node address is 14) and a sign bit which is asserted if the result is negative. This positive number is then forwarded to another sign-bit subtractor, SUB2, which subtracts this number from m_l. The result of this subtraction is always positive since, $0 \le \beta_i < m_i$ and $0 \le \xi_i < m_i$ therefore $\beta_i - \xi_i < m_i$

The two positive numbers are then compared at the minimum value selection unit MSU1. This circuit compares two four bit numbers and forwards the lowest valued one, also generating signals to indicate if the two numbers are equal in value, EQUAL, and if the second input, B, is the low value, B_LOW.

The resulting signal emanating from MSU1 is the minimum distance to the destination. This can now be compared to ρ_i at MSU2. The minimum of these two values is the longest step possible given the configuration, clockwise or counterclockwise, to the destination. After MSU2, the direction of travel must be resolved. This is accomplished by exclusive-ORing the sign bit from SUB1 with the B_LOW signal from MSU1 to yield a signal which indicates the direction of travel for the minimum distance. This signal is valid in all situations except when both directions are valid (e.g. $|\beta_i - \xi_i| = m_i - |\beta_i - \xi_i|$). Such a situation occurs when m_i is even, $2\rho_i = m_i$ and a message from node γ_i is bound for node $(\gamma_i \pm m_i/2) \bmod m_i$. An example of such a situation is a binary n-dimensional cube in which $\rho_i = 2$ for all i and $m_i = 2$ for all i. In this case, the port from γ_i to $(\gamma_i \pm m_i/2) \bmod m_i$ must either be numbered as ρ_i or as $2\rho_i$ - 1. This is an arbitrary decision so the cross-cycle link was assigned to port ρ_i. Since this is the case, the signal derived above, which sends a message clockwise or counterclockwise, would have to be quantified with a signal which could detect the equality of the two directions. Such a signal is available from MSU1. The EQUAL signal of MSU1 is asserted if both inputs are identical, as would be the case if $|\beta_i - \xi_i| = m_i - |\beta_i - \xi_i|$. This signal, when inverted, can be used as a gating trigger for the directional signal, an indeed, this is what is done at the AND gate U0136 A in Figure 5. The resulting signal is itself a gate for ρ_i, which is simply added to the desired port if the direction is clockwise. The result of MSU2, now the largest possible step toward the destination in this dimension, is added to the gated ρ_i signal through U012. The result of this is a logical port number which

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 the routing presented in section 2.1. The backtrack-to-the-origin-and-retry routing is a type of two-phase locking, where as resources we consider the various links necessary for the completion of the source-to-destination circuit.

We have studied backtrack-to-the-origin-and-retry routing on Hypercycle-based interconnection networks [3] and established its superior performance as compared to the hypercube based networks.

The backtrack-to-the-origin-and-retry routing is currently under implementation. Figure 1 gives the structure of a computational node in a concurrent computer that incorporates Hypercycles and backtrack-to-the-origin-and-retry routing. Figure 2 gives a block diagram of an r-dimensional Hypercycle router.

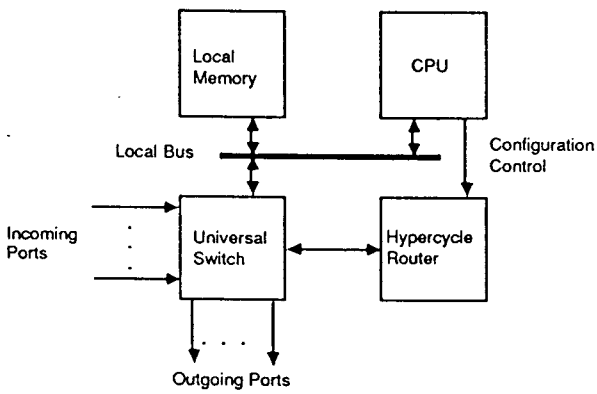

Figure 1. The configuration of a computing node that includes the Hypercycle Router.

3.0 General Design

As it can be seen in Figure 2. we are implementing our routing as a system having four modules. A destination address is presented at the Destination Address Register. The Next Port Generators then determine which ports can be used for the continuation of the path. The Port Validator ensures that the ports indicated are available for use. Then the port selector randomly chooses a single port to which the originating port is connected and to which the destination address is forwarded. The major functional units of the router,are described below:

3.1 Next Port Generator:

The subsystem termed the Next Port Generator provides the next port to which a message may be routed. This is done by using a modulo arithmetic circuit based on the routing presented in section 2.1 to obtain a relative address with respect to the current address. In each dimension i, there are $2\rho_i$ ($2\rho_i$ -1 if $\rho_i=m_i/2$) possible edges. To each of these edges, there is a corresponding communication channel and a logical port. We number the logical ports from 1 to $2\rho_i$ ($2\rho_i$-1) so that the minimum step in the counterclockwise direction will correspond to logical port 1, the maximum step to logical port ρ_i,, while the minimum step in the clockwise direction will correspond to logical port ρ_i+1 and the maximum step to port $2\rho_i$ ($2\rho_i$-1). Finally, this logical port number is mapped to a physical port number by accounting for all the ports used by previous dimensions and then adding this as an offset to the logical port.

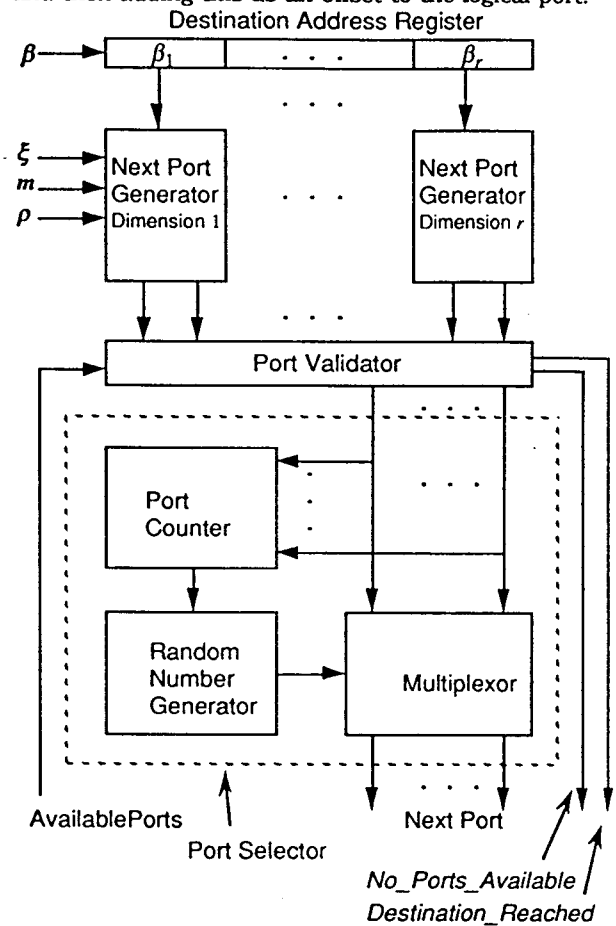

Figure 2. Block diagram of the Hypercycle routing engine. Routing in each dimension is done in parallel.

As an example, consider the case given in Figure 3. If a message starts at node 2 bound for node 5, the first step is to determine the direction of the shortest route. In this case the clockwise direction is selected and this yields a largest possible step of 2. Since the route is clockwise, the step size, 2, is offset by the connectivity, ρ_i, which produces a logical port of 4. If this were the first, or only, dimension of the Hypercycle, then this number would also correspond to the physical port. If other dimensions existed before this dimension, then the logical port would be offset by the number of ports used in all previous dimensions to yield a physical port number.

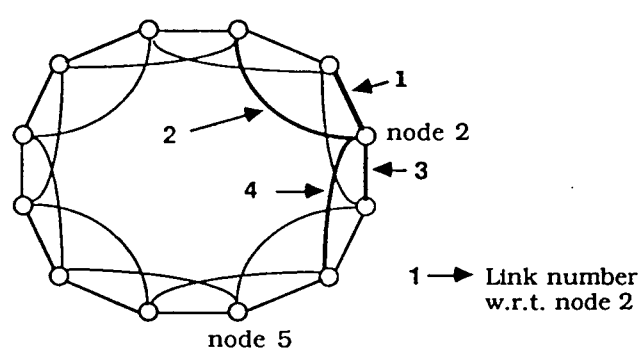

Figure 3. Determining the shortest route from node 2 to node 5.

can be mapped to a physical port number by adding it to the sum of all previous ports, as mentioned above.

5.0 Discussion

We have completed the design of the Next Port Generator and implemented it in 74LS logic. We have extensively tested this circuit and obtained maximum propagation delays of less than 125 ns. We used this implementation technology to validate our design. We are currently implementing the Next Port Generator in VLSI, and completing the design of the Port Selector.

We expect that the VLSI embodiment of our designs will be considerably faster.

REFERENCES

1. 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.
2. Dimopoulos, N., 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.
3. Dimopoulos, N. , Don Radvan, Kin F. Li "Performance Evaluation of the Backtrack-to-the-origin-and-retry routing for Hypercycle-based Intreconnection Networks" *Proceedings of the tenth International Conference on Distributed Systems*, pp. 278-284 Paris, June 1990.
4. *CASE User's Manual*, CASE Technology, Inc., Mountain View, Calif., 1988.
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. Waltz, D. L. "Applications of the Connection Machine" *IEEE Computer* January 1987, pp.85-97

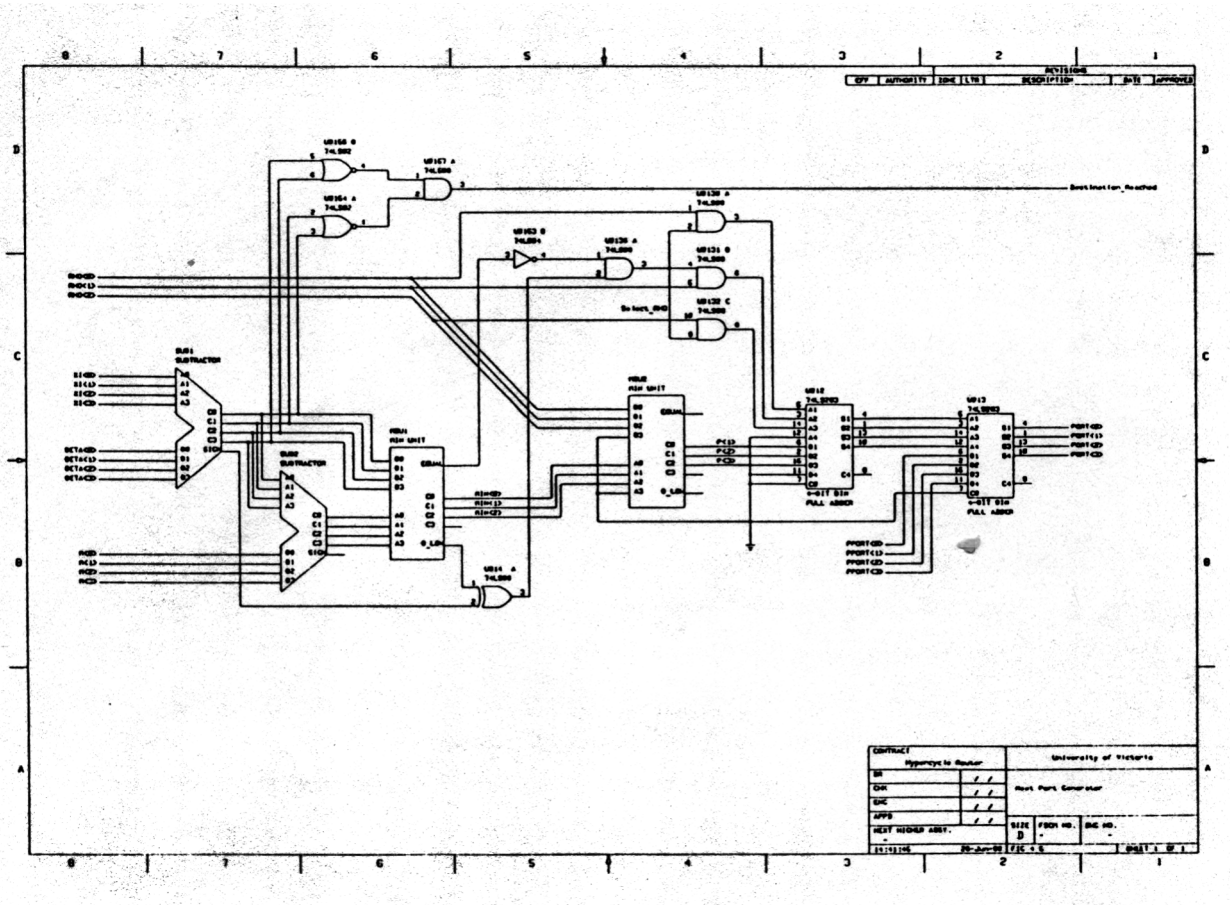

Figure 5. Implementation of the Next Port Generator