Deadlock Free Message Routing in Multiprocessor Interconnection Networks

William J Dally

and

Charles L Seitz

Computer Science Department
California Institute of Technology

5206:TR:86

Deadlock Free Message Routing in Multiprocessor Interconnection Networks

William J. Dally
Charles L. Seitz

5206:TR:86

May 10, 1985

Abstract

A deadlock-free routing algorithm can be generated for arbitrary interconnection networks using the concept of virtual channels. A necessary and sufficient condition for deadlock-free routing is the absence of cycles in the channel dependency graph. Given an arbitrary network and a routing function, the cycles of the channel dependency graph can be removed by splitting physical channels into groups of virtual channels. This method is used to develop deadlock-free routing algorithms for k-ary n-cubes, for cube connected cycles, and for shuffle-exchange networks.

Index Terms - Interconnection networks, communication networks, concurrent computation, parallel processing, message passing multiprocessors, graph model.

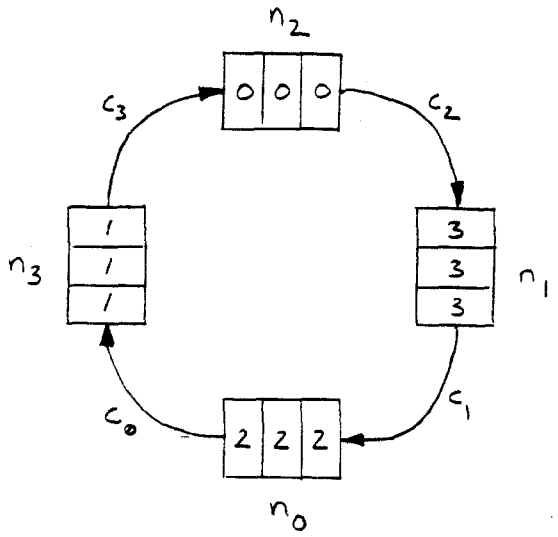


Figure 1: Deadlock in a 4-Cycle

1 Introduction

Message passing concurrent computers such as the Caltech Cosmic Cube [1] consist of many processing nodes that interact by sending messages over communication channels between the nodes. Deadlock in the interconnection network of a concurrent computer occurs when no message can advance toward its destination because the queues of the message system are full [2]. Consider the example shown in Figure 1. The queues of each node in the 4-cycle are filled with messages destined for the opposite node. No message can advance toward its destination; thus the cycle is deadlocked. In this locked state, no communication can occur over the deadlocked channels until exceptional action is taken to break the deadlock.

Reliable concurrent computation requires a routing algorithm that is provably free of deadlock. In this paper we present a general method for constructing deadlock-free routing algorithms for arbitrary networks. In Section 2 we state our assumptions and develop the necessary and sufficient conditions on a *channel dependency graph* for a routing to be deadlock-free. These conditions are used in Section 3 to develop a method of constructing deadlock-free algorithms for arbitrary networks. This method is based on the concept of virtual channels: groups of channels that share a physical channel but each have their own queue. Using virtual channels, we develop deadlock-free routing algorithms for arbitrary k-ary n-cubes in Section 4, for cube-connected cycles in Section 5, and for shuffle-exchange networks in Section 6. In each of these examples, physical channels belonging to cycles are split into a group of virtual channels. The virtual channels are ordered; routing is restricted to visit channels in decreasing order to eliminate cycles in the channel dependency graph.

2 Deadlock Free Routing

We assume the following:

- A message arriving at its destination node is eventually consumed.
- A node can generate messages destined for any other node.
- Between any two processing nodes i and j the order of messages transmitted directly from i to j must be preserved.
- There is room to queue at least two message fragments (possibly a single bit each) for each channel.
- Nodes can produce messages at any rate subject to the constraint of available queue space (source queued).

The following definitions develop a notation for describing networks, routing functions and configurations. A summary of notation is given below.

Definition 1 An *interconnection network*, I is a strongly connected *directed* graph, $I = G(N, C)$. The vertices of the graph, N, represent the set of processing nodes. The edges of the graph, C, represent the set of communication channels. The source node of channel c_i is denoted s_i and the destination node d_i. Associated with each channel, c_i, is a queue with capacity $\text{cap}(c_i)$ containing $\text{size}(c_i)$ messages. If the queue for channel c_i contains a message destined for node n_d then $\text{member}(n_d, c_i)$ is true. There may be redundant channels between two nodes. For purposes of analysis, we add an internal channel c_{n_i} to C for each node n_i. Messages originating at node n_i are considered to have arrived on c_{n_i}.

Definition 2 A *routing function* $\mathbf{R} : C \times N \mapsto C$ maps the current channel, c_c, and destination node, n_d, to the next channel c_n on the route from c_c to n_d, $\mathbf{R}(c_c, n_d) = c_n$. A channel is not allowed to route to itself, $c_c \neq c_n$. Note that this definition restricts the routing to be memoryless in the sense that a message arriving in channel c_c, destined for n_d has no memory of the route that brought it to c_c. However, this formulation of routing as a function from $C \times N$ to C has more memory than the conventional definition of routing as a function from $N \times N$ to C. Making routing dependent on the current channel rather than the current node allows us to develop the idea of channel dependence.

Definition 3 A *channel dependency graph*, D, for a given interconnection network, I, and routing function, $\mathbf{R}$, is a directed graph, $D = G(C, E)$. The vertices of D are the channels of I. The edges of D, are the pairs of channels connected by $\mathbf{R}$:

$$E = \{(c_i, c_j) | \mathbf{R}(c_i, n) = c_j \text{ for some } n \in N\}. \tag{1}$$

Note that, since channels are not allowed to route to themselves, there are no 1-cycles in D.

Definition 4 A *configuration* is an assignment of a subset of N to each queue. A configuration is legal if

$$\forall c_i \in C,\ \text{size}(c_i) < \text{cap}(c_i). \tag{2}$$

Definition 5 A *deadlocked configuration* for a routing function, **R**, is a non-empty legal configuration of channel queues $\ni$

$$\forall c_i \in C,\ (\forall n \ni \text{member}(n, c_i),\ n \neq d_i \text{ and } c_j = \mathbf{R}(c_i, n) \ \Rightarrow\ \text{size}(c_j) = \text{cap}(c_j)) \qquad (3)$$

In this configuration no message is one step from its destination and no message can advance because the queue for the next channel is full. A routing function, **R**, is *deadlock-free* on an interconnection network, I, if no deadlock configuration exists for that function on that network.

Summary of Notation

I	interconnection network, a directed graph $I = G(N, C)$,
N	the set of nodes,
n_i	a node,
C	the set of channels,
c_i	a channel,
c_{n_i}	the internal channel of node n_i,
s_i	the source node of channel c_i,
d_i	the destination node of channel c_i,
$\text{cap}(c_i)$	the capacity of the queue of channel c_i,
$\text{size}(c_i)$	the number of messages enqueued for channel c_i,
$\text{member}(n, c_i)$	true if a message destined for node n is enqueued for channel c_i,
$\mathbf{R}$	a routing function $\mathbf{R} : C \times N \mapsto C$,
D	the channel dependency graph.

Theorem 1 A routing function, **R**, for an interconnection network, I, is deadlock-free iff there are no cycles in the channel dependency graph, D.

Proof:

$\Rightarrow$ Suppose a deadlock-free network has a cycle in D. Since there are no 1-cycles in D, this cycle must be of length two or more. Thus one can construct a deadlocked configuration by filling the queues of each node in the cycle with messages destined for a node two channels away where the first channel of the route is along the cycle. Contradiction.

$\Leftarrow$ Suppose a network with no cycles in D is deadlocked. Since D is acyclic one can assign a total order to the channels of C so that if $(c_i, c_j) \in E$ then $c_i > c_j$. Consider the least channel in this order with a full queue, c_l. Every channel, c_n, that c_l feeds is less than c_l, and thus does not have a full queue. Thus, no message in the queue for c_l is blocked and one do not have deadlock. ■

3 Constructing Deadlock-Free Routing Algorithms

Given a routing function, **R**, and a network, I, that is not deadlock-free, we can construct a deadlock-free routing ($\mathbf{R}'$,I) by removing the cycles from (**R**,I). In some cases, such as a cycle with channels in only one direction, it is not possible to remove edges without reducing the set of nodes that can be reached from a given node. In these cases we can break cycles by splitting each physical channel along the cycle into a group of *virtual channels*. Each group of virtual channels shares a physical communication channel; however, each virtual channel requires its own queue. The purpose of virtual channels is to associate different priorities to messages traversing the same physical channel depending on the channel from which a message arrives and the node to which the message is destined. By assigning priorities so that a message's priority always increases as it moves closer to its destination, we can construct deadlock-free routing algorithms.

Consider for example the case of a unidirectional four-cycle as shown in Figure 2A, $N = \{n_0, \ldots, n_3\}$, $C = \{c_0, \ldots, c_3\}$. The interconnection graph, I is shown on the left and the dependency graph, D is shown on the right. We pick channel c_0 to be the dividing channel of the cycle and split each channel into high virtual channels, $c_{10}, \ldots, c_{13}$, and low virtual channels, $c_{00}, \ldots, c_{03}$, as shown in Figure 2B. We can consider the internal channel of node n_i to be numbered c_{2i}. Priority in use of the physical channel is given to the low virtual channel.

Messages at a node numbered less than their destination node are routed on the high channels and messages at a node numbered greater than their destination node are routed on the low channels. Channel c_{00} is not used. We now have a total ordering of the virtual channels according to their subscripts: $c_{13} > c_{12} > c_{11} > c_{10} > c_{03} > c_{02} > c_{01}$. Thus, there is no cycle in D and the routing function is deadlock-free. In the next three sections we apply this technique to three practical communications networks. In each case we add virtual channels and restrict the routing to route messages in order of decreasing channel subscripts.

Some possible implementations of virtual channels are shown in Figure 3. A parallel implementation of virtual channels is shown in Figure 3A. Virtual channels between nodes n_i and n_j are multiplexed over a physical channel c_p. Queues $Q_{0p}, \ldots, Q_{kp}$ in n_i contain messages enqueued for transmission over virtual channels $c_{0p}, \ldots, c_{kp}$ all of which are multiplexed onto physical channel c_p. A priority encoder (PE), selects the highest priority non-empty queue, Q_{lp}, and enables the next packet of its data onto the physical channel. The data includes routing information which selects the next queue along the route for the message in n_j. When there is room in this queue, the packet is acknowledged and removed from Q_{lp}. For multi-packet messages, the routing information can be transmitted just once and stored in node n_j.

As long as a high-priority queue, Q_{hp}, is non-empty it blocks any lower priority queues, $Q_{0p}, \ldots, Q_{(h-1)p}$ from transmitting their contents on the physical channel. This blocking of lower priority channels may be undesirable for message-flow performance, but does not introduce a deadlock. In a practical system a more complex protocol could be used to grant lower priority virtual channels access to the physical channel while a high priority channel is blocked.

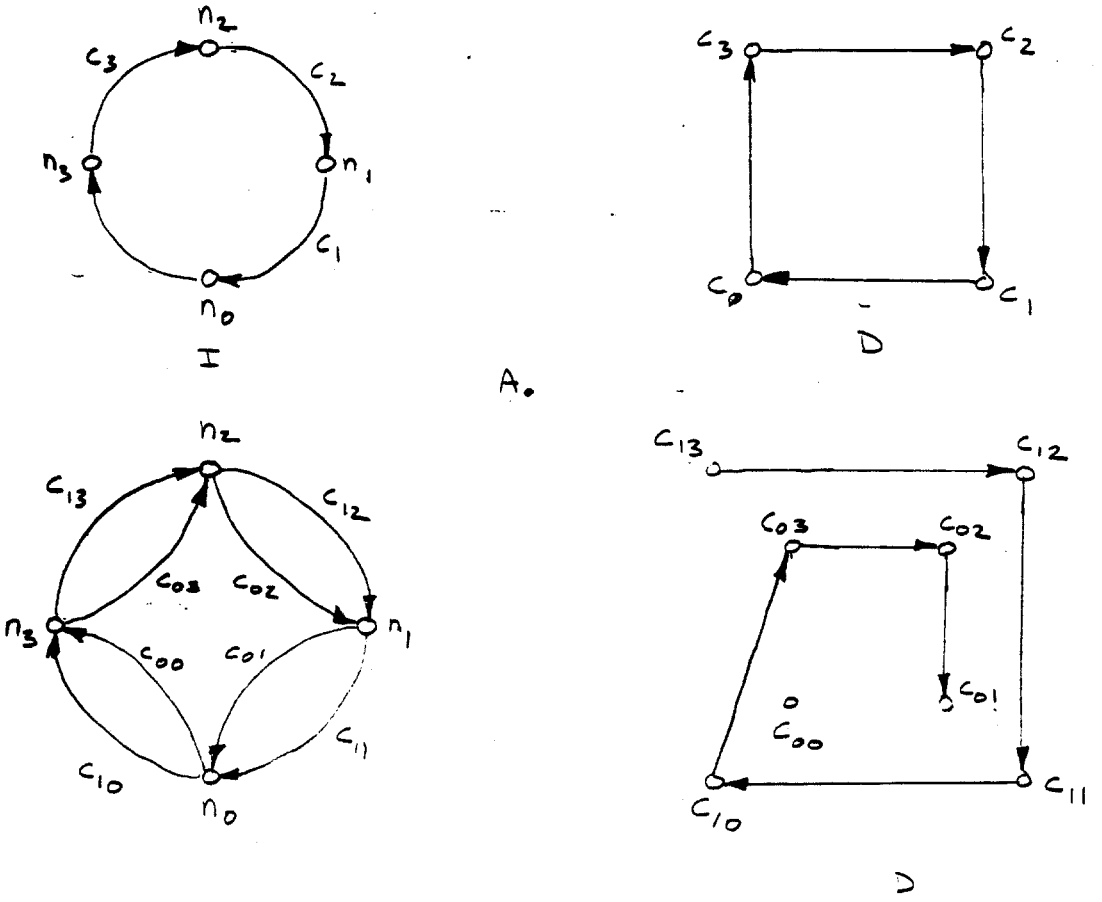
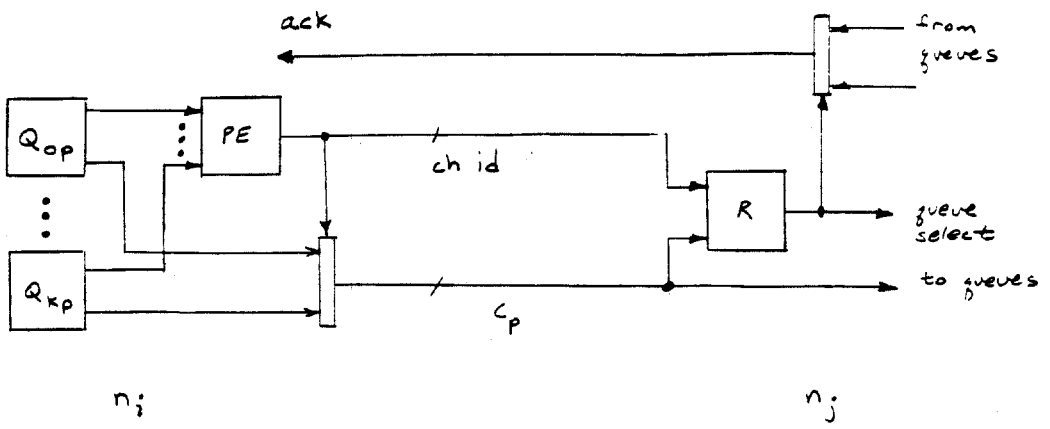
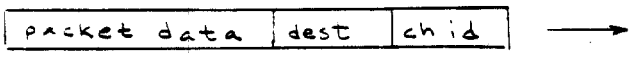


Figure 2: Breaking Deadlock by Adding Virtual Edges



A - Parallel Implementation



B - Format for Serial Implementation

Figure 3: Implementation of Virtual Channels

The same protocol can be implemented serially using the format shown in Figure 3B. The virtual channel id, routing information and data are transmitted serially from n_i to n_j. When this packet is accepted by n_j, an acknowledge signal is transmitted back to n_i to remove the packet from its queue and enable transmission of the next packet.

4 K-ary n-cubes

The E-cube routing algorithm [3,4] guarantees deadlock free routing in binary n-cubes. In a cube of dimension d, we denote a node as n_k where k is an d-digit binary number. Node n_k has d output channels, one for each dimension, labeled $c_{0k}, \ldots, c_{(d-1)k}$. The E-cube algorithm routes in decreasing order of dimension. A message arriving at node n_k

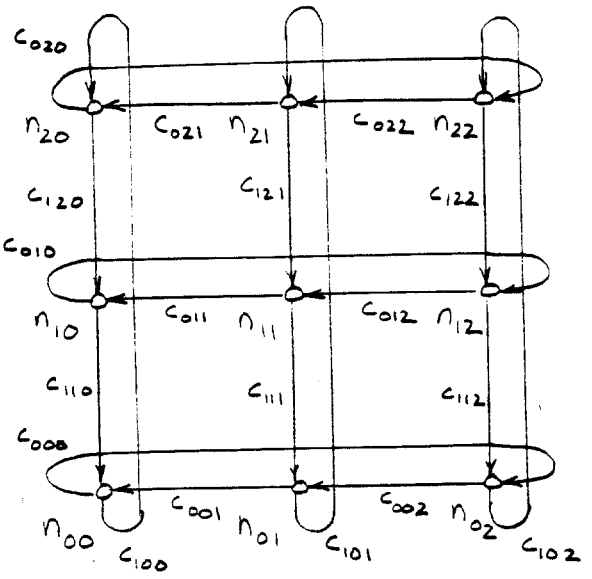


Figure 4: 3-ary 2-cube

destined for node n_l is routed on channel c_{ik} where i is the position of the most significant bit in which k and l differ. Since messages are routed in order of decreasing dimension and hence deceasing channel subscript, there are no cycles in the channel dependency graph and E-cube routing is deadlock-free.

Using the technique of virtual channels, this routing algorithm can be extended to handle all k-ary n-cubes: cubes with dimension n and k nodes in each dimension. Rings and toroidal meshes are included in this class of networks. This algorithm can also handle mixed radix cubes. Each node of a k-ary n-cube is identified by an n-digit radix k number. The i^{th} digit of the number represents the node's position in the i^{th} dimension. For example, the center processor in the 3-ary 2-cube of Figure 4 is n_{11}. The channels are identified by the number of their source node and their dimension. For example, the dimension 0 (horizontal) channel from n_{11} to n_{10} is c_{011}. To break cycles we divide each channel into an upper and lower virtual channel. The upper virtual channel of c_{011} will be labeled c_{0111}, and the lower virtual channel will be labeled c_{0011}. To give internal channels the lowest priority, they are labeled with a dimension higher than the dimension of the cube. To assure that the routing is deadlock-free, we restrict it to route through channels in order of descending subscripts. Priority is always given to the message from the channel with a lower subscript.

As in the E-cube algorithm we route in order of dimension, most significant dimension first. In each dimension, i, a message is routed in that dimension until it reaches a node whose subscript matches the destination address in the i^{th} position. The message is routed on the high channel if the i^{th} digit of the destination address is greater than the i^{th} digit of the present node's address. Otherwise the message is routed on the low channel. It is

easy to see that this routing algorithm routes in order of descending subscripts, and is thus deadlock-free.

Formally, we define the routing function:

$$\mathbf{R_{KNC}}(c_{dvn}, n_j) = \begin{cases} c_{d1(n-r^d)} \text{ if } (\mathrm{dig}(n,d) < \mathrm{dig}(j,d)) \wedge (\mathrm{dig}(n,d) \neq 0), \\ c_{d0(n-r^d)} \text{ if } (\mathrm{dig}(n,d) > \mathrm{dig}(j,d)) \vee (\mathrm{dig}(n,d) = 0), \\ c_{i1(n-r^d)} \text{ if } (\forall k > i, \mathrm{dig}(n,k) = \mathrm{dig}(j,k)) \wedge \\ \qquad\qquad (\mathrm{dig}(n,i) \neq \mathrm{dig}(j,i)). \end{cases} \tag{4}$$

Where $\mathrm{dig}(n,d)$ extracts the d^{th} digit of n, and r is the radix of the cube. The subtraction, $n - r^d$, is assumed to be performed modulo r.

Assertion **1** The routing function, $\mathbf{R_{KNC}}$, correctly routes messages from any node to any other node in a k-ary n-cube.

Proof: By induction on dimension, d.

For $d = 1$, a message, destined for n_j, enters the system at n_i on the internal channel, c_{d0i}. If $i < j$, the message is forwarded on channels, $c_{01i}, \ldots, c_{010}, c_{00r}, \ldots, c_{00(j+1)}$ to node n_j. If $i > j$, the path taken is, $c_{00i}, \ldots, c_{00(j+1)}$. In both cases the route reaches node n_j.

Assume that the routing works for dimensions $\leq d$ Then for dimension $d+1$ there are two cases. If $\mathrm{dig}(i,d) \neq \mathrm{dig}(j,d)$, then the message is routed around the most significant cycle to a node $n_k \ni \mathrm{dig}(k,d) = \mathrm{dig}(j,d)$, as in the $d = 1$ case above. If $\mathrm{dig}(i,d) = \mathrm{dig}(j,d)$, then the routing need only be performed in dimensions d and lower. In each of these cases, once the message reaches a node, n_k, $\ni \mathrm{dig}(k,d) = \mathrm{dig}(j,d)$, the third routing rule is used to route the message to a lower dimensional channel. The problem has then been reduced to one of dimension $\leq n$ and the routing reaches the correct node by induction. ∎

Assertion **2** The routing function $\mathbf{R_{KNC}}$ on a k-ary n-cube interconnection network, I, is deadlock-free.

Proof: Since routing is performed in decreasing order of channel subscripts, $\forall c_i, c_j, n_c \ni \mathbf{R}(c_i, n_c) = c_j$, $i > j$, the channel dependency graph, D is acyclic. Thus by Theorem 1 the route is deadlock-free. ∎

5 Cube Connected Cycles

The cube-connected cycle (CCC) [5] is an interconnection network based on the binary n-cube. In the CCC, each node of a a binary n-cube is replaced with an n-cycle and the cube connection in the n^{th} dimension is attached to the n^{th} node in the cycle. A CCC of dimension 3 is shown in Figure 5.

Each node in the CCC is labeled with the position of its cycle, (an n-bit binary number), and its position within the cycle. For example, in Figure 5, processor 2 in cycle 111 is labeled n_{2111}. There are two channels out of each node: an in-cycle channel and an out-of-cycle channel. The in-cycle channel is split into three virtual channels. One set of virtual channels is used to rotate a message around the cycle to get to the most significant node in the cycle. These channels are labeled c_{2d0ccc} where d is the dimension of the node, (its position in the cycle), and ccc is the cycle address. The next set of virtual channels is used to decrement the dimension during the E-cube routing of the message between cycles. These channels are labeled c_{1d1ccc}. The out-of-cycle channels, labeled c_{1d0ccc}, are used to toggle the bit of the current cycle address corresponding to dimension d. Note that the channels c_{1d0ccc} are actually physical channels. These connections are not shared with any other channels. The third set of virtual channels is used to rotate the message around the cycle to its destination once it is in the proper cycle. These channels are labeled c_{0d0ccc}. As above, we will restrict our routing to route through channels in order of descending subscripts. Priority is given to messages from channels with lower subscripts.

The routing algorithm proceeds in three phases. During phase 1, messages are routed around the cycle using the first set of virtual channels until they reach a node with dimension greater than or equal to the position of the most significant bit in which the destination cycle address differs from the current cycle address. During phase 2 we route the message to the proper cycle using a variant of the E-cube algorithm. At each step of phase 2, we find the most significant dimension, i, in which the current cycle and destination cycle addresses differ. The message is routed around the current cycle until it reaches the node with dimension i and is then routed out of the cycle. When the message arrives in the destination cycle it is routed around the cycle using the third set of virtual channels to reach its destination node. It is easy to see that routing is always performed in order of decreasing channel numbers, and thus the routing is guaranteed to be deadlock-free.

While most cube-connected cycles are binary, this routing algorithm can be extended for k-ary cube-connected cycles, that is, cycles with k cycles in each dimension. The only modification required is to split each out-of-cycle channel into two virtual channels. For simplicity, however we will analyze the routing for the binary case only. Formally, we define the routing function:

$$\mathbf{R_{CCC}}(c_{vdxn}, n_{d'j}) = \begin{cases} c_{2(d-1)0n} & \text{if } (v \geq 2) \wedge (d > 0) \wedge \\ & \quad (\exists i > d \ni \operatorname{dig}(i,n) \neq \operatorname{dig}(i,j)), \\ c_{1(d-1)1n} & \text{if } (v \geq 1) \wedge (x = 0) \wedge \\ & \quad (\forall i > d \operatorname{dig}(i,n) = \operatorname{dig}(i,j)) \wedge \\ & \quad (\operatorname{dig}(d,n) \neq \operatorname{dig}(i,j)), \\ c_{1(d-1)0n} & \text{if } (v \geq 1) \wedge (x = 0) \wedge \\ & \quad (\forall i \geq d \operatorname{dig}(i,n) = \operatorname{dig}(i,j)), \\ c_{1d0(n-r^d)} & \text{if } (v = 1) \wedge (x = 1) \wedge \\ & \quad (\forall i \geq d \operatorname{dig}(i,n) = \operatorname{dig}(i,j)), \\ c_{1d1(n-r^d)} & \text{if } (v = 1) \wedge (x = 1) \wedge \\ & \quad (\forall i > d \operatorname{dig}(i,n) = \operatorname{dig}(i,j)) \wedge \\ & \quad (\operatorname{dig}(d,n) \neq \operatorname{dig}(i,j)), \\ c_{0(d-1)0n} & \text{if } (x = 0) \wedge (n = j) \wedge (d' < d), \\ c_{0(d)0n} & \text{if } (x = 1) \wedge (n = j) \wedge (d' < d). \end{cases} \tag{5}$$

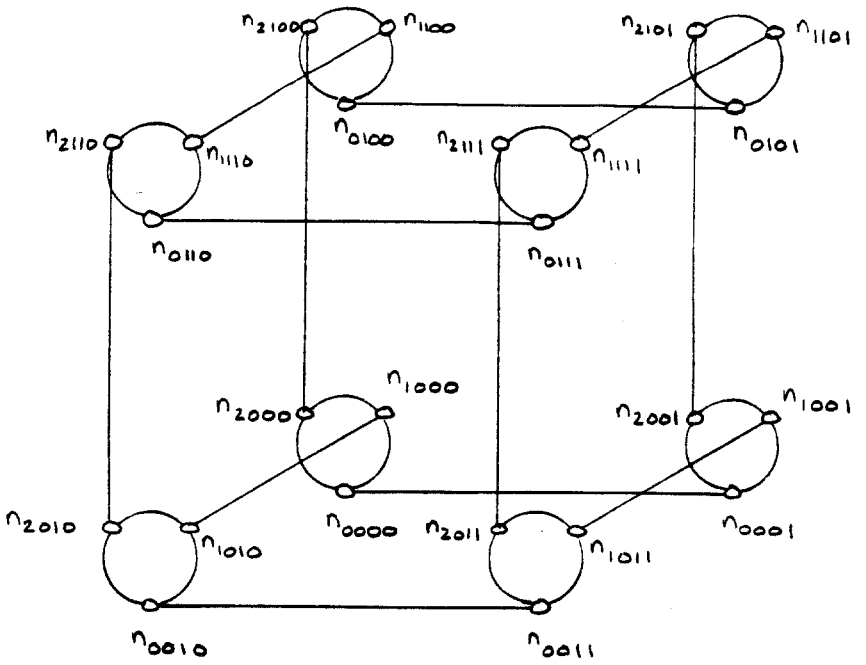


Figure 5: Cube Connected Cycle of Dimension 3

The following two assertions apply to the routing function, $\mathbf{R_{CCC}}$.

Assertion **3** $(v < 2) \Rightarrow \mathrm{dig}(i, n) = \mathrm{dig}(i, n')\ \forall i > d$.

Proof: The only way v can be decreased to less than 2 is for the right side of the assertion to hold. Initially $v = 3$, the message arrives on the internal channel. Then as long as $\exists i > d \ni \mathrm{dig}(i, n) \neq \mathrm{dig}(i, n')$, the first routing rule forwards the message along channels for which $v = 2$. When the arriving channel has $d = 0$, the current node has $d = m$ (where m is the dimension of the CCC). In this case the first routing rule forwards the message along a channel for which $v = 1$. The assertion holds since there are only m digits in n and n', so there is no $i > m$ for which the i^{th} digit of these two cycle addresses differ.

Once $v < 2$, the right side of the assertion continues to hold because of routing rules 2,3,4 and 5. We prove this by induction on d. For $d = m$, the assertion holds as stated above. If we assume the assertion holds for $d = i$, then for $d = i - 1$ it also holds since routing rules 2 and 5 route the messages out-of-cycle to toggle the d^{th} digit of n if the addresses disagree in that digit. ■

Assertion **4** $(v = 0) \Rightarrow (n' = n) \wedge (d' < d)$.

Proof: The only way to decrease v to zero is by routing rules 6 and 7 which require the right side of the assertion. By Assertion 3, when $v < 2$ and $d = 0$, after one or two more

traversals, the right side of the assertion will be met. Once $v = 0$, no out-of-cycle channels will be used so n does not change. ■

Assertion **5** The routing function, $\mathbf{R}_{CCC}$, correctly routes messages from any node to any other node in a k-ary cube-connected cycle.

Proof: Since there are only a finite number of channels, and $\mathbf{R}_{CCC}$ routes in order of decreasing channel subscripts, v will eventually be decreased to 0. Then by Assertion 4, $n' = n$ and $d' < d$, so the message will be rotated about the cycle until it reaches its destination. ■

Assertion **6** The routing function $\mathbf{R}_{CCC}$ on a k-ary cube-connected cycle interconnection network, I, is deadlock-free.

Proof: As in the case of k-ary n-cubes, since routing is performed in decreasing order of channel subscripts, $\forall c_i, c_j, n_c \ni \mathbf{R}(c_i, n_c) = c_j$, $i > j$, the channel dependency graph, D is acyclic. Thus by Theorem 1 the route is deadlock-free. ■

6 Shuffle-Exchange Networks

The shuffle-exchange network [6], shown in Figure 6, provides two channels out of each node: a shuffle channel and an exchange channel. The shuffle channel from node n_i has as its destination the node n_j where the binary representation of j is the left rotation of the binary representation of i, denoted here $j = \text{rol}(i)$. The exchange channel from n_i routes messages to n_k where the binary representations of k and i differ in the least significant bit.

The exchange channel out of n_i is labeled c_{1i}. The shuffle channel is labeled c_{0i}. For the shuffle-exchange network we split each channel into n virtual channels where $N = 2^n$. That is, we have one virtual channel for each bit of node address. Readers understanding the relationship between the binary n-cube and the shuffle will find this assignment of virtual channels unsuprising. The virtual channels are labeled c_{dxi}. Where $0 \leq d \leq n-1$, $x \in \{0, 1\}$, and $0 \leq i \leq N$. The internal channel at each node is labeled with $d = n$ to give it lower priority than any other channel.

The routing algorithm, like the E-cube algorithm, routes a message toward its destination one bit at a time beginning with the most significant bit. At the i^{th} step of the route, the $n - i^{\text{th}}$ bit of the destination address is compared to the least significant bit of the current node address. If the two bits agree, the message is forwarded over the shuffle channel to rotate the node address around to the next bit. Otherwise, the message is forwarded over the exchange channel to bring the two bits into agreement and then over the shuffle channel to rotate the address. At the i^{th} step messages are forwarded over channels with $d = n - i$. Since d is always decreasing and, during a single step, the exchange channel is used before the shuffle channel, messages are routed in order of decreasing virtual channel subscripts.

Formally, we define the routing function:

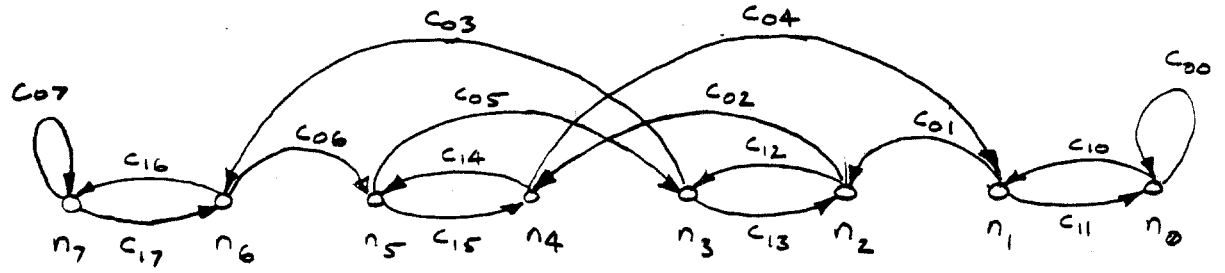


Figure 6: Shuffle-Exchange Network, $N = 8$

$$\mathbf{R_{SEN}}(c_{dxi}, n_j) = \begin{cases} c_{(d-1)0\mathrm{rol}(i)} & \text{if } (x = 0) \wedge (\mathrm{dig}(d-1, i) = \mathrm{dig}(0, j)), \\ c_{(d-1)1\mathrm{rol}(i)} & \text{if } (x = 0) \wedge (\mathrm{dig}(d-1, i) \neq \mathrm{dig}(0, j)), \\ c_{d0(i \oplus 1)} & \text{if } (x = 1). \end{cases} \tag{6}$$

Assertion **7** If a message is routed on channel c_{d0i} destined for node n_j, then $\forall m \geq d$, $\mathrm{dig}(m, j) = \mathrm{dig}(m, k)$, where k is i rotated left d bits.

Proof: By induction on dimension, d. For $d = n-1$, a message is only routed on the shuffle connection, $x = 0$, of dimension $n-1$ if $\mathrm{dig}(n-1, i) = \mathrm{dig}(0, j) \Rightarrow \mathrm{dig}(n-1, i) = \mathrm{dig}(n-1, k)$ by the definition of k. Since there is only one possible value for m, the assertion is satisfied.

If the assertion is true for dimension d, then after routing on connection $c_{(d-1)0i}$, the assertion also holds for $d-1$ by the same argument: a message is only routed on the shuffle connection, $x = 0$, of dimension $d-1$ if $\mathrm{dig}(d-1, i) = \mathrm{dig}(0, j) \Rightarrow \mathrm{dig}(d-1, i) = \mathrm{dig}(d-1, k)$. ■

Assertion **8** The routing function, $\mathbf{R_{SEN}}$, correctly routes messages from any node to any other node in a shuffle-exchange network.

Proof: From Assertion 7, after routing on channel c_{00i}, the message will be at its destination. It may reach its destination before this. Since the function routes in order of decreasing channel subscripts and there are a finite number of channels messages will reach their destinations. ■

Assertion **9** The routing function, $\mathbf{R_{SEN}}$, on a shuffle-exchange network, I, is deadlock-free.

Proof: Since routing is performed in order of decreasing channel numbers, D is acyclic and the routing is deadlock-free. ∎

7 Conclusion

We have shown how a deadlock-free routing algorithm can be constructed for an arbitrary communications network by introducing virtual channels. This technique has been applied to construct deadlock-free routing algorithms for k-ary n-cubes, for cube-connected cycles, and for shuffle exchange networks.

The use of virtual channels to construct deadlock-free routing functions is motivated by the definition of a routing function that maps $C \times N$ to C, rather than the conventional definition of a routing function that maps $N \times N$ to C. By including C in the domain of the routing function, we explicitly define the dependencies between channels. These dependencies are represented by a channel dependency graph D. A necessary and sufficient condition for deadlock-free routing is that D be acyclic.

To develop deadlock-free routing algorithms for specific networks we assign a subscript to each virtual channel using a mixed radix notation. Routing is performed in order of decreasing subscripts. Since the subscripts define a total order on the channels, there are no cyclic dependencies and the routing is deadlock-free.

The cost of implementing virtual channels need not be high. Each virtual channel requires its own queue, but the queue size can be as small as the unit of data that is transmitted on each handshake, possibly a single bit. With single bit queueing, virtual channels can be used to implement the low latency wormhole routing technique [7].

The availability of deadlock-free routing algorithms encourages the investigation of different interconnection topologies. While O(logN) diameter networks such as the binary n-cube and the shuffle are attractive because of their richness of interconnection, these networks are almost always embedded in a grid for physical implementation. In keeping with the VLSI imperative of making form fit function, high bandwidth grid interconnections may turn out to be more attractive.

References

[1] Seitz, Charles L., "The Cosmic Cube," *CACM*, 28(1), January 1985, pp. 22-33.

[2] Kleinrock, Leonard, ***Queueing Systems,*** Wiley, 1976, Vol. 2, pp. 438-440.

[3] Sullivan, H. and Brashkow, T.R., "A Large Scale Homogeneous Machine," *Proc. 4th Annual Symposium on Computer Architecture,* pp 105-124, 1977.

[4] Lang, Charles R., *The Extension of Object-Oriented Languages to a Homogeneous Concurrent Architecture,* Caltech Ph.D. Thesis, 5014:TR:82, 1982, pp. 118-124.

[5] Preparata, F.P. and Vuillemin, J.E., "The Cube-Connected Cycles: A Versatile Network for Parallel Computation," *Proc. 20th IEEE Symposium on the Foundations of Computer Science,* pp. 140-147.

[6] Stone, H.S., "Parallel Processing with the Perfect Shuffle," *IEEE Transactions on Computers,* vol. C-20, February 1971, pp. 153-161.

[7] Seitz, C. et. al, *Wormhole Chip Project Report,* Winter 1985.