Adaptive Multimodule Routers

Rajendra V. Boppana*
Computer Science Division
The Univ. of Texas at San Antonio
San Antonio, TX 78249-0667
boppana@cs.utsa.edu

Suresh Chalasani
ECE Department
University of Wisconsin-Madison
Madison, WI 53706-1691
suresh@cauchy.ece.wisc.edu

Abstract. Recent multiprocessors such as Cray T3D support interprocessor communication using partitioned dimension-order routers (PDRs). In a PDR implementation, the routing logic and switching hardware is partitioned into multiple modules, with each module suitable for implementation as a chip. This paper proposes a method to incorporate adaptivity into such routers with simple changes to the router structure and logic. We show that with as few as two virtual channels per physical channel, adaptivity can be provided to handle nonuniform traffic in multidimensional meshes.
Keywords: adaptive routing, mesh networks, multicomputers, multimodule routers, wormhole routing.

1 Introduction

Many recent experimental and commercial multicomputers and multiprocessors [6, 14, 18] use grid topology based networks such as meshes and tori. Majority of these multicomputers use the dimension-order or *e*-cube routing with *wormhole* (WH) switching [8]. Wormhole is a form of cut-through routing in which blocked messages hold on to the channels they already reserved.

In practice, the *e*-cube routing is implemented using multiple modules such that each module handles routing of messages in exactly one dimension. We refer to this implementation as the multimodule or partitioned dimension-order router (PDR) implementation [6, 7, 9, 14, 18].

For example, the Cray T3D uses a 3D torus network with each PDR implemented using three chips—one chip for each dimension module. An alternative router implementation is to use centralized crossbars to handle the switching in each router. While crossbar implementations can offer adaptivity and more flexibility, each crossbar chip requires more number of pins than the module chips used as the building block for PDR implementations. Thus, for the same technology, a PDR implementation yields wider channels compared to the crossbar implementation.

The *e*-cube is simple to implement and provides high throughput for uniform traffic. But it cannot handle well nonuniform traffic such as matrix transpose and bit reversal that occurs in parallel computing, due to its non-adaptive routing. Adaptive cut-through routing algorithms has been the subject of extensive research in recent years [1, 2, 10, 11, 12, 13, 15, 16, 17]. These results implicitly or explicitly assume routers with centralized crossbars. Therefore, such techniques are not suitable for multiprocessors with PDRs. Several other results on adaptive routing exploit the rich interconnection structure of hypercubes and are not suitable for high-radix, low-dimensional meshes and tori.

In this paper, we propose a technique to incorporate adaptivity into networks with PDRs implemented using multiple chips. Our approach is to provide partial adaptivity with a small increase in hardware and routing complexity, rather than provide full adaptivity, which is expensive to implement and requires extensive redesigning of the existing routers. The main contribution of this work is to show that partitioned dimension-order routers can be enhanced for adaptive routing *without using crossbars*. We show that with a small increase in the resources and simple changes to the router organization and routing logic, a router can be made versatile enough to handle uniform and nonuniform traffic well.

Section 2 gives an overview of dimension-order routers. Section 3 describes the changes to the router required for adaptive routing. Section 4 presents the proposed adaptive routing technique and the routing logic. Section 5 concludes this paper.

*Boppana's research has been partially supported by DOD/AFOSR grant F49620-96-1-0472, NSF grant CDA 9529541, and NSF grant CDA 9633299.

International Conference on High Performance Computing, pp. 342-347, December 1997

2 Partitioned Dimension-Order Routers

A (k, n)-mesh has n dimensions—DIM_0, ..., DIM_{n-1}, k nodes per dimension, and $N = k^n$ nodes. Each node is uniquely indexed by a radix-k n-tuple. Each node is connected via communication links to at most two other nodes in each dimension. The neighbors of node $x = (x_{n-1}, \ldots, x_0)$ in dimension i are $(x_{n-1}, \ldots, x_{i+1}, x_i \pm 1, x_{i-1}, \ldots, x_0)$, if they exist. Each link provides full-duplex communication using two unidirectional physical channels. A (k, n)-torus is a (k, n)-mesh with wraparound links; a link is said to be a *wraparound link* if it connects nodes $(x_{n-1}, \ldots, x_{i+1}, 0, x_{i-1}, \ldots, x_0)$ and $(x_{n-1}, \ldots, x_{i+1}, k-1, x_{i-1}, \ldots, x_0)$ in dimension i, $0 \leq i < n$. In this paper, we concentrate on (k, n)-mesh networks. Each node is a combination of processor, memory, and router. Since our interest in this paper is in the routing part of a node, we use node and router synonymously.

A DIM_i channel is one that connects two nodes whose addresses differ (by 1) only in DIM_i. We use NET_i to denote dimension i crosssection of the network, which consists of all nodes and DIM_i channels. Also, NET_{i+} is a subnetwork of the mesh consisting of all nodes and unidirectional DIM_i channels that start from a lower numbered node and end at a higher numbered node in dimension i. In addition, $\text{NET}_{i/j\,even}$ indicates the subnetwork consisting of all the nodes and DIM_i links among nodes with jth component of their addresses even. Similarly, NET_{i-} and $\text{NET}_{i/j\,odd}$ subnetworks are defined. Finally, MODULE_i denotes the module responsible for switching messages traveling in DIM_i.

To illustrate our technique, we use a 3D mesh as a typical network. However, our results can be extended to multidimensional tori and meshes in a straight forward manner. As per dimension order routing, each message completes the required hops in dimension DIM_i before taking any hops in DIM_j, $0 \leq i < j < n$, where n is the number of dimensions in the network. The router given in Figure 1(a), without the dashed lines, is a typcial 3D PDR.

The Cray T3D implements such a partitioned dimension-order router in each node using three identical router chips. A pair of 24-bit unidirectional lines (16-bit data + 8-bit control) interconnect appropriate dimension chips in adjacent nodes in the Cray T3D router. In addition, each chip has an input from the network interface (for injection of messages) or from previous dimension router chip and an output to the next dimension router chip or to the network interface (for delivery of messages). So, each router chip has three incoming 24-bit channels and three outgoing 24-bit channels. Not counting pins for power supply, ground, etc., each router chip requires at least 144 pins for data and control of virtual channels.

For a crossbar based router implementation, one chip is used per router. Such a chip requires at least 336 pins—2*6*24 = 288 pins for internode-connections and $2 * 24 = 48$ pins for injection and consumption channels. Thus PDR implementations have lower pin requirements per router chip. For the same number of pins per chip, PDRs can provide wider channels. The main disadvantages of PDRs are increased chip count and additional bottlenecks in the form of intermodule links used by messages that need to change their dimensions.

Since channels are the resources for which messages compete in wormhole routing, cyclic dependencies that arise in adaptive routing are avoided by simulating two or more virtual channels on each physical channel [8].

3 Modifications to Partitioned Dimension-Order Routers

To provide adaptive routing, the router design has to be modified. Two types of modifications to the router are needed: modifications to the router organization, and modifications to the routing logic. We discuss modifications to the router organization below, and modifications to the routing logic in the next section.

Since channels in a dimension other than the current dimension of travel are used for adaptive routing, there should be a mechanism for a message to travel from a higher dimension routing module to a lower dimension module. A simplest way to achieve this is to make the existing channel between MODULE_i and MODULE_{i+1}, $0 \leq i \leq n-2$, bidirectional. Even more flexibility can be provided by adding a new bidirectional channel between MODULE_{n-1} and MODULE_0. Both possibilities are shown for a 3D mesh router in Figure 1.

If the suggested connections are made, the chip pin count increases by the number of data and control lines required per channel. In the case of a Cray T3D type router, it is about 24 extra pins for each added connection. Additional buffers are needed to store flits at each new incoming channel. Alternatively, the pin count can be reduced by using multiplexers, as shown in Figure 2 for a 3D router. At the input of MODULE_i, a MUX is used to multiplex between the outputs from $\text{MODULE}_{i-1(\bmod n)}$ and $\text{MODULE}_{i-2(\bmod n)}$. At the input of MODULE_0, however, the multiplexer also has to include the injection channel as the input.

When multiplexers are used, a message sees an extra multiplexer delay at its injection into the network and whenever it changes its dimension of travel. However, we expect this delay to be not too significant compared to the queueing delay at moderate to high traffic loads. This additional delay may not affect the network throughput, since

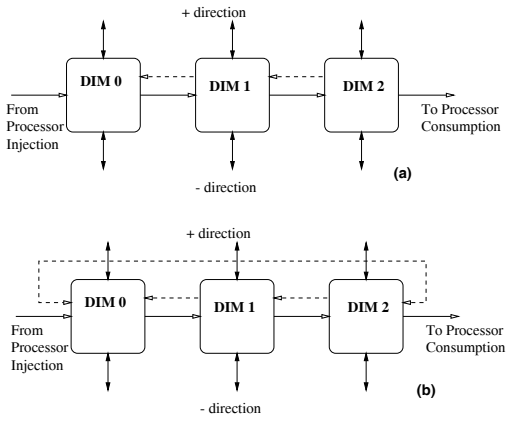

Figure 1: Modifications to the 3D PDR to support adaptive routing. The added connections are given by dashed lines. Part (a) shows making existing intermodule connections bidirectional, and part (b) adding a new connection between the highest and lowest numbered modules.

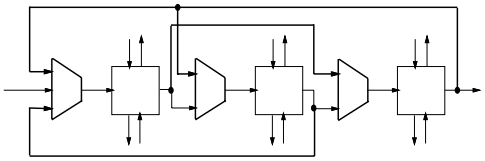

Figure 2: Example of implementing the required intranode connections using multiplexers.

router chips are designed to operate in a pipelined fashion [3, 4].

4 Adaptive Routing

We describe our technique for nD meshes with PDRs. First we define a few terms to facilitate the discussion. A profitable hop is one that takes a message closer to its destination. The *e*-cube hop of a message is the hop specified by the *e*-cube or dimension-order routing algorithm. It is the profitable hop in the lowest dimension in which the current node's address does not match that of the message destination. Our techniques can be classified as partially adaptive routing methods, since in general, they use a subset of available shortest paths to route messages. We consider only profitable internode hops, though our routing technique can handle nonprofitable hops taken by a message in the adaptive subnetwork.

Adaptive Routing Algorithm, Version 1. The first version of our adaptive algorithm is for PDRs that do not have channels between MODULE_{n-1} and MODULE_0.

First, let us describe the base dimension-order routing. In dimension-order routing, a message is always routed in the lowest dimension with a profitable hop. We use c_0 class of virtual channels for internode and intranode (on links between modules within a node) hops.

We enhance the basic dimension-order routing by letting a message have adaptivity while routing in DIM_0 through DIM_{n-2}. Messages route nonadaptively in the highest dimension, DIM_{n-1}. When a message is routed adaptively, it is routed so in two dimensions: the dimension of its *e*-cube hop, and the next dimension. More specifically a message routing in DIM_i, $0 \leq i \leq n-2$, uses the combination of two subnetworks. If the message's *e*-cube hop is to a higher numbered node in DIM_i, then it uses NET_{i+} and $\text{NET}_{i+1/ieven}$ subnetworks; otherwise, it uses NET_{i-} and $\text{NET}_{i+1/iodd}$ subnetworks. In adaptive routing, the sets of channels used by a message are internode channels in its lower dimension (for dimension-order routing), intranode channels to move from one dimension routing module to another (for changing dimension of travel), and internode channels in the higher dimension (for adaptive routing). All non-*e*-cube hops are taken using the c_1 class of virtual channels.

Table 1 indicates the subnetwork and the virtual channels used by various messages. An example of the subnetworks used by DIM_0 messages in a 2D mesh is given in Figure 3.

As an example, consider a message that needs to be routed from node (0,0) to node (1,3), middle node in the right-most column, in Figure 3. The available paths are as follows. The arrows indicate internode hops. The first path is due to *e*-cube routing.

$$(0,0) \rightarrow (0,1) \rightarrow (0,2) \rightarrow \textbf{(0,3)} \rightarrow (1,3)$$
$$\textbf{(0,0)} \stackrel{1}{\rightarrow} \textbf{(1,0)} \rightarrow (1,1) \rightarrow (1,2) \rightarrow (1,3)$$
$$(0,0) \rightarrow (0,1) \rightarrow \textbf{(0,2)} \stackrel{1}{\rightarrow} \textbf{(1,2)} \rightarrow (1,3)$$

All internode hops given above are on c_0 virtual channels unless otherwise indicated by a 1 on the corresponding arrow, in which case c_1 channels are used. Intranode transitions take place in the nodes indicated in bold. The internode hop is on c_0 if the intranode hops are due to *e*-cube routing—a message finished its hops in a given dimension and is going to the next dimension of routing. The intranode hops due to adaptive routing can be identified by looking at the channel used for internode hops prior to and after the intranode hop. Figure 4 shows the use of intranode channels for the second path given above. A fully-adaptive routing provides four shortest paths for routing this message.

Adaptive Routing Algorithm, Version 2. By noting that the c_1 channels in DIM_0 are unused, the algorithm can be improvised to provide even more adaptivity if MODULE_{n-1} and MODULE_0 of a router are connected. This is the second version of our algorithm. A message may forego routing completely in DIM_0 at any time and

Table 1: Adaptivity and channel allocation for messages in the first version of the routing algorithm

Msg. Type	Subnetworks	Internode Chs.	Intranode Chs.
DIM_{0+}	NET_{0+}, $\text{NET}_{1/0 even}$	c0 in DIM_0, c1 in DIM_1	c0 for *e*-cube, c1 for adaptive
DIM_{0-}	NET_{0-}, $\text{NET}_{1/0 odd}$	c0 in DIM_0, c1 in DIM_1	c0 for *e*-cube, c1 for adaptive
⋮			
DIM_{i+}	NET_{i+}, $\text{NET}_{i+1/i even}$	c0 in DIM_i, c1 in DIM_{i+1}	c0 for *e*-cube, c1 for adaptive
⋮	⋮	⋮	⋮
$\text{DIM}_{(n-1)+}$	$\text{NET}_{(n-1)+}$	c0	N/A
$\text{DIM}_{(n-1)-}$	$\text{NET}_{(n-1)-}$	c0	N/A

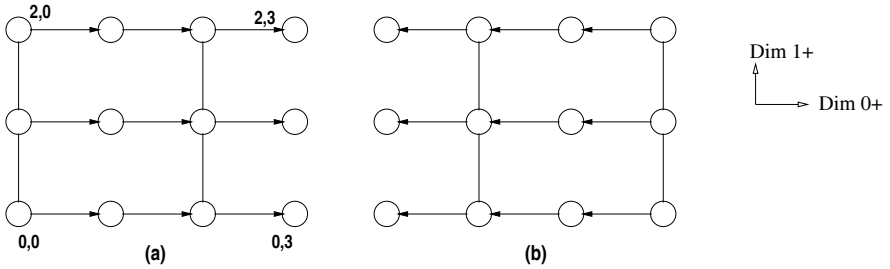

Figure 3: Subnetworks used by DIM_{0+} (part a) and DIM_{0-} (part b) messages in a 3×4 mesh network. c_0 virtual channelsare used on NET_{0+} and c_1 virtual channelson $\text{NET}_{1/0\,even}$.

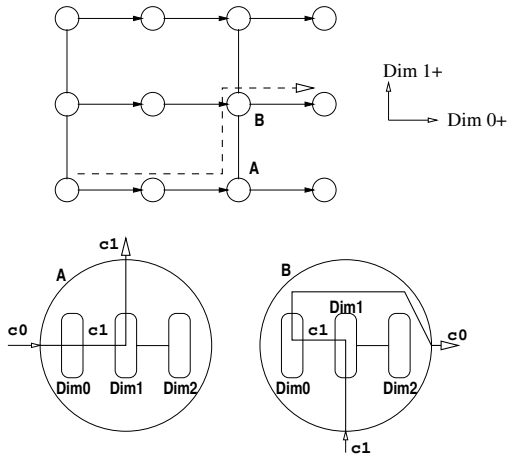

Figure 4: Use of intranode channels in routing a message from node (0,0) to node (1,3). The intranode paths are indicated in bold.

revisit it after completing the routing in DIM_1 through DIM_{n-2} adaptively as in the original version. To apply the above discussion, it is helpful to assume that such messages need to take hops in DIM_n rather than DIM_0 after finishing routing in DIM_{n-1}. After completing its routing in DIM_1 through DIM_{n-2} adaptively, the message is routed nonadaptively in NET_{n-1} using c_0 channels and in NET_0 (a revisit of DIM_0 subnetwork) using c_1 channels.

Adaptive Routing Algorithm, Version 3. Even more adaptivity can be provided by considering DIM_{n-1} and DIM_0 planes. However, the routing logic becomes slightly more complicated. Now a $\text{DIM}_{(n-1)+}$ message that needs to revisit NET_0 routes adaptively in $\text{NET}_{(n-1)+}$ and $\text{NET}_{0/(n-1)even}$ using c_0 for DIM_{n-1} hops and c_1 for intranode and DIM_0 hops. If a message that skipped DIM_0 has $\text{DIM}_{(n-1)+}$ hops to take when it becomes a DIM_{n-1} message, then it should be routed such that it can complete all of its DIM_0 hops on $\text{NET}_{0/(n-1)even}$. Similarly, a $\text{DIM}_{(n-1)-}$ messages travels adaptively in $\text{NET}_{(n-1)-}$ and $\text{NET}_{0/(n-1)odd}$ using c_0 and c_1 channels adaptively. If a message that skipped DIM_0 does not need to travel in DIM_{n-1} after completing its routing in DIM_{n-2}, then it can directly go through intranode channels from MODULE_{n-2} to MODULE_{n-1} and from MODULE_{n-1} to MODULE_0 and route to its destination using c_1 channels in NET_0.

Comparisons with other adaptive routing algorithms. There are no previous results on adaptive routing methods that work with multimodule routers. So, we compare the routing algorithms with some of the previously known crossbar based adaptive algorithms. Version 1 of our algorithm is similar to the planar-adaptive routing (PAR) algorithm proposed by Chien and Kim [15]. Our algorithm uses only 2 virtual channels per physical channel instead of 3 by PAR, which also requires a crossbar for router implementation. Versions 2 and 3 of the proposed routing al-

gorithms are more versatile than the PAR algorithm, since with these algorithms messages can skip routing in DIM_0 when the network is congested in DIM_0. In fact, Chien and Kim [15] report that PAR does not work very well for 3 or higher dimension meshes with nonuniform traffic such as bit reversal.

As an example take the bit reversal traffic in an $8 \times 8 \times 8$ mesh. Here, node (0,0,1) wants to communicate with node (4,0,0), node (0,0,2) with node (2,0,0), node (0,0,3) with node (6,0,0), and so on. (The destination of a node in bit reversal communication is obtained by taking the node address in binary, in this case a 9-bit address, and reversing the bits. For example node (0,0,1) has the bit address 000 000 001, and its destination will have the bit address 100 000 000.) With PAR, also with version 1 of our algorithm, all these messages have to go to node (0,0,0) before traveling in DIM_2. Since they do not need to travel in DIM_1, they have no adaptivity, unless the shortest path constraint is relaxed. With version 2 of our algorithm, however, such messages have the option of directly going to DIM_2, completing routing in DIM_2 and then coming back to DIM_0 and complete the routing. With version 3, such messages travel adaptively in the DIM_3-DIM_0 plane. This sort of adaptivity leads to better performance for nonuniform traffic.

Another point worth noting is that versions 2 and 3 of the proposed algorithm use all available virtual channels for routing even for messages that do not have adaptivity—those that need to travel in one dimension only. As an example consider routing of a message between nodes in a row of a 2D mesh, see Figure 5. This message travels only on DIM_0 links and does not have adaptivity. If it is routed using DIM_0 routing, then it uses c_0 channels. If it skips DIM_0 to comeback to it later, then it revisits DIM_0 (immediately, since there are no DIM_1 hops to take) and completes its routing using c_1 channels. This illustrates the versatility of the proposed routing technique in providing additional paths and additional virtual channels for adaptive routing. Furthermore, a DIM_i, $i > 0$, message can always use c_1 virtual channels on DIM_i links without creating deadlocks. In contrast, the PAR utilizes only one of the three classes of virtual channels used in DIM_0. The previously known adaptive wormhole routing schemes with such flexible channel allocation are crossbar based adaptive schemes [10, 17].

4.1 Proof of deadlock-free routing

The above routing algorithm works for routers implemented using a full crossbar, which can connect any input channel of a node to any output channel. The previous results on adaptive routing assume that the router in a node is implemented using a crossbar, which provides full switching capability among multiple dimensions. In a multimodule dimension-order router, changing dimensions of travel by messages is complicated, since intermodule channels will be shared among different types of messages in adaptive routing. We prove below that these additional dependencies are carefully controlled such that deadlocks are avoided.

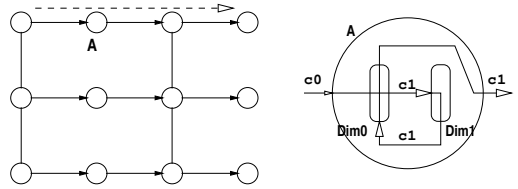

Figure 5: Using adaptive channels for routing messages with no adaptivity. The message finding that c_0 channel busy at node A, can skip DIM_0 and go to DIM_1 and back to DIM_0, since there are no DIM_1 hops to take, all in one node. After this it can use c_1 channels to reach its destination. The unneeded intranode transitions can easily be eliminated for such messages.

Proof sketch to show that version 1 is deadlock free: There are $2n$ types of messages: DIM_{i+} and DIM_{i-}, $i = 0, \ldots, n-1$ and n is the number of dimensions of the mesh. We simply show that a each message type uses a specific acyclic virtual network made up of virtual channels not used by other message types, and that these acyclic virtual networks are used by messages according to some partial order. Our key argument for first part is to show that when a message of type, say, DIM_{i+} uses the same physical channels as other message types, then it uses different virtual channels, and it never has common physical channels with other message types that use the same class of virtual channels. The argument for the second part follows directly from the dimension-order routing.

It is easy to show that the proposed algorithm is free of livelocks. We use only shortest paths, and a message can choose and hold for a virtual channel indefinitely without creating deadlocks. When a fair queueing policy such as FIFO is used, a message gets its channel in finite time.

The proof for version 1 of the algorithm can be extended to show that versions 2 and 3 of the algorithm are also deadlock free. Versions 2 and 3 are different from version 1 in that they use c_1 channels in DIM_0, and c_1 channels on links between DIM_{n-1} and DIM_0 modules while (or after, depending on the version) routing in DIM_{n-1}.

5 Concluding Remarks

We have presented a technique to enhance the dimension-order routers for adaptive routing in multicomputer networks. Particular attention has been paid to the applicability of the proposed techniques for current multicomput-

ers which use partitioned dimension-order routers (PDRs). The proposed technique guarantees deadlock free routing with only a modest increase in the number of virtual channels used. The changes to the router increase the pins on each router chip by a small constant. Alternatively, multiplexers may be used to reduce the chip pin count. This increases the chip count and transit delays for messages cutting-through intermediate nodes are higher. Neither is a severe disadvantage, however. Multiplexers are simple and inexpensive. With extensive pipelining for the router chip, increased transit delays do not affect the network throughput.

Since PDRs do not have centralized crossbars, the previously known adaptive routing techniques, which implicitly assume crossbars for switching, cannot be implemented without redesigning the existing routers. Indeed, the crossbar-based adaptive router used in Cray T3E is a complete redesign of the PDR used in the earlier generation machine, Cray T3D. Studies have shown that large crossbars adversely impact the speed of a router [5]. Also for the same technology, multichip implementations provide wider internode channels. Therefore, the proposed technique is attractive for implementing faster adaptive routers in future.

Another advantage is that the proposed technique uses only two virtual channels to provide adaptivity. Also, it allows message skip DIM_0 initially and revisit it later. This type of advantage is most beneficial in handling nonuniform traffic higher dimension networks, more so for 3D networks.

In our earlier work, we have shown that PDRs can be enhanced for fault-tolerant routing in the presence of block faults [4]. The technique presented here can be combined with our earlier work to provide adaptive, fault-tolerant routing in multicomputers with partitioned dimension-order router routers. We are currently evaluating the performance of the algorithm with the dimension-order and other adaptive routing algorithms.

References

[1] K. V. Anajan and T. M. Pinkston, "An efficient, fully adaptive deadlock recovery scheme: DISHA," in *Proc. 22nd Ann. Int. Symp. on Comput. Arch.*, pp. 201–210, June 1995.

[2] K. Bolding and L. Snyder, "Mesh and torus chaotic routing," in *Proc. of Advanced Research in VLSI and Parallel Systems*, 1992.

[3] R. V. Boppana and S. Chalasani, "A framework for designing deadlock-free wormhole routing algorithms," *IEEE Trans. on Parallel and Distributed Systems*, vol. 7, pp. 169–183, Feb. 1996.

[4] S. Chalasani and R. V. Boppana, "Fault-tolerance with multimodule routers," in *Proc. 2nd Int. Symp. on High-Performance Comput. Arch.*, pp. 201–210, Feb. 1996.

[5] A. A. Chien, "A cost and speed model for k-ary n-cube wormhole routers." Presented at Hot Interconnects 1993, Mar. 1993.

[6] Cray Research Inc., *Cray T3D System Architecture Overview*, Sept. 1993.

[7] W. J. Dally, "Network and processor architecture for message-driven computers," in *VLSI and Parallel Computation* (R. Suaya and G. Birtwislte, eds.), ch. 3, pp. 140–222, San Mateo, California: Morgan-Kaufman Publishers, Inc., 1990.

[8] W. J. Dally and C. L. Seitz, "Deadlock-free message routing in multiprocessor interconnection networks," *IEEE Trans. on Computers*, vol. C-36, no. 5, pp. 547–553, 1987.

[9] W. J. Dally and P. Song, "Design of a self-timed VLSI multicomputer communication controller," in *Int'l Conf. on Computer Design*, pp. 230–234, 1987.

[10] J. Duato, "A new theory of deadlock-free adaptive routing in wormhole networks," *IEEE Trans. on Parallel and Distributed Systems*, vol. 4, pp. 1320–1331, Dec. 1993.

[11] M. Fulgham and L. Snyder, "Integrated multi-class routing," in *Proc. of Parallel Routing and Communication Workshop*, pp. 17–28, June 1997.

[12] P. T. Gaughan and S. Yalamanchili, "A family of fault-tolerant routing protocols for direct multiprocessor networks," *IEEE Trans. on Parallel and Distributed Systems*, vol. 6, pp. 482–497, May 1995.

[13] C. J. Glass and L. M. Ni, "The turn model for adaptive routing," in *Proc. 19th Ann. Int. Symp. on Comput. Arch.*, pp. 278–287, 1992.

[14] Intel Corporation, *Paragon XP/S Product Overview*, 1991.

[15] J. H. Kim and A. A. Chien, "An evaluation of planar-adaptive routing (PAR)," in *Proc. Fourth IEEE Symp. on Par. and Distr. Processing*, 1992.

[16] J. Y. Ngai and C. L. Seitz, "A framework for adaptive routing in multicomputer networks," in *Proc. First Symp. on Parallel Algorithms and Architectures*, pp. 1–9, 1989.

[17] L. Schwiebert and D. N. Jayasimha, "Optimally fully adaptive routing for meshes," in *Proc. Supercomputing '93*, pp. 782–791, 1993.

[18] C. L. Seitz, "Concurrent architectures," in *VLSI and Parallel Computation* (R. Suaya and G. Birtwistle, eds.), ch. 1, pp. 1–84, San Mateo, California: Morgan-Kaufman Publishers, Inc., 1990.