Fault-Tolerance with Multimodule Routers*

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

Rajendra V. Boppana
Computer Science Division
The Univ. of Texas at San Antonio
San Antonio, TX 78249-0664
boppana@runner.utsa.edu

Abstract. The current 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 fault-tolerance into such routers with simple changes to the router structure and logic. The previously known fault-tolerant routing methods assume centralized crossbar based routers and are not applicable to multiprocessors with PDRs. The proposed technique works for convex fault model, using only local knowledge of faults. Using the proposed techniques and as few as four virtual channels per physical channel, torus networks with PDRs can handle faults without compromising deadlock- and livelock-freedom. Simulations for 2-dimensional torus and mesh networks show that the resulting fault-tolerant PDRs have performances similar to those of the crossbar based routers.

1 Introduction

Many recent experimental and commercial multicomputers and multiprocessors [26, 22, 12] use direct-connected networks with grid topology. A (k, n)-torus network has an n-dimensional grid structure with k nodes (a node is a processor-memory-router element) in each dimension such that every node is connected to two other nodes in each dimension by direct communication links. Majority of these multicomputers use dimension-order or e-cube routing with *wormhole* (WH) switching [17]. 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 [18, 13, 26, 22, 12]. 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 bulding block for PDR implementations. Thus, for the same technology, a PDR implementation yields wider channels compared to the crossbar implementation.

While the e-cube is simple to implement and provides high throughput for uniform traffic, it cannot handle even simple node or link faults due to its nonadaptive routing. Adaptive, fault-tolerant cut-through routing algorithms has been the subject of extensive research in recent years [11, 19, 15, 21, 24, 1, 4, 7, 20, 2, 16, 6]. These results implicitly or explicitly assume routers with centralized crossbars. Therefore, such techniques are not suitable for multiprocessors with PDRs. Several other results (see, for example, [23, 25] and the references therein) exploit the rich interconnection structure of hypercubes and are not suitable for high-radix, low-dimensional tori.

In this paper, we propose a technique to incorportate fault-tolerance into networks with PDRs implemented using multiple chips. Our approach is to provide interprocessor communication among the fault-free nodes, rather than to recreate or simulate the original topology of a faulty network. We have previously proposed similar techniques for fault-tolerant routing in multicomputer networks with crossbar based routers [4, 7, 9]. The main contribution of this work is to show that partitioned dimension-order routers also can be enhanced for fault-tolerant routing *without using crossbars*. We show that with a small increase in the resources and simple changes to the router organization and routing logic, multiple block faults can be handled without compromising livelock and deadlock freedom.

Our technique works with local knowledge of faults (each fault-free node knows only the status of its and its neighbors' links), handles multiple faults, and guarantees livelock- and deadlock-free routing of all messages. Our fault model allows multiple node and link faults as long as the fault regions are convex in shape—rectangular in 2D tori, cubic in 3D tori, etc. The convex fault model is simple enough to provide modular routing, yet powerful enough to model node and printed-circuit-board level faults. We apply our techniques to torus networks and show that, with as few as four virtual channels per physical channel, and some modifications to the interconnection of dimension-modules, PDRs can be used for fault-tolerant routing.

Section 2 gives an overview of dimension-order routers. Section 3 describes the fault-model. Section 4 describes the changes to the router required to handle faults. Section 5 gives the required modifications to the routing logic. Section 6 presents simulation results on the performance of mesh and torus networks under faults. Section 7 concludes this paper.

*Chalasani's research has been supported in part by a grant from the Graduate School of UW-Madison and the NSF grant CCR-9308966. Boppana's research has been partially supported by NSF Grant CCR-9208784.

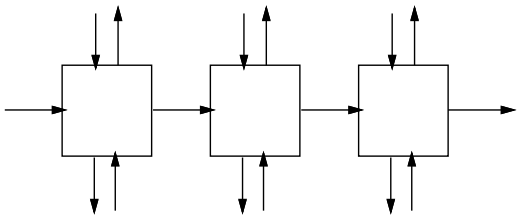

Figure 1: Organization of a partitioned dimension-order router for 3D torus and mesh networks.

2 Partitioned Dimension-Order Routers

A (k, n)-torus 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 two other nodes in each dimension. The neighbors of the node $x = (x_{n-1}, \ldots, x_0)$ in dimension i are $(x_{n-1}, \ldots, x_{i+1}, x_i \pm 1, x_{i-1}, \ldots, x_0)$, where addition and subtraction are performed modulo k. Each link provides full-duplex communication using two unidirectional physical channels. 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, $0 \leq i < n$. 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. To illustrate our technique, we use a 3D torus 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. A three-dimensional PDR is shown in Figure 1.

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 interchip links used by messages that need to change their dimensions.

Since channels are the resources for which messages compete in wormhole routing, cyclic dependencies and deadlocks in a torus are avoided by simulating two virtual channels on each physical channel [17]. (The Cray T3D actually simulates four virtual channels to handle two distinct classes of messages with two virtual channels per class of messages.)

Another interesting feature of the Cray T3D router is that its routing logic is programmable. Routing tables, which contain routes for each destination, can be loaded into the network interface by software. In fact, this ability to alter routing tables together with the wraparound links in the torus topology can be used to provide a rudimentary fault-tolerant routing to handle one fault, for example, in a row [12].

3 Faults in Networks

We consider permanent failures of nodes and links that do not disconnect the network. We model multiple simultaneous faults, which could be connected or disjoint. We assume that the mean time to repair faults is quite large and that the existing fault-free nodes shoud be used productively in the mean time. We assume that all faults are nonmalicious faults; that is, a failed component simply ceases to work. Therefore, messages are generated by and for processors with non-faulty nodes.

Detection and isolation of faults is done easily because fault information is kept locally. Each node is required to detect faults on its incoming physical channels and its router chips. A node can detect its faulty components, if any, using a suitable self-test sequence periodically. When node detects a fault within its processor, router or other component, it simply stops sending signals on all of its outgoing channels. A healthy node sends status signals to its neighbors on its outgoing physical channels and monitors status signals sent by its neighbors on its incoming physical channels. Missing or incorrect sequences of signals indicate malfunction of the link or the node sending them. The nodes at the end of a malfunctioning link stop using that link. When a node detects a faulty link, it reports this fault to its neighbors reachable via the fault-free links. We develop fault-tolerant algorithms, for which it is sufficient if each non-faulty node knows the status of the links incident on it and its neighbors. Another approach for fault detection is given in [2].

We model *block* faults, as per which the set of faulty nodes can be partitioned into disjoint subsets such that each subset forms an n-D cube in an nD torus. Examples of block faults include a 3D cube in a 3D torus, a rectangle in a 2D torus, and so on. Figure 2 shows example block faults in 2D and 3D torus networks.

The block model is simple, yet models two common fault scenarios: single faults, and multiple dependent faults, which can occur, for example, if a board (which has a block of nodes) loses its power-supply or is removed for repair. In addition, the routing techniques developed here can be used to provide a secure computation environment within a multiprogramming mode, where several users share the processors and the network. To provide a secure computing environment, a block of nodes may be allocated such that the nodes and the links among them are not used by other computations or messages resulting from them. By treating such a block of processors and links as faulty in routing the other messages, the proposed techniques can

be applied for on-the-fly allocation and release of blocks of nodes for special-purpose computations.

A simple characterization of block faults is that a fault-free node may have at most one faulty neighbor. Using this rule, any fault pattern can be *blocked*: if a node has more than one neighbor faulty, it marks itself faulty. Thus a fault is blocked within a finite number of steps, bounded by the diameter of the network.

Fault rings. Consider a 2D torus with a block fault. This block fault is enclosed by a ring of nonfaulty nodes and links; the smallest such ring is known as the fault-ring for the block fault. In a 3D torus, the block fault is a 3D cube such that any 2D cross-section of the fault is a 2D block fault. Therefore, for a block fault in a 3D torus, several fault rings are formed, one for each possible 2D cross section of the fault. Examples of fault rings for 2D and 3D tori are shown in Figure 2.

A fault-ring corresponding to a 2D fault-block can be formed in a distributed manner using a two-step process. In the first step, each node that detected a faulty neighbor sends this information to its neighbors in other dimensions. In the second step, based on the set of messages received in the first step, each node that is to be on the f-ring determines its neighbors on the f-ring [4]. In 3D or higher dimensional tori, a node may be on multiple fault rings of a fault block. But, a link can be on at most one fault ring.

The concept of fault rings extends to faults in higher dimensional tori. For each nD block fault, several fault rings, one for each 2D cross section of the fault, are formed and used in routing messages around faults.

An f-ring represents a two-lane path to a message that needs to go through the block fault contained by the f-ring. Thus an f-ring simulates 4 paths to route messages in two dimensions. Therefore, some physical channels in an f-ring may need to handle traffic many times the traffic of a channel not on any f-ring. Routing around one or more fault-rings creates additional possibilities for deadlocks to occur. Hence, wormhole routing algorithms must be designed to avoid these deadlocks also.

Two f-rings are said to overlap if they share one or more links. In this paper we consider block faults that give rise to nonoverlapping f-rings. The techniques used for nonoverlapping f-rings can be extended to handle overlapping f-rings [8].

4 Modifications to Partitioned Dimension-Order Routers

To tolerate faults under the block-fault model, 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 router logic in the next section.

If a message is blocked by a fault, then alternate dimensions must be used to route the message around the fault. Therefore, a message may need to travel in dimension i after traveling in dimension j, where $j > i$. To handle this, we provide new connections from the output of router chip i to the inputs of router chips $(i+1) \bmod n$ and $(i+2) \bmod n$, for $0 \leq i < n$, where n is the number of dimensions. These additional connections require multiplexers. At the input of router chip i, a multiplexer is used to multiplex between the outputs from router chips $(i-1) \bmod n$ and $(i-2) \bmod n$. If a node is not on an f-ring, this multiplexer is permanently set to route the output from $(i-1) \bmod n$ router chip to the input of router chip i. At the input of router chip 0, however, the multiplexer also has to include the injection channel as the input. These changes are indicated in Figure 3, where the required changes are shown using thick lines.

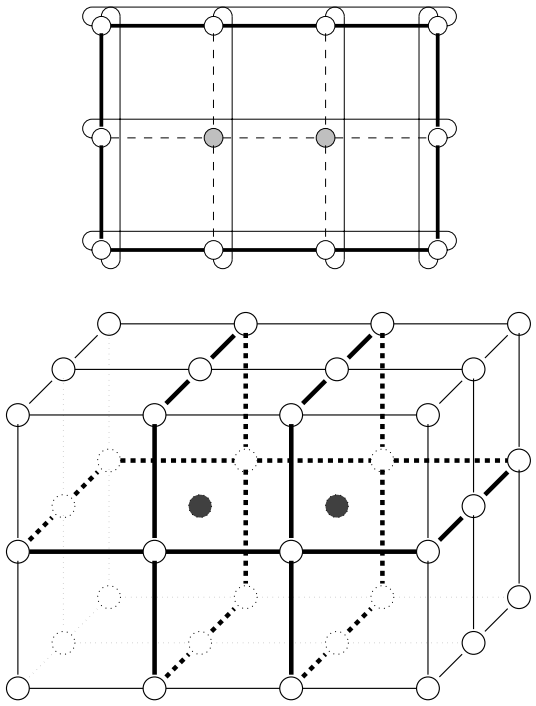

Figure 2: Examples of block faults in 2D and 3D torus networks. Shaded nodes and the links incident on them are faulty. Dashed lines in the 2D torus indicate faulty links. For the 3D torus, wraparound links and faulty links are not shown for clarity, and internal healthy links are indicated by dotted lines. Thick lines indicate the corresponding fault rings.

Cost of modifications. We consider cost in terms of the increase in the signal pins to router chips and additional hardware required by these modifications. The extra hardware required is one multiplexer per dimension and additional routing of wires among the chips. Also, 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 router chips are designed to operate in a pipelined fashion. This issue is further discussed in Section 6.

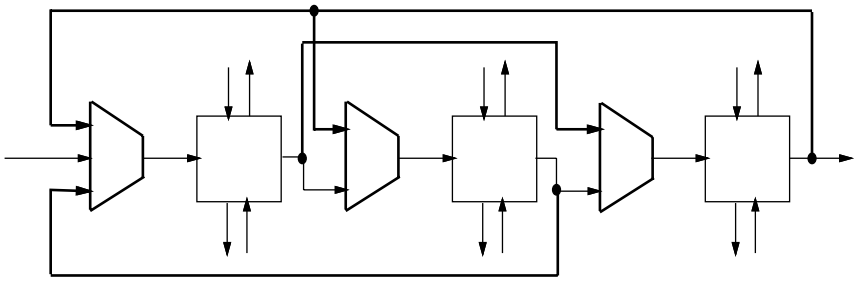

Figure 3: Modifications to the 3D PDR to support fault-tolerant routing.

5 Fault-Tolerant Routing

We describe our technique for the PDRs used in a 3D torus network. Extension of these techniques to higher dimensions is discussed later. When there are no faults, the original dimension-order routing is used. Even when there are faults in the network, each message is routed using the dimension-order routing as much as possible. When a message arrives at a node, the next hop for that message is specified by this algorithm. If that hop is on a faulty link, then the message is blocked by the fault. The routing logic is enhanced to handle such situations so that the message is routed around faults. Once the message is routed around faults, the dimension-order routing is used to route the message until it reaches its destination or is blocked again. It is noteworthy that the modified routing logic is used only when a message is blocked by a fault. A message is a misrouted message if it is being routed around an f-ring; otherwise it is a normal message.

Modifications to the routing logic. A normal message is routed by the original dimension-order routing. A normal message may be blocked by a fault while traveling in any of the three dimensions: DIM_0, DIM_1, or DIM_2. In that case it becomes a misrouted message and routed around an appropriate f-ring. We specify different ways for different dimension messages to get around a fault. A message blocked from taking its DIM_0 (respectively, DIM_1) hop travels on two sides of the corresponding f-ring formed from a DIM_0-DIM_1 (respectively, DIM_1-DIM_2) cross-section of the fault. A message blocked from taking its DIM_2 hop travels on *three* sides of an f-ring formed from DIM_2-DIM_0 cross-section of the fault. Figure 4 illustrates the routing of messages on f-rings. Messages blocked in DIM_0 and DIM_1 may choose one of two possible orientations to get around faults, but messages blocked in DIM_2 can use only one particular orientation as indicated in Figure 4. If a misrouted DIM_0 or DIM_1 message reaches a corner node of the f-ring, then it takes the turn and continues to travel on it as a normal message, since the original dimension-order routing can be used. On the other hand, a misrouted DIM_2 message becomes normal and original dimension-order routing is used to route it again only after it reaches the other side of the fault block with only DIM_2 hops left for it to reach its destination.

In a fault-free network with dimension-order routing, each physical channel is used by a specific type of message. With faults, however, blocked messages are misrouted, and some physical channels around f-rings are used by multiple types of messages. This creates cyclic dependencies.

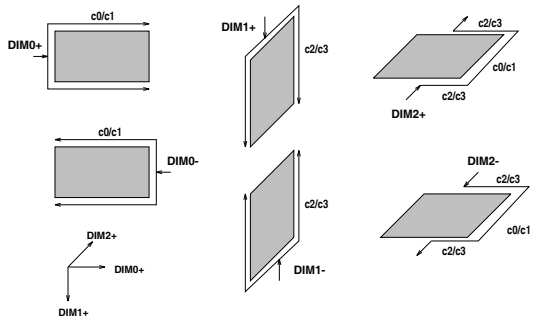

Figure 4: Routing of six different message types around a fault. The shaded area represents a faulty block, and directed lines indicate the paths of messages on the f-ring around the fault. The type of virtual channels used are also indicated.

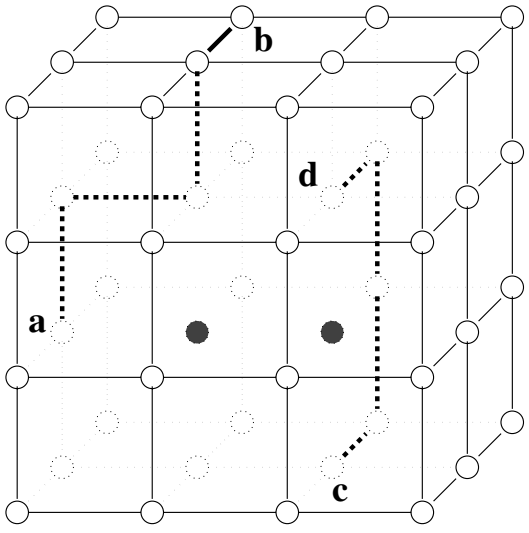

Figure 5: Example of the proposed fault-tolerant routing. There is one fault block indicated by shaded nodes. The links incident on the faulty nodes and wraparound links are not shown for clarity. Routes for two messages, one from **a** to **b** and another from **c** to **d**, are shown by thick dashed lines.

To break these new dependencies and to keep the routing deadlock free, we use two new classes in addition to the original two classes required for deadlock free routing in a fault-free torus. These additional virtual channels require additional flit-buffers in each router chip. Let the four classes of virtual channels be c_0, c_1, c_2, c_3. On each physical channel (internode as well as intranode—between router chips), a virtual channel of each class is simulated. Depending on the direction and dimension a message is traveling before being blocked by a fault, it can be one of 6 possible types: DIM_{i+} and DIM_{i-}, $i = 0, 1, 2$. The channel allocation is such that any new dependencies among the six types of messages caused by sharing of the physical channels on f-rings are broken. DIM_0 messages use virtual channels of class c_0 before taking a hop on a DIM_0 wraparound channel and c_1 virtual channels there after. A DIM_0 message that has completed hops in DIM_0 but not reached its destination will turn into a DIM_1 message and continues routing as a DIM_1 message. Similarly, a DIM_1 message uses c_2 or c_3 virtual channels, and turns into a DIM_2 message after completing hops in DIM_1. A DIM_2 message uses c_0 or c_1 while traveling in DIM_2 and c_2 or c_3 while traveling in DIM_0 (because of misrouting). These rules for fault-tolerant routing of messages are summarized in Table 1.

A message that has completed hops in a dimension goes to the next dimension router chip using a virtual channel on the interchip link to that chip. The virtual channel class used is the same as the virtual channel class used for the hop it just completed. If it does not need to travel in the next dimension, then it can use any virtual channel that can be used by a message of that dimension on the interchip link to the following dimension chip. For example, if a DIM_0 message has completed its hops, then it goes to DIM_1 chip using c_0 or c_1 channel on the interchip link. If it does not need to take any hops in DIM_1, then it uses c_2 or c_3 on the link from DIM_1 chip to DIM_2 chip.

An example of the proposed fault-tolerant routing method is shown in Figure 5. The proof of deadlock free routing is given next.

5.1 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. As mentioned earlier, the previous works on fault-tolerant wormhole routing algorithms implicitly assumed that the router in a node is implemented using a crossbar, which provides full switching capability among multiple dimensions. In a multichip dimension-order router, changing dimensions of travel by messages is complicated, since interchip channels are shared among different types of messages. We prove below that these additional dependencies resulting from sharing interchip links do not cause deadlocks.

Lemma 1 *The proposed fault tolerant routing algorithm is deadlock-free.*

Proof. There are six types of messages: DIM_{i+} and DIM_{i-}, $i = 0, 1, 2$. A particular set of virtual channels are used for each message type. So, messages of a type travel in a particular virtual network formed by all the nodes and the set of virtual channels used by them. During the routing, a message of one type may change into another type (for example, a DIM_{0+} message may change into DIM_{1-} after completing its hops in DIM_0). Our proof technique relies on showing that there is a partial order among all the virtual channels of the network and messages acquire them in an increasing order of ranks [17]. For this we need to show that (a) the sets of virtual channels used by the various types of messages are pairwise disjoint, (b) the virtual network for each type is acyclic, and (c) the virtual networks are used by messages as per some partial order. The multiplexers do not cause any new dependencies, since they simply connect inputs to outputs in a demand time-multiplexed manner.

The sets of virtual channels used by various types of messages are pairwise disjoint. First, we show that the set of virtual channels used by DIM_{i+} messages is disjoint from the set of virtual channels used by DIM_{i-} messages.

Consider DIM_{0+} and DIM_{0-} messages. If the sets of virtual channels used by DIM_{0+} and DIM_{0-} messages are not disjoint, then they share virtual channels on internode physical channels or interchip physical channels. The DIM_{0+} messages travel on positive direction DIM_0 physical channels and the DIM_1 physical channels among nodes on the left boundary columns of f-rings. Similarly, DIM_{0-} messages travel on negative direction DIM_0 physical channels and DIM_1 physical channels among nodes on the right boundary columns of f-rings. This is illustrated in Figure 6 for an example f-ring. Since f-rings do not overlap, the column channels are used by exactly one or none of the two classes of messages. So, DIM_{0+} and DIM_{0-} do not share virtual channels on internode physical channels, since the physical channels they use are disjoint.

Because of multimodule implementation, DIM_{0+} and DIM_{0-} messages may share some virtual channels on interchip links. With our fault-tolerant routing logic, this cannot occur, however. Referring to Figure 6 again, we note that only DIM_{0+} messages can reserve the c_0 channel from DIM_0 chip to DIM_1 chip in the middle nodes (node A in Figure 6) on the left column and the c_0 channel from DIM_1 chip to DIM_0 chip in the corner nodes on the left boundary column of the f-ring. A DIM_{0-} message does not use them. Similarly, the c_0 channels between DIM_0 and DIM_1 chips in the nodes on the right boundary column of an f-ring are used only by DIM_{0-} messages. For messages that took hops on wraparound links in DIM_0, the above argument repeats with c_1 as the virtual channel. Hence, there cannot be sharing of virtual channels among $\text{DIM}_{0\pm}$ messages.

From the above argument, it is clear that there are no dependencies among DIM_{0+} and DIM_{0-} classes of messages. A normal message that completes its hops in DIM_0 becomes a DIM_1 message and moves to the DIM_1 chip in the current node. The use of a virtual channel into DIM_1 chip is similar to the use of the injection channel form processor into DIM_0 chip and does not cause any dependencies.

This argument can be repeated to show that DIM_{1+} and DIM_{1-} messages use disjoint sets of physical channels.

The argument to show that DIM_{2+} and DIM_{2-} messages use disjoint sets of physical channels must take into consideration that a DIM_2 misrouted message travels on three sides of the f-ring. However, the principal argument is unchanged. As in the case of DIM_0 messages, the interchip channel between DIM_2 and DIM_0 router chips in a node on an f-ring can be used by only one of the DIM_{2+} and DIM_{2-}

Table 1: Planes and virtual channels used by various messages in a 3D torus with fault-tolerant PDRs.

Message type	Plane type	Virtual channel classes
DIM_{0+}, DIM_{0-}	DIM_0-DIM_1	c_0 before reserving a wraparound link in DIM_0, c_1 after reserving a wraparound link in DIM_0
DIM_{1+}, DIM_{1-}	DIM_1-DIM_2	c_2 before reserving a wraparound link in DIM_1, c_3 after reserving a wraparound link in DIM_1
DIM_{2+}, DIM_{2-}	DIM_2-DIM_1	c_0 (c_1) while traveling in DIM_2 before (after) reserving a wraparound link in DIM_2, and c_2 (c_3) while traveling in DIM_0 before (after) reserving a wraparound link in DIM_2

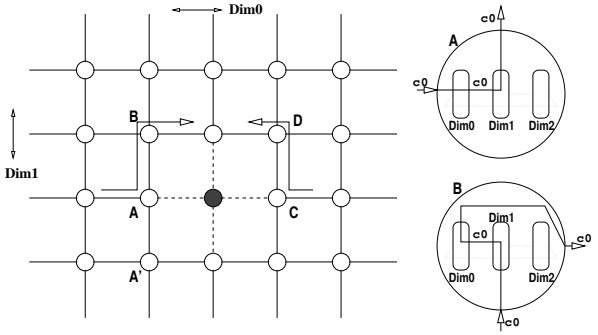

Figure 6: Illustration of virtual channels used on interchip links. Nodes A and B are the places where the misrouted DIM_{0+} message uses interchip links. Similarly a misrouted DIM_{0-} message uses interchip links in nodes C and D. The interchip channels used are given by labeled, directed thick lines.

types of messages. The interchip channel usage by a DIM_{2+} message on an f-ring is shown in Figure 7.

Now, consider messages of two different dimensions. By our virtual channel allocation given in Table 1, they use different classes of virtual channels on the physical channels they share. Only on disjoint physical channels they may use virtual channels of the same class.

The virtual network for each message type is acyclic. Consider DIM_{0+} messages. A normal DIM_{0+} message always progresses towards its destination. It uses c_0 channels before taking a hop on a DIM_0 wraparound link and c_1 thereafter. Since the original dimension-order routing is deadlock free, there are no cycles among these virtual channels. A misrouted DIM_{0+} message travels on the left column of an f-ring reserving channels c_0 (or c_1) as per a linear order. Let $c_j(x, y)$ denote the virtual channel of class c_j on the physical channel that starts from node x and ends in node y, one of its neighbors. If y is a DIM_{i+} neighbor of x ($y_i = (x_i + 1) \bmod k$), then $x \rightarrow y$ is the DIM_{i+} physical channel between them. Similarly DIM_{i-} channels are defined.

The linear order of c_0 channels on an f-ring is given by $c_0(x, y) \prec c_0(y, z)$ where x, y, z are nodes on the left column of the f-ring, and both $x \rightarrow y$ and $y \rightarrow z$ are DIM_{1+} or DIM_{1-} physical channels. For example, for the f-ring in Figure 6, $c_0(A', A) \prec c_0(A, B)$ and $c_0(B, A) \prec c_0(A, A')$. Now a partial order can be defined among the virtual channels used by DIM_{0+} messages using the following rules:

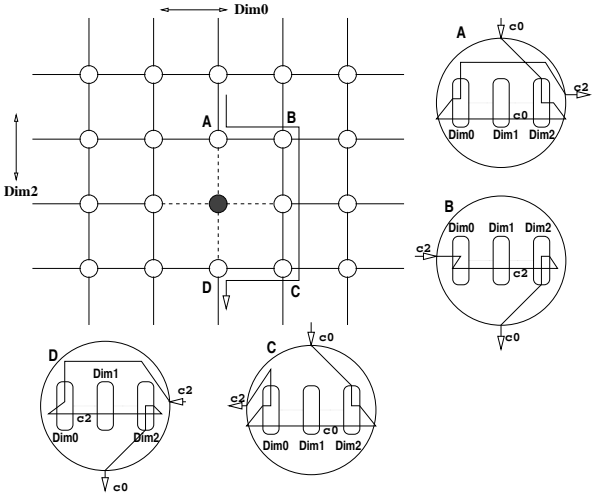

Figure 7: Illustration of virtual channels used on interchip links. Nodes A, B, C and D are the places where the misrouted DIM_{2+} message uses interchip links. The interchip channels used are given by labeled, directed thick lines.

- c_0 channels are ranked lower than c_1 channels.
- $c_0(w, x) \prec c_0(x, y)$ and $c_1(w, x) \prec c_1(x, y)$ if both $w \rightarrow x$ and $x \rightarrow y$ are DIM_{0+} or DIM_{0-}.
- $c_0(w, x) \prec c_0(x, y)$ and $c_1(w, x) \prec c_1(x, y)$ if (a) both $w \rightarrow x$ and $x \rightarrow y$ are DIM_{1+} or DIM_{1-} and (b) w, x, y are nodes on the left column of an f-ring.
- c_0 (respectively, c_1) channels on the left column of an f-ring in a DIM_0-DIM_1 plane are ranked higher than the c_0 (c_1) channels on the DIM_{0+} physical channels that end in the left column nodes of the f-ring, and are ranked lower than the c_0 (c_1) channels on the DIM_{0+} channels that start from the left column nodes.

Therefore, the virtual network of DIM_{0+} messages is acyclic. Similar rankings can be given to show that the virtual networks for other message types are acyclic.

The virtual networks are used according to a partial order. This directly follows from the dimension-order routing. A DIM_{i+}, $i = 0, 1, 2$ message never uses the virtual network of a DIM_{i-} message and the virtual networks of DIM_{j+} or DIM_{j-} messages, $j < i$. It can be easily verified that this defines a partial order. ∎

Table 2: Planes and virtual channels used by various messages for fault-tolerant routing in an nD torus.

Message type	Plane type	Virtual channel classes
M_0	$A_{0,1}$	c_0 and c_1
M_1	$A_{1,2}$	c_2 and c_3
M_2	$A_{2,3}$	c_0 and c_1
$\vdots$		
M_{n-1}, n even	$A_{n-1,0}$	c_2 and c_3
M_{n-1}, n odd	$A_{n-1,0}$	c_0 and c_1 in DIM_{n-1}, and c_2 and c_3 in DIM_0

Lemma 2 *The proposed fault tolerant routing algorithm is livelock-free and correctly routes all messages.*

Proof. To see that messages are correctly delivered without introducing livelocks in the faulty network, observe that (a) a message is misrouted only around an f-ring, (b) a message, once it leaves an f-ring will never revisit it, (c) there are a finite number of f-rings in the mesh, (d) a normal message progresses towards its destination with each hop, and (e) the destination node is accessible, since all non-faulty nodes are connected. Since a message is misrouted only by a finite number of hops on each f-ring and it never visits an f-ring twice, the extent of misrouting is limited. This together with the fact that each normal hop takes a message closer to the destination proves that messages are correctly delivered and that livelocks do not occur. ∎

5.2 Extensions

Extension to multidimensional tori. The algorithm for routing in the presence of nonoverlapping f-rings can be extended to n-dimensional (nD) tori using the planar-adaptive routing (PAR) technique [11].

The block-fault model for nD tori assumes that fault-regions are in multiple, disjoint nD boxes. The routing algorithm to handle nonoverlapping f-rings still needs only four virtual channels per physical channel. The key issue is how virtual channels and planes are used to route messages. Let $A_{i,j}$, where $0 \leq i < j < n$, denote the set of all 2D planes formed using dimensions i and j. Further, $A_{i,j} = A_{j,i}$. We use only planes in sets $A_{i,j}$, where $0 \leq i < n$ and $j = i + 1 \pmod n$.

A normal message that needs to travel in DIM_i, $0 \leq i < n$, as per the e-cube is an M_i message and is routed in a plane of the type $A_{i,j}, j = i + 1 \pmod n$. A M_i message that completed its hops in dimension DIM_i becomes a DIM_j message, where $j > i$ is the next dimension of travel as per the e-cube algorithm. A blocked message uses the f-ring in its current 2D plane to get around faults. Table 2 indicates the types of planes and virtual channels used in routing various types of messages by the routing algorithm. A message that still needs hops in dimension 0 uses virtual channel c_0 (before reserving a wraparound link in dimension 0), and virtual channel c_1 (after reserving a wraparound link in dimension 0). For $n = 3$, this usage is the same as described in the previous section for 3D tori. Using this virtual channel assignment, one can show that dependencies do not arise on internode and interchip links, using arguments similar to those given in Lemma 1.

Extension to meshes. The above techniques for fault-tolerant routing can be applied to mesh networks as well. For mesh networks, due to the absence of wraparound links, the required number of virtual channels is smaller than that for tori. For example, two virtual channels per physical channel are sufficient to handle nonoverlapping f-rings in meshes. However, in meshes, faults on the boundaries of the network (for example, topmost row in a 2D mesh) require special handling. The treatment of meshes is similar to that given in [3, 4]. The proof of deadlock free routing is similar to that given above for torus routers.

6 Simulation Based Performance Study

We have used a flit-level simulator to study the performance of the fault-tolerant PDRs proposed in this paper. We have simulated 16×16 mesh and torus networks for the uniform traffic pattern with geometrically distributed message interarrival times. In practice, fixed length messages give better manageability of resources such as injection and consumption buffers, and small message sizes are more suitable for fine-grain computations. Hence, we have used fixed length messages of 20 flits, which could be suitable for transmitting four 64-bit words together with header, checksum and other information on 16-bit wide physical channels.

For torus simulations, we have simulated four virtual channels on each internode and interchip physical channel, and for mesh simulations, two virtual channels per physical channel. On physical channels that are neither faulty nor part of f-rings, all the simulated virtual channels are used to route normal messages. Since on each such physical channel only one dimension messages travel, extra channels are available to reduce channel congestion. Recent studies [5, 14] have shown that using more virtual channels than those necessary for deadlock free routing improves the performance of the e-cube considerably. On physical channels that are part of f-rings, each virtual channel is used for a specific type of message. The virtual channels on a physical channel (internode as well as interchip) are demand time-multiplexed, and it takes one cycle to transfer a flit on a physical channel.

We have assumed that messages experience processing delays when passing through intermediate nodes. We used a delay of 3 cycles to process header flits, and a delay of 2 cycles to route data flits from an incoming channel to an outgoing channel of an intermediate node. This is in addition to any other delays that a flit may see because of either round robin processing of one incoming header at a time by the router or virtual channel bandwidth allocation.

Each virtual channel has a buffer of depth four (holds four flits) to pipeline message transmission smoothly. Because of asynchronous pipelining of message transmission among nodes, bubbles are created with shallow buffers of depth 1 or 2. So, mesh routers have 32 flits of storage and torus routers 64 flits of storage.

To facilitate simulations at and beyond the normal saturation points for each routing algorithm, we have limited the injection by each node. This injection limit is independent of the message interarrival time. After some experimentation, we have set the injection limit to 2, which means that a node may inject a new message if fewer than two of its previously injected messages are still in the node. When there are faults in the network, the injection limit

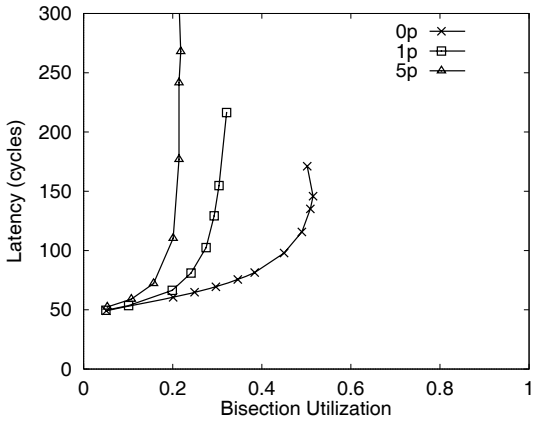

Figure 8: Performance of the fault-tolerant PDR for a 2D torus network with four virtual channels per physical channel. The label dp indicates results for d% faults.

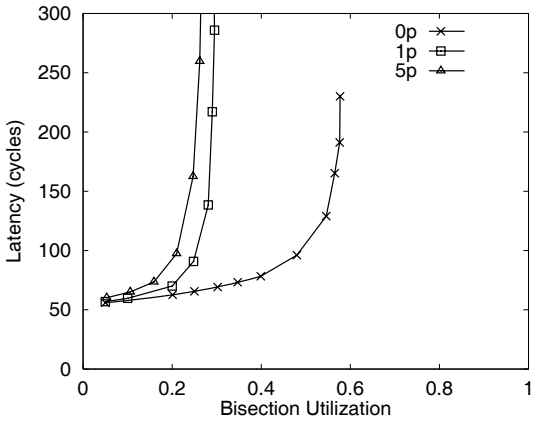

Figure 9: Performance of the fault-tolerant PDR for a 2D mesh network with two virtual channels per physical channel. The label dp indicates results for d% faults.

has little effect on the latency and throughput values prior to the saturation.

We use bisection utilization and average message latency as the performance metrics. The bisection utilization (ρ_b) is defined as follows.

$$\rho_b = \text{Number of bisection messages delivered/cycle} \times \frac{\text{Message length}}{\text{Bisection bandwidth}}$$

The bisection bandwidth is defined as the maximum number of flits that can be transferred across the bisection in a cycle, and is proportional to the number of nonfaulty links in the bisection of the network—for example, the row links connecting nodes in the middle two columns of a 16×16 mesh. A message is a bisection message if its source and destination are on the opposite sides of the bisection of the fault-free network. The average message latency is the average duration from a message's injection to its consumption.

We have simulated the mesh and torus networks with no faults and with approximately 1% and 5% of the total network links faulty. We have used a mixture of node and link faults. Node faults cause more severe congestion, since a node fault blocks both row and column messages while a link fault blocks only one type of messages. We have set one node and one link faulty for the 1%-faults case, and 4 nodes and 10 links faulty for the 5%-faults case. In each case, we have randomly generated the required number of faulty nodes and links such that isolated faults with nonoverlapping f-rings are formed.

Performance under faults. Figures 8 and 9 give the simulation results for torus and mesh networks, respectively. For each value reported in these graphs, the 95% confidence interval is within 10% of the value.

In each case, the performance for fault-free routing is much higher than the performance with faults. The peak utilization for torus PDR without faults is 52%, but drops to 32% with 1% faults and to 22% with 5% faults. Similarly, the peak utilization for mesh PDR without faults is 58%, but drops to 30% with 1% faults and to 27% with 5% faults. These results are consistent with the performance degradations seen for crossbar based fault-tolerant dimension-order routers [4].

The high performance for fault-free networks is due to use of the extra channels to avoid congestion. Even with a single fault, the number of types of messages traveling on f-ring links is increased and severe congestion occurs. Thus an f-ring becomes a hotspot causing performance degradation. Therefore, for graceful degradation of performance, some form of adaptivity should be considered. We believe that it should be feasible to provide limited adaptivity while retaining the multimodule implementation of the router.

The performances of torus and mesh networks are not directly comparable for several reasons. Mesh routers are simulated with 32 flits of storage, while torus routers are simulated with twice as much storage. Bisection utilization is a ratio of achieved throughput to bisection bandwidth, which is influenced by the topology. In terms of raw throughput, the fault-free torus delivered messages at the rate of 66 flits or 3.3 messages/cycle, while the fault-free mesh delivered at the rate of 36 flits/cycle.

Impact on fault-free performance. Our approach for deadlock-free routing uses a few extra virtual channels to break cyclic dependencies among channels. If a fault-free network already uses virtual channels, then the impact of using a few more virtual channels to provide fault-tolerant routing is not too severe. Otherwise, adding virtual channels for fault-tolerance may affect the fault-free performance. For example, dimension-order routing on a fault-free mesh is deadlock free without using any virtual channels. Since we need two virtual channels per physical channel to provide fault-tolerant routing, the cost and speed of PDRs are affected. The cost is increased because of the additional switching and virtual channel controllers at the outgoing channels. The speed may be reduced because of the increased complexity of selecting an outgoing channel and additional delays through virtual channel controllers.

In this paper, we address the impact of the reduced speed on message delays and network throughput. For the sake of simplicity, assume that the node delays for flits is 1

cycle in the PDR without virtual channels. The reduction of router speed can be handled in one of the following two ways.

- Unpipelined routers: Transit time for each flit through an intermediate node is still one cycle. But the router operates with a slower clock.
- Pipelined routers: Clock rate of the router is kept the same. However, the transit time for a flit through a node equals multiple clock cycles.

Chien [10] analyzed several wormhole router organizations and concluded that adding virtual channels could increase the clock cycle time of a router substantially. This analysis is based on the assumption that routers are unpipelined. An unpipelined router has to examine the header of a message on an incoming channel, select an appropriate outgoing channel, and place the header on the selected outgoing channel (in the absence of contention) in one clock cycle. So, introducing virtual channels increases the delays seen by messages in their intermediate nodes.

An alternative is to pipeline the message path within a router. By pipelining the message path, the clock rate need not be reduced when virtual channels are introduced. A message still sees larger node delays in the form of multiple clock cycles. For example, a message header may see a three stage processing: buffering at input channel, selecting and switching to an appropriate outgoing channel, and virtual channel controller at the output channel. Once a path is established for a message, its subsequent data flits cut-through each intermediate node in two stages: buffering at input channel and virtual channel controller at the output channel.

We have conducted simulations to compare message delays and throughputs for both types of routers. Figure 10 gives the results. For the pipelined router, at each intermediate node visited, header flits see 3-cycle delays and data flits 2-cycle delays. For the unpipelined router, the node delays are constant—1 cycle.

If the unpipelined router has the same clock rate as the pipelined router, then the former has about 30 cycles lower latency and 5% higher bisection utilization. If clock cycle time of the unpipelined router is about 30% more than the pipelined router, then both give rise to the same message delays. However, the pipelined router gives over 20% higher throughput in terms of bytes/second.

In our earlier simulations for a 16×16 mesh with crossbar based routers, one virtual channel (same as no virtual channels) per physical channel, and channel buffer depths of 8, we obtained about 60% bisection utilization [3]. From the results in Figure 10, a pipelined PDR with two virtual channels and channel buffer depth of 4 achieves similar throughputs. The main difference is message delays are higher in the pipelined router. In both cases, the router clock cycle time and amount of buffer space per physical channel are the same.

This shows that adding virtual channels to a network that does not already have virtual channels may not reduce the throughput. Pipelining the message path within a router is the key. Thus, adding additional virtual channels to provide fault-tolerant communication does not necessarily reduce the performance for the fault-free case.

If a network already uses virtual channels for the fault-free case, then adding a few more virtual channels causes only small increases to the cost and clock cycle time. In such cases, the throughput increased by adding a few extra virtual channels (as in the torus case) usually outweighs any small increases in message delays.

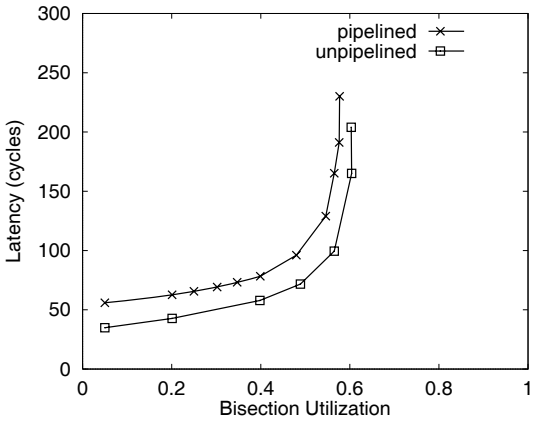

Figure 10: Performances of unpipelined and pipelined PDRs in a (16,2)-mesh with two virtual channels per physical channel. The unpipelined router is simulated with 1-cycle delay for flits going through an intermediate node. The pipelined router is simulated with 3-cycles delay for header flits and 2-cycles delay for data flits.

7 Concluding Remarks

We have presented a technique to enhance the dimension-order routers in multicomputer networks to tolerate block faults. Particular attention has been paid to the applicability of proposed techniques for current multicomputers which use partitioned dimension-order routers (PDRs). Since PDRs do not have centralized crossbars, the previously proposed techniques for fault-tolerant routing cannot be implemented without redesigning the existing routers. With the proposed technique, however, multiple faults can be tolerated while retaining the PDR implementation. 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.

There are two possible disadvantages. The chip count is increased with the use of multiplexers. These multiplexers themselves may be prone to faults. Also, transit delays for messages cutting-through intermediate nodes are higher. Neither is a severe disadvantage, however. Multiplexers are simple and inexpensive, so their cost and reliability do not affect the overall cost and reliability of the router. With extensive pipelining for the router chip, increased transit delays do not affect the network throughput.

In this paper, we have shown how to handle faults that give rise to nonoverlapping f-rings. Overlapping f-rings can be handled using more virtual channels than in the case of nonoverlapping f-rings. To make the length of all links in a given dimension of the torus the same, often alternate nodes in a given dimension are placed physically close on the same circuit board [12]. In this case, the faults on a board lead to overlapping f-rings, which can be handled using more virtual channels than in the case of nonoverlapping f-rings [8].

We have presented simulation results on the performance of the fault-tolerant routing in mesh and torus networks with PDRs. As in the case of meshes with crossbar based routers [4], the first fault causes substantial performance degradation. Additional faults cause only a little performance degradation. This is due to the hotspot effect created by faults on the nodes and channels on f-rings and the nonadaptive nature of dimension-order routing, which does not allow messages to take alternate paths when the dimension-order path is congested.

Using adaptivity provides graceful degradation of performance. But at the present, there are no known adaptive routing methods that are suitable for multichip implementations. Future multiprocessors are more likely to use multichip routers than single-chip routers, since for the same technology, multichip implementations provide wider internode channels. Therefore, further research on combining adaptivity and fault-tolerance for multimodule routers is needed.

Acknowledgments

We would like to thank Professors James E. Smith and P. Ramanathan for many insightful comments on this work.

References

[1] K. Bolding and L. Snyder, "Overview of fault handling for the chaos router," in *Proceedings of the 1991 IEEE International Workshop on Defect and Fault Tolerance in VLSI Systems*, pp. 124–127, 1991.

[2] K. Bolding and W. Yost, "Design of a router for fault-tolerant networks," in *Proc. of Parallel Routing and Communication Workshop*, pp. 226–240, May 1994.

[3] R. V. Boppana and S. Chalasani, "Fault-tolerant routing with non-adaptive wormhole algorithms in mesh networks," in *Proc. Supercomputing '94*, Nov. 1994.

[4] R. V. Boppana and S. Chalasani, "Fault-tolerant wormhole routing algorithms for mesh networks," *IEEE Trans. on Computers*, vol. 44, pp. 848–864, July 1995.

[5] S. Borkar et al., "iWarp: An integrated solution to high-speed parallel computing," in *Proc. Supercomputing '88*, pp. 330–339, 1988.

[6] Y. M. Boura and C. R. Das, "Fault-tolerant routing in mesh networks," in *Proc. 1995 Int. Conf. on Parallel Processing*, pp. I.106–109, Aug. 1995.

[7] S. Chalasani and R. V. Boppana, "Adaptive wormhole routing in tori with faults," *IEE Proceedings: Computers and Digital Techniques*. To appear. Preliminary results presented at International Conference on Supercomputing '94.

[8] S. Chalasani and R. V. Boppana, "Fault-tolerant communication with partitioned dimension-order routers." February 1995.

[9] S. Chalasani and R. V. Boppana, "Adaptive fault-tolerant wormhole routing algorithms with low virtual channel requirements," in *Proc. Int'l Symp. on Parallel Architectures, Algorithms and Networks*, pp. 214–221, Dec. 1994.

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

[11] A. A. Chien and J. H. Kim, "Planar-adaptive routing: Low-cost adaptive networks for multiprocessors," in *Proc. 19th Ann. Int. Symp. on Comput. Arch.*, pp. 268–277, 1992.

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

[13] 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.

[14] W. J. Dally, "Virtual-channel flow control," *IEEE Trans. on Parallel and Distributed Systems*, vol. 3, pp. 194–205, Mar. 1992.

[15] W. J. Dally and H. Aoki, "Deadlock-free adaptive routing in multicomputer networks using virtual channels," *IEEE Trans. on Parallel and Distributed Systems*, vol. 4, pp. 466–475, April 1993.

[16] W. J. Dally, L. R. Dennison, D. Harris, K. Kan, and T. Xanthopoulos, "The reliable router: A reliable and high-performance communication substrate for parallel computers," in *Proc. of Parallel Routing and Communication Workshop*, pp. 241–255, May 1994.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] C. J. Glass and L. M. Ni, "Fault-tolerant wormhole routing in meshes," in *Twenty-Third Annual Int. Symp. on Fault-Tolerant Computing*, pp. 240–249, 1993.

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

[23] T. Lee and J. Hayes, "A fault-tolerant communication scheme for hypercube computers," *IEEE Trans. on Computers*, vol. 41, pp. 1242–1256, Oct. 1992.

[24] 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.

[25] C. S. Raghavendra, P.-J. Yang, and S.-B. Tien, "Free dimensions – an effective approach to achieving fault tolerance in hypercubes," in *Twenty-Second Annual Int. Symp. on Fault-Tolerant Computing*, pp. 170–177, 1992.

[26] 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.