VLSI Design
1995, Vol. 2, No. 4, pp. 315–333
Reprints available directly from the publisher
Photocopying permitted by license only

The Cost of Adaptivity and Virtual Lanes in a Wormhole Router

KAZUHIRO AOYAMA
Mitsubishi Electric Corporation, Kamakura, Japan

ANDREW A. CHIEN
Department of Computer Science, University of Illinois, Urbana, IL 61801

We examine the cost in router complexity of adaptivity and virtual lanes in wormhole routers, using f-flat adaptive routers (based on a generalization of *planar-adaptive routing*) which include routers with a range of routing freedom. Our studies show that adaptivity is expensive because it requires additional virtual channels and much larger crossbar switches for both adaptivity and deadlock prevention. Increases of 50 to 100% in channel utilization are required to justify additional degrees of routing freedom.

Three internal router architectures for virtual lanes are examined and the fully expanded crossbar is found to be most effective because it gives simplest control and minimal internal blocking. Examining router designs with from 1–16 virtual lanes indicates that 30% improvements in channel utilization are required to justify each additional virtual lane. These studies combined with published simulation results indicate that only modest numbers of virtual lanes are likely to be cost effective.

Key Words: *Routing Networks, Interconnection Networks, Adaptive routing, Parallel processing, cut-through routing, wormhole routing.*

1 INTRODUCTION

In concurrent computers, interconnection networks are used by the processing nodes to exchange data and synchronize with each other. Network performance is often critical, as the performance of large-scale parallel machines is sensitive to network latency and throughput. While multicomputers have been touted as scalable parallel architectures, their scalability is limited by the performance of their interconnection networks.

An interconnection network is defined by its topology, routing, and flow control. The topology is the pattern of network node interconnection via physical communication channels. The routing algorithm specifies how packets choose paths through the network. Flow control deals with the allocation of channel and buffer resources to packets as they proceed through the network. This paper focuses evaluating the cost of a variety of routing features involving routing and flow control.

Deterministic, dimension-order routers are used in a variety of multicomputers. Such routers use only one path through the network, deterministically routing each packet from source to destination. Deterministic routers are attractive because they are exceedingly simple and provide low latency and high bandwidth. However, deterministic routers have a number of significant disadvantages: poor performance under non-uniform traffic loads and poor fault tolerance [16, 32, 25]. Adaptive routing is a promising approach to alleviate these problems, but adaptive routers can be more complex and this complexity leads to legitimate concerns about their speed and tangible benefits. Adaptive routers allow many paths between source and destination to be used, generally choosing based on network load conditions. Recently, a number of dramatically simpler adaptive routing algorithms have been proposed [32, 9, 5]. This breakthrough makes adaptive routing feasible, but not without cost. Deciding whether or not to incorporate adaptive routing is still a complex cost-performance tradeoff with the cost side of the equation still largely undefined. Virtual lanes provide multiple lanes for messages along any particular path in a routing algorithm [14, 8]. As with adaptive rout-

ing, virtual lanes can improve router performance by increasing channel utilization. However, they also increase router complexity, slowing implementations.

In this paper, we examine the cost of adaptivity and virtual lanes in one family of deterministic and adaptive routers based on a series of gate-level router designs. While study of a wider range of adaptive routing algorithms is of interest, such is beyond the scope of the paper. This paper makes two significant contributions. First, it gives a detailed description of an adaptive wormhole router, characterizing the functionality and speed of each module and providing a basis for estimating router speeds. Second, it examines the speed of a baseline deterministic router and a family of enhanced routers with increased routing freedom and numbers of virtual lanes. This not only allows the speed of adaptive routers to be compared to existing deterministic router designs, it also provides a basis for assessing the cost of adaptivity and virtual lanes, admitting a cost performance tradeoff.

To assess the cost of adaptivity, we examine a series of router designs with a range of adaptivity. In this context, we define adaptivity as the maximum number of routing choices at an intermediate routing node. We examine routers with from one to eight degrees of routing freedom. Our router designs show that the cost of a few degrees of routing freedom can be modest. However, higher degrees of adaptivity incur much greater costs; 50% or greater increases in channel utilization are required to justify each additional degree of routing freedom beyond two.

To assess the cost of virtual lanes, we examine router designs with from one to sixteen virtual lanes. Several router architectures have been proposed for virtual lanes, so we first examine each of these and then select the most attractive, a fully expanded crossbar. Our design studies of this architecture show that while virtual lanes are expensive, they are less expensive than increased adaptivity. Each additional virtual lane requires an increase in channel utilization of 30% to be cost effective. The majority of the increased cost is in larger crossbars and much larger virtual channel controllers. Given published studies of the benefits of virtual lanes, a few virtual lanes may give enough of a throughput increase to justify this cost, but large numbers of virtual lanes are unacceptably expensive.

Overview The remainder of the paper is organized as follows. Section 2 describes the context for this work, defining wormhole routing, planar-adaptive routing, and describing previous router implementation studies. Section 3 describes our base router design, a planar-adaptive router. Section 4 presents cost-performance metrics for router designs and applies them to the base router. With a baseline established, Sections 5 and 6 consider the cost of adaptive routing and virtual lanes. The overall performance results are summarized in Section 7. Finally, Section 8 summarize the paper and discusses several possible directions for future research.

2 BACKGROUND

Communication performance depends critically on a network's topology, flow control, and routing algorithm. We focus on k-ary n-cubes, direct networks with radix k and dimension n [13]. By varying choice of k and n, this family of networks represent a wide range of choices in density of interconnection. We also focus only on routers that use wormhole routing, a low cost approach to flow control that allows small simple routers. Wormhole routers for k-ary n-cubes have been used in a variety of commercial and research machines [22, 31, 15, 2, 1, 27, 3].

In recent years, an increasing number of interconnection networks have made use of *wormhole routing,* a fine-grained flow control technique which requires only small amounts of hardware [*18*]. Consequently, wormhole routers are small, cheap, and fast. The basic idea behind wormhole routing is to begin forming the path from source to destination, sending the data right behind the message header. If the message is blocked, all of the data flits (flow control units) are stopped in place in the network. Because the flits are stopped in place, wormhole routers require only a modest amount of storage and can support messages of arbitrary size. However, stopping the flits in place requires holding the channels along the path, giving rise to a plethora of possible deadlocks and conflicts. Addressing these issues effectively has been the subject of much research [12, 29, 19, 9, 5, 32].

Routing approaches can be divided into two categories: *deterministic* and *adaptive* routing. In deterministic routers, each message is routed along a fixed path, determined by the source and destination of the message. Deterministic routing's main advantage, hardware simplicity, is directly tied to its primary disadvantage, a lack of routing flexibility which limits network performance and fault tolerance. Any particular fixed choice of routes will produce poor performance for some communication patterns.

Adaptive routing can alleviate such problems by mapping communications to paths flexibly, based on

network loading. The flexibility in routing improves performance on non-uniform workloads [25] and can provide a measure of fault tolerance [9, 23]. The major disadvantage of adaptive routing is the greater complexity required to support the additional routing flexibility while assuring deadlock-freedom. This increase in hardware complexity can significantly reduce router speed, decreasing total network performance. To reduce the cost of adaptive routing, many approaches based on limited adaptivity have been proposed.

Virtual lanes can also be added to deterministic or adaptive routers. The idea behind virtual lanes is to increase the utilization of physical channels by decoupling two types of network resources: buffers and physical channels [14]. If a message holding a buffer is blocked, the message releases the physical channel so that other messages can use it. Adaptive routing and virtual lanes complement each other: adaptive routing attempts to distribute traffic uniformly over the physical channels and virtual lanes attempt to maximize the utilization of each physical channel.

In this paper, we focus on one family of adaptive routing algorithms which encompass a range of adaptivity and hardware complexity. This family allows routing freedom to be traded off against router speed while assuring deadlock-freedom. The simplest adaptive router in this family, a planar-adaptive router, is described below. In this context, we evaluate the cost of adaptivity and virtual lanes.

2.1 Planar-adaptive Routing

Planar-Adaptive Routing (PAR) is a limited adaptivity routing algorithm. PAR has many implementation advantages, most notably hardware simplicity. PAR uses only three virtual channels for deadlock prevention and small crossbar switches regardless of the number of dimensions [9, 24, 4].

The idea in planar-adaptive routing is to provide limited adaptivity by routing adaptively in a series of two-dimensional planes. As the packet progresses towards its destination, it passes through a series of adaptive planes and eventually, the packet completes routing in all dimensions and is delivered to the destination. By limiting adaptivity to two dimensions and structuring the passage from one adaptive plane to the next, network cost can be reduced while maintaining deadlock-freedom.

Define $n - 1$ adaptive planes, A_0 to A_{n-2}, as the combination of several sets of virtual channels. $d_{i,j}$ denotes the jth virtual channel in the ith dimension

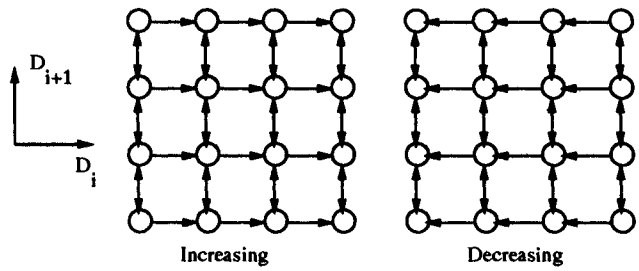


FIGURE 1 The plane A_i is divided into two virtual planes, increasing A_i+, and decreasing A_i-, They are completely decoupled.

$d_{i,j}+$ and $d_{i,j}-$ denote subsets of those channels in the increasing and decreasing directions respectively.

$$A_i = d_{i,2} + d_{i+1,0} + d_{i+1,1}$$

Each adaptive plane involves only two dimensions.[1] Three virtual channels in each dimension are needed to support the $n - 1$ adaptive planes.

Planar-Adaptive Routing Algorithm:

High-level (between adaptive planes)

1. For $i = 0$, $i < (n - 1)$ do
   Route adaptively in A_i, see low-level routing.
   end
2. After exiting the loop, it can only be necessary to correct the address in d_{n-1}. If necessary, route in $d_{n-1,2}$ to the destination.

Low-level (within adaptive plane A_i)

Adaptive plane A_i contains virtual channels $d_{i,2}$, $d_{i+1,0}$, and $d_{i+1,1}$. Within the plane, route adaptively with respect to dimensions d_i and d_{i+1} by choosing any channel that leads closer to the destination. In order to prevent deadlock, the traffic is divided into two classes: packets *increasing* and *decreasing* their d_i address. Route them in the increasing and decreasing networks respectively that are shown in Figure 1.

Increasing Network: $d_{i,2}+$, $d_{i+1,0}$

Decreasing Network: $d_{i,2}-$, $d_{i+1,1}$

Within each network, traffic is routed adaptively towards its destination in any of the productive channels. When the d_i address is correct, routing is completed in plane A_i, so proceed to the next high-level step.

[1]The order of dimensions is arbitrary.

In the high-level routing, the basic idea is to route successively in the adaptive planes. Routing in adaptive plane A_i reduces the distance in d_i to zero. After routing in all of the adaptive planes, the packet has reached its destination. For d_{n-1}, there cannot be any adaptivity left for a minimal router, so the packet is routed directly to its destination. In the low-level routing, the scheme is adaptive, as multiple paths can be chosen within each adaptive plane.

2.2 F-flat Adaptive Routing

The planar-adaptive routing algorithm can be generalized to support higher degrees of adaptivity. The basic idea is to increase the degree of routing freedom at each low-level routing step (adaptive planes to adaptive cubes, etc.), producing the class of f-flat routers. F-flat adaptive routers are deadlock-free and allow a range of routing freedom choices. An f-flat adaptive router allows routing in the f-flat subspace of the n-dimensional space, giving f degrees of adaptivity. Thus, a planar-adaptive router is a 2-flat adaptive router. The composition of adaptive spaces is handled in an analogous fashion to planar-adaptive routing. Any deadlock-free adaptive routing algorithm can be used with the f-flat. Increasing the routing freedom by increasing f can improve channel utilization, but each such increase requires additional hardware, incurring increases in router latency and router clock periods. The dimension-order router is a 1-flat adaptive router.

2.3 Related Work

In this section, we survey the related router design studies. All of the work surveyed here involves deterministic routers. While differences in router functionality and implementation technology make direct comparisons difficult, the comparison shows that our router design is competitive. Recently, there others have studied the complexity of adaptive routers [6], but these routers use virtual-cut through and misrouting. While their results are consonant with ours, the different context makes the results somewhat incomparable.

Caltech Routing Chips The *Torus routing chip* (TRC) is a dimension-order router for *k-ary n-cube* networks [18] which implemented wormhole routing and used virtual channels to prevent deadlock. Channel throughput was 8 MB/s with byte wide self-timed communication channels (8 Mhz). This was about an order of magnitude better than contemporary communication networks used by the Caltech Cosmic Cube or Intel iPSC. The router setup latency was 150ns per routing step. The mesh routing chip (MRC) is the second generation Caltech routing chip [21]. The MRC supports mesh networks with dimension-order-routing. The MRC is a self-timed circuit which based on 3×3 crossbar switches (one port for positive direction, the negative direction and the processor element) in each dimension. Router latency is approximately 50ns and channel data rates are as high as 90MB/s, using byte-wide links. Derivatives of MRCs are used in several research machines [1, 28, 34].

The J-Machine Router The J-Machine is a fine-grained concurrent computer developed at MIT [17, 33]. The J-machine network is a three-dimensional mesh, with bidirectional 9-bit channels, and dimension-order, wormhole routing. The J-Machine network uses two virtual channels to support two logically independent message priorities and a globally synchronous clock. The data throughput is 36 MB/s (32 Mhz). The latency of the routing is 62.5 ns per hop.

Recent Router Designs The latest Caltech EMRC routing chips are also dimension-order, wormhole routers [35]. These chips are self-timed and use byte wide channels to achieve 66 MB/s. The typical path formation latency for the head of a packet is approximately 30*ns*. The Intel Paragon router is descended from the original Caltech MRCs [22, 20]. The Paragon router is a deterministic router and comparable to our designs, as it is implemented in a similar technoolgy (0.8 micron CMOS gate array) and gives performance comparable to our designs. Published figures for its delay and channel bandwidth are 40 ns and 200 MB/second respectively.

3 BASE ROUTER DESIGN

In this section, we describe the basic router design which is used as a baseline for our cost studies. We describe the architecture of the router, describing the functionality of each module in detail. The goal is to provide insight as to how and why router changes affect performance. In the following sections, the cost of each additional router feature is calculated by comparing the performance of the enhanced router to the baseline design. Our baseline router is a *planar-adaptive router* (PAR) as described in Section 2.1.

A planar-adaptive router consists of a series of composable modules, one for each adaptive plane.

The external interface of one such plane consists of four bidirectional links and two ports which are typically connected to the local processor (see Figure 2). These two ports can also be used to compose routers for a higher dimensional network. The bidirectional connections to neighbors, denoted L1, L2, L3, and L4 are each implemented with dual unidirectional channels, and a similar interface is used for the processor ports. PAR requires two virtual channels in the y-dimension to support deadlock-free adaptive routing (labeled *iyp* and *iyn* for increasing x and *dyp* and *dyn* for decreasing x, in the positive and negative y directions respectively).

Design and Technology Assumptions Because pin-limitations are a concern affecting router throughput, our designs use data channels with 16 bits of data and 7 additional control signals.[2] This produces a router with well below 250 data pins, well within the range feasible for a pin grid array or more advanced packaging technology. Our designs are based on a 0.8 micron CMOS gate array library from Mitsubishi Electric Corporation. All timing estimates are based on conservative routing estimates, nominal processing, and nominal operating temperature.

3.1 Overall Design

A complete block diagram showing router internals can be found in Figure 3. Packets flow from left to right. Incoming links are attached on the left and outgoing links on the right. For each link there are data lines and control lines in the forward direction (left to right, shown with thick lines) and flow control signals in the reverse direction (right to left, shown with thin lines). The upper crossbar and routing decision logic support the increasing x subnetwork and the lower crossbar and routing decision logic the decreasing x subnetwork (see Figure 1). The basic function of each block is described below. A complete description of the router can be found in Aoyama's thesis [4].

- External flow controller (XFC) supports asynchronous internode communication by synchronizing inputs to the local node clock.
- Address Decoder (AD) decodes the packet header, generating requests for permissible outputs.
- Internal flow controller (IFC) controls data flow across the switch and updates packet headers (relative addressing) based on the routing decision signals from the RD. Header update is overlapped with address decoding in the AD to decrease router latency.
- Routing Decision block (RD) receives request signals from all of the AD's and arbitrates amongst these signals, generating routing decisions which control the IFC's and the crossbar switch.
- Crossbar Switch (CB) connects input channels to output channels.
- Virtual Channel controller (VC) multiplexes virtual channels onto the physical channels. Because the router interfaces are asynchronous, the VC also synchronizes the flow control signal. The VC and the IFC manage intranode data flow cooperatively to minimize internal delays.

The router is internally synchronous, but externally asynchronous. Internal synchrony makes inter-

[2]A few more control signals are needed for each additional virtual channel.

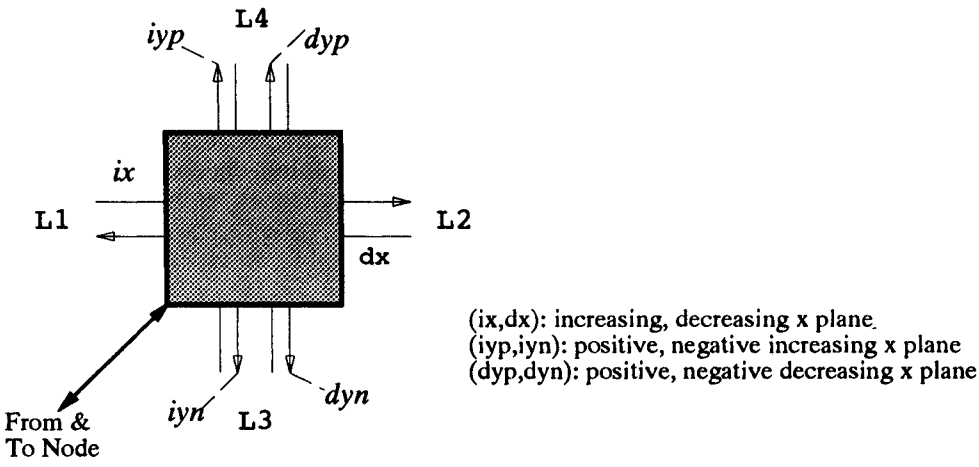


FIGURE 2 External interfaces of a *planar-adaptive router.*

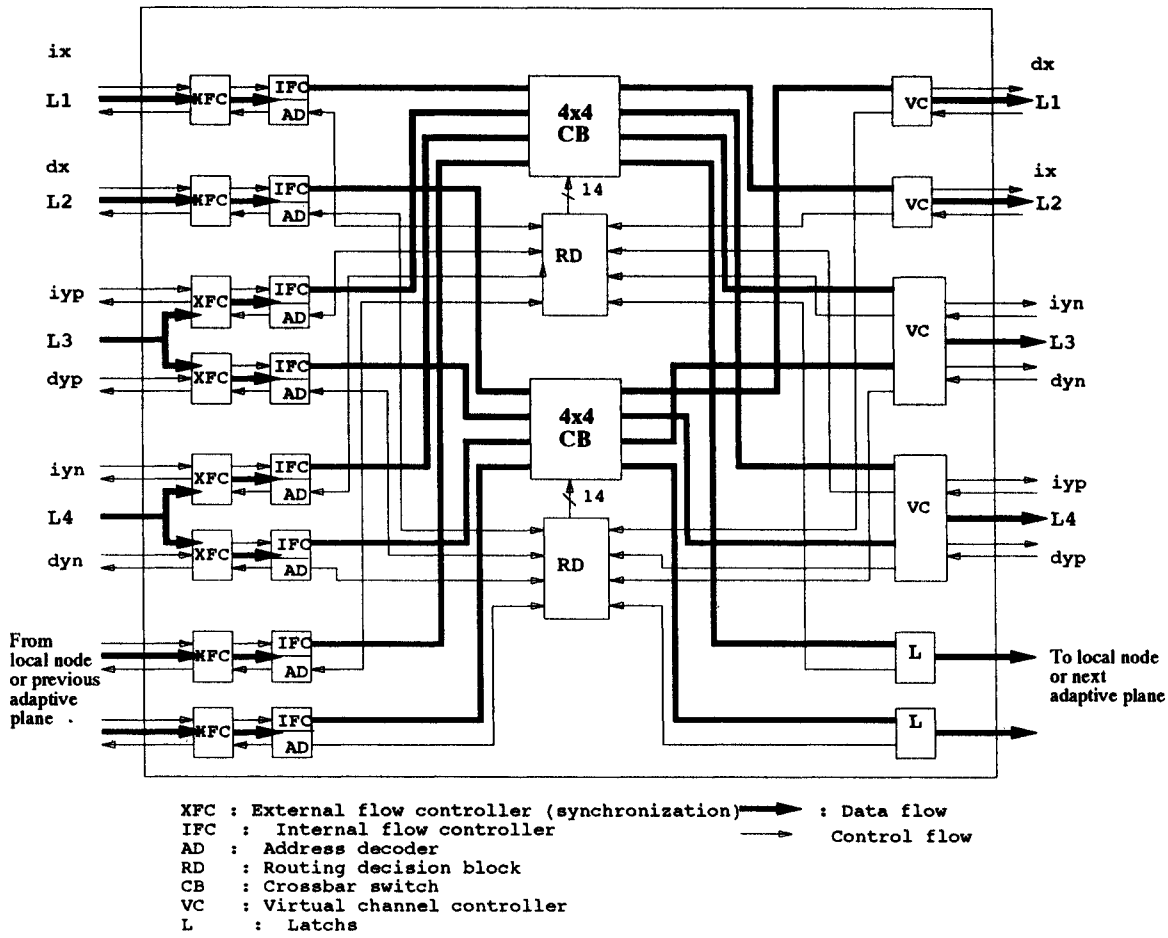


FIGURE 3 Internal organization of a 2D planar-adaptive router.

nal coordination, particularly fair arbitration and selection for virtual channels, inexpensive. An asynchronous interface between routers is a consequence of the difficulty of distributing a high speed, low skew clock to a large system.[3] We assume that all routers operate with a clock of identical frequency but with differing and even slightly variable skew. Differences in clock phase between nodes are handled by synchronizers.

The critical internal router operations which affect router setup latency and achievable clock rate in the network router are *path setup* and *data through,* respectively. Path setup is the delay incurred by each packet as it is forming a path to the destination. Data through is the delay incurred by each flit as it moves through the router. A path setup operation involves the following steps: the AD generates requests based on the header, the RD assigns a path and sets up the switch, and data flows through the switch to the output (VC if appropriate). The data through operation determines the achievable channel bandwidth because it defines the rate at which flits can move through the router. A data through operation consists of the following steps: the IFC sends data forward, the data moves through the CB, and the data is accepted for transmission at the VC. To ensure correct operation our planar-adaptive, wormhole router must manage the following tasks: flow control, routing, and virtual channel multiplexing. In the following sections, we discuss how each of the tasks are accomplished.

3.2 Flow Control

The major benefit of wormhole routing is that routers can have extremely low buffer requirements. How-

[3]An alternative would be to use phase-locked loops, which does not change the synchronization cost. It simply fixes the synchronization penalty to an integral number of clock periods between routers.

ever, one consequence of this is that flow control performed on small units of data, and must be performed extremely rapidly to prevent buffer overflow. A competing goal is to maximize the channel bandwidth usable by a single packet. To achieve both of these goals, our design fully pipelines flow control operations, allowing a single packet to use the full bandwidth of a physical channel.

Between routers, both the data and flow control signals are asynchronous, so the synchronization time increases the effective delay in both directions. The XFC synchronizes the incoming data, and the VC synchronizes backward flow control signals using a synchronizer based on a Muller C element [30] to sample the input signal.

Pipelined flow control, synchronization penalties, channel delay, and clock skew all increase the buffer requirements of wormhole routing. If the channel delay is one clock cycle, synchronous pipelined flow control with unit delay channels requires two flit buffers, and the synchronization delay increases the buffer requirement to four flits. Thus, the minimum configuration for unit delay channels is four flit buffers per channel. Adding one more flit buffer dramatically simplifies buffer control, so our designs all include five flit buffers.

3.3 Routing Decision

In an adaptive router, routing decisions are based on the packet destination address and current router state. If several messages arrive simultaneously, all of those packets will have their paths set up in a single cycle. For each channel, the *AD* decodes the message header and generates requests for the permissible paths; this is trivial for a planar-adaptive router. The *RD* arbitrates between simultaneous requests and enforces resource constraints (no more than one packet connected to each output). Our RD design uses the straight-first selection policy [24] when there is no contention, giving the packets going straight priority over those turning. Fair arbitration is used to prevent starvation when a packet has already been forced to wait.

3.4 Switching

Router must switch packets, conveying data from appropriate inputs to outputs. This is done by the two crossbars (CB) which not only form the forward paths for data, but also the reverse paths for flow control signals.

3.5 Multiplexing Physical Channels

Virtual channels are implemented by multiplexing the physical channels. The VC, virtual channel controller, manages this multiplexing, preventing starvation for any virtual channel and attempting to utilize the physical channel efficiently. However, to achieve these goals, a VC must coordinate the movement of data from the router inputs, through the crossbar, as well as the scheduling of virtual channels onto the physical channel. Figure 4 shows three virtual channels sharing a single physical channel. At the left side there are three IFC's (internal Flow Controllers) and at the right side there is a single VC. Between them is the crossbar for this subnetwork.

The VC and IFCs cooperate to move data through the crossbar, attempting to keep the physical channel busy while fairly allocating resources to all virtual channels. When the channel is about to become idle, the VC requests data from all virtual channels which have empty downstream buffers (based on the E1-3 signals). Each such virtual channel sends data across the crossbar which is buffered, then sequenced over the physical channel. To achieve this collection and sequencing without losing cycles, the VC needs IFC buffer status and flow control information. *ready* signals from IFCs indicate the upstream buffer states, and the *empty* signals which indicate the downstream buffer states, allowing the VC to schedule only those virtual channels which can use the physical channel.

Once the data flits have reached the VC, they are sequenced across the physical channel. In Figure 5, the VC inputs come from the left, and the physical channel is on the right. Our *VC* uses two levels of arbitration; one in a collection phase and one in the delivery phase. First, during the collection phase (when data is accepted from all of the virtual channels into the staging buffers) arbitration decides which virtual channel sends first. Second, during the delivery phase, arbitration sequences the data from the staging buffers (losers in the first round arbitration) over the channel. We use fixed priority arbitration in both cases simplifies the VC. Starvation is prevented by assuring that all collected flits are transmitted before the next collection phase.

4 PERFORMANCE OF THE BASE ROUTER

To evaluate the performance impact of adaptive routing on multiprocessor routing networks, we first de-

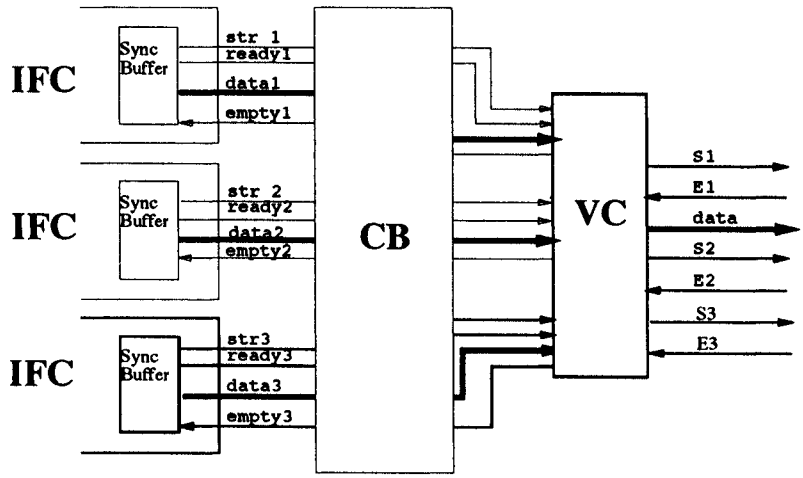


FIGURE 4 Internal flow controllers (IFC) and virtual-channel controllers (VC) managing the flow of data across the crossbar.

fine two router performance metrics. These metrics are affected by topology, routing algorithm, and implementation technology. We first focus on the internal router issues where speed is determined largely by routing algorithm and technology, producing a characterization of the internal router delay for a variety of router designs. These estimates quantify the performance impact of including advanced router features such as adaptivity and virtual lanes. Subsequently, we consider the system level issues—clocking scheme, clock synchronization, and channel delay—, and how to relate them to the internal delay measures for our base router design. Router setup latency includes both internode and intranode delay; each of which can be broken down into component delays. In this section, we estimate both contributions to router setup latency. The achievable clock rate depends on the clocking scheme and the data through delay which characterizes the basic rate at which flits can move internally.

4.1 Cost and Performance Metrics

Router performance can be characterized by several metrics: channel utilization, router setup latency and achievable clock rate. These metrics are defined below.

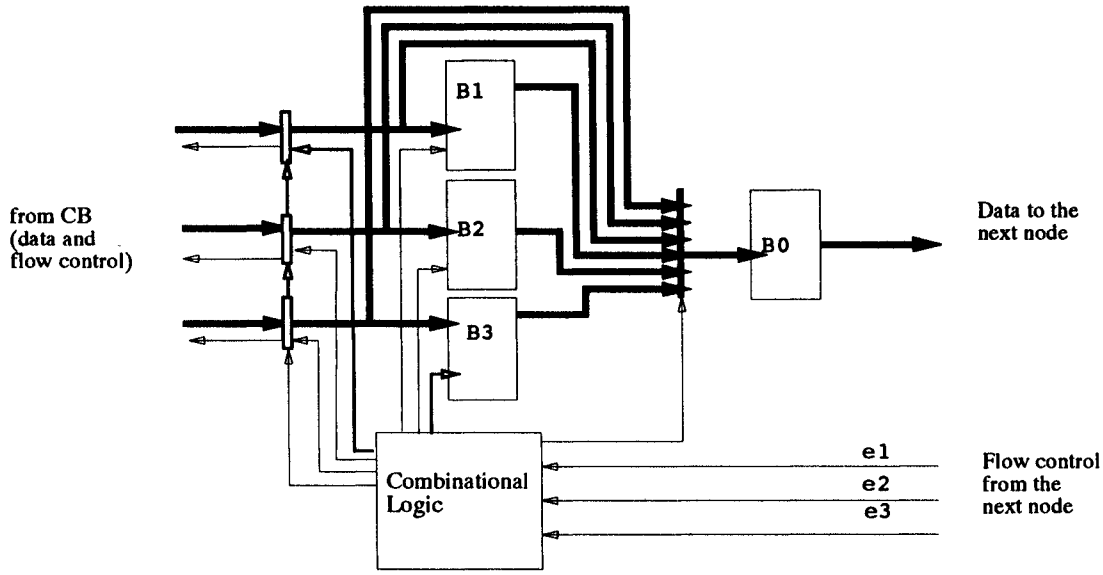


FIGURE 5 Internal structure of a virtual channel controller (VC). B1–3 are staging buffers, and B0 holds data currently traversing the physical channel.

- Channel utilization measures the router's ability to make productive use of the physical channel resources. It is characterized by the fraction of channels utilized for a given traffic load.
- Router setup latency is the delay from router input to output. Combined with the channel delay, router setup latency determines the network's zero-load delay. For wormhole-routed networks, this is often close to the typical network delay.
- Achievable clock rate is the maximum rate for which the router operates correctly. For an asynchronous designs, this is the maximum operation rate. For synchronous designs, it is the maximum clock rate. In both cases, this is the primary determinant of the channel clock rate.

Most previous studies of router enhancements have focused on channel utilization, a measure of performance improvement [32, 25, 16, 5, 7]. One reason for this is that channel utilization can be studied independent of implementation issues. We focus on the cost of adaptive routing and how it affects the router setup latency and achievable clock rate. Increases in setup latency and achievable clock period can also be viewed as the cost of router enhancements. Setup latency depends on intra-router delay (logic delays within the router) and inter-router delay (channel latency determined by topology, packaging, and synchronization time). Achievable clock rate depends on both the clocking scheme, inter and intra-router latency. For now, we focus on two internal router performance measures *path setup,* the intranode setup latency, and *data through,* the intranode flow control time, closely tied to achievable clock rate.

4.2 Performance Analysis

Internode Delay Internode delay contributes to router setup latency. Internode delay includes the time to get off chip, across the wires, and onto the destination chip (buffer, propagation, input latch, synchronizer and synchronization delays). For standard output buffers and input latches, the nominal performance of gate array library gives the delays shown in Figure 6. The output buffer delay includes line charging time, characterized by the loading. Our analysis makes no attempt to account for long channel delays. The synchronizer delay is due to gate delay in the synchronizer. This is in addition to the synchronization delay depends on the clock skew. Based on these numbers, we can estimate the best case and worst case skew.

Stage	Delay
Output buffer	2.5 ns with 25pF output load.
Input buffer	0.6 ns
Input latch	0.8 ns (setup time)
Synchronizer	1.0 ns
Total	4.9 ns

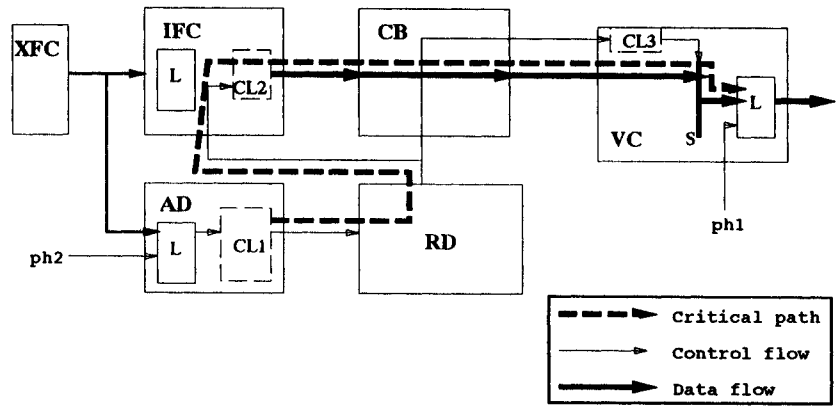
FIGURE 6 Elements of internode delay.

Based on the fixed components of internode delay, if skew is less than $T - 4.9ns$ along the forward path (where T is one half a clock period and is greater than 4.9ns), this is the best case, and the channel crossing takes only one cycle. If the skew is greater than $T - 4.9ns$ but less than $2T - 4.9ns$, the crossing will take two cycles. If T is less than $4.9ns$, the channel crossing may take two or more cycles. Even for clock rates of several hundred megahertz, $2T - 4.9ns$ is an achievable skew for a large scale system.

Intranode Delay There are two important types of intranode delay: *path setup* and *data through* delay which contribute to router setup latency and achievable clock rate respectively. While the internode delay depends primarily on topology and packaging, the intranode delay depends strongly on router features. The data through delay determines the flow control rate, thereby affecting the maximum achievable clock rate. In this section, we characterize the intranode delay for the base router, using these delays as a point of reference for the remainder of the paper.

Figure 7 (a) and (b) show the critical path and timing of a planar-adaptive router at path setup. The critical path starts at the entry to the AD, passes through the RD, through the IFC, CB, and finally the VC. Figure 7 (b) breaks the overall delay down into constituent module delays. The majority of the path setup delay is in the AD, which latches the header from the XFC then generates route requests. Most of delay in the AD is due to the data latch, L. The RD arbitrates the request signals, generates crossbar control signals, and tells the IFC which path was chosen. With knowledge of which path will be taken, the IFC selects the appropriate new header (all possible updated headers are waiting). Simultaneously, the CB is setup, and a data ready signal from the IFC passes through the CB, arriving at the VC. The crossbar setup and data ready signal operations are not on the critical path; their delay is masked by the larger header selection time. The updated header flit passes through the CB, arriving at the VC where it is latched at $ph1$.

Figure 8 shows the critical path and components

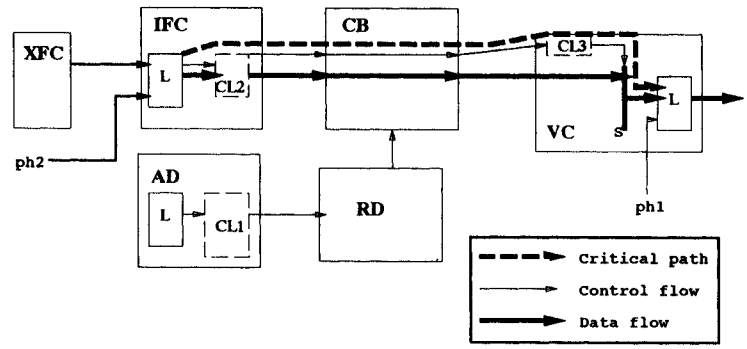


(a)

Block name	Delay
AD	3.3 ns
RD	2.1 ns
IFC	2.6 ns
CB	1.1 ns
VC	1.2 ns
Total	10.3 ns

(b)

FIGURE 7 The critical path and delay at path setup in the base router. L indicates a latch, and CL1–3 denote combinational logic blocks.



(a)

Block name	Delay
IFC	2.2 ns
CB	1.0 ns
VC	2.5 ns
Total	5.7 ns

(b)

FIGURE 8 The critical path and delay at data through in the base router. As before, L denotes latches and CL1–3 denote combinational logic blocks.

of delay of a planar-adaptive router at *data through.*[4] The data from the XFC is latched inside the IFC (in *L*), and then sent through the CB to the VC. In the VC, the data must wait for the arbitration amongst the virtual channels, even if no others are trying to send at this time. The arbiter's output controls selector *S*. After passing through the selector, the data is latched in *L* in the VC by *ph*1. It can now be transmitted to the next node.

4.3 Discussion

Determining the router setup latency is fairly straightforward, but determining achievable clock rate based on data through delay involves consideration of internode and intranode delays as well as clock skew margins. In our design, a flit crosses the channel and the router in a single clock period (channel and synchronization delay is a half period from ph1 to ph2, and router delay is the other half period from ph2 to ph1). If we assume a two-phase clock with equal length phases, then whichever delay is larger, the internode delay or intranode delay, determines the network clock rate. A more thorough description of the assumptions for determining achievable clock rate are given in Section 7.

For our base line router, the data-through delay of 5.7 ns for our baseline router dominates likely internode delays and skew margins, and thus a clock period of 2*5.7 = 11.4 ns is achievable. Such a clock period would allow a generous clock skew margin of 0.8 ns. Channel delays of two cycles would allow and easily achievable skew margin of 6.5ns.

While our adaptive router can sustain high speed and is low latency, it is slower than a comparable deterministic router. Based on the same assumptions, a dimension-order router design has delays of 5.68*ns* and 3.0*ns* at path setup and data through, respectively. The major reasons the deterministic router is faster is the lack of serialization between routing decision and header selection (routing choices and header updates are fixed) and virtual channel controllers. In terms of intranode speed, the DOR is nearly twice fast as our baseline adaptive router. However, since intranode delay at data through in DOR (3.0*ns*) is less than internode delay (4.9*ns*), the router clock rate can be dominated by internode delay, limiting the achievable clock period for the DOR to 9.8*ns*, only 16% faster than the planar-adaptive router.

[4]In our simplified reporting of performance figures, the VC appears to have a different delay in the two situations: path setup and data through. This is because the overlap of operations is slightly different in each case.

5 THE COST OF ADAPTIVITY

In this section, we characterize the cost of adaptivity by examining router designs with a range of adaptivity. These routers are all taken from the class of f-flat adaptive routers [10]. The f-flat adaptive routing framework can be used with any deadlock-free adaptive routing algorithm within each f-flat; we assume a Linder-Harden router [29] with fully adaptive minimal routing.

Increasing routing freedom not only increases the complexity of individual router modules, many more modules are needed. The hardware module requirements are generalized to f-flat routers below:

$$\#\text{ of } CB\text{'s} = 2^{f-1}$$

$$\#\text{ of } CB \text{ ports} = (f + 2)$$

$$\#\text{ of } VC\text{'s} = 2f$$

$$\#\text{ of virtual channels per } VC = 2^{f-1} + (f - 1)*2^{f-2}$$

$$\#\text{ of XFC's} = \#\text{ of IFC's} = \#of\ AD's = (\#\text{ of } CB's)\ *(\#\text{ of } CB \text{ ports}) = 2^{f-1}*(2 + f)$$

$$\#\text{ of RD's} = \#\text{ of } CB\text{'s} = 2^{f-1}$$

$$\#\text{ of request signals from } AD = (f + 2)$$

$$\#\text{ of } cnct \text{ signals from } RD = (f + 2)^2$$

To give the reader an idea of how the resource requirements (crossbars and virtual channels) increase, consider a 3-flat adaptive router. Each 3-flat (or cube) corresponds to a plane in planar-adaptive routing and is divided into four virtual subnetworks (x+, y+, z), (x+, y−, z), (x−, y+, z) and (x−, y−, z) for 3-flat deadlock-free routing. This requires two virtual channels in the first two dimensions and four

virtual channels in the third dimension. If a series of 3-flats are composed for a higher-dimensional network, eight virtual channels per physical channel are needed.

As routing freedom is increased, not only does the number of router modules required increase, some of the router modules become more complex, thereby becoming slower. We consider each router module in turn. IFC and AD delays do not change because their designs require only modest changes for higher degrees of routing freedom.

The delay in *RD* with f-flat adaptivity may be estimated a follows:

$$T_{RD_f} = \frac{f+2}{4} * T_{RD} + \left\lceil \frac{f+2}{4} - 1 \right\rceil * T_{gate}$$

T_{gate} denotes the basic gate delay. The basic structure of the *RD* consists of $f + 2$ connection controllers, whose inputs feed into $f + 2$-input priority encoders. Each controller controls ith priority signal, and the outputs of the priority controller are used to determine the *cnct* signals.[5] The term on the right comes from the lowest priority connection controller, and is proportional to f because the controllers are daisy-chained together. The term on the right arises from the combining logic which grows in proportion to the number of *cnct* signals.

The *CB* delay increases because even with partitioned crossbars, $f + 2$ ports are required per CB. The CB delay is described below:

$$T_{CB_f} = T_{CB} + \left\lceil \frac{f+2}{4} - 1 \right\rceil * T_{gate}$$

The *VC* delay for f-flat routers increases because the number of virtual channels to be multiplexed on each physical channel increases, requiring larger (deeper) arbitration circuits and selectors $T_{VC(3)}$ denotes the basic delay of a virtual channel controller for three virtual channels.

$$T_{VC_f} = T_{VC(3)} + \left\lceil \left(\frac{2^{f-1} + (f-1)*2^{f-2}}{3} - 1 \right) \right\rceil * T_{gate}*2$$

In the *VC*, there are two arbiters. The delay of each arbiter increases in discrete jumps with the number of virtual lanes in a *VC*. This increase is represented by the last term in the equation. For path setup, the overlap of the IFC delay causes the last term to be irrelevant when f is smaller than four. Combining these terms gives overall formulas for router delay with f-flat adaptivity:

At path setup:

$$\begin{aligned} T_{f\text{-}flat} &= T_{AD} + T_{RD_f} + T_{IFC} + T_{CB_f} + T_{VC_f} \\ &= T_{AD} \\ &\quad + \left(\frac{f+2}{4} * T_{RD} + \left\lceil \frac{f+2}{4} - 1 \right\rceil * T_{gate} \right) \\ &\quad + T_{IFC} \\ &\quad + T_{CB} + \left\lceil \frac{f+2}{4} - 1 \right\rceil * T_{gate} \\ &\quad + (\text{basic delay in VC}) \\ &\quad + \left\lceil \left(\frac{2^{f-1} + 2^{f-2}(f-1)}{3} - 1 \right) \right\rceil \\ &\quad * T_{gate} * 2 \end{aligned}$$

At data through:

$$\begin{aligned} T_{f\text{-}flat} &= T_{IFC} + T_{CB_f} + T_{VC_f} \\ &= T_{IFC} \\ &\quad + T_{CB} + \left\lceil \frac{f+2}{4} - 1 \right\rceil * T_{gate} \\ &\quad + T_{VC(3)} + \left\lceil \left(\frac{2^{f-1} + 2^{f-2}(f-1)}{3} - 1 \right) \right\rceil \\ &\quad * T_{gate}*2 \end{aligned}$$

Based on our cost model we can estimate the speed of routers with a range of adaptivity (see Figure 9). From these estimates, it is clear that router delay increases significantly with adaptivity, but not very rapidly. This is because each f-flat can be partitioned, requiring much smaller crossbars. For example, the CB in the 3-flat router is only 5×5. Instead, the number of virtual channels required for deadlock prevention the primary source of increased delay. The 3-flat adaptive router requires eight virtual channels just to prevent deadlock. The increased crossbar sizes and large numbers of virtual channels make routers with higher adaptivity much slower. The 3-flat router is 50% and the 4-flat is 190% slower than the planar-adaptive router at data through with much of the additional delay coming

[5]The *cnct* signals are used grant output port request from the ADs.

# of f-flat	The delay at *path setup*	Delay Ratio *path setup*	The delay at *data through*	Delay Ratio *data through*
1	5.7*ns*	0.7	3.0*ns*	0.5
2	10.3*ns*	1.0	5.7*ns*	1.0
3	14.6*ns*	1.4	8.8*ns*	1.5
4	17.1*ns*	1.7	16.5*ns*	2.9
5	29.8*ns*	2.9	26.1*ns*	4.6
6	56.5*ns*	5.5	51.9*ns*	9.1
7	114.9*ns*	11.2	109.5*ns*	19.2
8	243.8*ns*	23.7	237.9*ns*	41.7

FIGURE 9 Router delays with 1 to 8 degrees of routing freedom. Delay ratios are with respect to the base router (two-flat adaptive router).

from exponential increases in the numbers of virtual channels for deadlock prevention. These increases are much larger than the throughput benefits claimed by most adaptive routers, reducing or eliminating the overall performance benefits of high degrees of adaptivity.

6 THE COST OF VIRTUAL LANES

In this section, we characterize the cost of virtual lanes by examining router designs with from one to sixteen virtual lanes. Virtual lanes can increase channel utilization in a network by multiplexing the physical channels, allowing packets to pass one another [14, 26]. Though both virtual channels and virtual lanes use additional hardware buffers, virtual lanes require greater connectivity in the router as each virtual lane is interchangeable within its virtual channel class. Essentially, this means that the crossbars cannot be partitioned. In this section, we first consider the pros and cons of several proposed architectures for virtual lanes, then estimate the speed and cost of the most attractive architecture.

6.1 Architectural Alternatives for Virtual Lanes

In [14], Dally proposes three alternatives for implementing virtual lanes, which differ primarily in the size of the *crossbar switch* and how it is multiplexed (Figure 10). A 2-input, 2-output CB is used for illustrative purposes. Adding virtual lanes to a planar adaptive router would require beginning with a 4-input, 4-output CB. The basic options for adding m virtual lanes to a basic $p \times p$ crossbar are:

A. An $m\ p \times m\ p$ crossbar switch.

B. A fully multiplexed $p \times p$ crossbar switch.

C. A partially multiplexed $m\ p \times p$ crossbar switch.

Option A uses no multiplexing, adding crossbar ports for each virtual lane. Because there is no multiplexing, this approach requires the simplest control and arbitration. Option B shares the switch amongst the virtual lanes. Option C represents a compromise, providing inputs for each virtual lane, but sharing the CB outputs. The major distinguishing characteristics of these options are internal blocking and switching time for flits.

Internal Blocking Because the physical channels are often a critical limiting resource, router designs minimize internal blocking. Both options A and C are internally nonblocking. On the other hand, option B does have internal blockage (see Figure 11 for an example). Blocking on option B arises from the interaction of flow control and crossbar multiplexing, and can cause performance losses. Because of the importance of minimizing internal blocking, we rule out option B.

Switching Speed Switch multiplexing affects the critical path length for data through and thus the achievable router speed. Option B has been eliminated on the basis of blocking, so we consider options A and C. As shown in Figure 12, option A requires only one pass through the CB for each data transmission while option C requires several passes. In option A, the crossbar configuration is fixed, and the fixed connections operate identically to the base router. In option C, because the outputs are shared, the switch settings for each cycle are determined by which virtual channels will use the physical channels this cycle. Thus, the VC and IFC's must collaborate to control the switch based on data status information from the IFC's as well as the empty signals from downstream nodes. This approach requires three passes through the switch, the first to get the data

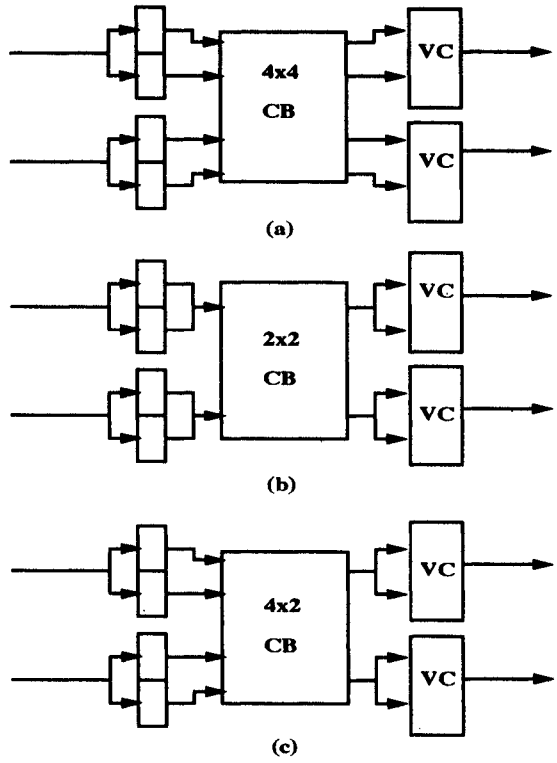


FIGURE 10 Three proposed architectures for virtual lanes.

status information to the VC's, the second to setup the switch and send the enable signals to the chosen IFC's and finally for the data to pass through the crossbar.

In addition, option C requires extra arbiters to manage the switch multiplexing; one for each switch output, managing the virtual lanes in a single virtual channel class (see Figure 13). These additional arbitration steps are more expensive for larger numbers of virtual lanes and increase not only the path setup time, but also the data through delay (switching speed), directly reducing network throughput.

Because latency and bandwidth are first priorities, option A is the most attractive. Gate count is not a major constraint for most router designs, and for modest dimension networks and virtual lanes, the required crossbar switches are feasible. For example, going from one virtual lane (base router) to two virtual lanes produces a router design with 8×8 crossbar switches and 6 input virtual channel controllers (see Figure 14). Alternatives which multiplex the crossbar switch may be more attractive for routers with large numbers of virtual lanes.

6.2 Performance of Routers with Virtual Lanes

In this section, we characterize the speed of routers supporting virtual lanes with architectural alternative option A. Adding virtual lanes requires minor modifications to the RD, CB and VC. First, the *RD* must connect messages to virtual lanes, not just physical channels. Second, adding one virtual lane requires VC's that can support six virtual channels (the former three virtual channels multiplied by two virtual lanes). Finally, the crossbar size is increased. A modified base router with two virtual lanes is shown in Figure 14. Comparing to Figure 3 shows the significant increase in complexity. Based our designs, the speed of a router with m virtual lanes can be estimated as follows:

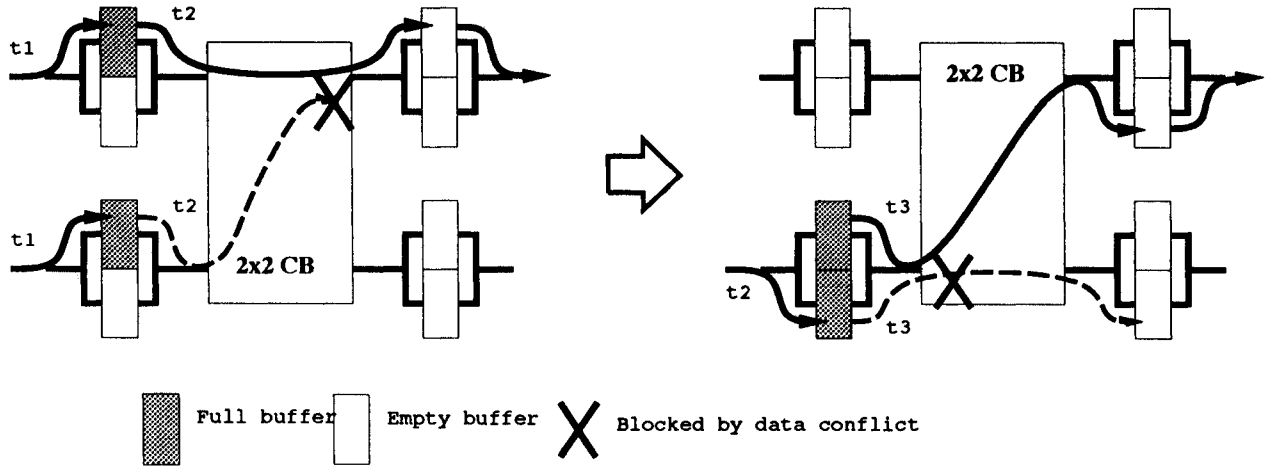


FIGURE 11 An internal data conflict using Option B. At time t3, one of the flits on the lower port is blocked because the CB input is busy.

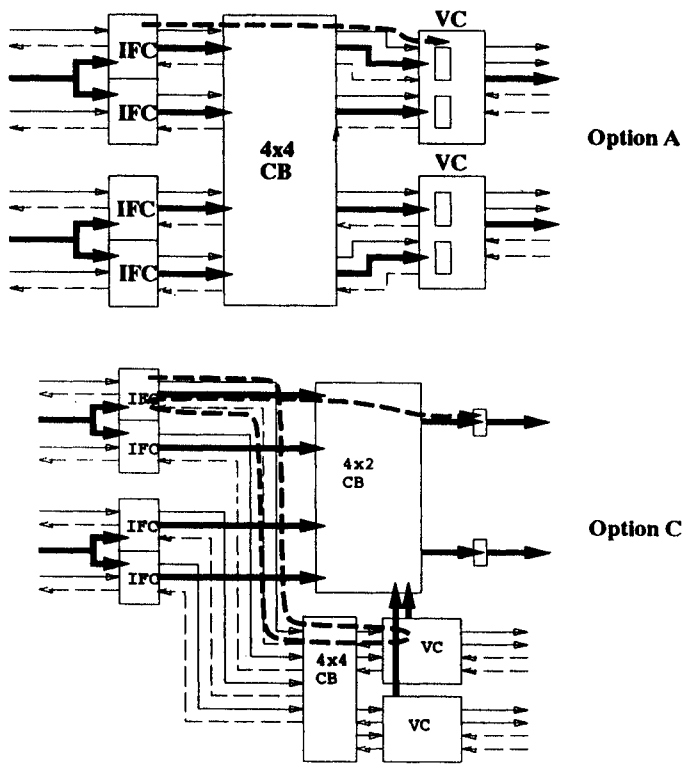


FIGURE 12 Critical paths for data through in Options A and C.

$$\begin{aligned}\text{Path setup} &= AD_{delay} + m * RD_{delay} \\ &\quad + IFC_{delay} \\ &\quad + CB_{delay} + (m-1) * T_{gate} \\ &\quad + VC_{delay} + (m-1) * T_{gate} \\ &= 3.3 + m * 2.1 + 2.6 + 1.1 \\ &\quad + (m-1) * 0.6 \\ &\quad + 1.2 + (m-1) * 0.6 \\ &= 7.0 + 3.3m\end{aligned}$$

$$\begin{aligned}\text{Data through} &= IFC_{delay} \\ &\quad + CB_{delay} + (m-1) * T_{gate} \\ &\quad + VC_{delay} + (m-1) * 2 * T_{gate} \\ &= 2.2 + 1.0 + (m-1) \\ &\quad * 0.6 + 2.5 + (m-1) * 1.2 \\ &= 3.9 + 1.8 * m\end{aligned}$$

The RD delay at path setup increases linearly in m, the number of virtual lanes, because there are m times more inputs to the crossbar switch. The CB and VC delay increase slowly based on the number of virtual lanes due to increasing depth of the switching and arbitration circuits. The second term in VC delay for path setup is a factor of two smaller than that for data through because of the different overlap at path setup (see section 5). These delays are summarized for a planar-adaptive router with a range of virtual lanes from one to sixteen in Figure 15. Clearly, adding virtual lanes significantly increases router setup and data through latency. For example, if data through latency determines throughput, going from one virtual lane to two virtual lanes requires more than a 30% throughput improvement to be worthwhile. In conjunction with measured performance benefits of virtual lanes, this indicates that only modest numbers of virtual lanes are likely to give performance benefits.

7 OVERALL COST SUMMARY

Our studies show clearly that adaptive routing and virtual lanes can have a significant impact on router delays. To relate these results to existing channel

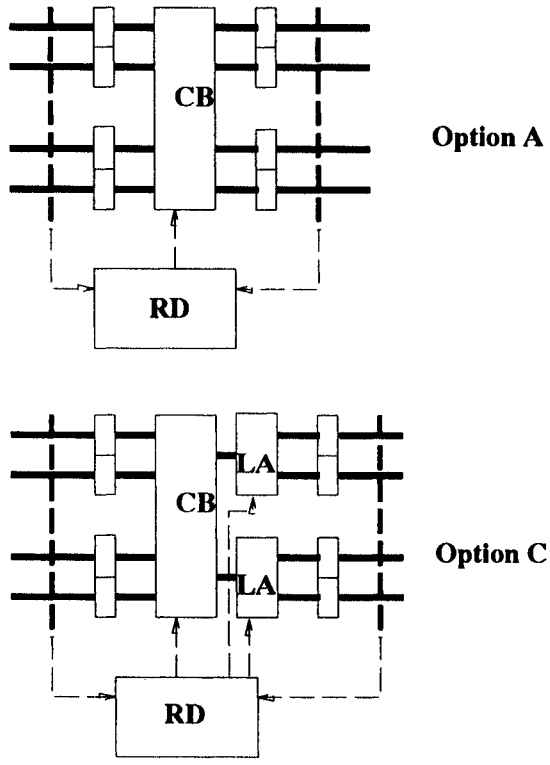


FIGURE 13 Arbiter requirements for Options A and C. Option A requires no additional arbiters while Option C requires a local arbiter (LA) for each set of virtual channels and uses the arbiters to control the crossbar switch.

utilization studies for routers, we translate the delays into achievable clock rates, allowing us to estimate how much better the channel utilization numbers would have to be to justify each type of router feature.

We convert the router delays to achievable network clock rates based on four assumptions: two phase clocking, equal length phases (approximately 50% duty cycle clocks), path setup in one and a half clock cycles, and the achievable clock rate determined by the delay at data through. The first two conditions match the intranode delay (at data through) and internode delay. The third condition assumes we can achieve a router setup latency of two clock periods. The final condition assumes a simple router architecture with identical flit and phit (physical transfer unit) size.

Figures 16 and 17 summarize the intranode delay, achievable clock rate, delay per hop and required channel utilization improvement rate of routers with f-flat adaptivity and m virtual lanes, respectively. The required channel utilization figures show the performance increase required to justify addition of the feature. The numbers in Figure 17 are based on adding virtual lanes to a planar-adaptive (2-flat) router.

Using the collected information, one can compare the cost of adaptivity and virtual lanes. Previous simulation studies show that network channel utilization benefits from a mix of the two features [25]. Our results show that adaptivity based on the Linder-Harden algorithm is more expensive than virtual lanes, and higher degrees of adaptivity based on this approach are probably not feasible. A 3-flat adaptive router incurs an increase in delay as large as a router with three virtual lanes. A 4-flat router's delay is nearly as large as a router with *seven* virtual lanes. In summary, routers with modest adaptivity and larger numbers of virtual lanes are most attractive. Further, higher degrees of adaptivity or large numbers of virtual lanes are probably not viable, as their cost-effectiveness depends on four-fold increases in channel utilization.

8 SUMMARY AND FUTURE WORK

In this paper, we have described the design of a planar-adaptive router, and used that design to analyze the cost of a basic adaptive router. Our router is internally synchronous and externally asynchronous. Based on a 0.8 micron gate array technology, we characterized the speed of our design. The intranode router delay is 10.3*ns* and 5.7*ns* for path setup and data through respectively, supporting a maximum signalling rate of 87 Mhz or 174 MB/s per physical channel (sixteen-bit channels). Our design could be improved by using a single-phase clock and edge-triggered latches, potentially raising performance to 348 *MB/sec.*

Using the planar-adaptive router design as a baseline, we explored the cost of adaptivity and virtual lanes. Our studies show that higher degrees of adaptivity can be extremely expensive. Justifying the increased cycle time due to adaptivity requires that it deliver huge increases in channel utilization. For example, to justify an increase in adaptivity from a two-flat to a 3-flat router requires a 50% improvement in channel utilization. To justify an increase from a two-flat to four-flat router the improvement must be extremely large, 190%. Simulation studies show that improvements in channel utilization due to adaptive routing are likely to be more modest. Consequently, only low degrees of adaptivity are attractive.

Our studies show that virtual lanes are less expensive than adaptivity, but still quite expensive. To justify the increased cycle time, the first additional virtual lane must provide at least a 30% increase in

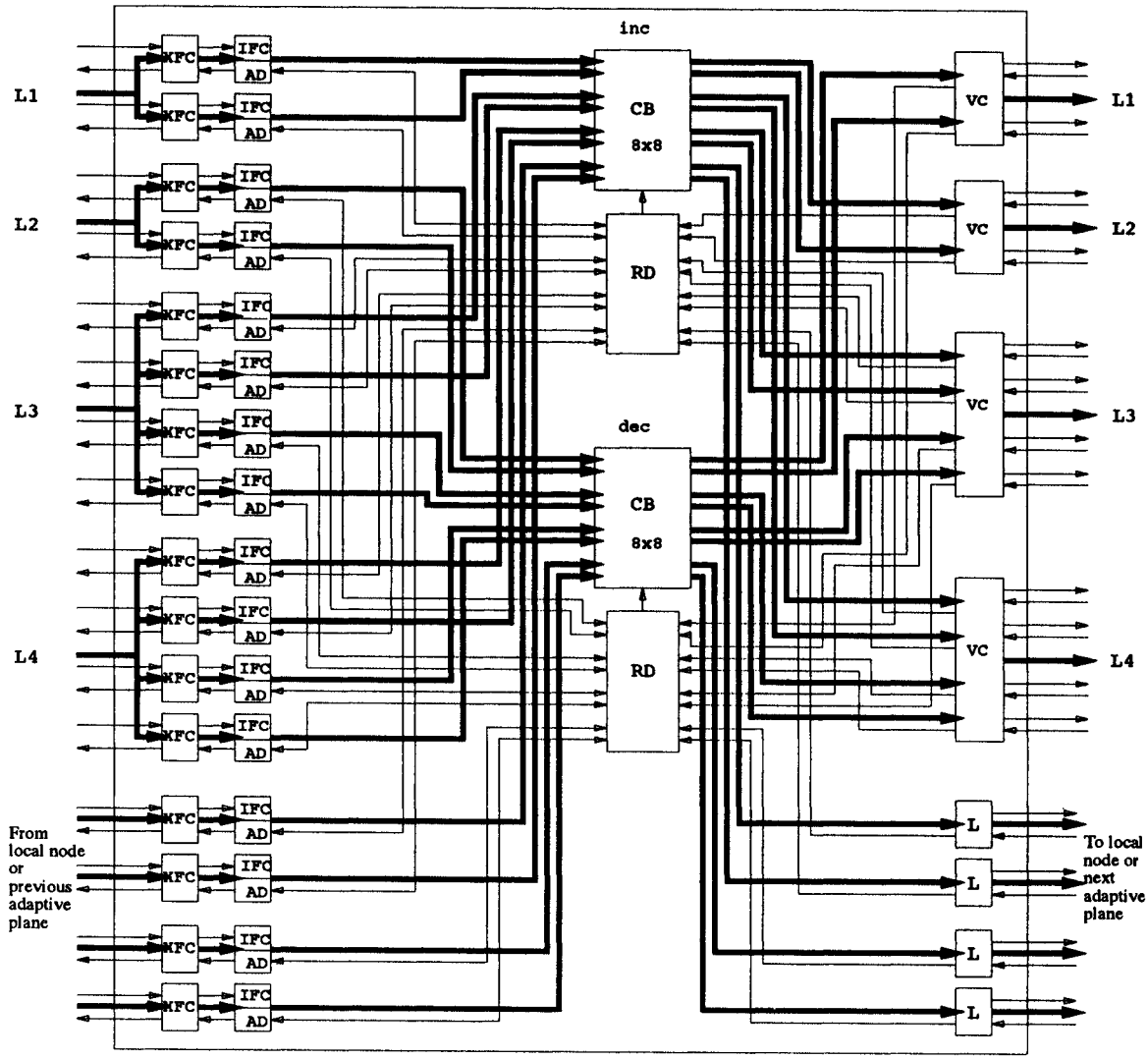


FIGURE 14 A 2-dimensional planar-adaptive router with two virtual lanes.

# of virtual lanes	The delay at *path setup*	Delay Ratio *path setup*	The delay at *data through*	Delay Ratio *data through*
1	10.3*ns*	1.0	5.7*ns*	1.0
2	13.6*ns*	1.3	7.5*ns*	1.3
3	16.9*ns*	1.6	9.3*ns*	1.6
4	20.2*ns*	2.0	11.1*ns*	1.9
8	33.4*ns*	3.2	18.3*ns*	3.2
12	46.6*ns*	4.5	25.5*ns*	4.5
16	59.8*ns*	5.8	32.7*ns*	5.7

FIGURE 15 The delay of planar-adaptive routers with from one to sixteen virtual lanes.

# of f-flat	Internal delay Data Through	Achievable clock rate	Reqd Util. Improvement	# of virt. channels per link
1	3.0*ns*	102.0 MHz	0.86	1
2	5.7*ns*	87.7 MHz	1.0	3
3	8.8*ns*	56.8 MHz	1.5	8
4	16.5*ns*	30.3 MHz	2.9	20
5	26.1*ns*	19.2 MHz	4.6	48
6	51.9*ns*	9.6 MHz	9.1	112
7	109.5*ns*	4.6 MHz	19.2	2240
8	237.9*ns*	2.1 MHz	41.7	8640

FIGURE 16 The delay, achievable clock rate, delay per hop and required channel utilization improvement for routers with from one to eight-flat adaptivity.

channel utilization. Justifying second and third virtual lanes require 30% further increases in channel utilization for each. Published simulations studies show that such increases are possible for a modest number of virtual lanes, but performance increases sufficient to justify larger numbers of virtual lanes appear unlikely [14, 25]. Consequently, small numbers of virtual lanes should produce the best performance. In summary, routers with modest adaptivity and larger numbers of virtual lanes are most attractive.

By examining the implementation complexity of adaptive routing and virtual lanes, we seek to balance their cost and benefit. While much research has been published on the advantages of these features, we hope to provoke debate on their cost and real benefits. For example, some proponents of adaptive routing have claimed lower latency at low loads as a performance advantage. Our design studies show that increases in router complexity and intrarouter latency are likely to overwhelm such benefits. Both adaptive routing and virtual lanes can increase network throughput, but our design studies show that their complexity can produce compensating reductions in network bandwidth. Measuring the cost of network features allows us to weigh their benefit against their cost and make informed tradeoffs.

There are still many avenues open for future work in this area. Though our design presents a basic evaluation of the cost of adaptivity and virtual lanes, it is based on a single technology point and a particular class of router architectures. Other technology points and router architectures should be examined to see if they give qualitatively different results. This study also examined basically one approach to routing, others studies of this type [11] will certainly explore the cost of alternative approaches to adaptive routing. The ultimate goal is to integrate the optimization of concerns to include routing algorithm, network topology, routing freedom, and virtual lanes, allowing the major choices which affect network cost and performance to be related in a global perspective on network design.

Acknowledgments

This paper has benefited greatly from careful readings by Jae Kim, Ziqiang Liu, and other members of the Concurrent Systems Architecture Group. The research described in this document and the authors of this paper are supported in part by grants from the National Science Foundation, grants CCR-9209336 and MIP-92-23732, Office of Naval Research, grant N00014-92-J-1961, and the National Aeronautics and Space Administration, grant NAG 1-613. Additional support has been provided by a generous special-purpose grant from AT&T Foundation.

References

[1] A. Agarwal et al. The mit alewife machine: A large-scale distributed-memory multiprocessor. Lcs-technical memo

# of virtual lanes	Internal delay Data Through	Achievable clock rate	Reqd Util. Improvement	# of virt. channels per link
1	5.7*ns*	87.7 MHz	1.0	3
2	7.5*ns*	66.7 MHz	1.3	6
3	9.3*ns*	53.8 MHz	1.6	9
4	11.1*ns*	45.0 MHz	1.9	12
8	18.3*ns*	27.3 MHz	3.2	24
12	25.5*ns*	19.6 MHz	4.5	36
16	32.7*ns*	15.3 MHz	5.7	48

FIGURE 17 The delay, achievable clock rate, delay per hop and required channel utilization improvement for planar adaptive (2-flat) routers for a range of virtual lanes.

454, Massachusetts Institute of Technology, Laboratory for Computer Science, 1991.

[2] A. Agarwal. Limits on interconnection network performance. *IEEE Transactions on Parallel and Distributed Systems,* 2(4), 398–412, 1991.

[3] G. Alverson, R. Alverson, D. Callahan, B. Koblenz, A. Porterfield, and B. Smith. Exploiting heterogeneous parallelism on a multithreaded multiprocessor. In *Proceedings of the 6th ACM International Conference on Supercomputing,* 1992.

[4] K. Aoyama. Design issues in implementing an adaptive router. Master's thesis, University of Illinois, Department of Computer Science, 1304 W. Springfield Avenue, Urbana, Illinois, January 1993.

[5] P. Berman, L. Gravano, G. Pifarre, and J. Sanz. Adaptive deadlock and livelock free routing with all minimal paths in torus networks. In *Proceedings of the Symposium on Parallel Algorithms and Architectures,* 1992.

[6] K. Bolding, S. Cheung, S. Choi, S. Hassoun, T. Ngo, and R. Wille. The chaos router chip: Design and implementation of an adaptive router. In *Proceedings of the Grenoble Conference on VLSI,* 1993.

[7] R. Boppana and S. Chalasani. A comparison of adaptive wormhole routing algorithms. In *International Symposium on Computer Architecture,* 1993.

[8] S. Borkar, R. Cohn, G. Cox, T. Gross, H. T. Kung, M. Lam, M. Levine, B. Moore, W. Moore, C. Peterson, J. Susman, J. Sutton, J. Urbanski, and J. Webb. Supporting systolic and memory communication in iWarp. In *Proceedings of the 17th International Symposium on Computer Architecture.* IEEE Computer Society, 1990.

[9] A. A. Chien and J. H. Kim. Planar-adaptive routing: Low-cost adaptive networks for multiprocessors. In *Proceedings of the International Symposium on Computer Architecture,* pages 268–77, May 1992.

[10] A. A. Chien and J. H. Kim. Planar-adaptive routing: Low-cost adaptive networks for multiprocessors. *Journal of the Association for Computing Machinery,* to appear.

[11] Andrew A. Chien. A cost and performance model for k-ary n-cube wormhole routers. In *Proceedings of Hot Interconnects Workshop,* August 1993.

[12] W. Dally and C. Seitz. Deadlock-free message routing in multiprocessor interconnection networks. *IEEE Transactions on Computers,* C-36(5), 1987.

[13] W. J. Dally. Performance analysis of k-ary n-cube interconnection networks. *IEEE Tranactions on Computers,* 39(6), June 1990.

[14] W. J. Dally. Virtual channel flow control. *IEEE Transactions on Parallel and Distributed Systems,* 3(2), 194–205, 1992.

[15] W. J. Dally, S. Ahmed, P. Carrick, A. Chien, R. Davison, J. Fiske, G. Fyler, W. Horwat, J. Keen, S. Lear, R. Lethin, M. Vestrich, T. Nguyen, M. Noakes, P. Nuth, and D. Wills. Design and implementation of the message-driven processor. In *Proceedings of the 1992 Brown/MIT Conference on Advanced Research in VLSI and Parallel Systems,* T. Knight and J. Savage, eds., pages 5–25. MIT Press, 1992.

[16] W. J. Dally and H. Aoki. Deadlock-free adaptive routing in multicomputer networks using virtual channels. *IEEE Transactions on Parallel and Distributed Systems,* 4(4), 466–74, April 1993.

[17] W. J. Dally, A. Chien, S. Fiske, W. Horwat, J. Keen, M. Larivee, R. Lethin, P. Nuth, S. Wills, P. Carrick, and G. Fyler. The J-Machine: A fine-grain concurrent computer. In *Information Processing 89, Proceedings of the IFIP Congress,* pages 1147–1153, August 1989.

[18] W. J. Dally and C. Seitz. The torus routing chip. *Distributed Computing,* pages 187–96, 1986.

[19] J. Duato. On the design of deadlock-free adaptive routing algorithms for multicomputers: design methodologies. In *Proceedings of Parallel Architectures and Languages Europe,* 1991.

[20] Dave Dunning. The intel paragon router. VLIS Seminar Talk at the Massachusetts Institute of Technology, 1992.

[21] Charles M. Flaig. Vlsi mesh routing systems. Master's thesis, California Institute of Technology, Department of Computer Science, May 1987.

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

[23] J. Kim, Z. Liu, and A. Chien. Compressionless routing (cr). In *Proceedings of the International Symposium on Computer Architecture,* 1994.

[24] J. H. Kim. Planar-adaptive routing: Low-cost adaptive networks for multiprocessors. Master's thesis, University of Illinois, Department of Electrical and Computer Engineering, January 1993.

[25] J. H. Kim and A. A. Chien. An evaluation of planar-adaptive routing (par). In *Proceedings of the Fourth Symposium on Parallel and Distributed Processing,* December 1992.

[26] J. H. Kim and A. A. Chien. Evaluation of wormhole routed networks under hybrid traffic loads. In *Proceedings of the Hawaii International Conference on System Sciences,* January 1993.

[27] D. Lenoski, J. Laudon, K. Gharacharloo, W. Weber, A. Gupta, J. Hennessy, M. Horowitz, and M. Lam. The stanford dash multiprocessor. *IEEE Computer,* March 1992.

[28] Daniel Lenoski and et al. The Stanford DASH Multiprocessor. *IEEE Computer,* pages 63–79, Mar 1992.

[29] D. Linder and J. Harden. An adaptive and fault tolerant wormhole routing strategy for k-ary n-cubes. *IEEE Transactions on Computers,* C-40(1), 2–12, January 1991.

[30] C. Mead and L. Conway, *Introduction to VLSI Systems,* chapter Highly Concurrent Systems, pages 263–92, Addison-Wesley, 1980.

[31] NCUBE, Beaverton, Oregon. *NCUBE 2 6400 Series Supercomputer: Technical Overview,* 1989.

[32] L. Ni and C. Glass. The turn model for adaptive routing. In *Proceedings of the International Symposium on Computer Architecture,* 1992.

[33] P. Nuth and W. Dally. The j-machine network. In *Proceedings of the 1992 IEEE International Conference on Computer Design: VLSI in Computers and Processors,* pages 420–23, 1992.

[34] S. Lillevik, Intel Corporation. Touchstone program overview. In *Proceedings of the Fifth Distributed Memory Computers Conference,* 1990.

[35] C. Seitz and W. Su. A family of routing and communication chips based on the mosaic. In *Proceedings of the University of Washington Symposium on Integrated Systems,* 1993.

Biographies

KAZUHIRO AOYAMA received the M.Eng. degree in instrumentation engineering from Keio University, Tokyo, Japan, in 1985, and the M.S. degree in computer science from the University of Illinois at Urbana-Champaign, in 1993. Since April 1985, he has been with Mitsubishi Electric Corporation, Kamakura, Japan, where he is currently designing parallel computers. He is a member of the Institute of Electronics, Information and Communication Engineers. Electronic mail to *aoyama@kama.melco.co.jp*

ANDREW A. CHIEN received his B.S. degree in Electrical Engineering (1984) and his M.S. (1987) and Ph.D. (1990) degrees in Computer Science all from the Massachusetts Institute of Technology. He is currently an Assistant Professor in the Department of Computer Science at the University of Illinois where he leads the Concurrent Systems Architecture Group. His research interests involve the interaction of architecture, system software, compilers, and programming languages in high-performance parallel systems. Andrew Chien has authored numerous research papers in those areas and recently published *Concurrent Aggregates: Supporting Modularity in Massively-Parallel Programs* with MIT Press. Electronic mail to *achien@cs.uiuc.edu*