Appears in the Proceedings of Hot Interconnects II, August 1994
Stanford University, Palo Alto, CA

Architecture and Implementation of the Reliable Router [1]

William J. Dally, Larry R. Dennison, David Harris, Kinhong Kan and Thucydides Xanthopoulos.
Artificial Intelligence Laboratory
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139

Abstract *The Reliable Router (RR) is a network switching element targeted to two-dimensional mesh interconnection network topologies. It is designed to run at 100 MHz and reach a useful link bandwidth of 3.2 Gbit/sec. The Reliable Router uses adaptive routing coupled with link-level retransmission and a unique-token protocol to increase both performance and reliability. The RR can handle a single node or link failure* anywhere *in the network without interruption of service. Other unique features include a queueless low-latency plesiochronous channel interface and simultaneous bidirectional signalling.*

1 Introduction

Interconnection networks significantly affect the performance and reliability of massively parallel processors (MPPs). Previous work on network switching elements which employed oblivious routing such as the J-Machine Router [2] and the Caltech Mesh-Routing Chips [11] did not address the issue of reliability in part because of the inherent non-adaptivity of oblivious routing. Past work on adaptive elements such as the Chaos Router [9] exploited adaptivity for performance reasons only without providing fault handling mechanisms.

The Reliable Router (RR) exploits adaptive routing for both performance and reliability purposes. It also has mechanisms for continuous link monitoring and link-level retransmission when a link parity error is detected. It also employs a forwarding protocol at the flit level that facilitates packet reconstruction and duplicate detection at the receiving end when a fault occurs. We call this protocol the Unique Token Protocol (UTP) [5].

The coupling of these features (adaptive routing, link monitoring, link-level retransmission and the UTP) enable the RR to handle a single node or link failure *anywhere* in the network without interruption of service.

The Reliable Router avoids the problem of global clock distribution and skew management by using a queueless low-latency plesiochronous channel interface. This mechanism allows each RR chip to be clocked by its own local clock without the need for FIFO buffering on either side of the channel. The penalty is 0.1% of link bandwidth. Plesiochronous timing is also a reliability feature because there is no single point of failure in the clock net.

The RR solves the high pinout problem by using current-mode simultaneous bidirectional signalling [6]. This method has reduced the chip signal pin count from 311 signal pins to 215. This is a saving of 96 signal pins over a conventional signalling mechanism.

2 Architectural Description

The Reliable Router is designed for two-dimensional mesh topologies. Its organization is shown in Figure 1. There is one Input Controller and one Output Controller for every direction. Moreover, there is a processor input/output and a diagnostic input/output. Communication between an input and an output port occurs through a crossbar switch. The switch is a full crossbar and allows each Input Controller to connect to every Output Controller. There are actually some restrictions regarding the possible connections imposed by the routing relation.

2.1 Major Object Types

Three types of objects are handled by the architecture. The *packet* is the main unit of information exchange between the sending and receiving end. The Reliable Router can handle packets of arbitrary size. Virtual channels [4] are allocated on a packet basis.

Every packet is broken into 75-bit *flits*. Buffering, forwarding and flow control within the system is performed at the flit level (wormhole routing). Crossbar bandwidth and physical channels are also allocated at the flit level. The first flit of each packet is called *head* flit and contains the address of the packet destination. The format of RR head flits is shown in Table 1.

[1]This research is supported by the Advanced Research Projects Agency under contracts N00014-88K-0738, F19628-92-C-0045, and N00014-91-J-1698.

Bit Field	63:12	11	10	9:5	4:0
Contents	User Info	Priority	Diagnostic	Address in y	Address in x

Table 1: Head Flit Format

Bit Field	22	21	20:18	17	16	15:0
Frame 0	PE	USR0	VCI	BP1	BP0	Data [15:0]
Frame 1	Copied Kind		Copied VCI	BP3	BP2	Data [31:16]
Frame 2	U/D	USR1	Kind	BP5	BP4	Data [47:32]
Frame 3	Freed			BP7	BP6	Data [63:48]

Table 2: Frame Format

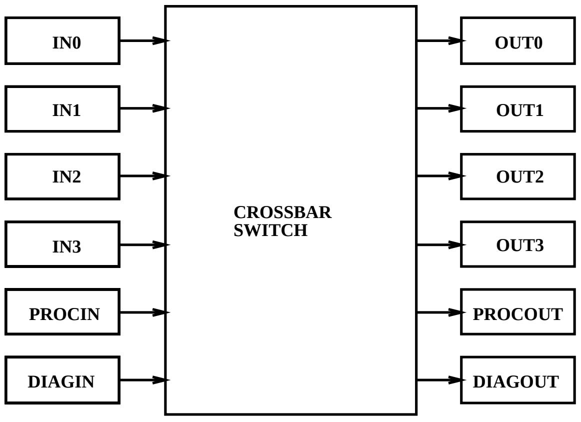

Figure 1: Organization of the Reliable Router

Subsequent flits are of type *data* and carry user data. The final flit of the packet is of type *tail* and marks the end of the packet. There is also a flit of type *token* that is injected at the end of each packet to implement the Unique Token Protocol. Each flit contains 64 bits of user data, 8 byte-parity bits for end-to-end error detection, and 3 bits that indicate the flit type.

Flits are further broken into *frames* so that they can be transmitted across physical channel links which are only 23 bits wide. Each flit is decomposed into four separate frames as shown in Table 2.

In addition to the data, byte parity (BP7-BP0) and kind bits (Kind), the four frames also include the virtual channel identifier (VCI), flow control information (Copied Kind, Copied VCI, Freed), link status information (U/D, PE) and two user bits (USR1, USR0).

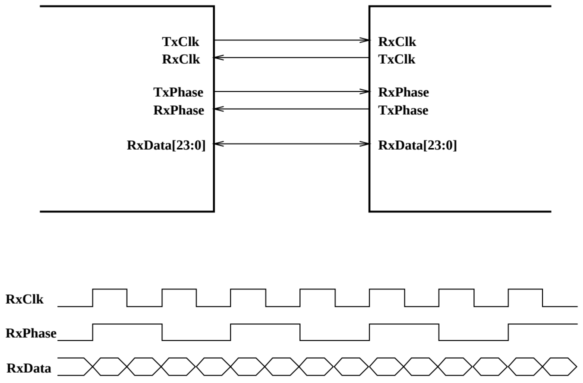

Figure 2: Channel Interface

2.2 Channel Interface Architecture

The channel interface architecture between two adjacent routers is shown in Figure 2.

The transmitting node drives the simultaneous bidirectional 24-bit *RxData* lines (23 data + 1 parity) and also passes its clock *TxClk* to the receiver. The frames are driven on the data lines on both clock edges. The transmitter needs to send the *TxPhase* pulse as well so that the receiver can distinguish which frame (0-3) of the flit is being received and be able to reassemble the frames and produce the transmitted flit. The *TxPhase* pulse is produced by toggling a flip-flop on the positive edge of *TxClk* and making sure that it has the same timing as the *RxData* lines.

The receiver samples the incoming frames using the transmitter clock. After having assembled all four frames into a flit, the receiver decides whether or not to accept the assembled block based on whether

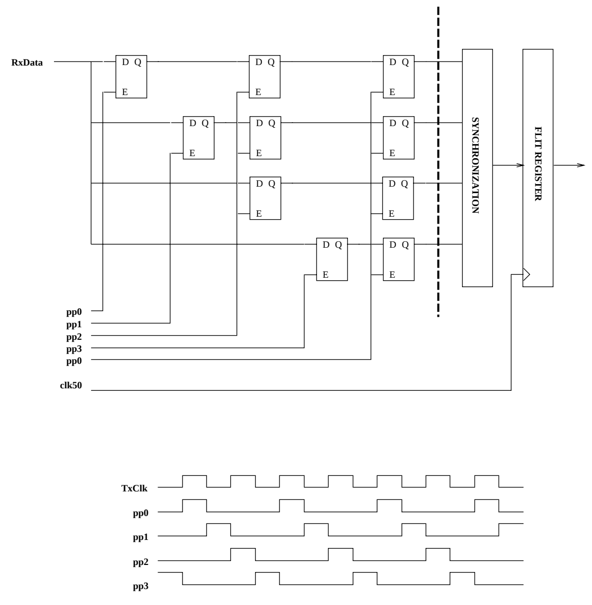

Figure 3: Flit Assembly at the Receiver.

a parity error occurred during the reception of all frames. If such an error occurred, the flit is discarded and the link is marked as faulty. Otherwise, the flit is synchronized in the receiver clock domain and then passed to the rest of the system. The receiver assembles each incoming flit using the ranks of level-sensitive latches shown in Figure 3.

Waveforms *pp0, pp1, pp2, pp3* are produced by delaying and gating *TxClk* and *TxPhase*. Flit assembly costs two clock cycles as shown in the figure. Yet, this firewall of protection is necessary for reliability purposes. The rest of the system should not be allowed to look at parts of the received flit, unless all the parts have been received without any errors. The block labeled *Synchronization* implements clock domain crossing in a very efficient fashion. The latency penalty is 0 cycles in the best case and 1 clock cycle in the worst case for an average of 0.5 cycles. The final stage of this module involves sampling the whole flit in the receiving clock domain. The *clk50* pulse has half the frequency of the local *TxClk*, since a new flit is received only once every two cycles.

We name the structure of Figure 3 the Front End.

2.3 Functional Description

2.3.1 Terms and Definitions

The following terminology will be used throught the rest of the architectural description:

Flit Time The time that a flit stays in the Flit Register in Figure 3 before being replaced by the next received flit. This period of time is equal to two clock periods (20 ns assuming a 100 MHz clock.)

Routing Problem The result of comparing the address of the packet destination in the head flit with the local node address. The Routing Problem contains all the necessary information for a decision regarding which way to take the next hop.

Route (noun) The result of a routing computation. The route consists of an Output Controller Identifier (0-5 including the processor output and the diagnostic output) and an Output Virtual Channel Identifier (0-4).

Most of the functionality of the chip has been pushed into the Input Controller. A simplified block diagram of the Input Controller is shown in Figure 4. This block diagram closely matches the current layout. The Input Controller supports five separate virtual channels with decoupled resources. Virtual channels are used to increase performance and also ensure deadlock freedom. The functionality of the Input Controller can be summarized as follows:

- It buffers flits in the FIFO module. The FIFO is divided into five separate banks, one for every virtual channel. Each bank is 16-flits deep. Each bank behaves as a regular first-in-first-out buffer along with some special state and functionality to implement backtracking and retransmission in case of a fault.

- It computes the next step route of each packet based on head flit information and current output virtual channel state. The route is stored in dedicated registers so that it can be used by subsequent data flits. Route computation and storage resources reside in each Virtual Channel Module.

- It picks one of the five virtual channels to drive a flit across the crossbar. This functionality resides in the Virtual Channel Select Module.

- It keeps track of the virtual channel buffer size in the receiving node and stops the corresponding virtual channel from transmitting any more flits across the crossbar in order to prevent FIFO overruns.

Flits enter the Input Controller after being assembled and synchronized in the Front End shown in Figure 3.

If the flit is a head flit and its data payload contains routing information, it goes through the Compute Routing Problem Module which processes the header and produces a Routing Problem as defined above. The Routing Problem is fed through the corresponding Virtual Channel Module and a copy of the original flit is stored in the head position of the FIFO bank that corresponds to the virtual channel identifier of the incoming flit. Dedicated logic within the Virtual Channel Module calculates a possible next step Route for the packet that depends on the Routing Problem and on current availability of output virtual channels. This routing decision is not final and we call it "optimistic". The associated block of combinational logic that computes the route is called the "Optimistic Router." If another Input Controller wants to route an incoming head flit to the same Output Controller, then the Crossbar Allocator has to decide which one of the two head flits is allocated the resource.

If, on the other hand, the incoming flit is of type data it does not go through the Routing Problem and Route computation. It is just stored in the appropriate FIFO bank waiting to be popped and forwarded to the next node.

Every Flit Time a decision must be made as to which FIFO Bank – virtual channel – gets a chance to push a flit through the crossbar. A round robin scheduler picks among the eligible virtual channels. A virtual channel is considered eligible in two cases:

Routed case: If the next flit of the channel is data or tail and a route has already been established by assigning an output virtual channel, then the virtual channel is eligible if there are available flit buffers in the neighbor node.

Unrouted case: If the next flit is a head and the Optimistic Router indicates that a possible next step route can be computed based on channel state information at that time, then the virtual channel is eligible to bid for crossbar bandwidth.

The selection of the virtual channel to be popped is made by the Virtual Channel Select module. The selected virtual channel (Read VC) is used to read the appropriate FIFO bank and also to select the appropriate output controller identifier to be submitted to the crossbar allocator. This selection is done through the Route Multiplexer. When a flit is picked to be pushed through the crossbar, the new virtual channel identifier which is computed as part of the next step route is appended by the Flit Munger.

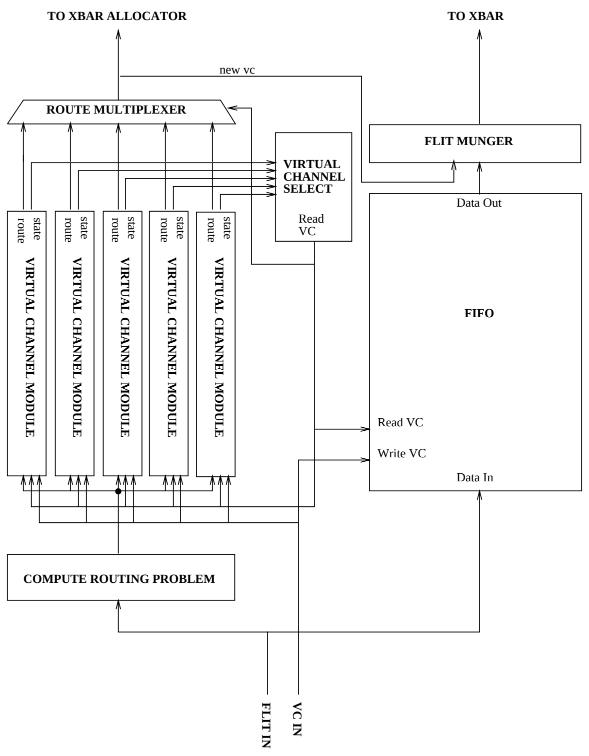

Figure 4: Input Controller Block Diagram.

Each Output Controller simply appends certain acknowledgment information, computes parity, breaks the flit into four frames and transmits them across the physical link.

2.3.2 The Virtual Channel Module

The Virtual Channel Module is shown in Figure 5. Both the Routing Problem module and the Route module are level-sensitive latches used to store the Routing Problem and the Route respectively of the packet occupying the virtual channel. The Route must be stored so that it can be used by subsequent data flits that do not carry the destination address. The Routing Problem must be stored because it may be reused if at the time the head flit arrived, a Route

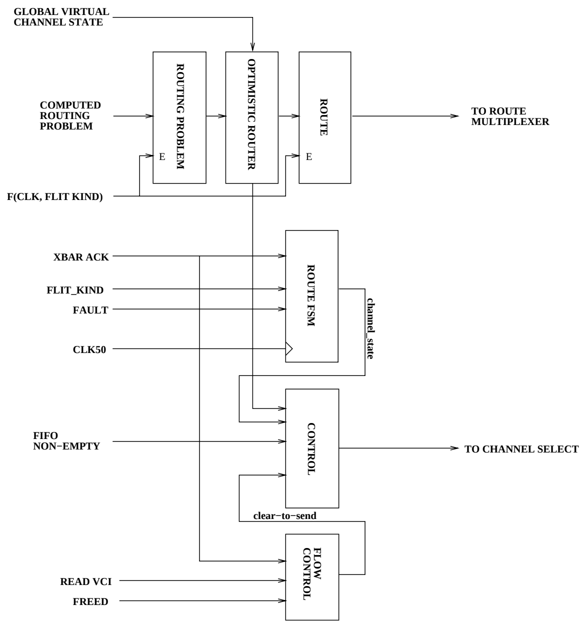

Figure 5: Virtual Channel Module.

could not be computed because appropriate output virtual channels were not available.

Both of these latches are put in transparent mode at the same time. The common enable of these latches goes high half a clock cycle (5 ns) after the beginning of the Flit Time period and gets deasserted half a clock cycle before the end of the Flit Time. This design allocates a maximum of 10 ns to the computation of the Route within the Optimistic Router block. Level-sensitive latches have been used to budget computation time more efficiently among the combinational modules and allow the Crossbar Allocator to see as early as possible the Route coming out of the Input Controller.

This design generates the Routing Problem, computes the Route, selects a virtual channel and allocates the crossbar within 20 ns (2 cycles.)

The rest of the circuitry in the Virtual Channel Module is mainly concerned with internal bookeeping. A finite state machine which we call the Route FSM keeps track of the virtual channel state at all times. A simplified version of its state diagram is shown in Figure 6. Virtual channels are initialized to the IDLE state. When a head flit appears in the Flit Register, the channel state switches to either NEEDS ROUTE or ROUTED state depending on whether the flit was actually accepted by the crossbar (assertion of the *xbar ack* signal). A channel is marked as routed only when the head flit succeeds

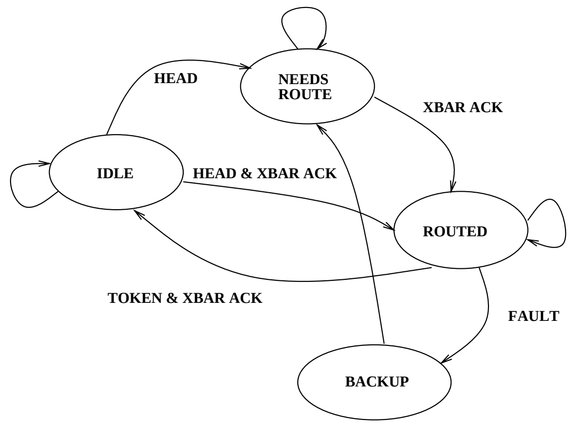

Figure 6: State Diagram of the Route FSM.

in reserving an output controller and gets transmitted across the crossbar. Otherwise, the Optimistic Router will try to recompute a Route during the next Flit Time. The virtual channel switches back to the IDLE state when a token flit goes through and the resource is deallocated. If a fault occurs while the message is ROUTED, the channel switches to the BACKUP state where internal preparation for retransmission occurs. After a Flit Time, the channel goes back to the NEEDS ROUTE state and starts looking for an alternate route among the non-faulty output virtual channels.

The Flow Control Module contains a counter that is incremented each time a data flit goes across the crossbar into the virtual channel buffer storage of the neighboring node and gets decremented each time the neighboring node forwards a flit and frees up buffer storage. This information propagates to the original node in the form of a *freed* acknowledgment shown in Figure 5 and on Table 2. When there is no available buffer space in the receiving router, the Flow Control Module deasserts the *clear-to-send* signal.

Finally, the Control Module collects information from the Optimistic Router, the Route FSM and the Flow Control Module and decides whether it should declare the virtual channel eligible (as described in the previous section) to transmit a flit across the crossbar. A channel is considered eligible if the FSM indicates that the channel is routed, the FIFO indicates that the channel is non-empty and the *clear-to-send signal* is asserted. Moreover, a channel is also considered eligible if it is not routed but the Optimistic Router indicates that a route can be successfully computed based on available output channel status at the time.

It is the Virtual Channel Select Module shown in Figure 4 which decides which one of the five virtual channels can bid for crossbar bandwidth based on information supplied by the five separate Control modules.

2.3.3 Crossbar Allocator

The Crossbar Allocator is responsible for resolving conflicts among Input Controllers that want to use the same Output Controller. An Input Controller can submit a bid to the Allocator in one of 3 priorities. The Allocator makes a random decision among Input Controllers within each priority level. The first two priority levels correspond to user packet priority. The third (highest) priority level is reserved for system use. Priority 3 is set after an Input Controller has been refused Crossbar bandwidth seven times. In this way, starvation is prevented.

2.4 Architectural Support for Virtual Channels

The Reliable Router has five virtual channels associated with every physical link. Managing five virtual channels at the architectural level may require extra levels of arbitration for allocation of shared resources. Such serialization can cause performance degradation. Early in the architectural design, it was decided to replicate one of the most sensitive resources: The Optimistic Router logic. An Optimistic Router was placed in each one of the five Virtual Channel Modules. This decision has eliminated the arbitration for the router along with inefficiencies associated with picking a virtual channel that could not be routed. With just one copy of the routing logic there was no way to determine whether a particular channel could be routed before feeding its head flit through the router.

The only place in the RR where serialization of virtual channels occurs is when the non-idle virtual channels of an Input Controller compete for physical link bandwidth through the crossbar switch. In this case, a round robin scheduler which picks only non-idle virtual channels ensures a fair and efficient arbitration. This scheduler resides in the Virtual Channel Select module.

3 Latency and Throughput

The latency through a Reliable Router ranges from 7 to 8 cycles. A cycle count is shown on Table 3. Cycle 2 may not be incurred in the best case as mentioned in a previous section. The horizontal line after cycle 2 indicates domain crossing from transmitter clock to receiver clock. Flit Assembly costs two cycles, and plesiochronous timing costs another one to two cycles. Yet, it was decided to pay the extra price in terms of latency and gain in reliability and scalability. The actual computation in the chip takes only two cycles. This is impressive, given that it is a fully adaptive router with virtual channels and retransmission features.

The projected clock period is 100 MHz which gives a worst case latency of less than 80 ns.

The datapath of the chip is quite wide and each link can achieve a useful unidirectional bandwidth of 3.2 Gbit/sec – 16 user data lines changing on both edges of a 100 MHz clock.

4 Adaptive Routing

Past research on adaptive routing has suffered from exponential dependence of resources on network dimension [10], carrying and updating message state along the route [3], and ad hoc fault-handling mechanisms incapable of handling faults along the edges of the network [10], [1]. The RR routing algorithm minimizes resource requirements and message state by using Duato's method [7]. The fault-handling properties of the routing algorithm are decoupled from the adaptive properties by using a different set of virtual channels and a different adaptive algorithm – the Turn Model [8] – for fault-handling. One can think of a network of RRs as the superposition of three separate virtual networks:

A minimally adaptive network. A packet using this network is able to route to any productive channel – a channel which will bring it closer to its destination. This virtual network tries to exploit adaptivity for performance reasons and is susceptible to deadlocks. An example message trace in such a network is shown in Figure 7 (a). The RR allocates two virtual channels to the adaptive network.

A dimension-ordered network. Packets in this network are routed in strict dimension order: In every dimension d a packet is routed along that dimension

Cycle No.	Description
0	Flit Assembly
1	Flit Assembly
2	Synchronization (worst case)
3	Sampling in local clock domain
4	Routing problem and Route computation
5	Route Computation, Channel Selection and Crossbar Allocation
6	Transmission across Crossbar and Parity Computation
7	Fragmentation and Transmission off Chip

Table 3: Cycle Count Through the System.

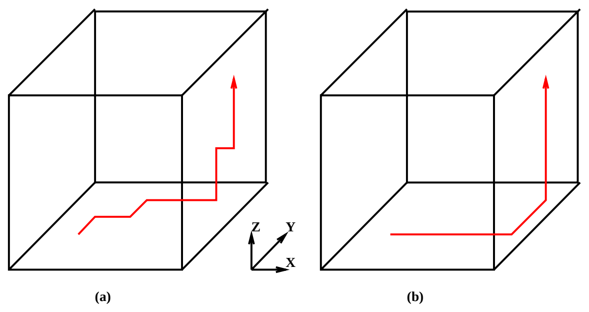

Figure 7: Minimally adaptive vs. dimension-ordered routing.

until it reaches a node whose address in dimension d matches the address of the packet destination node in the same dimension. If the addresses match, then the packet continues to route in the next lower dimension where the current channel address and the destination address differ. An example of dimension-ordered routing is shown in Figure 7 (b). The order of dimensions in this example is x, y, z. Dimension-ordered routing is provably deadlock-free [4]. This virtual network exists in order to break deadlocks introduced in the previous network as suggested in [7]. The RR has two dimension-ordered virtual channels – one used by packets of priority 0 and one used by packets of priority one. Two packet priorities have been implemented with associated decoupled resources to avoid software deadlocks.

A fault-handling network. This network permits non-minimal adaptive steps and it is used to exploit the fault-handling properties of adaptive routing. The number of turns a packet can make is restricted to make the network deadlock-free [8]. The RR allocates one channel for fault-handling purposes.

All three virtual networks share the same physical network by using different sets of virtual channels. The routing algorithm consists of three separate computations occuring in parallel. The whole route computation takes a single clock cycle. Each computation results in a next step virtual channel from one of the three virtual networks described above:

Adaptive Computation The switching node attempts to find a non-busy minimally adaptive channel. If such a channel is found, then the message will use it as the next step of its path.

Dimension-Ordered Computation If no minimal adaptive channel is available, the packet is routed to the unique dimension-ordered channel corresponding to its current position and its destination. If this channel is busy, the packet blocks for one cycle and tries again to find a channel using the Adaptive Computation.

Fault-Handling Computation If the dimension-ordered channel is faulty, the packet is routed to a fault-handling channel. This can be any channel, productive or unproductive, except for a channel that will cause the message to make a 180-degree turn. After the packet has been forwarded to the next switching node the algorithm either reverts to picking channels starting from the Adaptive Computation or sticks to Fault-Handling Computation channels only and the packet reaches its destination through fault-handling channels. This decision depends on the packet dimension at the time the Dimension-Ordered Computation fails to select a channel and is essential for the algorithm to be deadlock-free.

If the packet makes a side step to go around a faulty link, this side step will have to be reversed at some point in the future. If the side step is along the y dimension, then the side step can be reversed without violating the order of dimensions x, y, and the packet is allowed to route to dimension-ordered channels after routing on the non-minimal fault-handling channel. If, on the other hand, the side step is along the x-dimension, then the order of dimensions will be violated when the packet tries to reverse this non-minimal step. For this reason the packet continues to route on fault-handling channels until it reaches the destination.

5 The Unique Token Protocol

End-to-end protocols may solve the reliability problem when coupled with adaptive routing, but require extra overhead and the necessary resources do not scale linearly with the size of the machine [5]. For this reason, link-level retransmission is used in combination with a unique-token protocol (UTP) [5] to guarantee fault-tolerant exactly-once delivery of all packets in the network. This link-level protocol offers significant advantages over end-to-end protocols because it does not require acknowledgment packets and does not keep copies for possible retransmissions at the packet source. In this way, effective network bandwidth is increased and storage requirements at the nodes decrease. Moreover, the protocol reduces the amount of storage required at the destination nodes for duplicate message detection. These properties allow the protocol resources to scale linearly with the number of network nodes as opposed to end-to-end protocols.

An example of packet forwarding under the UTP is shown in Figure 8 where source node A sends a packet to destination node C. The packet is buffered and forwarded through switching node B. The process must ensure that at least two copies exist in the path between the source node and the destination node at all times. This can be achieved by first copying the packet forward one node, and then allowing the release of the storage in the rearmost node as shown in Figure 8. When the packet is first injected into the network, a token is injected right behind the packet. The invariant that no copies of the packet exist behind the token is always preserved. Packet copying and token passing are carried out exactly as shown in Figure 8. *Note that although multiple copies of the packet are kept in the network, every node receives a packet only once.* Thus, in the absence of faults, the arrival of the token at the destination implies that the packet has been delivered exactly once. For simplicity, the UTP was described at the packet level only. The Reliable Router implements the UTP at the flit level, and the protocol is a bit more involved.

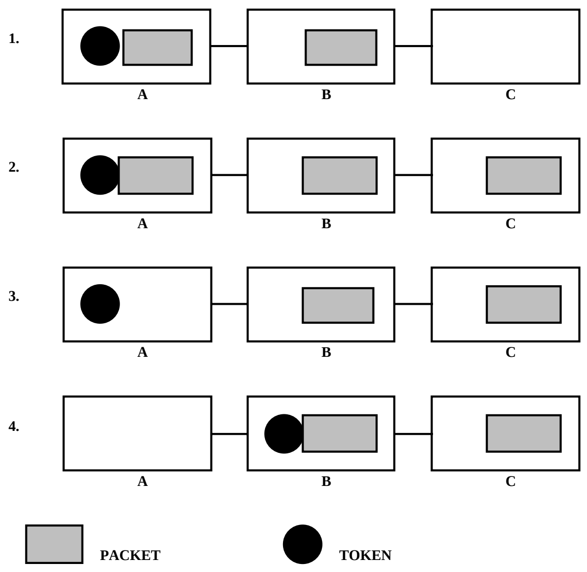

Figure 8: Buffering and forwarding under the UTP.

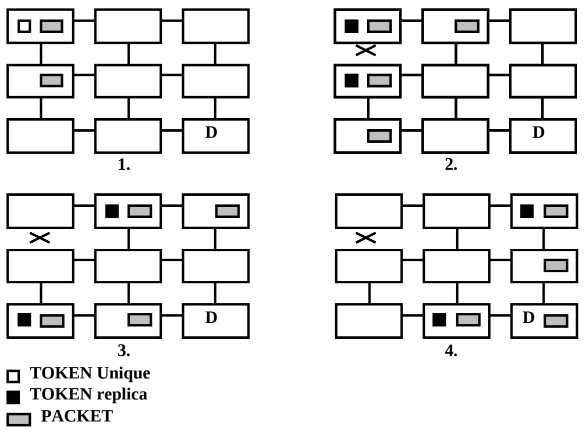

Figure 9: Fault Handling under the UTP

5.1 Fault Handling

When a node in the network fails, communication between the advance and rear copies of the packet may be severed. Each copy now must make its way to the destination without knowing the fate of the other copy. When packets arrive at the destination they must be marked in such a way so that the destination knows that it needs to look for duplicates. For this reason, two types of tokens are defined: *Unique*, and *Replica*. If the network needs to use multiple paths while forwarding the packet, the token is changed to type Replica for all copies of the packet. After the token is changed, forwarding proceeds in the usual way of always keeping two copies of the packet per path.

Such an example is shown in Figure 9. Due to a faulty link, communication between the two copies of the packet has been broken. As a result, each copy changes its token to Replica, or generates a Replica token and proceeds to the destination using different paths. When the destination receives a packet with a Replica token, it knows that it should be looking for duplicates. This scheme is based on the assumption that packets have unique identifiers so that duplicates can be detected and eliminated.

5.2 Flit-Level Implementation of the UTP

The actual implementation of the Unique Token Protocol occurs at the flit level rather than at the packet level. A long packet may span a number of nodes. The flit-level UTP guarantees that each flit of the packet has a copy for retransmission purposes in the neighboring node. A snapshot of a packet in flight under the flit-level UTP is shown in Figure 10. The figure shows a packet that consists of one head flit, seven data flits, and one tail flit. There are a number of things to notice from that figure:

1. There exists a second copy of every data or tail flit in the preceding node.
2. There is only one copy of the token flit. There exist no flits of the specific packet behind its token.
3. The head flit is stored at the head of the flit queue in every node spanned by the packet. It is deallocated only when the token flit leaves the node.

Every node needs a copy of the head flit to ensure retransmission of the partial packet when a link fails

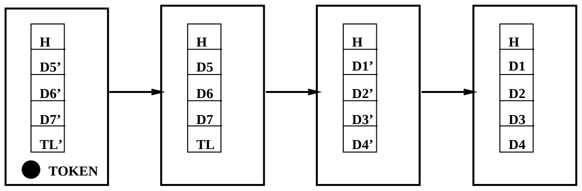

Figure 10: The UTP at the flit-level

after only part of a packet has been transmitted to the next node. The head flit is used to encapsulate the partial packet in the regular packet format and send it to the destination through an alternate route. The head flit of the trailing piece of a partial packet is tagged as a special kind of head flit – in the Reliable Router terminology it is called a *head:restart* flit as opposed to a *head:original* flit. This is necessary for the destination to reconstruct the original packet. It is also assumed that the tail flit of each packet contains the length of the message in flits. Given two partial pieces of a packet, the packet length and the the relative order of the two pieces, the destination can reconstruct unambiguously the original packet.

5.3 Architectural Support for the UTP

The implementation of the Unique Token Protocol presents a number of design challenges.

The token must be handled differently from the other flit types. The token can only be forwarded when the Flit Queues have unloaded and invalidated all resident data flits. Moreover, there must be a mechanism to generate a token, back up the pointers in the Flit Queues and reroute the virtual channel when a fault occurs. This functionality is partitioned among the Control, Virtual Channel State and Flit Munger modules of the Input Controller.

The greatest design challenge is that in order to keep two copies of flits at all times within the network, flow control information must make two steps to the back using separate flow control paths. This is shown in Figure 11. Let us assume that node C copies a flit across to node D. It sends a *copied* message to node B using the Copied Kind and Copied VCI fields shown in Table 2. Node B will use this information to invalidate its own copy of the flit. Moreover, reception of a *copied* field in an incoming frame will make node B send a *freed* message to node A using the Freed field of an outgoing frame. Node A will receive the *freed* information and use it to decrement the appropriate counter in its Counter Bank, indicating that

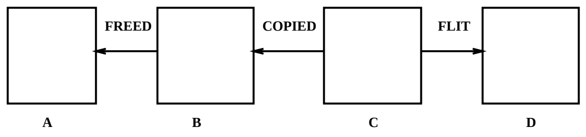

Figure 11: Flow Control in the Reliable Router.

the neighboring node (in this case node B) has one flit less in that particular Flit Queue. This backward flow control path is implemented using large buses that broadcast the *copied* and *freed* information to all Input Controllers. Only the Input Controllers that have a virtual channel which is routed to the port where the *copied* or *freed* message was received act upon this information.

6 Plesiochronous Data Recovery

One of the more difficult tasks in putting together a large scale MPP has been the global clock distribution. The RR addresses this problem by using plesiochronous timing. Each router is clocked using a different local oscillator with the same nominal frequency. There is no single global clock – and associated single point of failure – and the clock distribution problem is entirely avoided.

In a system of Reliable Routers, the clocks are all free running. To move data bits, the transmitter sends the clock along with the data. The receiver uses this clock to sample the data wires. Two latches are used to move data from the transmit clock domain to the receive clock domain. However, without some additional protocol, these latches will very quickly undersample or oversample flits.

A low-level protocol is used that imposes a maximum data rate on any wire which is below the minimum carrier rate of any link in the system. The minimum carrier rate is determined by the lowest actual frequency of any of the local oscillators. The transmit limit is implemented in the router by turning off the crossbar one out of a thousand times. The Output Controllers send a non-data frame called a *padding flit* whenever there is no data to send.

The principle of the interface operation is shown in Figure 12. The receiver delays the input waveform A by 180 degrees and produces waveform B. At any point in time it is always safe to sample from one of the two waveforms. Signals $KoutA$ (Keep out A)

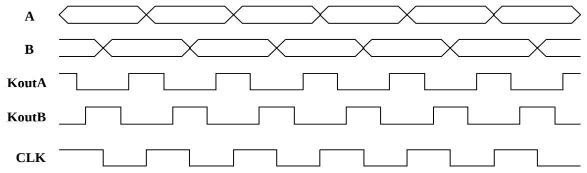

Figure 12: Plesiochronous Interface

and $KoutB$ (Keep out B) are produced by the transmitter clock and indicate the time windows where it is undesirable to sample from the A or B waveforms respectively. The receiver is sampling from one of the two waveforms. When the receiver clock samples a high on the corresponding Keep-out line, the gray area has been entered. As soon as the receiver detects a padding flit on the lines, it will switch and lock on the other waveform. If undersampling occurs, the receiver has missed a padding flit that does not carry any information. If oversampling occurs, the receiver has sampled an extra padding flit which is simply ignored. The finite state machine that controls the multiplexer of the A and B buses ensures that switching occurs only when both buses carry the same data.

This exact scheme has been implemented in the Synchronization module of Figure 3. In this case, waveform A is delayed by 180 degrees with respect to a pulse having half the frequency of *TxClk*. This is why the worst case latency is 1 clock cycle. In general, this scheme can achieve a worst case latency penalty of only half a clock cycle. Essentially this design minimizes the synchronization penalty.

7 Bidirectional Signalling

The 6 input and output ports shown in Figure 1 require a large amount of interchip bandwidth. To allow the use of a conventional chip carrier with fewer than 300 pins, simultaneous bidirectional signalling [6] is employed. This method allows bidirectional point to point digital signal communication on the same chip carrier pin and at the same time.

Figure 13 shows the block diagram of a pair of simultaneous bidirectional transceivers. The printed circuit board connection is modelled as a transmission line, with both ends of the line terminated with an internal resistor. A logic 1 or a logic 0 is transmitted as a positive current or a negative current respectively; the signal across the chip boundary is thus a superposition of the two current streams. To receive the digital signal from another transmitter, a

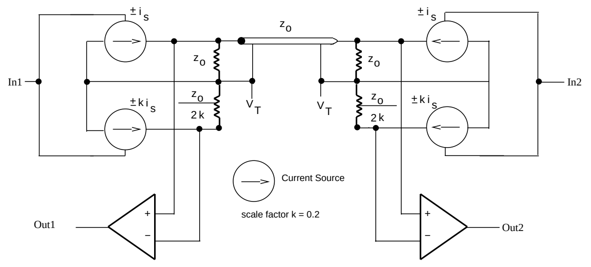

Figure 13: Simultaneous Bidirectional Signalling

local transmitter generates an internal copy of its own transmitted signal as a reference. The reference voltage is then subtracted from the superimposed signal to generate the received signal.

The small voltage swing (i.e. 250mV) of the signalling method calls for different noise reduction techniques:

1. *On-Chip Termination Control* allows the drivers' termination resistors to be fine-tuned for process variations.
2. *Current-Steering* keeps the current drawn by the drivers roughly constant to reduce the noise on the power planes.
3. *Rise-Time Control* staggers the turn-on of successive stages of current drivers to reduce the $L\frac{di}{dt}$ noise induced by the package pins.

To minimize power consumption, the current source for generating the local reference voltage is scaled down, while the reference resistor is scaled up accordingly. All of the 24 bidirectional signals that connect an adjacent pair of Reliable Routers employ the described simultaneous bidirectional signalling method, which amounts to a savings of 96 signal pins over conventional signalling methods. There is also substantial reduction in power and noise over full swing signalling.

8 Physical Implementation

The Reliable Router will be fabricated on a 144 mm^2 die in a 3-metal layer 1μ CMOS process. The current floorplan is shown on Figure 14.

The gray boxes comprise the Input Controller. The figure shows the FIFO, Virtual Channel Modules, and the Flit Munger (FM). The white boxes named OC

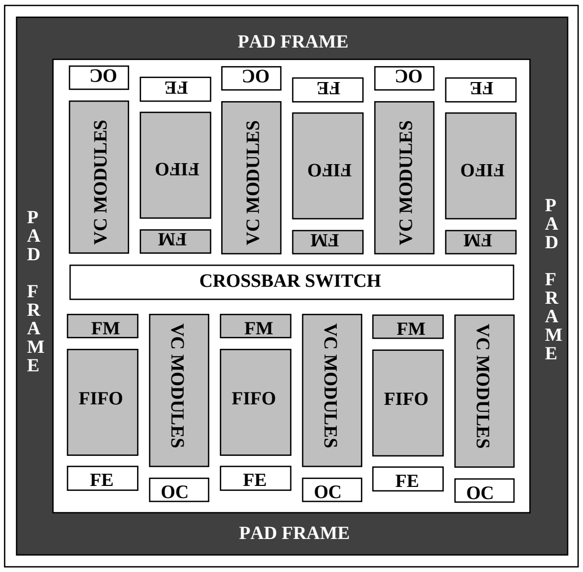

Figure 14: Reliable Router Floorplan

are the Output Controllers. The white boxes named FE are the Front Ends of Figure 3. Just by the relative sizing one can tell that most of the functionality in this chip lies in the Input Controllers.

Full custom layout has been employed for the FIFO module (virtual channel flit queues) because of size and special functionality required. This module is the most area-intensive block in the whole design. The Virtual Channel Module was layed out in a datapath design style. For all other modules, semi-custom layout was used.

90% of the layout has been completed. We plan to tape out the design in the very near future.

9 Conclusion

The internal architecture of the Reliable Router, a network switching element which provides reliable and high performance communication between nodes of parallel computers, has been presented. The Reliable Router uses a simple and efficient adaptive routing algorithm with minimal resource requirements. It also employs a link-oriented retransmission protocol with significant advantages over conventional end-to-end protocols. The coupling of these two features enable the RR to handle a single node or link failure without interruption of service. Other performance features include a queueless low-latency plesiochronous channel interface and simultaneous bidirectional signalling.

References

[1] Andrew A. Chien and Jae H. Kim. Planar Adaptive Routing: Low-cost Adaptive Networks for Multiprocessors. In *Proceedings of the 19th International Symposium on Computer Architecture*, pages 268–277, May 1992.

[2] William J. Dally et al. The Message-Driven Processor: A Multicomputer Processing Node with Efficient Mechanisms. *IEEE Micro*, April 1992.

[3] William J. Dally and Hiromichi Aoki. Deadlock-Free Adaptive Routing in Multicomputer Networks using Virtual Channels. *IEEE Transactions on Parallel and Distributed Systems*, Vol. 4, No.4, April 1993.

[4] William J. Dally and Charles L. Seitz. Deadlock Free Routing in Multiprocessor Interconnection Networks. *IEEE Transactions on Computers*, C-36(5):547–53, May 1987.

[5] Larry R. Dennison. Reliable Interconnection Networks for Parallel Computers. MIT AI Laboratory Technical Report 1294, October 1991.

[6] Larry R. Dennison, Whay S. Lee and William J. Dally. High Performance Bidirectional Signalling in VLSI Systems. In *Research on Integrated Systems: Proceedings of the 1993 Symposium.*

[7] Jose Duato. On the Design of Deadlock-Free Adaptive Routing Algorithms for Multicomputers: Design Methodologies. In *Parallel Architectures and Languages Europe Proceedings*, pages 390-405, June 1991.

[8] Cristopher J. Glass and Lionel M. Ni. The Turn Model for Adaptive Routing. In *Proceedings of the 19th International Symposium on Computer Architecture*, pages 278–287, May 1992.

[9] S. Konstantinidou and L. Snyder. Chaos Router: Architecture and Performance. In *Proceedings of the 18th International Symposium on Computer Architecture*, pages 212–221, May 1991.

[10] Daniel H. Linder and Jim C. 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.

[11] Charles L. Seitz and Wen-King Su. A Family of Routing and Communication Chips Based on the Mosaic. In *Research on Integrated Systems: Proceedings of the 1993 Symposium.*