The Alpha 21364 Network Architecture

Shubhendu S. Mukherjee, Peter Bannon, Steven Lang, Aaron Spink, and David Webb

Alpha Development Group
Compaq Computer Corporation
Shrewsbury, Massachusetts
Shubu.Mukherjee@compaq.com

ABSTRACT

The Alpha 21364 processor provides a high-performance, highly scalable, and highly reliable network architecture. The router runs at 1.2GHz and routes packets at a peak bandwidth of 22.4 GB/s. The network architecture scales up to a 128-processor configuration, which can support up to four terabytes of distributed Rambus memory and hundreds of terabytes of disk storage. The distributed Rambus memory is kept coherent via a scalable, directory-based, cache coherence scheme. The network also provides a variety of reliability features, such as per-flit ECC. These features make the 21364 network architecture well-suited to support communication-intensive server applications.

1. INTRODUCTION

Advances in semiconductor technology have allowed microprocessors to integrate more than a hundred million transistors on a single chip. The Alpha 21364 microprocessor uses 152 million transistors to integrate an aggressive Alpha 21264 processor core, a 1.75 megabyte second-level cache, cache coherence hardware, two memory controllers, and a multiprocessor router on a single die (Figure 1a). In the 0.18 µm bulk CMOS process, the 21364 will run at 1.2 GHz and provide 12.8 gigabytes/second of local memory bandwidth and 22.4 gigabytes/second of router bandwidth (Figure 2). This paper describes the Alpha 21364 network and router architectures.

The Alpha 21364's tightly-coupled multiprocessor network connects up to 128 such processors[1] in a two-dimensional torus network (Figure 1b). Such a fully-configured 128-processor shared-memory system can support up to four terabytes of Rambus memory and hundreds of terabytes of disk storage. Such an aggressive multiprocessor configuration allows us to support the massive computation and communication requirements of a variety of application domains, such as high-performance technical computing, database servers, web servers, and telecommunication applications. We designed the Alpha 21364 network architecture to meet the communication demands of these memory- and I/O-intensive applications.

The Alpha 21364's router architecture's novelty lies in its extremely low latency, enormous bandwidth, and support for directory-based cache coherence. The router offers extremely low latency because it operates at 1.2 GHz, which is the same clock speed as the processor core. The pin-to-pin latency within the router is 13 cycles or 10.8 nanoseconds. In comparison, the ASIC-based SGI Spider router runs at 100MHz and offers a 40 nanosecond pin-to-pin latency [2].

Similarly, the Alpha 21364 offers an enormous amount of peak and sustained bandwidth. The 21364 router can sustain between 70% and 90% of the peak bandwidth of 22.4 gigabytes/second. The 21364's router can offer such enormous bandwidth because of aggressive routing algorithms, carefully crafted distributed arbitration schemes, large amount of on-chip buffering, and a fully-pipelined router implementation to allow an aggressive operational clock rate of 1.2 GHz.

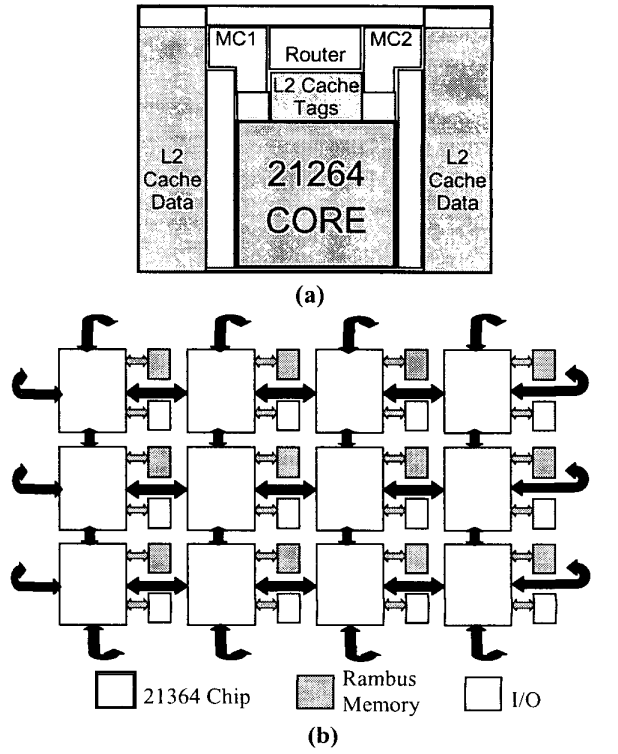

Figure 1. (a) Alpha 21364 Floorplan (b) A 12-processor configuration with Alpha 21364s. MC1 = memory controller 1, MC2 = memory controller 2.

Finally, the network and router architectures have explicit support for directory-based cache coherence, such as separate virtual channels for different coherence protocol packet classes. This helps to avoid deadlocks and improves the performance of the 21364's coherence protocol.

The rest of the paper is organized as follows. Section 2 briefly describes the 21364 network's packet classes. Section 3 and Section 4 describe the 21364's network and router architectures, respectively.

2. NETWORK PACKET CLASSES

Network packets and flits are the basic units of data transfer in the 21364's network. A packet is a message transported across the network from one router to another and is composed of one or more flits. A flit is a portion of a packet transported in parallel on a single clock edge. The size of a flit is 39 bits, 32 of which are for payload, and 7 bits are for per-flit ECC. Thus, each of the incoming and outgoing interprocessor ports (Figure 1b) is 39 bits wide. The 21364 network supports packet sizes of one, two, three, 18, and 19 flits. The first one to three flits of a packet contains the

[1] The Alpha 21364 can be easily redesigned to support a much larger configuration.

0-7695-1357-3/01 $10.00 © 2001 IEEE

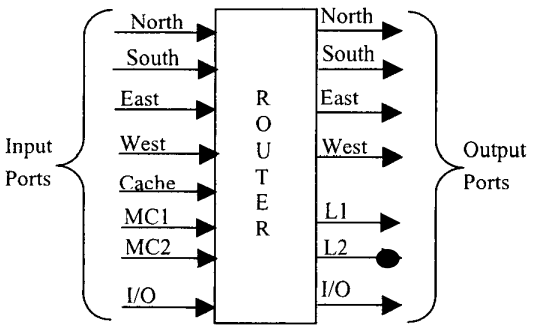

Figure 2. Router Ports. This figure shows the eight input ports and the seven output ports of the 21364's router. The north, south, east, and west interprocessor ports correspond to off-chip connections to the two-dimensional torus network. MC1 and MC2 are the two on-chip memory controllers (shown in Figure 1a). The Cache Input Port corresponds to the on-chip second-level cache. The L1 output port connects to the second-level cache as well as MC1. Similarly, the L2 output port connects to the cache and MC2. Finally, the I/O ports connect to the I/O chip external to the 21364 processor. The total aggregate bandwidth of the seven output ports is 22.4 GB/s.

packet's header (see Section 4.1). Additionally, the 18- or 19-flit packets typically contain 64-byte (or 16-flit) cache blocks or up to 64 bytes of I/O data.

The 21364 network supports seven packet classes:

- *Request Class (three flits).* A processor or I/O device uses a request packet to obtain data in and/or ownership of a cache block or up to 64 bytes of I/O data.
- *Forward Class (three flits).* A memory controller (MC1 or MC2 in Figure 1a) uses a forward packet to forward a request packet to the current owner or sharer (processor or I/O device) of a cache block.
- *Block Response Class (18 or 19 flits).* A processor or I/O device uses a block response packet to return the data requested by a request class packet or to send a modified cache block back to memory.
- *Non-Block Response Class (two or three flits).* A processor, memory controller, or I/O device uses a non-block response packet to acknowledge coherence protocol actions, such as a request for a cache block.
- *Write IO Class (19 flits).* A processor or I/O device generates a Write IO packet when it stores data to I/O space.
- *Read IO Class (three flits).* A processor generates Read IO packets to load data from I/O space.
- *Special Class (one or three flits).* These are packets used by the network and coherence protocol. The special class includes *Noop* packets, which can carry buffer deallocation information between routers.

The packet header (one to three flits long) identifies the packet's class and function. The header also contains routing information (see Section 4.1) for the packet, (optionally) the physical address of cache block or data block in I/O space corresponding to this packet, and flow control information between neighboring routers. Besides the header, the Block Response and Write IO packets also contain 16 flits or 64 bytes of data.

The 21364's coherence protocol and I/O devices use these packet classes to communicate between processors, memory, and I/O devices. A description of the 21364's directory-based coherence protocol can be found in Bannon, et al [6].

3. NETWORK ARCHITECTURE

The 21364 network is a two-dimensional torus (Figure 1b). In addition, the network can also support limited configurations of imperfect tori, which can map out faulty routers in the network.

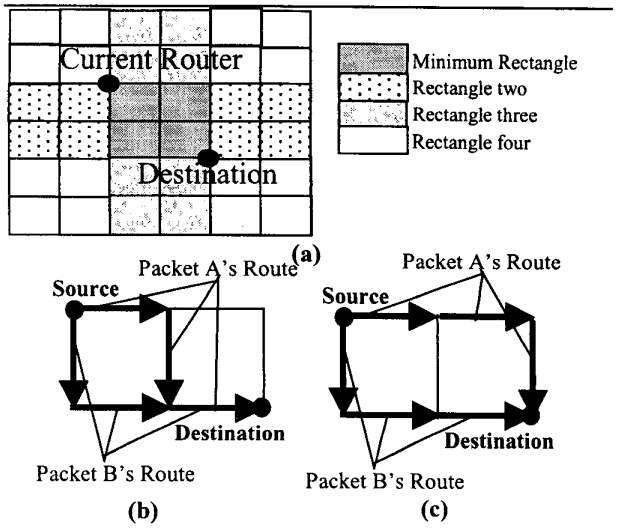

Figure 3. (a) Adaptive Routing in the Minimum Rectangle (b) Preference to turn (for packet A's route) makes the last hop a bottleneck link (c) Preference to continue straight in the same dimension allows a source-destination pair to maximize bandwidth between them.

This section discusses the 21364's virtual cut-through routing, adaptive routing algorithm, and deadlock avoidance techniques.

3.1 Virtual Cut-Through Routing

The 21364 uses virtual cut-through routing in which flits of a packet proceed through multiple routers until the header flit gets blocked at a router. Then, all flits of the packet are buffered at the blocking router until the congestion clears. Subsequently, the packet is scheduled for delivery through the router to the next router and the same pattern repeats. To support virtual cut-through routing, the 21364's router provides buffer space for 316 packets (see Section 4.3).

3.2 Adaptive Routing

The 21364's network uses adaptive routing to maximize the sustained bandwidth. However, the adaptive routing algorithm is very simple, which enables a simpler implementation of the arbitration scheme compared to more elaborate fully-adaptive routing algorithms. In the 21364 scheme, packets adaptively route within the *minimum rectangle*. Given two points in a torus (in this case, the current router and the destination processor), one can draw four rectangles that contain these two points as their diagonally opposite vertices. (Figure 3a). The minimum rectangle is the one with the minimum diagonal distance between the current router and the destination.

The adaptive routing algorithm picks one output port among a maximum of two output ports that a packet can route in at any router. Thus, each packet at the current router's input port and destined for a network output port has only two choices: either it can continue in the same dimension (e.g., North Input to South Output) or it can turn (e.g., North Input to East Output). This is because with every hop a packet will reduce its Manhattan distance to its destination. This shrinks the size of the minimum rectangle the packet is routing in.

If the adaptive algorithm has a choice between both the available network output ports (i.e., neither of the output ports is congested), then it gives preference to the route that continues in the same dimension. This allows a source and destination pair of processors to maximize the bandwidth between them by allowing multiple packets to route on separate routes (Figure 3b & Figure 3c).

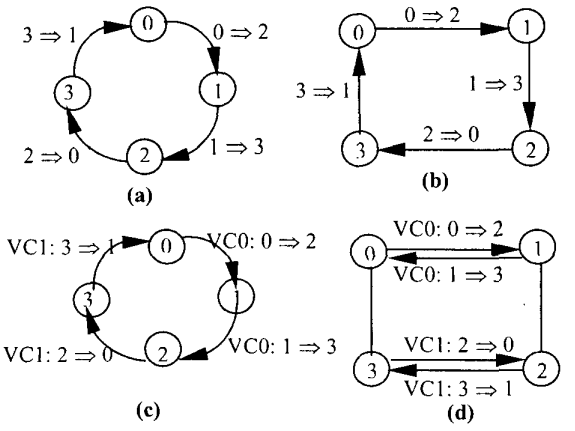

Figure 4. Potential deadlocks and 21364's solutions to break the deadlocks. Each arc represents a packet and is labeled with its source and destination processor. (a) shows a potential deadlock within a dimension (i.e., processor 0 to 3 are in the same dimension) of the two-dimensional torus network. The network is deadlocked because each packet is waiting for a buffer in the forward path to free up. (b) shows a potential deadlock across dimensions (with processors 0 and 1, 1 and 2, 2 and 3, and 3 and 0 in respectively different dimensions). This deadlock arises because each packet is waiting for buffers in the next dimension to free up. (c) shows how the 21364 breaks the intra-dimension deadlock by dividing up the buffers into virtual channels VC0 and VC1. Because the buffers are divided up into VC0 and VC1, the cyclic dependence between the packets is broken. (d) shows how the 21364 breaks the inter-dimension deadlock. In the VC0 and VC1 channels, each packet routes first along the primary axis (horizontally) and then routes along the secondary axis (vertically). When packets change dimensions, they recompute their virtual channels in the new dimension. Thus, packets in VC0 and VC1 along the primary axis depend on packets in the secondary axis, but packets in the secondary axis do not depend on packets in the primary axis. This scheme avoids the cyclic dependence across dimensions and removes the inter-dimension deadlock.

3.3 Deadlock Avoidance Rules

Both coherence and adaptive routing protocols can introduce deadlocks in a network because of cyclic dependences created by these protocols. This section describes the 21364's deadlock avoidance rules.

3.4 Avoiding Deadlocks in the Coherence Protocol

The coherence protocol can introduce deadlocks due to cyclic dependence between different packet classes. For example, request packets can fill up a network and prevent block response packets from ever reaching their destinations. The 21364 breaks this cyclic dependence by creating virtual channels [5] for each class of coherence packets and assigning (by design) an ordering constraint among these classes. By creating separate virtual channels for each class of packets, the 21364's router guarantees that each class of packets can be routed independent of other classes. Thus, a Block Response packet can never be blocked by a Request packet. The classes are ordered as: Read IO, Write IO, Request, Forward, Special, Non-Block Response, and Block Response. Thus, a Request can generate a Block Response, but a Block Response cannot generate a Request.

Additionally, the 21364 takes three measures to preserve I/O consistency rules. First, the router uses only deadlock-free virtual channels (Section 3.5) to force I/O requests of the same class to follow the same route and, thereby, arrive in order. Second, the router has separate virtual channels for I/O writes and reads (i.e., Write IO and Read IO respectively) to allow I/O writes to pass I/O reads. Finally, the router prevents I/O reads from bypassing I/O writes to support I/O ordering rules.

3.5 Avoiding Deadlocks in Adaptive Routing

Adaptive routing can generate two types of deadlocks, namely, intra-dimension and inter-dimension. Figure 4a and Figure 4b show examples of these two kinds of deadlocks. 21364 breaks these two deadlocks using Jose Duato's theory [4], which states that adaptive routing will not deadlock a network as long as packets can drain via a deadlock-free path.

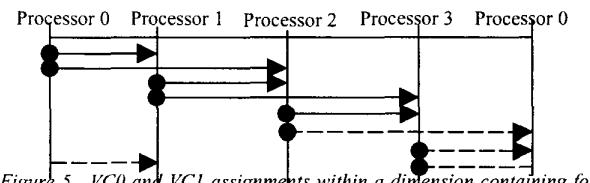

Figure 5. VC0 and VC1 assignments within a dimension containing four processors. The solid arrows represent VC0s and dashed arrows represent VC1s. Because the 21364 routes only within the minimum rectangle, an arrow does not span more than two hops in a dimension with four processors. The assignments in this figure are based on Dally's scheme (Section 3.5). For the consecutive physical links, the VC0 to VC1 ratios are: 2/1, 3/0, 2/1, and 0/3.

The 21364 creates logically distinct adaptive and deadlock-free networks using virtual channels. Each of the virtual channels corresponding to a packet class, except the *special class* (described in Section 2), is further subdivided into three sets of virtual channels, namely, *adaptive*, *VC0*, and *VC1*. Thus, the 21364 has a total of 19 virtual channels (three each for the six non-special classes and one for the special class).

The adaptive virtual channels form the adaptive network and have the bulk of a router's buffers associated with them. The VC0 and VC1 combination creates the deadlock-free network, which provides a guaranteed deadlock-free path from any source to any destination within the network. Thus, packets blocked in the adaptive channel can drain via VC0 and VC1.

The VC0 and VC1 virtual channels must be mapped carefully on to the physical links to create the deadlock-free network. The 21364 has separate rules to break deadlocks within a dimension and across dimensions. Within a dimension, the 21364 maps the VC0s and VC1s in such a way that there is at least one processor that dependence chains formed by VC0s do not cross. The same applies to VC1 mappings. This ensures that there is no cyclic dependence in a virtual channel within a dimension.

The 21364 can choose among a variety of such virtual channel mappings because the virtual channel assignments are programmable at boot time. Perhaps the simplest scheme that satisfies the property stated above was proposed by Dally [5] in which all processors in a dimension are numbered incrementally. Then, for all source and destination processors, we can make the following virtual channel assignments: if source is less than the destination, that source-destination pair is assigned VC0. If source is greater than destination, then that pair is assigned VC1.

Unfortunately, in this scheme, the virtual channel to physical link assignments are not well-balanced (Figure 5), which can cause under-utilization of network link bandwidth under heavy load. In the 21364, we search for an optimal virtual channel to physical link assignment using a hill climbing algorithm. This scheme does not incur any overhead because we run the algorithm off-line and only once for a dimension with a specific size (ranging from two to 16 processors).

Figure 4d shows how 21364 breaks inter-dimensional deadlocks by designating one of the two directions as the primary axis and the other as the secondary axis. A packet can always proceed along the primary axis in VC0, VC1, or adaptive channels. However, along the secondary axis, a packet can proceed only via the adaptive channel, unless the row or column (in which the packet is routing) of the secondary axis also contains the destination 21364. If the secondary axis contains the destination

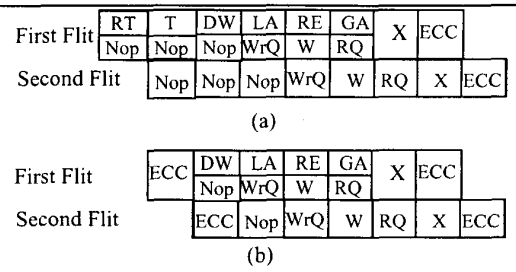

Figure 6. Two of the nine logical router pipelines in the 21364. (a) shows the router pipeline for a local input port (cache or memory controller) to an interprocessor output port (b) shows the router pipeline from an interprocessor (north, south, east, or west) input port to an interprocessor output port. The first flit goes through two pipelines: the scheduling pipeline (upper pipeline) and data pipeline (lower pipeline). Second and subsequent flits follow the data pipeline. RT = Router Table Lookup, Nop = No operation, T = Transport (wire delay), DW = Decode and Write Entry Table, LA = Local Arbitration, RE = Read Entry Table and Transport, GA = Global Arbitration, W = Wait, WrQ = Write Input Queue, RQ = Read Input Queue, X = Crossbar, and ECC = Error Correction Code.

21364, then the packet can route in VC0, VC1, or Adaptive channels.

21364's deadlock-avoidance rules, however, do not prevent a packet in VC0 or VC1 from returning to the adaptive channel. Thus, a packet blocked in the adaptive channel can drop down to VC0 or VC1. However, in subsequent routers along the packet's path, the packet can return to the adaptive channel, if the adaptive channel is not congested. This works for the 21364 because virtual cut-through routing buffers entire packets at a router, even though packets that are not blocked can span multiple routers at the same time. The 21364's ability to buffer an entire packet at a router removes dependence between consecutive routers, which allows packets to move from the deadlock-free VC0 and VC1 channels to the adaptive channel. Additionally, a 21364's choice of direction and virtual channel are independent of a packet's prior route in the network, which helps remove cyclic dependences among routers.

4. ROUTER ARCHITECTURE

The 21364's router has nine pipeline types based on the input and output ports. An input or an output port can be of three types: local port (cache and memory controller), interprocessor port (off-chip network), and I/O. Any type of input port can route packets to any type of output port, which leads to nine types of pipelines. Figure 6 shows two such pipeline types.

In addition to the pipeline latency, there are a total of six cycles of synchronization delay, pad receiver and driver delay, and transport delay from the pins to the router and from the router back to the pins. Thus, the on-chip pin-to-pin latency from a network input to a network output is 13 cycles. At 1.2 GHz, this leads to a pin-to-pin latency of 10.8 nanoseconds.

The network links that connect the different 21364 chips run at 0.8 GHz, which is 33% slower than the internal router clock. The 21364 chip runs synchronously with the outgoing links, but asynchronously with the incoming links. The 21364 sends its clock with the packet along the outgoing links. Such clock forwarding provides rapid transport of bits between connected 21364 chips and minimizes synchronization time between them.

4.1 Router Table Lookup and Decode Stages

The 21364's router table consists of three parts:

- a 128-entry configuration table, with one entry for each destination processor in the network,
- a virtual channel table consisting of two 16-bit vectors, which contains the deadlock-free virtual channel assignments, and
- an additional table to support broadcasting invalidations to clusters of processors (as required by 21364's coherence protocol [6]).

Class	Interprocessor Ports			Cache Port	MC Ports	IO Port
	ADP	VC0	VC1			
Request	8	1	1	8	0	8
Forward	8	1	1	0	8	0
Block Response	3	1	1	6	4	5
Non-Block Response	8	1	1	8	9	9
Write IO	1	2	2	4	0	2
Read IO	1	2	2	4	0	2
Special	8	0	0	0	0	0

Table 1. Input Buffers in the 21364 router. Each buffer can hold a complete packet of the specified class, except for local ports for which the packet's payload can reside in the 21364's internal buffers. Buffers at the interprocessor input ports are subdivided into ADP (adaptive), VC0, and VC1 channels. The local ports—one cache, two MC (memory controller), and one IO ports—do not need virtual channels. The adaptive channel for Write IO and Read IO classes are only used optionally for the first hop in a network with faulty nodes. Subsequent hops are always in order and strictly follow the VC0 and VC1 channels. There are six other special buffers not shown in the table. The total number of buffers in the 21364 router is 316.

Like the SGI Spider switch [2], the 21364 router table is programmable by software at boot-time, which allows software the flexibility to optimize the desired routing for maximal performance and to map out faulty nodes in the network.

The first flit of a packet entering from a local or I/O port accesses the configuration table and sets up most of the 16-bit routing information in a packet's header. These 16 bits are:

- two bits for the east-west and north-south directions (positive or negative),
- eight bits for the destination coordinates along the two dimensions,
- one bit (used by incomplete torus networks) to indicate if the packet can route in the adaptive channel,
- one bit to indicate if the packet is an IO packet,
- two bits to encode the virtual channel number (Adaptive, VC0, and VC1), and
- two reserved bits.

Each configuration table entry contains 24 bits that include the header's routing information (except the two bits that encode the virtual channel number), six access control bits, three bits to encode routing information for incomplete tori networks (with mapped out nodes), and one bit of parity.

The decode stage identifies the packet class, determines the virtual channel (by accessing the virtual channel table), computes the output port, and figures out the deadlock-free direction. The decode phase also prepares the packet for subsequent operations in the pipeline.

4.2 Error Correction Code Manipulation

Each 32-bit flit of a 21364 network packet is protected by 7-bit ECC. The router checks ECC for every flit of a packet arriving through an interprocessor or an I/O port. The router regenerates ECC for every flit of a packet leaving through an interprocessor or an I/O output port (Figure 6). ECC regeneration is necessary particularly for the first flit of the packet because the router pipeline can modify the header before forwarding the packet.

If the router pipeline detects a single-bit error, it corrects the error and reports it back to the operating system via an interrupt. However, it does not correct double bit errors. Instead, if it detects a double-bit error, the 21364 alerts every reachable 21364 of the

occurrence of such an error and enters into an error recovery mode.

4.3 Input Buffering

The 21364 router provides buffering only at each of the input ports. Table 1 shows the distribution of input buffers at each input port. Each input buffer can hold a complete packet of the specific packet class, except for local ports for which packet payloads reside in the 21364's internal buffers. The interprocessor ports are subdivided into adaptive and deadlock-free channels, whereas the local ports have a single monolithic buffer space. The router has a total of 316 packet buffers.

Each input port has an entry table that holds the in-flight status for each packet and input buffers that hold the packets. The first flit of a packet writes the corresponding entry table entry in the DW stage (Figure 6). An entry table entry contains a valid bit, bits for the target output ports (Figure 2), bits to indicate if the packet can adapt and/or route in the adaptive channels, bits supporting the anti-starvation algorithm[2], and other miscellaneous information. This information is used by the readiness tests in the LA phase of arbitration and read in the RE phase to decide the routing path of each packet (Section 4.4). Flits are written to and read from the packet buffers in the WrQ (Write Input Queue) and RQ (Read Input Queue) stages, respectively, after scheduling pipeline has made the routing decision for the first flit.

Either the previous 21364 router in a packet's path or the cache, memory controller, or I/O chip where the packet originated controls the allocation of input buffers at a router. Thus, each resource delivering a packet to the router knows the number of occupied packet buffers in the next hop. When a router deallocates a packet buffer, it sends the deallocation information to the previous router or I/O chip via *Noop* packets (Section 2) or by piggybacking the deallocation information on packets routed for the previous router or I/O port.

4.4 Arbitration

The most challenging component of the 21364 router is the arbitration mechanism that schedules the dispatch of packets arriving at its input ports. To avoid making the arbitration mechanism a central bottleneck, the 21364 breaks the arbitration logic into local and global arbitration (Figure 7). There are 16 local arbiters, two for each input port. There are seven global arbiters, one for each output port. In each cycle, a local arbiter may speculatively schedule a packet for dispatch to an output port. Two cycles following the local arbitration (Figure 6), each global arbiter selects one out of up to seven packets speculatively scheduled for dispatch through the output port. Once such a selection is made, all flits in the X (Crossbar) stage (Figure 6) follow the input port to the output port connection, as shown in Figure 7.

The local arbiters perform a variety of readiness tests to determine if a packet can be speculatively scheduled for dispatch via the router. These tests ensure that:

- the nominated packet is valid at the input buffer and has not been dispatched yet,
- the necessary dispatch path from the input buffer to the output port is free,
- the packet is dispatched in only one of the routes allowed,
- the target router, I/O chip, or local resource (in the next hop) has a free input buffer in the specific virtual channel,

[2] Because of the distributed and speculative nature of 21364's arbitration mechanism, packets residing at the input buffers can be starved. The 21364 provides a sophisticated anti-starvation mechanism, which detects starved packets and drains them via the output ports.

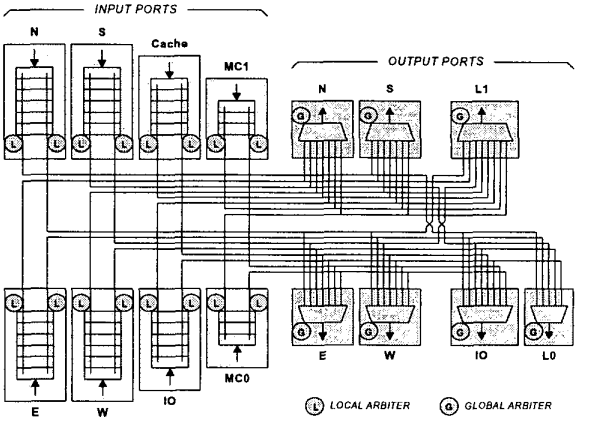

Figure 7. Connections between Local and Global Arbiters.

- the target output port is free,
- the anti-starvation mechanism is not blocking the packet, and
- a Read IO packet does not pass a Write IO packet.

A global arbiter selects packets speculatively scheduled for dispatch through its output port. The local arbiters speculatively schedule packets not selected by any global arbiter again in subsequent cycles.

To insure fairness, the local and global arbiters use a *least-recently selected (LRS)* scheme to select a packet. Each local arbiter uses the LRS scheme to select both a class (among the several packet classes) and a virtual channel (among VC0, VC1, and Adaptive) within the class. Similarly, the global arbiter uses the LRS policy to select an input port (among the several input ports that each output port sees).

Additionally, the 21364 provides two special modes—the *Rotary Rule* mode and *CDP Rule* mode—in which packets are prioritized according to the input port they arrive from and the packet class they belong to. The Rotary Rule gives priority to packets arriving from an interprocessor port to allow older packets residing in the network to move sooner than younger packets generated from the local or I/O ports. The CDP (Coherence Dependence Priority) Rule prioritizes the packets according to their class ordering (Section 3.4). Thus, the CDP Rule prioritizes Block Response packets over Request packets.

ACKNOWLEDGMENTS

The Alpha 21364 network architecture has been made possible many engineers. Richard Kessler was one of the most important contributors towards the design of the 21364's network architecture. Jim Burnette provided valuable feedback during the initial stages of the design. Zarka Cvetanovic and Simon Steely provided valuable performance simulation and feedback. Keith Farkas, Geoff Lowney, Paul Rubinfeld, and Simon Steely provided helpful feedback on early drafts of this paper.

REFERENCES

[1] Peter Bannon, "Alpha 21364: A Scalable Single Chip SMP," Microprocessor Forum Talk, 1997.

[2] Mike Galles, "Scalable Pipelined Interconnect for Distributed Endpoint Routing: The SGI SPIDER Chip," Hot Interconnects IV, Palo Alto, August 1996.

[3] Anil Jain, et al, "A 1.2 GHz Alpha Microprocessor with 44.8 GB/sec of chip pin bandwidth", International Solid-State Circuits Conference Digest of Technical Papers, February, 2001, Vol 44, pp. 240.

[4] Jose Duato, "A New Theory of Deadlock-Free Adaptive Routing in Wormhole Networks," *IEEE Transactions on Parallel and Distributed Systems*, vol .4 , no. 12, pp 1320-1331, December, 1993.

[5] William J. Dally, "Virtual Channel Flow Control," Proceedings of the 17th Annual International Symposium on Computer Architecture (ISCA), May 1990.

[6] Peter Bannon, Brian Lilly, David Asher, Maurice Steinman, David Webb, Rishan Tan, and Timothe Litt, "Alpha 21364: A Single-Chip Shared Memory Multiprocessor," GOMAC 2001 Digest of Papers, March 2001, pp 334- 337.