A Fast, Simple Router for the Data-Intensive Architecture (DIVA) System

Chang Woo Kang and Jeffrey Draper
USC-Information Sciences Institute
4676 Admiralty Way
Marina Del Rey, CA 95292 USA

Abstract—**This paper presents a fast, simple router design for implementing the Red Rover algorithm for a bidirectional ring. This design is very suitable for the Data-Intensive Architecture (DIVA) system, a system which demonstrates the benefits of embedded DRAM technology, because of its high performance as well as simple architecture and low cost. The key attributes of this router are one clock node-to-node latency, high channel throughput, and simple hardware implementation. The router architecture employs short-cut FIFO data paths, which makes the router speed independent of the channel buffer size (in terms of flits). A prototype implementation of the router achieves a maximum channel bandwidth of 5.12 Gb/s and runs at 80 MHz using 3.3V CMOS signaling in 0.5μm technology. This high throughput and low latency were achieved without resorting to the use of complex high-speed signaling technologies.**

I. Introduction

Embedded DRAM technology is growing in popularity, as it appears to be a promising solution to the increasing gap between processor and memory speeds [6]. Integrating processor logic and memory in processing-in-memory (PIM) chips offers dramatically increased memory bandwidths (up to 2 orders of magnitude) over conventional systems. Furthermore, memory latency is also reduced because internal memory accesses avoid the delays associated with communicating off chip. The Data-Intensive Architecture (DIVA) system aims to exploit this technology by combining PIM devices with one or more external host processors and a PIM-to-PIM interconnect [9]. The DIVA system design imposes a unique set of requirements on the PIM-to-PIM interconnect. PIM chips will be physically grouped as conventional memory chips, mounted on DIMM modules. The number of PIM chips in a hosted cluster is therefore in the range of 32 to 64 chips, depending on how many PIM chips can be packed onto a DIMM module. The PIM-to-PIM interconnect must then be amenable to the dense packing requirement of DIMM modules. Low latency and high throughput are also desirable properties of this interconnect. Furthermore, this network must be scalable to allow the addition or removal of modules. This combination of requirements favors a one-dimensional network. Recently implemented routers such as SGI SPIDER [5] and the Cray T3E network router [8] are not suitable to be embedded in PIM devices because of complexity and size. The resulting PIM Routing Component (PiRC) is a one-dimensional wormhole router which implements the Red Rover routing algorithm to effect deadlock-free routing in bidirectional rings [1], [3]. The Red Rover algorithm provides a more even, symmetric distribution of message traffic among virtual channels in a bidirectional ring and therefore attains lower latencies and higher throughput than Dally's spiral algorithm [4]. Additionally, the PiRC routes fixed-size packets and uses source routing to achieve low latency. The PiRC architecture is presented in detail in Section II.. Section III. describes implementation and performance issues. Simulation scenarios for testing functionality are presented in Section IV., and concluding remarks are given in Section V..

II. Router Architecture

Because it employs the *Red Rover* algorithm, the *PIM Routing Component (PiRC)* has a very simple architecture and may be viewed as two identical routers which are time-multiplexed. One router operates on the rising transition of the clock while the other operates on the falling transition. In this manner, two virtual channels (A and B) are time-multiplexed onto each physical channel. Each virtual router contains controlling logic, consisting of an input controller, switch, and output controller, and short-cut FIFO data paths (see Figure 1). A channel *input controller* receives control signals from a sender and generates control signals for storing data into a short-cut FIFO. The *switch and output controller* determines to which output port input data should be forwarded and arbitrates fairly among contending requests for a particular output port. The handshaking protocol between sending and receiving PiRC channels, described in Section A., is also very simple and efficient.

Other factors also contribute to the simplicity of the PiRC architecture. A packet is constrained to a fixed length of ten 32-bit flits, and the phit size is the same as the flit size. All operation including receiving, switching, arbitrating, and sending is done in a half clock cycle. Thus, only one clock is needed for a flit to traverse from one node to the next in the non-blocking case. The PiRC implements wormhole routing [7] so that flits of a blocked packet remain in place in the network channels. However, each PiRC FIFO contains enough space to buffer a complete 320-bit packet. This ability simplifies the handshaking so that handshakes need only occur on packet boundaries rather than on every flit.

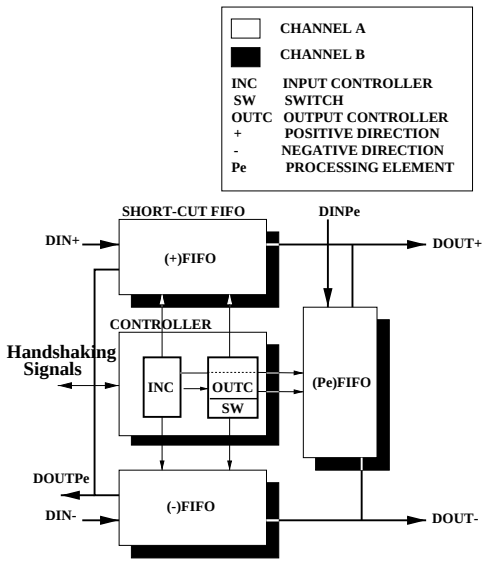

Fig. 1. PIM Routing Component (PiRC) Block Diagram

Figure 2 shows the internal interface for one virtual level of the PiRC (the other level is identical). This figure shows how the FIFOs, input controllers(INC), switches(SW), and output controllers(OUTC) interact for the positive(+), negative(-), and processing element(Pe) directions. Note especially the switching and merging combinations in the data paths. A packet entering the (+) FIFO may continue in the (+) direction or exit the network through the Pe port. Similarly, a packet entering the (-) FIFO may continue in the (-) direction or exit the network through the Pe port. Finally, a packet which is injected via the Pe FIFO may enter the network via the (+) or (-) port. These routing restrictions result in 2-way switchers and 2-way mergers at every point of contention. This artifact simplifies the router design, requiring the design of only one merge and one switch element that are then replicated as needed. The SI and RI signals are *send* and *ready* handshaking signals for the input channels, while the SO and RO signals correspond to output channels. More detail about their operation is given in the following section.

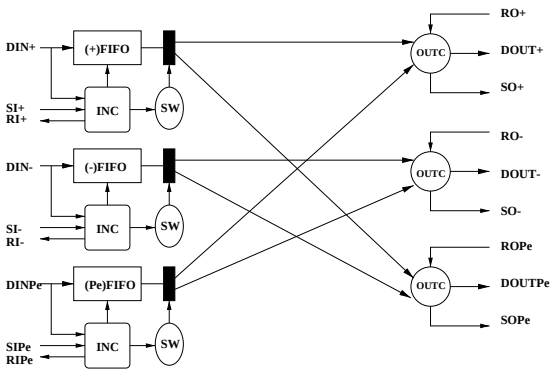

Fig. 2. PiRC Internal Interface

A. Handshaking Protocol

The handshaking protocol is really simple and efficient. First, note that the SI and RI signals of a receiving PiRC channel are connected to the SO and RO signals, respectively, of a neighboring sending PiRC channel. The receiver keeps asserting the RI signal as long as its corresponding FIFO is not full. The sender keeps sampling the corresponding RO signal at every edge of the clock and starts sending a pending message whenever the receiver is ready. By using this protocol, the sender constantly monitors the state of the receiver and does not waste time to explicitly request the status of the receiver FIFO. As depicted in Figure 3, this protocol makes it possible for the sender to make a decision to send data as soon as an asserted RO is sampled. To indicate it is sending data, the sender asserts SO, and the receiver latches DIN data into the FIFO upon sampling the corresponding asserted SI signal. The receiver then latches data on the next nine clock cycles to receive the entire packet.

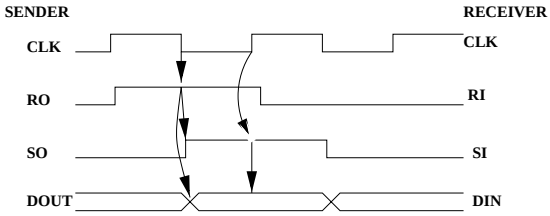

Fig. 3. Handshaking between a sender and a receiver

B. Short-Cut FIFO

In order to achieve high-speed data transmission along the physical channel, fast switching activity between channels is essential. A previous implementation of a Red Rover router, the *PDSS router* [2], specified a simple controller and complex flit buffer design and is suitable for only a small number of flits per packet. In the PDSS router, there are a large number of flit buffers that can drive the final output stage bus, as shown in Figure 4. This arrangement results in a large capacitive load. The controller is, however, very simple such that finite state machines without peripheral logic are sufficient for controlling the register-tristate buffer pairs. In contrast, the *PiRC* implements a complex controller and simple FIFOs in order to accommodate a large number of flit buffers in the channel buffer. In fact, the output stage load capacitance is independent of the number of flit buffers in a short-cut FIFO because only the top element of the FIFO is capable of driving the output stage bus. This characteristic makes the design very flexible with regard to channel size and is important as different package types impose different pin-count, and therefore channel size, constraints. With this design, every flit in the FIFO shifts toward the top of the FIFO as long as the path is not blocked. Also, incoming flits are placed in the first empty flit buffer (from the top of the FIFO). Figure 5 illustrates the cell of the FIFO, block diagram, and an example of flit movement.

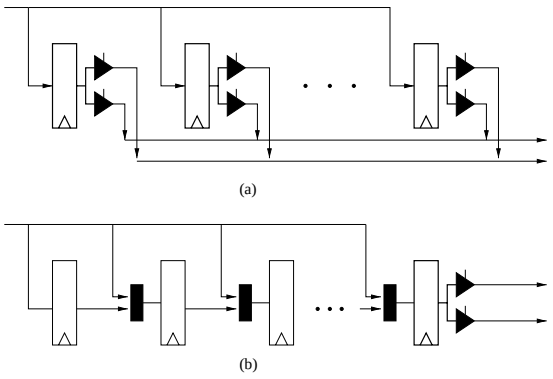

Fig. 4. Channel Buffer Design:(a) Register-Tristate Buffer in PDSS Router and (b) Short-cut FIFO in PiRC

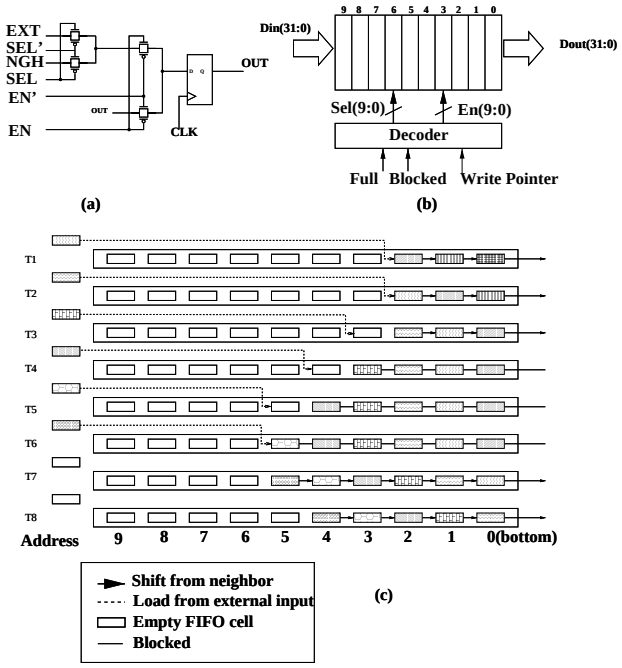

Fig. 5. Short-Cut FIFO: (a) FIFO Cell, (b) Block Diagram, and (c) Movements of Flits

The cell in (a) has two data inputs, one from the neighboring flit in the FIFO (NGH) and the other from the external input for this router channel(EXT). The decoder in (b) generates proper control signals for the FIFO based on current conditions and a write pointer indicator. (c) is an example showing the movement of flits. Until T3 there is no blocking; therefore, the flit in flit buffer 0 goes out and the other flits shift toward the top of the FIFO so that the FIFO depth is constant. New flits from the external input are loaded into flit buffer 2 (this example assumes some residual flits exist in the FIFO initially). Flits do not shift if the output path is blocked, as shown during T4, T5, and T6. However, the write pointer increments so that subsequent incoming flits begin to fill up the FIFO. When the path becomes unblocked, flits drain out as shown from T7.

In order to keep track of the header flit of a packet, the SI signal flows through a one-bit FIFO as the header flit moves. The operation of this one-bit FIFO is identical to that of the data FIFO described above. This signal becomes the output signal SO in the final output stage, indicating that the router channel is sending a header flit of a new packet.

C. Input Controller

The input controller, shown in Figure 6, is simple counter-based logic that directs the loading of flits into the short-cut FIFO. When the input controller samples an asserted SI signal, it begins latching the flits of an incoming packet. The *up/down counter* dynamically changes the write pointer value, which always points to the first empty space in the FIFO, as the router reads and writes flits. The wen (write enable) generator causes the $up/down$ counter to increment the write pointer value when SI arrives from the sender. The ren (read enable) generator is activated when the output controller starts forwarding flits from the FIFO to an output channel, and it also prevents the reading of garbage in an empty FIFO. The counter operates at both clock edges so that it can increase the write pointer at the rising edge when a new flit is written and decrease the pointer at the falling edge when a flit is read from the FIFO (these clock edges apply for A virtual channels—for B virtual channels, the opposite clock edges apply). The *full-empty detector* indicates the status of the FIFO. The RI handshaking signal is merely the inverse of the $full$ signal. As mentioned earlier, the decoder translates the write pointer value into proper control signals for the FIFO.

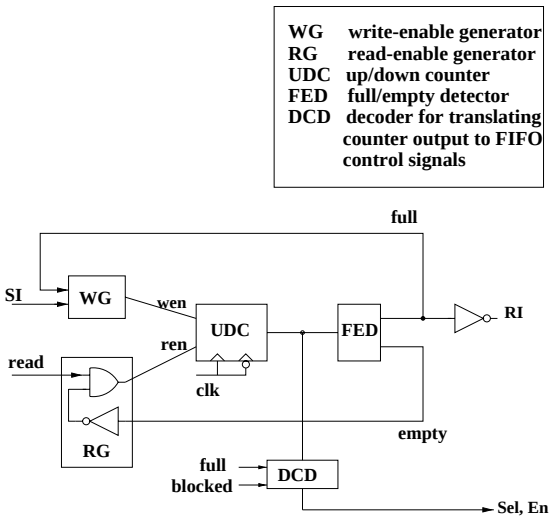

Fig. 6. Input Controller

D. Switch and Output Controller

The output controller samples RO at every clock edge so that it can send a pending packet as soon as possible. Once RO is asserted and detected by the output controller, the header flit of a pending packet and SO are sent immediately. While flits

are being transmitted to the receiver, the write pointer of the sending FIFO decrements if there are no incoming flits from the neighboring PiRC. On the other hand, the pointer keeps pointing to the same flit buffer in the sending FIFO if the sending FIFO is simultaneously receiving data from its neighbor. The switch determines the direction in which a packet is to be forwarded. The first flit of a packet, the header, contains routing information for the switch. The header is unary encoded such that the number of hops a packet is to traverse is indicated by the number of 1's set in the header. The header is shifted at each hop so that this value is decremented. Therefore, the switch simply inspects the first bit of the routing header to determine which output port to request for a given packet. Using a first-come-first-served policy, the output controller arbitrates fairly between requests from two FIFOs contending for usage of the same output physical channel. If contending requests arrive in the same clock cycle to an idle output controller, an arbitrary selection is performed; however, the FIFO which is not granted access during this arbitration is guaranteed access when the current FIFO completes based on the first-come-first-served policy.

III. Implementation and Performance

The PiRC design was begun by behavioral modeling in VHDL and compiled with Synopsys. Cascade EPOCH was used for routing and placement as well as layout generation for a prototype implementation. Control blocks were synthesized, while the short-cut FIFO was generated using custom layout to achieve high density. We tested our design at the behavioral level, pre-synthesis level, and post-synthesis level with Synopsys, and transistor level with Powermill.

The resulting PiRC prototype layout is for the HP14b process available through MOSIS. This process uses 0.5μm, 3-layer metal CMOS technology. The PiRC has a die size of 2.76 mm X 2.36 mm and contains 75,276 transistors. Simple hardware based on an efficient routing algorithm allows us to achieve a clock frequency of 80MHz. The router operates on both clock edges, leading to a channel bandwidth of 5.12Gb/s. Only one clock is required for a flit to move from one node to the next, resulting in a node-to-node delay of 12.5ns. Figure 7 shows the layout of the PiRC, placed and routed with the floor plan of Figure 1. Although this prototype achieves respectable performance, we expect performance to improve significantly when we migrate to a currently available embedded DRAM process using 0.25μm or even 0.18μm technology, such as the IBM SA27-E or TSMC process.

IV. Simulation

Five critical scenarios were used to verify the PiRC design. The external PiRC connections used for simulation are shown in Figure 8. This configuration allows short-cut FIFOs to be cascaded together so that one FIFO essentially feeds another. The header flit of a packet is set in simulation to specify

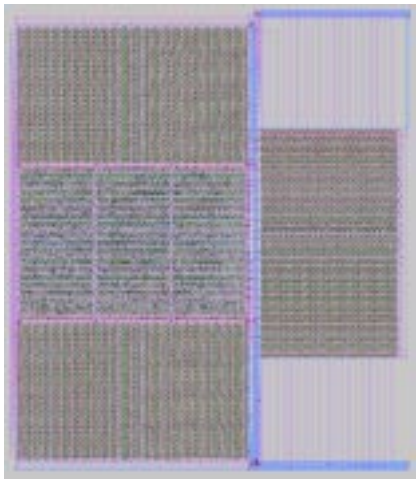

Fig. 7. Layout

whether the corresponding packet travels from the (Pe) FIFO to the (+) FIFO or the (-) FIFO. Test vectors are injected on the Tester terminals indicated in Figure 8, which essentially serve as processing element signals. The scenarios are as following:

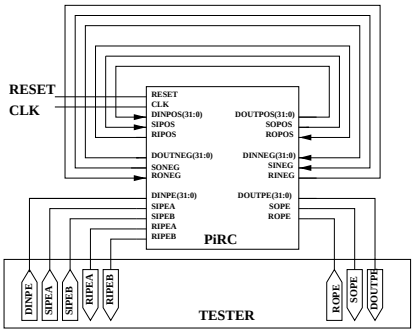

Fig. 8. Router Configuration for Testing

1. Two messages move back-to-back without blocking.
2. Two messages move back-to-back. The first message is blocked until the (+,-) FIFO is full. Consequently, the second message is blocked in the (Pe) FIFO and starts filling it. Then, the first message becomes unblocked and drains out. As soon as the first message starts moving out, the second message follows it along the path.
3. Two messages move back-to-back. The first message is blocked until the (+,-) FIFO gets half-way full, and then the first message drains out.
4. The first message is blocked until the (+,-) FIFO fills half-way, and when the first message starts draining out of the (+,-) FIFO, the second message is injected to the (Pe) FIFO from the tester. Due to the short-cut FIFO design, the second message quickly traverses the (Pe) FIFO to trail the first message.

5. Two packets in (+,-) FIFO and (Pe) FIFO request the same channel concurrently. This scenario ensures that fair arbitration is performed when resolving conflicts.

The *PiRC* performed successfully for all possible combinations of the above scenarios for two sets of virtual channels.

V. Conclusion

A fast, simple router for the Data-Intensive Architecture (DIVA) system has been presented. This device, the *PIM Routing Component (PiRC)*, implements the *Red Rover* routing algorithm to achieve high performance with minimal complexity. The PiRC has advantages of simple logic, one clock node-to-node delay, high channel throughput, and robust speed consistency, regardless of the number of flit buffers in a channel buffer. This combination of attributes makes the PiRC ideal for the DIVA system.

VI. Acknowledgments

The authors would like to acknowledge the support of DARPA (Contract No. F30602-98-2-0180).

References

[1] Jeff Draper and Fabrizio Petrini, "Routing in Bidirectional k-ary n-cubes with the Red Rover Algorithm," *Proceedings of the International conference on Parallel and Distributed Processing Techniques and Applications*, Jun 1997, pp. 1184-93.

[2] Jeff Draper, "PDSS Router Design," *Technical Report. Information Sciences Institute/USC*, 1996.

[3] Jeff Draper, "The Red Rover Algorithm for Deadlock-Free Routing on Bidirectional Rings," *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications*, August 1996, pp. 345-54.

[4] William J. Dally and Charles L. Seitz, "Deadlock-Free Message Routing in Multiprocessor Interconnection Networks," *IEEE Transactions on Computers*, May 1987, pp. 547-553.

[5] Mike Galles, "Scalable Pipelined Interconnect for Distributed Endpoint Routing: The SGI SPIDER Chip," *Proceedings of the Symposium on Hot Interconnects*, August 1996, pp. 141-146.

[6] Subramanian S. Lyer and Howard L. Kalter, "Embedded DRAM technology," *IEEE Spectrum*, April 1999, pp. 56-64.

[7] Lionel M. Ni and Philip K. McKinley, "A Survey of Wormhole Routing Techniques in Direct Networks," *IEEE Computer*, February 1993, pp. 62-76.

[8] Steven L. Scott and Gregory M. Thorson, "The Cray T3E Network: Adaptive Routing in a High Performance 3D Torus," *Proceedings of the Symposium on Hot Interconnects*, August 1996, pp. 147-156.

[9] Mary Hall et al., "Mapping Irregular Application to DIVA, a PIM-based Data-Intensive Architecture," *Supercomputing*, 1999.