

An Area-Efficient Router for the Data-Intensive Architecture (DIVA) System

Sumit Mediratta, Jeff Sondeen, Jeffrey Draper
USC Information Sciences Institute
4676 Admiralty Way, Suite 1001
Marina del Rey, California-90292, USA
{sumitm, sondeen, draper}@ISI.EDU

Abstract

A key component of the Data-Intensive Architecture (DIVA) is the Processing-In-Memory (PIM) Routing Component (PiRC) that is responsible for efficient communication between PIM chips. This paper presents the design of a low area, delay and power router for DIVA. A 58.5% saving in area and 86% reduction in load on the clock as compared to an earlier PIM router design makes the presented design ideal for use in the second version of DIVA, with low area being a critical design requirement for DIVA. This paper also gives a comparison of the presented design with an earlier PIM router design in terms of delay and power to justify the new design choice.

1. Introduction

The role of interconnection networks is becoming dominant in defining the performance of architecture. The current generation of symmetric multiprocessors (SMP) and massively parallel processors (MPP) depends heavily on the characteristics of underlying interconnection networks and communication mechanism for performing efficiently [1-6]. Interconnection networks have also become crucial for PIM systems such as DIVA [7] that aims to mitigate the processor-memory speed gap. The presented router forms a key component (Fig.1) of DIVA.

Much research has been done in the implementation of high-performance routers, but either the size or complexity of these general-purpose routers [8] or their design for specific purpose [9] makes them unsuitable for use in DIVA. Although some low-area and high-performance interconnection strategies are reported [2 & 6], emphasis in those architectures is on on-chip communication while the router described in this paper focuses on off-chip communication of data.

The router presented in this paper improves on earlier PIM router designs in both key subcomponents that comprise most routers: data FIFOs and FIFO controllers. Besides an obvious improvement of implementing router data FIFOs with SRAM rather than flip-flops, a more efficient FIFO controller was also desired. The router design presented in this paper uses a much more simplified FIFO controller by exploiting the characteristics of the dual-ported SRAM used in the data FIFOs. The result is dramatic improvements in area, speed, and power not only for the controller but also for the router as a whole, especially when coupled with the improvements offered by the SRAM FIFOs.

Section 2 gives a background on the communication mechanism employed in DIVA and current status, section 3 describes the architecture of the SRAM FIFO based router, section 4 reports the performance results of the current implementation and compares them with an earlier short-cut FIFO based design and section 5 draws conclusions from the performance comparisons.

2. DIVA communication mechanisms background

DIVA PIMs represent the first smart-memory coprocessors designed to support general memory-to-memory communication between independent threads of control. Each DIVA PIM contains a commercial DRAM interface, enabling PIM memory to be accessed by host software either as smart-memory coprocessors or as conventional memory.

A separate memory-to-memory interconnect enables communication between PIMs without involving the host processor and is implemented using the PiRC. This interconnect is amenable to the dense packing requirement of memory devices and should allow scalability of the system. For workstation class system sizes (on the order of 32 PIMs), this combination of

requirements favors a one-dimensional network [10]. Future generations of large DIVA-like systems will require a more complex interconnection network and will be addressed in future research.

The communication mechanism in DIVA focuses on system-wide support for memory-to-memory communication, via parcels. A parcel is similar to an active message [11] as it is a relatively lightweight communication mechanism containing a reference to a function to be invoked when the parcel is received. Parcels are distinguished from active messages in that the destination of a parcel is an object in memory, not a specific processor. From a programmer's view, parcels, together with the global address space supported in DIVA, provide a compromise between the ease of programming a shared-memory system and the architectural simplicity of pure message passing.

The basic mechanism used in the DIVA system to support parcel communication is a parcel buffer (or pbuf) (Fig. 1). The pbuf has a virtual as well as a physical abstraction. The pbuf locations appear as regular memory locations to the application. Although the pbuf could be implemented as registers within the PIM node processor, a memory-mapped mechanism for the parcel buffer allows a uniform implementation for both the node and host pbufs. Hence, a pbuf within the PIM chip host interface is memory-mapped into the host processor's address space to allow the host processor to communicate with PIM nodes via the parcel mechanism. Another key point is that the PIM node contains a wide datapath (256-bits) from the processing logic to both memory and the node pbuf that enables high-bandwidth parcel communication.

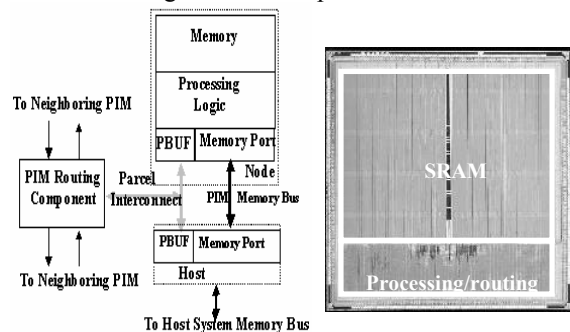


Figure 1. DIVA chip architecture and microphotograph

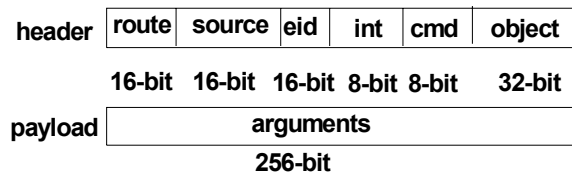


Figure 2. DIVA Parcel Format

The physical parcel format is shown in Fig. 2. A parcel consists of a 96-bit header and 256-bit payload. The user writes most of the parcel contents during a parcel launch; however, the system is responsible for generating the route, source, eid, and int fields. The route is a 16-bit value that is used by the PiRC to direct a parcel to the correct PIM chip and node. The source is a 16-bit value that represents the node ID of the sender and can be used by kernel software to correct routing errors. The eid is the 16-bit environment identifier of the process that launched the parcel, and the 8-bit int field indicates whether the parcel should generate an interrupt at the receiving pbuf. The object is a 32-bit virtual address of the object to which the parcel is directed, and the cmd is an 8-bit identifier that the user can use to index into a table of commands for the specified object. The 256-bit payload consists of arguments for the command task or other data associated with the action specified by the parcel.

Data is written to or read from the pbuf in 256-bit increments via WideWord registers in the processing logic. The pbuf address space can then be viewed as a set of 256-bit registers. Besides the header and payload registers, there are also status and configuration registers. Although the payload is the only true physical 256-bit register, each register is allocated 256 bits of the address space and is aligned to the least significant bit boundary. At least two register sets are needed: one for sending and one for receiving. In addition, it is desirable to have multiple address mappings (aliases) of these sets to support different access privileges, launching and non-launching writes to the send registers, destructive and non-destructive reads from the receive registers, and interrupt capability. The DIVA design includes several aliases to support such mechanisms.

The unique requirements of communication using PIM-to-PIM interconnect led to the design of a short-cut FIFO based router in an initial prototype PIM implementation [10]. An SRAM-based prototype of DIVA with the short-cut FIFO based router design has been fabricated in TSMC 0.18 μ m technology and is operating at 160MHz [7]. The total area of the processing/routing logic is 15.51% of the area of the whole chip (Fig. 1).

The current prototype of the DIVA chip with the short-cut FIFO design is being tested in a development system. SODIMM boards containing 2 PIMs each were fabricated and inserted into an existing, custom PPC603e computer board (Fig. 3) to demonstrate system operation. Many demonstration codes have been ported to this system, including Cornerturn, Transitive Closure, Field, Pointer, and StreamAdd.

The PIM prototype has demonstrated a 35x speedup on the Cornerturn application as compared to the PPC603 of the development system. Instrumented measurement experiments are being conducted for other codes.

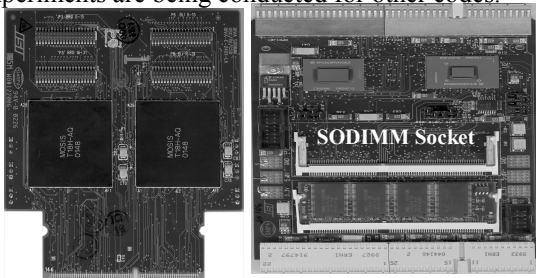


Figure 3. (a) 2-PIM SODIMM Module (b) PPC603e-Based Development Board

The router design using short-cut FIFOs fulfilled the communication requirements of the PIM architecture, but it resulted in large area and power characteristics. Low area and power are crucial design requirements for the processing logic in a PIM chip as all the area and power requirements for the processing logic are overhead in what is first and foremost a memory chip. Moreover, future enhancements in the functionality of the DIVA PIM design, e.g. addition of single-precision floating-point units, demanded all the subcomponents to be area efficient. Additionally, lab measurements have indicated the short-cut FIFO implementation of the router is the limiting factor to operating speed of the first DIVA PIM prototype. Reduced area, power, and delay requirements motivated the search for the improved PIM router design presented in this paper.

3. PiRC architecture overview

The presented router is a one-dimensional wormhole router, which implements the Red Rover routing algorithm for deadlock-free routing in bi-directional rings [10][12][13]. It uses source routing to route fixed-size packets and achieves low latency and area as compared to adaptive routing [14]. The PiRC contains three input physical channels and three output physical channels (Fig.4). There is one output and input physical channel to and from the + and - direction each, and one output and input physical channel to and from the processing element (PE). Further, two virtual channels ('A' and 'B') are time-multiplexed onto each physical channel that can be viewed as each router operating with the defined handshaking protocol at the opposite edges of the clock. The control block is responsible for handshaking with the neighbouring PiRCs and processing elements, routing of packets between the input and output channels and fair allocation of the output channel to prevent starvation. The packet length

is of fixed size of eleven 32-bit flits only [7], with the flit size equal to the phit size. This reduces the overhead incurred in handshaking, as handshaking is required only for the transfer of the header flit.

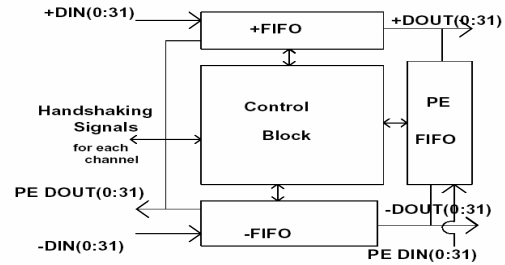


Figure 4. Router Block Diagram

A packet may enter the network from the PE through the + or - output channels, and a packet may leave the network from the + or - direction FIFO through the PE output channel. A packet travelling in the + or - direction cannot reverse directions and go to the - or + output channel, respectively (see [10] Fig. 2), because of restrictions provided by the Red Rover routing algorithm. The following description describes the control logic and functioning of the 'A' level of the router virtual channel. For the 'B' virtual channel, the control logic and data flow are identical, with events triggering at the opposite edges of the clock.

3.1. Switch and output channel control logic

The functionality of the control logic, to determine the next candidate for an output channel in a fair manner, is realized by a finite state machine (FSM). There is a separate FSM for each output channel. The FSM for a + output channel is shown in Fig. 5.

At reset, each output channel starts in the 'Idle' state. The handshaking signal RO, which indicates that the destination router connected to that output channel is ready to receive data, should be asserted for any request to be converted into a grant signal. The priority is given to the +, - and + input channels for +, - and PE output channels, respectively, in case two simultaneous requests occur for an output channel. However, it is assured that the other channel (PE, PE and - input channel in case of +, - and PE output channel, respectively) will be granted the output channel after the current owner finishes sending its packet. It can be seen from the state diagram that after sending the current packet the control logic moves from '+ Grant' state to the 'PE Grant' state if there is a request from the PE input channel, irrespective of the waiting packet on the + input channel to be sent on the + output channel. In this way, this design ensures fair arbitration and prevents starvation.

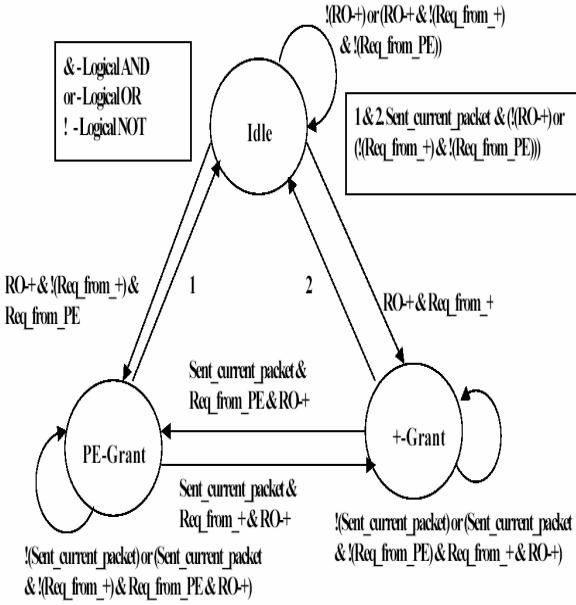


Figure 5. FSM for + Output Channel

The control logic moves to the 'Idle' state if there is no pending request for the current output channel. Otherwise, it remains in the same state if the other candidate input channel is not requesting the output channel and the current owner input channel wants to send another packet. Let us denote the grant signal for +, - and PE input channel to send the data on the +, - and + output channels, respectively, by 'Upper output channel enabled' and grant signal for PE, PE and - output channels, respectively, by 'Down output channel enabled'. Please notice that these signals signify the enabling of sending data on 'Upper output channel' and 'Down output channel' from the input channel point of view, and they are just the grant signals generated by the shown FSM and two other FSMs (for -ve and PE output channel) connected appropriately to control each FIFO's output data.

3.2. Handshaking signals control logic

Two handshaking signals RI (ready signal at the input channel) and SO (send signal at the output channel) needs to be generated at each node. The other two-handshaking signals RO (next node ready signal at the output channel) and SI (previous node send signal at the input channel) are in fact the former two signals coming from different nodes. The RI signal is kept asserted as long as the corresponding FIFO is empty. The sender keeps on polling the RO signal at every edge of the clock, and starts sending a message upon detecting it to be asserted. The sender asserts SO as the next step of the protocol to indicate sending of valid

data. The receiver stores corresponding DIN data into the FIFO upon sampling the corresponding asserted SI signal, and then it captures data on the next ten clock cycles to receive the entire packet.

The control logic for SO generation is shown in Fig. 6. The SI signal, indicating the arrival of a header flit, is latched (signal 'SI_Lat') and is not allowed to assert a signal, 'SI_Header', (used for header processing like decrementing the hop count, etc.) until any current packet is read out completely (this is controlled by the read pointer of the SRAM). Here, a dead cycle in turning around from reading of one packet to the reading of another packet is prevented by initiating the control logic for the assertion of 'SI_Header' while the last flit of any previous packet is still being read. The 'Upper output channel' and 'Down output channel' are then used in conjunction with 'SI_Header' to produce the SO signal (produced by performing an OR of the SO_up and/or SO_down signals coming from different input channels) for the desired and granted output channel. The 'SI_Header' and 'SI_Lat' are then reset to the initial position when the FIFO starts sending data to the destined output channel.

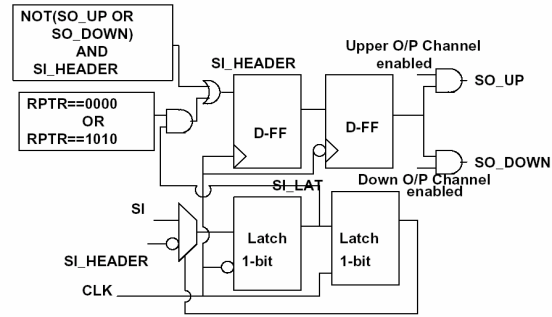


Figure 6. Control Logic for SO Signal

The control logic for the generation of RI is shown in Fig. 7. It is asserted at reset to show that the corresponding FIFO is empty and it is ready to accept a packet. When the 'SI_Header' signal is asserted then the control logic determines the next state of RI depending on the blocked or unblocked state of the current packet.

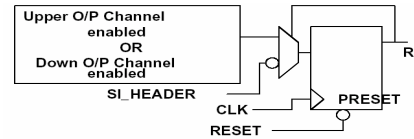


Figure 7. RI Signal Generation

3.3. Read/write control logic of SRAM FIFO

The control logic for an SRAM-based FIFO (Fig. 8) is much simpler and results in less area and delay as compared to the short-cut FIFO based design. There is a 12x32-bit dual-port (one read port and one write port)

SRAM for each of the six FIFOs to fulfill the design requirement of having enough buffer space to hold a packet. Two 4-bit up-counters, which count from 0 to 10(decimal), generate the write and read pointers for the writing and reading of the SRAM. Two separate counters for the above purposes ensure that read and write operations can be carried out independently. This is necessary to write a new packet while a current packet is being read out.

The writing logic consists of generating a 'wptr' (write-pointer) and a 'wen' (write enable) signal. The writing of 'A' virtual channel data occurs at the +ve edge of the clock. So, the 'wptr' is incremented at the -ve edge of the clock and 'wen' also becomes available sufficiently before the setup time requirements of the 'wen' signal for the SRAM. The condition for the generation of 'wen' signal is that the SI signal should be asserted when 'wptr' is 0. It thus initiates a series of ten writes for the immediately following ten flits. The 'wen' again gets de-asserted when 'wptr' returns to 0 after counting up to 10 if the SI signal is not activated.

The reading of the SRAM occurs at the -ve edge of the clock for 'A' virtual channels. This design requirement to fulfill the write-read clock latency of the SRAM is well suited for the specified handshaking protocol, which dictates the writing of the data in the FIFO at the +ve edge, and expects data at the -ve edge of the clock. The 'rptr' (read-pointer) value is updated at the +ve edge of the clock to make the read address available to the SRAM and adequately meets the setup and hold time requirement on the address port. The 'rptr' is incremented if the desired output channel is granted to this input channel. This requirement is fulfilled by using the 'Up output channel enabled' and 'Down output channel enabled' signals.

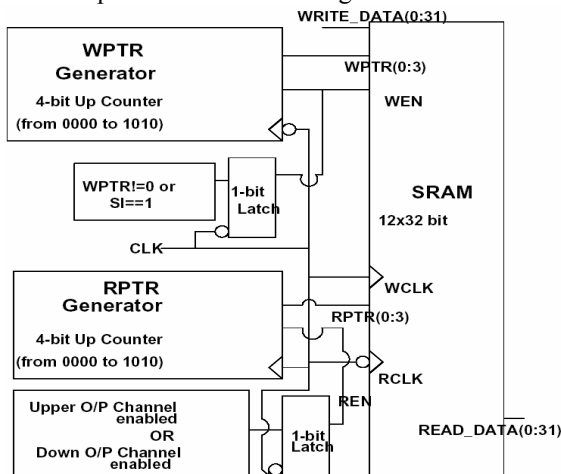


Figure 8. SRAM Read/Write Control Logic

4. Performance evaluation

The current design is specified in RTL-level behavioral VHDL code. The code was synthesized using Synopsys Design Compiler targeting the Artisan standard cell library for TSMC 0.18 μ m technology. The resulting netlist was placed and routed using Cadence Silicon Ensemble. A Silicon Ensemble layout of the router design is shown in Fig. 9. The current design's area constitutes 1.61% of the total processing logic and 0.25% of the total chip area.

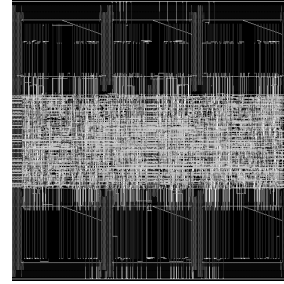


Figure 9. Layout of SRAM FIFO based PiRC

To quantify and justify the assumption of better area, delay and power performance in the proposed design, a similar procedure was followed to obtain performance metrics of the short-cut FIFO based design in TSMC 0.18 μ m technology instead of its reported metrics in HP14b process that is a 0.5 μ m, 3-metal process. The results are summarized in Table 1.

	Short-cut FIFO based Design	SRAM FIFO based Design
Transistors	125360	55188
Area	895 μ m X 645 μ m	665 μ m X 360 μ m
Delay (ns)	5.42	4.00
Power (mW)	116.43	53.96

Table 1. Performance Comparison of Two Designs

It is self-evident from the performance comparison of the two designs that the proposed design of this paper is better in terms of all the metrics listed above. The huge savings in area (58.5% of the short-cut FIFO based design) is the main asset of this design because of emphasis on lower space utilization for the processing logic as a key design specification for DIVA. The saving in area is mainly due to the smaller relative size of an SRAM cell (6 transistors) as compared to a flip-flop (approximately 20 transistors). Simpler control logic as compared to the previous design, where a significant amount of logic was needed for keeping track of the header flit, also contributes to the area reduction.

Also, the load on the clock has decreased to 86% of its value from the previous design. This improvement arises because of the great reduction in the number of transistors in the clock tree. This also justifies the reduction in power dissipation (measured assuming 20% utilization factor).

Delay reduction is justified by the simpler control logic and use of high-speed SRAMs with lesser load on the clock signal. The presented router is able to operate at 250MHz using 1.8V CMOS signaling. Its operation at both the clock edges leads to a channel bandwidth of 16Gb/s for a 32-bit wide channel.

5. Conclusion

This paper has presented the design of a PIM router with a simplified controller exploiting SRAM-based FIFOs for the DIVA project. This design has lower area, delay and power; much simpler control logic; higher throughput and lesser load on the clock than a previously reported and implemented PIM router design. In addition, it features one clock node-to-node delay. These features make this design a natural choice for assimilation into the next generation of the DIVA PIM prototype.

A PIM chip design incorporating this router has taped out for fabrication in October of 2003. DIMMs containing these PIMs are to be inserted into a Hewlett-Packard Itanium2-based Long's Peak server. The quantification of the performance enhancement achieved with the new design will be of great significance to the designers of the modules with communication requirements similar to DIVA. Although the DIVA project employs the router as part of a tightly integrated PIM chip design, the current router is also suitable for stand-alone embedded implementations.

6. Acknowledgments

This research was supported by DARPA contract F30602-98-2-0180.

7. References

[1] Duato J., Lopez P. et al. "A High Performance Router Architecture for Interconnection Networks",

- International Conference on Parallel Processing*, 1996, pages 1-61 - 1-68
- [2] Mai K., Paaske T., Dally W. J., Horowitz M. et al. "Smart Memories: A modular Reconfigurable Architecture", *ISCA*, Vancouver BC, Canada, 2000, pages 161-171
- [3] Charlesworth A., "The Sun Fireplane SMP Interconnect in the Sun Fire 3800-6800", *Hot Interconnects 9*, 2001, pages 37-42
- [4] Charlesworth A., "STARFIRE: Extending the SMP Envelope", *IEEE Micro*, Volume 18, 1998, pages 39-49
- [5] Kessler R.E., Schwarzmeier J.L., "CRAY T3D: A New Dimension for Cray Research", *Comcon*, Spring 1993, pages 176-182
- [6] Taylor M. B., "The Raw Processor Specification (LATEST)", *Comprehensive specification for the Raw processor*, Cambridge, MA, 2002
- [7] Draper J., et al., "The Architecture of the DIVA Processing In Memory Chip", *International Conference on Supercomputing*, June 2002
- [8] Koyanagi Y. et al, "Synfinity II - A High-Speed Interconnect with 2GBytes/sec Self-Configurable Physical Link", *Hot Interconnects 9*, 2001, pages 23-29
- [9] Galles M., "SPIDER: A High - Speed Network Interconnect", *IEEE Micro*, Volume 17, 1997, pages 34-39
- [10] Kang C.W., Draper J., "A Fast, Simple Router for the Data-Intensive Architecture (DIVA) System", *IEEE Midwest Symposium on Circuits and Systems*, 2000, pages 188-192
- [11] Eicken T. von, Culler D. et al, "Active messages: a mechanism for integrated communication and computation." *Proceedings of the 19th Annual International Symposium on Computer Architecture*, May 1992.
- [12] Draper J., "The Red Rover Algorithm for Deadlock-Free Routing on Bidirectional Rings", *International Conference on Parallel and Distributed Processing Techniques and Applications*, August 1996, pages 345-354
- [13] Draper J., Petrini F., "Routing in Bidirectional k-ary n-cubes with the Red rover Algorithm", *International Conference on Parallel and Distributed Processing Techniques and Applications*, June 1997, pages 1184-1193
- [14] Yoshinaga T. et al., "A Cost and Performance Comparison for Wormhole Routers based on HDL Designs", *International Conference on Parallel and Distributed Systems*, 1998, pages 375-382