Design and Implementation of a Multicomputer Interconnection Network Using FPGAs*

Chun-Chao Yeh, Chun-Hsing Wu, and Jie-Yong Juang
Department of Computer Science and Information Engineering
National Taiwan University
Taipei, Taiwan, ROC
Fax:886-2-3628167
Email: juang@csie.ntu.edu.tw

Abstract

In this article, we present an experience of using FPGAs in the design and implementation of a multicomputer interconnection network. The switching element, the router and the network controllers are all designed and implemented with FPGAs, and cooperatively form a four-port by four-port interconnection network. The switching elements were designed with ASIC before, but were not very successful. Advantages of using FPGAs over traditional ASIC design will be discussed in this article.

Keywords: *FPGAs, Multicomputer, Cell-based Interprocessor Communication*

1 Introduction

Distributed memory multicomputer systems provide a cost-effective way toward high performance computing [1, 2]. Performance of such a system highly depends on the efficiency of the underlining interconnection network. Many interconnection networks have been proposed in the literatures[3, 4, 5, 6]. But, only a few were implemented, and most of them were realized in ASIC chips. Traditional ASIC (custom IC or standard cell) development processes are expensive and complex. Moreover, these ASIC chips are not programmable. Even a minor modification would require a new iteration of design and fabrication. The problem will be worse for a complicated design such as a multiprocessor interconnection network since it involves complex interaction between hardware and software. Thus, most research efforts were conducted with software simulation only. No actual machine was build. However, a simulated architecture with simulated network traffic patterns is usually difficult to reveal subtle situations vital to the behavior of the machine. It is even difficult to codesign with system software based on simulation. Thus, it is usually necessary to verify a design with a prototype system after simulation.

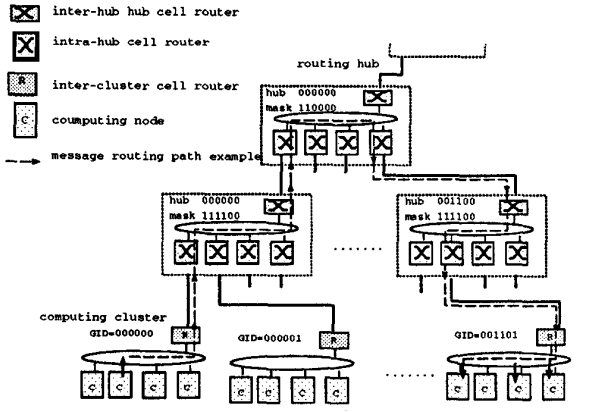
The advent of FPGAs technologies makes it possible to prototype novel systems with minimal efforts[7]. In the past few years, we were working on the development of a multicomputer system for I/O-intensive applications[8]. The interconnection network of the multicomputer system is a cell-based architecture with fix-sized cell format which makes network controllers simple and easy to design. It also results in a data transfer latency shorter than that of traditional message passing interconnection network. Message flow is also easier to control. The cell-based interconnection network was implemented with FPGAs. Our experience confirms that design with FPGAs provides many benefits in the areas of hardware/software codesign, architecture trade-off studying, and system debugging. We will discuss these points in more detail in the following sections.

2 Interconnection network

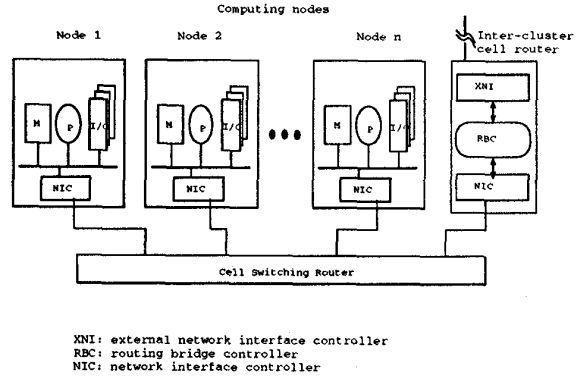
2.1 Interconnection network architecture

In our multicomputer design, system is partitioned into several clusters interconnected by an external network composed of cell routing hubs(Fig. 1.a). The structure of a cluster is shown in Fig. 1.b. An internal interconnection network was designed to connect computing nodes and a inter-cluster cell router in a cluster. The internal network and the external network share the same architecture which supports cell-based communication. A message may be divided into multiple cells of fixed size. Each cell carries a routing tag in the cell header to identify destination nodes.

*This work was partially supported by the National Science Council under grants NSC82-0408-E-002-575, NSC83-0408-E-002-002, and NSC84-2221-E-002-004



(a) Architecture of a scalable multicomputer



(b) A computing cluster of the multicomputer system

Figure 1: Architecture of a scalable multicomputer

A destination can be in the same cluster or in a remote cluster. Cells for the nodes in the same cluster will be received by the nodes directly. Cells for the nodes in remote clusters will be forwarded to routing hubs in the external network by the inter-cluster cell router , and then be routed to the destination clusters through the network. Routing hubs can be collectively organized into a tree, a ring or other topologies.

Both interconnection networks were designed and implemented with Xilinx FPGAs. Due to the gate count limitation of the FPGAs devices, most of the CAL(Cell Adaption Layer) functions of the cell-based communication network (such as cell partition/assemble, checksum, etc.) are implemented in software. Underlining hardware realizes only the essential parts of the networks. Four major

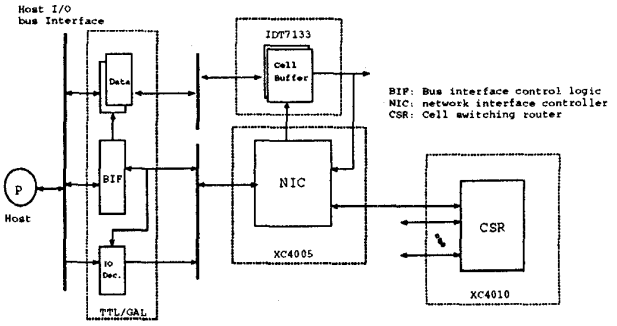


Figure 2: Block diagram of the internal interconnection network

parts of them are: Network Interface Controller(NIC), Cell Switch Router(CSR), Cell Buffer(CB), and Bus Interface Control Logic(BIF)(Fig. 2).

2.2 Cell-based communication

For transmission of a message, the cellized data units are forwarded to the Cell Buffer by the host DMA. Once these cells are ready for sending, the Network Interface Controller(NIC) sends a routing request to the Cell Switch Router(CSR), and waits for an acknowledgement from it to enable the transmission. The CSR tries to set up a connection upon the request. When it is done, the NIC is signified to start delivering the cells continuously until no more left in the buffer. However, it may be interrupted upon events such as out of buffer space for incoming cells. When it is interrupted , the transmitting connection will be disconnected temporarily to release the channel resources, and will be reconnected after all of the events are handled. The cell format is shown in Fig. 3. It consists of sixty four bytes, four bytes for header and sixty bytes for data. In our prototype, a cluster consists of four nodes, each of which can be addressed by setting a corresponding bit in the destination node(DNO) field. Multicasting addressing can be done by setting the bits corresponding to the destinations in the DNO. The destination group ID(DGD) field carries the address of the destination cluster.

3 Design with FPGAs

We use commercially available dual port RAM as the cell buffer, and host bus interface logics were implemented with TTL/PLD devices. The two core control modules, NIC and CSR, were designed with FPGAs.

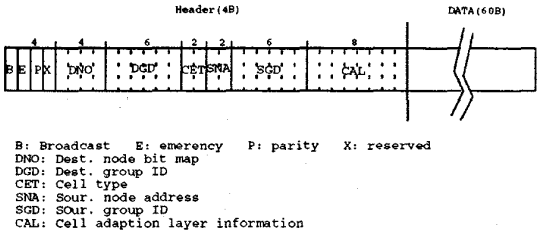


Figure 3: Cell format

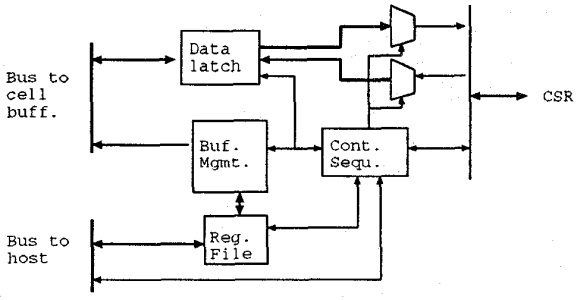
3.1 Network Interface Controller(NIC)

Functional block diagram of a NIC is shown in Fig. 4.a. It consists of four major modules: data Mux/DeMux, cell buffer management, control sequencer, and a register file. The bus to the cell buffer memory is of sixteen bits in width, while the bus to the cell switching router contains only four data lines per port due to the limitation of the pin count of the FPGAs implementing the CSR. A data Mux/DeMux is needed to convert these two data streams of different widths. Cells in the cell buffer are organized into two ring structures, one for transmitting, the other for receiving. Cell access is managed by a FIFO mechanism. The data structures and access control are maintained by a cell buffer manager. The data structure as well as the two sets of ring buffer pointers are stored in the register file. These modules are coordinated by the control sequencer. The control sequencer also handles link flow control signals, routing requests, and cell arrival notification.

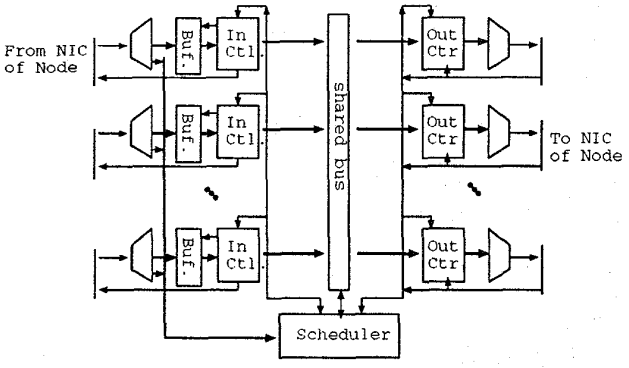
The NIC was implemented using Xilinx 4005PG156-6[10]. Some design statistics are presented in Table 1. Notice that many of the IO pins are allocated to provide two sets of address lines, control signals, and data lines of the cell buffer memory.

3.2 Cell Switching Router(CSR)

Routing of a cell is carried out by CSRs. The customized CSR is a four-port by four-port shared bus switch. Each port contains four data lines and two control lines; one is for data valid, the other is for port ready, and is connected to the NIC of a computing node. Block diagram of the switching router is shown in Fig. 4.b. The internal shared bus has sixteen data lines. Consequently, a 16 to 4 de-multiplexer is needed in the input module. Communication with NIC through a port of the switch is asynchronous, but the internal bus operates in TDMA(Time Division Multiple Access) mode and is synchronized with each port. Thus, a few latch buffers are required for slot matching. Besides, some control logic is designed to handle flow control and bus access.



(a) NIC (Network Interface Controller)



(b) CSR (Cell Switching Router)

Figure 4: NIC and CSR block diagrams

FPGAs Design Statistics						
Design	NIC			CSR		
FPGAs	XC4005PG156			XC4010PG191		
	Used	Max	Util	Used	Max	Util
CLBs	108	196	55	353	400	88
Pins	107	112	95	50	160	31
Flip Flops	125	616	20	612	1120	54
Equ. Gates	2319	-	-	7853	-	-

Table 1: Placement and route result statistics of the FPGAs design

The structure of the output module is similar to that of the input module. Since the output port is well synchronized with internal shared bus, no slot matching buffer is needed. Instead, Some bus data fetch logic is needed to receive correct data from source input port. Cell routing requests are scheduled by the scheduler module. If the destination port is not ready for the request, the scheduler will reject the request by asserting a port-not-ready signal to the requesting NIC. Otherwise, it replies to the request with proper control signals to allow the NIC to put the data in the channel. In the mean time, it notifies the destination port to listen to the channel. The scheduler also monitors port-ready signals from all NICs. When a destination NIC is not ready, the scheduler disconnects the connection and sends a notice to the source NIC.

The CSR is implemented using Xilinx 4010PG191-6[10]. Some design statistics are presented in Table 1. Comparing with NIC, CSR needs more logic to realize. Significant portion of Flip Flops is used for Mux/DeMux, shared bus buffer latch , and latches for channel slot matching. The Table shows resource usage for only one CSR.

The CSR described above has only 4x4 ports. Larger scale switch can be constructed from the basic switching elements to form a multistage switch network.

4 System development with FPGAs

As system software and hardware getting more complex, design confirms with software simulation might not be enough and efficient. Short design turnaround time and reprogrammable design with FPGAs allow us to prototype a novel system architecture more efficiently. In our previous work, we have designed and implemented a switching element of 8x8 ports using standard cell ASIC[9]. Unfortunately, we found some architecture features missing in the design of the ASIC chip when we applied it to implement the interconnection network. We were forced to redesign it with anther expensive processing cost, and still couldn't get it work properly. Contrast to this, our experiences show that design with FPGAs has many benefits in system development, including: system hardware/software codesign, architecture trade-off study, and system debugging.

In the development of a high performance system, usually it is needed to tune the software and hardware iteratively, and make choices between alternative architectures. For example, in our prototyping system, system parameters such as cell size, the time to issue an interrupt, buffer size, etc. are designed into hardware to minimize the overhead, but their effects are hard to evaluate precisely in the earlier stage of system development. Tuning of these parameters has to be done when software and applications are developed. There are also other architecture features crucial to software, and whether they can deliver the expected performance heavily depends on the behaver of the software.

Up to now, our interconnection network has been modified several times upon the feedback from software development. Major changes are summarized in Fig. 5. The first version provides only primitive cell-delivering functionalities. In this version, the processor was interrupted for every incoming cell to prevent cells from pending in the buffer. However, we found that it caused too many interrupts, and resulted in a heavy processing load for CPU to handle them. In version 2, we changed the design in which an interrupt was triggered by a watch dog timer so that enough number of incoming cells can be accumulated in the cell buffer before an interrupt is generated. An interrupt can also be generated when an urgent cell, indicated by an urgent flag in the cell header, arrives. Besides, in the first version, each cell is scheduled to route independently. If there are more than one cell from different sources destined to the same port, arbitration is made by routing scheduler with a random choice. Random choice is a fair scheme. However, we found that its overhead is too large comparing with a packet-based network, as cells associated with a message might be widely scattered in the cell buffer and connections are likely to be torn down and reconnected frequently during a message transmission. For this reason, we adopted a new scheduling strategy which takes channel connection into account. In version 3, once a connection is setup, the node owns the connection until it releases it. To avoid an idle connection, if cell transmission through connection stops for some reason(for example, receiving buffer overflow), the connection will be disconnected temporarily to release channel resource for others, and it will be reconnected automatically later. Also, we found that current cell size(64 bytes) is not efficient in transporting large volume of data as in file transfer applications. We are going to make the cell size programmable by CPU to maximize the data transfer efficiency for different applications.

These architecture modifications would had resulted in

Version 1	*Primitive functions for cell communication *Interrupt -one interrupt per cell *Routing arbitration: -random
Version 2	*Interrupt -watch dog timer -urgent cell
Version 3	*Routing arbitration -connection-based *Channel disconnection/reconnetion

Figure 5: Major changes in the evolution of the interconnection network

Nodes	Four nodes + One cell router
Sys. clock	10MHz
Network bandwidth	40 Mbps per port
Board size	32cm x 28cm
FPGAs chips	Four XC4005's + One XC4010

Table 2: System features of the interconnection network board

system redesign and reprocessing if conventional ASIC development method were used. However, we spent only several hours in the redesign of FPGAs to modify hardware and add new features in each modification. Furthermore, system debugging is much easier with FPGAs design. During the system development, we are able to start with a simplified design, and evolve it to the target architecture gradually. We can also introduce extra pins to expose crucial information out for testing to help debugging. For example, to debug a cell routing fault, we had reduced the CSR to a 2x2 switch, and turned off unrelated features in the early stage. With such a design, we were able to save FPGAs resource for implementing self-checking routines and latches to latch relevant internal status. Consequently, internal status can be monitored by a Logic Analyzer easily.

5 Remarks

Although, a single FPGAs chip is still difficult to host a complex system due to limited gate counts, with proper system partition and design trade off between hardware and software, we had successfully designed and implemented a multicomputer interconnection network with FPGAs. From our design and implementation experience, we showed that design with FPGAs has many advantages over a design with traditional nonprogrammable ASICs.

References

[1] J. M. Hsu, and P. Banerjee, "Performance Evaluation of Hardware Support for Message Passing in Distributed Memory Multicomputers," *in Proc. Int. Conf. on Para. Proc.*, 1991.

[2] M. D. Noakes, D. A. Wallach, and W. J. Dally, "The J-Machine Multicomputer: An Architectural Evaluation," *in Proc. Int. Symp. on Comp. Arch.*, pp. 224-235, 1993.

[3] H. Ahmadi, and W. E. Denzel, "A Survey of Modern High-Performance Switching Techniques," *IEEE J. Sel. Areas Commun.*, SAC-7, pp1091-1103, 1989.

[4] F. T. Leighton, "Introduction to Parallel Algorithms and Architectures: Arrays • Trees • Hypercubes," *Morgan Kaufmann Publishers*, San Mateo, CA, 1992.

[5] Y. Yeh, M. Jluchy, and A. Acampora, "The knowck-out switch: A simple Modular Architchture for High-Performance Packet Switching," *IEEE J. Sel. Areas Commun.*, vol. SAC-5, pp.1274-1283, 1987.

[6] H. Kuwahara, M. Ogino, N.Endo, and T. Kozaki, "A Shared Buffer Memory Switch for an ATM exchange," *in Proc. Int. Conf. on Commun.*, pp. 118-122, 1989.

[7] B. F. Fawcett, "System-Integration Features and Development Tools Key to FPGA Design," *J. Microprocessors and Microsystems*, vol 18, num. 9, pp. 547-560, 1994

[8] C. C. Yeh, J. T. Lin, W. C. Kao, C. H. Wu, and J. Y. Juang, "A Multicomputer Server for I/O-Intensive Apllications," *to appear in 12th IASTED Int. Conf. on Applied Informatics*, Austria, 1995.

[9] Y. J. Lin, J. M. Ho, C. C. Yeh, and J. Y. Juang, "Design of a Switching Module for Large-Scale ATM Switch," *in Proc. Int. Conf. Para. and Distri. Sys.*, Taiwan, pp. 399-408, 1993.

[10] Xilinx XC4000 Databook, Xilinx, inc. 1991.