DESIGN OF A CMOS LINK ADAPTER FOR CIRCUIT SWITCHED MULTICOMPUTER NETWORKS

Indraneel Sarkar
Wipro Information Technology Ltd.,
88, MG Road, Banglore, India

Rajat Moona
Department of Computer Science and Engineering
Indian Institute of Technology, Kanpur, India

Abstract – Communication speed and mode of communication are critical factors that determine the effectiveness of a multicomputer network system. The Link Adapter (LA), presented in this paper, aims to provide a simple and efficient way of interconnecting processors in a variety of multicomputer systems. It has been designed as a custom VLSI CMOS chip and supports many features like selective multicast, high scalability and a simple interface. The LA is versatile enough to be used in any multicomputer configuration with point to point communication links. This is illustrated by an example in this paper.

Introduction

Multicomputer systems provide a way of achieving high performance by exploiting parallelism inherent in applications. A variety of message passing multicomputer configurations have been proposed for various applications[1, 2, 3]. Many systems with point-to-point interconnection use coprocessors to handle the communication[3]. In this paper we present the design of a communication coprocessor for message passing multicomputers. The coprocessor provides four communication channels, one of which is attached to the host processor. Other channels are used to form the point-to-point interconnection network of processors. Multiple coprocessors can be used to expand the number of communication links at each processor. This coprocessor, called link adapter (LA), is versatile enough to be used in a number of configurations and can serve as a general purpose interconnection module in a multicomputer network.

Much work has been done on switch-controllers, routers and buffers primarily for packet-switched computer networks. MCN[4] and Nectar[5] are examples of crossbar switch controllers. MCN is based on the high performance parallel interface (HIPPI)[6, 7]. It uses a separate protocol processor and RAM buffers to store and forward packets on the network. Nectar[5] employs RISC processor based communication boards (CABs) to handle the protocol processing and transfer of data on its optic-fibre network and supports both, circuit-switched and packet-switched, data transfers. DAMQ[8] was proposed as a solution to some of the problems faced in routing and buffering of data in a packet switched network. It uses first-in first-out buffer with a single read port and a single write port to store packets at intermediate nodes. Apart from DAMQ, three other approaches have also been discussed and their relative performances analysed in [8].

The LA, presented in this paper, is designed for a circuit-switched network where data in transit is not stored anywhere at intermediate nodes. A first-in first-out (FIFO) buffer is used at each node to hold incoming data intended for it. For reasons like simplicity of design and ease of use, these buffers have been treated as external devices and are not considered a part of the LA. Any FIFO buffer that meets a simple data transfer protocol, may be used. This protocol is also discussed in this paper. A protocol used for connection set-up between the source and the destination processors, is very simple and naturally supports selective multicast.

The rest of the paper is organized as follows. In the next section, we describe block-level organization of the LA and signals required to interface it with other resources on the network. In this section, we also discuss various design issues, requirements to be met by the FIFO buffer, circuit setup and release protocol. Following this section, we present the implementation details, the simulation results and an example, illustrating the usefulness of the LA.

Design of the LA

As described earlier, LA supports four communication ports, called channels, numbered 0 to 3. A channel, apart from channel 0, is connected to another channel on a different LA via half-duplex links. Channel 0 of each LA provides interface to connect with the host. A FIFO buffer "sits" on the link connecting channel 0 of the LA to its host and stores the incoming communication data intended for the host. The host can use this data as and when required by it. Other channels are identical in all respect except for the priority used in resolving the conflicting requests.

The interface between two LAs requires 13 lines per channel summarized below.

- 8 bidirectional lines for the data. These lines also carry the control bits required during the circuit-

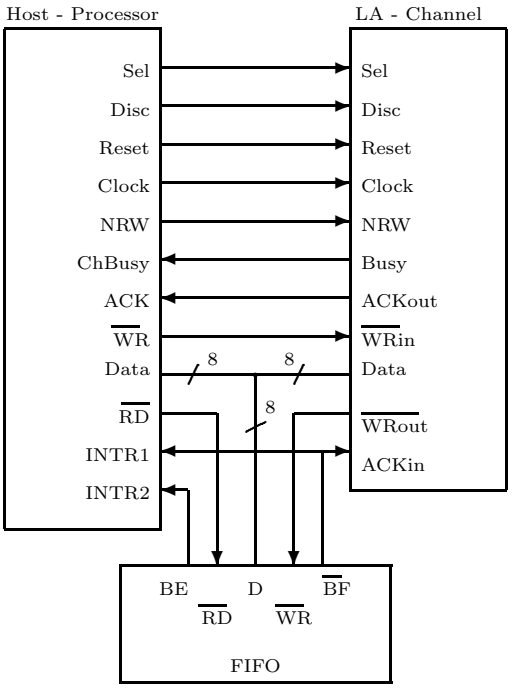

Figure 1: The LA–Host–FIFO Interface

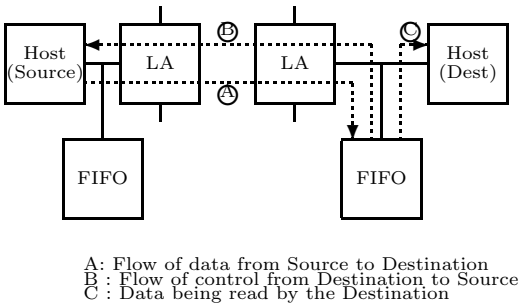

Figure 2: Flow of data and signals on the network

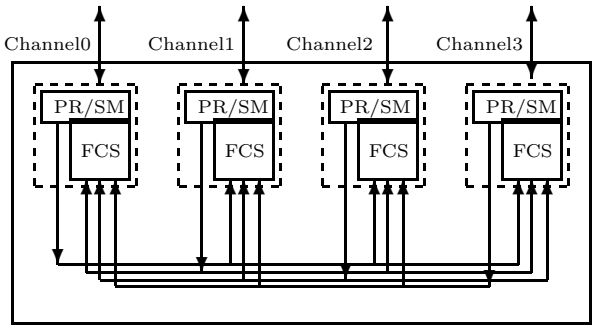

Figure 3: The LA overview

set-up phase of the communication.

- 1 $\overline{\text{WR}}_{in}$ line. Each channel of the LA receives the $\overline{\text{WR}}$ pulse on this line (from the adjacent LA).
- 1 $\overline{\text{WR}}_{out}$ line, over which the $\overline{\text{WR}}$ pulse from the source is relayed on to the next LA.
- 1 ACK_{in} line, required for accepting the ACK signal from the destination LA channel.
- 1 ACK_{out} line for sending ACK signal to the source LA.
- 1 DISC line to signal the release of the circuit.

The LA-Host-FIFO interface (see figure 1) needs few additional lines apart from the above. A message sent from one computer to another may have to pass through various intermediate LAs. After the circuit has been setup between the source and the destination processors, intermediate LAs merely act as relays for the message and the control signals. Transfer of data is carried out under the supervision of the source host computer. At the destination node data gets stored in the FIFO buffer which can later be used by its host. It is the responsibility of this destination FIFO buffer to supply necessary signals for flow-control and prevention of buffer overflow. This information is carried over the chain of ACK_{in} and ACK_{out} lines. A graphic description of the movement of data and the associated control signals from the source to the destination is provided in figure 2.

Figure 3 gives an overview of the LA. Primary components of a channel in the LA are as follows.

- Fully Connected Switch (FCS), through which the channel gets connected to one or more channels.
- Priority Resolver or the arbitration logic (PR), required in case of two different requests arriving simultaneously at a channel, and
- State Machine (SM), necessary to keep track of the present state of the channel.

Design Considerations

The factors which influenced the design of the LA are as follows.

- Our belief that a circuit-switched point-to-point network has lower delays than its corresponding packet-switched equivalent. Communication in message-passing environment being bursty in nature, it is preferable to allow the transfer to proceed to its completion before another is permitted access to the same buffer. This also prevents the interleaving of messages from different sources at the destination. Moreover, a circuit-switched approach obviates the need of buffers at intermediate nodes.
- The need to provide an efficient and natural way of multicasting. In multicast, a message is sent from a source processor to a set of processors on the network. Broadcast is a special case of multicast in which all processors on the network other than the source are included in this set.
- The necessity to keep the interface with the LA as simple as possible. Since the objective was to design a general purpose interconnection module, LA

to host interface is designed such that the signals needed can be provided by any computer. This ensures the adaptability of our design to various processors.

- The restriction placed by the maximum permissible number of pins for the chip. As the LA was conceived as a single chip device, we had to multiplex the data and the control lines to keep the pin number within an acceptable limit. Each channel requires 13 pins. In addition, channel 0 requires four lines for Reset, Clock and LA-Host bidirectional handshake (figure 1). Thus the LA chip can be packaged in a 68 pin die.

The FIFO buffer in LA-Host interface has to meet two very simple requirements. The data path should be 8-bit wide and apart from the usual $\overline{\text{RD}}$ and $\overline{\text{WR}}$, two signals, $\overline{\text{BF}}$ (buffer-full) and BE (buffer-empty), should be available externally. $\overline{\text{BF}}$ is connected to the ACK_{in} line of channel 0 of the LA, while BE is interpreted by the processor to find the presence of a packet in the buffer. These requirements are easily met by most of the commercially available FIFO buffers.

Circuit Set-up and Release

In circuit-switching, entire path from the source to the destination has to be set up before an attempt is made to transfer data. When a host needs to send message, it first sends request for connection to all intermediate LAs on the path. As a LA provides four channels, 4 bit control information is used which also enables a multicast. Thus circuit set-up within a LA is achieved by means of control nibbles. A *'1'* in bit position i of the control nibble indicates a request for connection with channel $3 - i$. Initially, path between the host and the channel 0 of LA is established. A control nibble sent on this establishes a path between channel 0 and other channels. This, therefore, sets up the path till the next LA. Using a succession of control nibbles, the entire path from the source to the destination is established.

Specifically, a source processor carries out the following steps in order to establish the path.

1. Activate the NRW (No Read Write) line (see figure 1). Internally, two signals, NRW from the host and $\overline{\text{BF}}$ from the FIFO buffer, are used at channel 0 to produce ACK_{out} signal. The activation of the NRW line causes a temporary cessation in data transmission.
2. Poll the ChBusy signal. When data is being written to the FIFO buffer, channel 0 asserts this signal indicating the use of the LA-Host-FIFO interface by the LA. If this interface is not in use by the LA, source processor proceeds with its set-up bid; otherwise it deactivates its NRW line and withdraws, to try later.
3. After getting the 'go-ahead' write the first control nibble on the link to the LA. The FCS(s) of the channel(s) whose bits have been set in the control nibble attempt to set up a connection with the requesting channel. The succeeding nibbles are forwarded to all enabled channels.

   A channel in the LA receives control nibble on the lower four lines of its data bus and forwards the succeeding ones on the upper. Thus the data lines are crossed over to interconnect two LAs. For data transmission, however this poses no problem as the cross-over occurs on both, the input and the output channel.

   Once all control nibbles have been written, destination channel 0 relays the status of its ACK_{in}($\overline{\text{BF}}$ bit of the FIFO) and NRW lines to the channel requesting the connection, which in turn, is relayed to the source computer. The entire path gets locked after an affirmative ACK is received. The data transfer may now proceed. If the FIFO overflows before the end of trasnsmission ($\overline{\text{BF}}$=0), the destination computer is notified and the transmission is temporarily suspended. In such a condition, the destination computer should activate its NRW line to ensure temporary cessation of data communication and read the FIFO buffer to create space.
4. After sending all data bytes, the source processor produces a DISC (disconnect) pulse. Following this the path gets unlocked giving access to other requests.

Multicast follows naturally by setting the required bits in the control nibble. Since there is possibility that more than one request for a channel might arrive simultaneously, a priority resolver is built into the switch controller of every channel. We have implemented it as a fixed priority arbiter. Due to such arbitration, a connection set-up bid might get preempted in case the corresponding ACK signal has not yet arrived. The source processor relinquishes its demand by generating a DISC pulse if it does not receive the ACK signal from the destination LA(s) immediately after sending all control nibbles. This clears the intermediate LAs in the path to allow other connection bids.

Implementation and Simulation Results

The LA discussed in this paper was designed on Nelsis IC Design System[9] using 1.6μ scalable CMOS process. Total area occupied by a LA design without the pads is $0.72\text{mm} \times 0.52\text{mm}(0.37\text{mm}^2)$. We carried out the simulation using switch level simulators. In the simulation setup, only a single channel is simulated due to the CPU constraints. The results are extrapolated to analyse the performance of the entire circuit.

- t_s and t_h : Set-up and hold times for the path registers of each channel of the LA = 5ns and 10ns respectively. This demands that a control nibble should be stable for a minimum of 15ns.

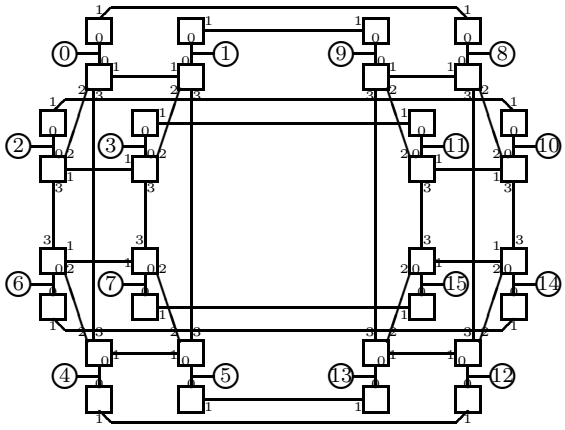

Figure 4: An example of a 16 node hypercube network

- t_{dwr} : Time delay between the appearance of the first $\overline{WR}$ pulse on the $\overline{WR}_{in}$ line of the channel and generation of a request for connection to the desired channel = 4.74ns. Thus to send control nibble to the second LA, it should be stable for $\approx$ 20ns.
- t_{dr} : Time delay between the appearance of an internal connection request (i.e., a request from a channel of the same LA) and its acceptance[1] = 26ns. Thus two control nibbles should be spaced at least 26ns apart.
- t_{dack} : Time required for the processing of the collected ACK signals to be relayed to the previous LA[2] = 3.07ns.
- t_{sw} : Switching time of the FCS = 1.0ns.

If the longest path in the circuit contain k LAs, then the control nibble to the last LA will require to be stable for $(15+5(k-1))$ns or $(10+5k)$ns. Assuming a 60ns write cycle of the processor and presence of buffers in between the LA and processor, a conservative estimate allows upto 5 or 6 LAs in the path. With such restrictions, LA can prove itself useful in moderate sized high speed multicomputer networks. The dynamic power dissipation of each channel is 2.196mW[3].

Applications

The LA can be used in any message passing multicomputer using point to point communication links. In figure 4, we demonstrate its utility in a 16 node hypercube network. In this system, two LAs are used at each processor thus providing upto 6 communication links to the other processors. Channel 0 of each LA is connected to the FIFO buffer and the processor. Two nodes in our implementation of the hypercube are connected, using channel 1 of first LA, in dimension 0; using channel 2 of first LA, in dimension 1; using channel 3 of first LA, in dimension 2; and using channel 1 of second LA in dimension 3. Control nibbles can be found easily. As an example, to establish a path from node 0000 to nodes 0100 and 0001 simultaneously, successive control nibbles will be 0101 and 1000.

Conclusion

The LA has been designed using CMOS technology. It supports a fast circuit setup and communication in circuit switched networks. Extensive simulations carried out at the Indian Institute of Technology, Kanpur, have revealed the usefulness of the LA in multicomputer networks.

References

1. Crowther, W. *et.al.* "The Butterfly Parallel Processor," *IEEE Comp. Arch. Newsletter*, Sep./Dec. 1985, pp. 18–45.
2. Gottlieb, A. *et.al.* "The NYU Ultracomputer - Designing an MIMD Shared Memory Parallel Computer," *IEEE Trans. on Comp.*, **C-32**(2), Feb. 1983, pp. 175–189.
3. Seitz, C. L., "The Cosmic Cube," *CACM*, 28(1), Jan. 1985, pp. 22–33.
4. DuBois, A. J. and J. Rasure, "Design and Evaluation of a Distributed Asynchronous VLSI Crossbar Switch Controller for a Packet Switched Supercomputer Network," *SIGARCH Comp. Arch. News*, 19(4), 1991, pp. 69–79.
5. Arnould, E. A., *et.al.* "The Design of Nectar:A Network Backplane for Heteregeneous Multicomputers," *Proc. of the 3rd Int. Conf. on Architectural Support for Programming Languages and Operating Systems*, Boston, Mass. Apr. 3-6, 1989, pp. 205–216.
6. High Performance Parallel Interface (HIPPI) Mechanical, Electrical and Signalling Requirements, ANSI X3T9/88-127 Rev 7.1.
7. High Performance Parallel Interface (HIPPI) Data Frame Control Requirements, ANSI X3T9/89-146 Rev 2.3.
8. Tamir, Y. and G. L. Frazier, "High-performance Multi-queue Buffers for VLSI Communication Switches," *Comp. Arch. News*, 6(2), May 1988, pp. 343–354.
9. "The Nelsis IC Design System Users' Manual," TU Delft Software Distribution, TU Delft, 1989.
10. de Graaf, A. C., A. J. van Genderen, "SLS: Switch-Level Simulator," The Nelsis IC Design System Documentation, TU Delft, 1988.

[1] if not preempted by a higher priority request

[2] measured from the time the last ACK was received

[3] obtained from switch-level simulation[10]