DESIGN AND IMPLEMENTATION OF A LINK ADAPTER FOR CIRCUIT SWITCHED MULTICOMPUTER NETWORKS

A thesis submitted
in partial fulfillment
of the requirements
for the degree of

Master of Technology

by

Indraneel Sarkar

to the

Department of Computer Science and Engineering
Indian Institute of Technology, Kanpur

February, 1993

CSE-1993-M
SAR-D

13 APR 1993

CENTRAL LIBRARY
Acc. No. A115620

CSE-1993-M-SAR-DES

POST GRADUATE
I.I.T. Kanpur
Submitted on

CERTIFICATE

This is to certify that the work contained in the thesis titled, **DESIGN AND IMPLEMENTATION OF A LINK ADAPTER FOR CIRCUIT SWITCHED MULTICOMPUTER NETWORKS**, was carried out under my supervision by **Indraneel Sarkar** and it has not been submitted elsewhere for a degree.

Dr. Rajat Moona
Asst. Professor
Dept. of Comp. Sc. and Engg.
I.I.T., Kanpur

Abstract

Communication speed and mode of communication are critical factors that determine the effectiveness of a multicomputer network system. The LINK ADAPTER (LA) designed at the Indian Institute of Technology, Kanpur, 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 for the processor and the communication links. This thesis provides the design and a brief behavioural description of the LA. We also provide examples which demonstrate the versatility of the LA for use in a variety of multicomputer configurations.

ACKNOWLEDGEMENTS

With the wheel having turned a full circle, I find myself at a task, my dread of which has not diminished inspite of having gone through it earlier. Exclusion of certain names from this page may be considered intentional.

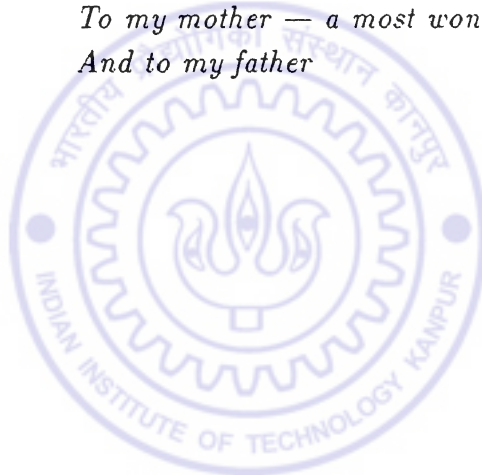
To Rajat (Dr. Rajat Moona, my thesis supervisor) goes the wholesome credit of keeping me constantly on my toes and ensuring the timely completion of my thesis. Whatever I have learnt in VLSI is due to him. Needless to say, it was his course on Advanced Computer Architectures that kindled my interest in the field and decided my thesis topic for me. This report has been typeset on LaTeX for which, Leslie Lamport and Donald Knuth are gratefully acknowledged.

Ananda takes the cake when it comes to entertaining friends. Incidently, it was Ananda who took the pains to go through the LaTeX manual and then patiently explain its subtleties to us ignoramuses. I 'thank' all my friends and classmates for making my stay at IITK whatever it was.

Lest He feel left out, I also acknowledge the tender mercies of the Providence in helping me get this far.

Indraneel
Kanpur
February 18, 1993

To my mother — a most wonderful person
And to my father



Contents

List of Figures

Chapter 1

Introduction

Multicomputer systems offer the potential to exploit parallelism inherent in applications to achieve high performance by decomposing the task into sub-tasks that may be executed in parallel. Dependencies between these sub-tasks are satisfied by means of inter-computer communications. This thesis presents the design and implementation of a coprocessor for communication between computers in a multicomputer network.

1.1 Need for a Communication Coprocessor

A variety of multicomputer configurations have been proposed and developed [2, 6, 10]. A multicomputer system is a collection of high performance computers sharing a common resource, namely, an interconnection network, and communicating through message passing. These configurations differ in the way computers communicate with each other and the manner in which they are interconnected. We shall mostly be concerned with multicomputers wherein the network comprises computers connected to each other by point-to-point links.

There are two major factors that influence the efficiency of communication between computers. They are:

- bandwidth of the communication channel
- latency in communication

The former is more a characteristic of the medium of exchange than of its controller. The latter depends on the demand for the communication

medium and the manner in which it is managed. Low latency is an important criterion in the design of a network of computers for a large parallel computing environment. It becomes a critical factor in the high throughput of the entire system when messages have to traverse multi-hop paths. Networks formed by point-to-point interconnection of computers require the intermediate nodes to process the message and reroute them on to their destinations. This involves fair amount of processing and consumes valuable computing time. In applications where messages have to be exchanged frequently, the intermediate computers would be hopelessly tied up acting as couriers between the source and destination. Moreover, the destination computers need not be involved in the actual data transfer. Messages may be stored in buffers from where they are read at will by the concerned computers. Such considerations necessitate a coprocessor on to which the entire work of message routing, reception and storing may be off-loaded leaving the computer free to carry on with its computational tasks. Thus the structure of the network changes from the simple interconnection of computers to that of computers linked to their respective coprocessors, which in turn are connected to each other to form the network. This is illustrated in Fig. 1.1.

1.2 Features of a Communication Coprocessor

Communication coprocessors (CCP) may be looked upon as small $n \times n$ switches with n ports, called channels. Of these, one channel is connected to the host computer and the remaining $(n-1)$ channels are attached to other coprocessors.

A communication coprocessor is evaluated on the basis of the services offered to the user. The commonly used parameters for evaluation are,

(a) ease of programming

(b) protocol processing efficiency

(c) modes of transfer

(d) pending time of requests

(e) switching times between paths

(f) message rerouting

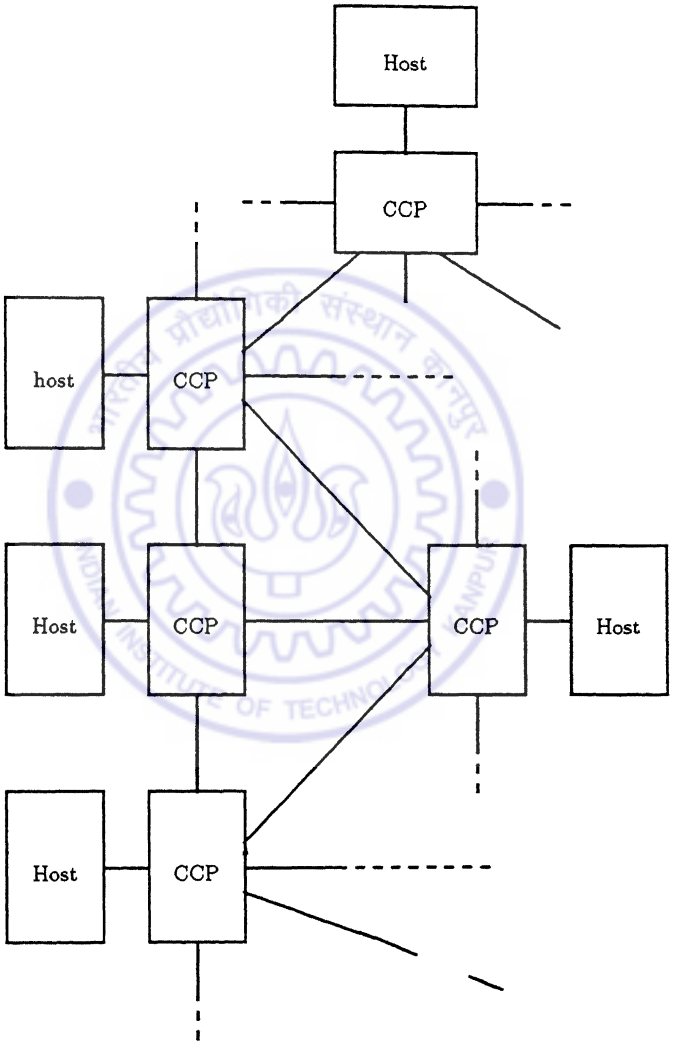


Figure 1.1: Computers linked via Communication Coprocessors

(g) buffering schemes

(h) flow control

Many of these are also dependent on the nature of the underlying network to a large extent.

1.3 Features of the Link Adapter

The communication coprocessor presented in this thesis was designed for point-to-point communication in a circuit switched network. It is named Link Adapter (hereafter referred to as LA) after its major function to manage the links over which its host communicates with other resources on the network. The salient features of the LA are listed below.

- Simple host-LA and LA-LA interface
- High scalability
- Efficient multicasting
- Low switching times
- Implicit rerouting and flow control

The LA was designed as a single chip custom VLSI CMOS circuit. Results obtained from extensive simulations carried out on the circuit demonstrate the usefulness and versatility of the LA in various multicomputer configurations.

1.4 Principles and Tools of VLSI Design

VLSI designs are classically represented in three manners.

- Behavioural specification, where outputs are defined as functions of the inputs.
- Structural specification, where interconnections between components (typically transistors, gates) in a functional block are specified. Once defined, these functional blocks are treated as composite components for hierarchical structural specification.

- Physical representation, where actual layout masks for implementation in silicon are designed. Geometry of the mask patterns are regulated by "design rules" that have to be strictly adhered to. These process dependent design rules may be expressed as:

  1. micron rules
  2. $\alpha - \beta$ rules
  3. λ rules

VLSI implementation of LA was carried out on the Nelsis IC Design System [11]. The Nelsis system is based on the principles outlined in the preceding paragraphs and is not oriented towards a particular technology or process. It provides tools for synthesis, verification and design data management, and supports full-custom, gate-array and standard-cell design techniques. The concept of hierarchical design is followed by means of "cell" definitions. Modularity is achieved with the cell-abutment facility, and cell arrays aid in maintaining regularity of design. Designs may be specified in circuit or layout "views". These correspond to the notions of structural and physical design representations respectively. Matching tools help in verifying that the representation in the two views conform with each other. Physical dimensions for the layout view are expressed in terms of λ-values.

The LA was designed for the c3tu scalable 1.6μ CMOS process with the value of λ set to 0.2μ.

1.5 Organisation of this report

In the next chapter we review some communication coprocessors and buffering strategies for multicomputer networks. The design and operation of LA is presented in Chapter 3. Some of the typical usages of LA are described through examples in Chapter 4. In Chapter 5, we present some simulation results for LA. Finally, we conclude this thesis by discussing the capabilities and limitations of the current design and methods to overcome them in the last chapter.

Two layout plots are included in Appendix A. These pertain to the layouts for the LA and an individual channel. Appendix B contains timing diagrams obtained after the circuit was simulated on the sls [4] simulator.

Chapter 2

Communication Coprocessors: A Survey

In this chapter we review two approaches for communication coprocessors, viz., Multiple Crossbar Network [3] and Nectar [1], and discuss their merits and bearing on our work.

2.1 Multiple Crossbar Network

The Multiple Crossbar Network (MCN) comprises a crossbar switch core and its controller [3]. It was motivated by the need to allow visualisation data to be sent at movie speed rates (800Mbits/s) to a user's workstation from a supercomputer. The MCN is based on the recommendations of the ANSI X3T9 committee, which proposed the High Performance Parallel Interface (HIPPI) in 1989. HIPPI allows for data rates of 800Mbits/s to 1.6Gbits/s. It uses a 32 or 64-bit wide data bus operating at 40ns clock speed and following a simplex data bus protocol to obtain such data rates.

The core of the network entity is a crossbar switch, which allows 32 simultaneous connections if there are no conflicting requests. MCN is meant to be a packet- switched network formed by point-to-point connections. The network has two major components, Crossbar Switch (CBS) and Crossbar Interface (CBI). CBS is a 32×32 crossbar switch with the controller, and the CBI includes protocol processing mechanism, buffers and the HIPPI interfaces. In MCN each host computer is attached to the network via its CBI, which takes care of protocol processing and the storing and forwarding of packets on the network. Figs. 2.1 and 2.2 illustrate MCN and CBI through their respective block-diagrams.

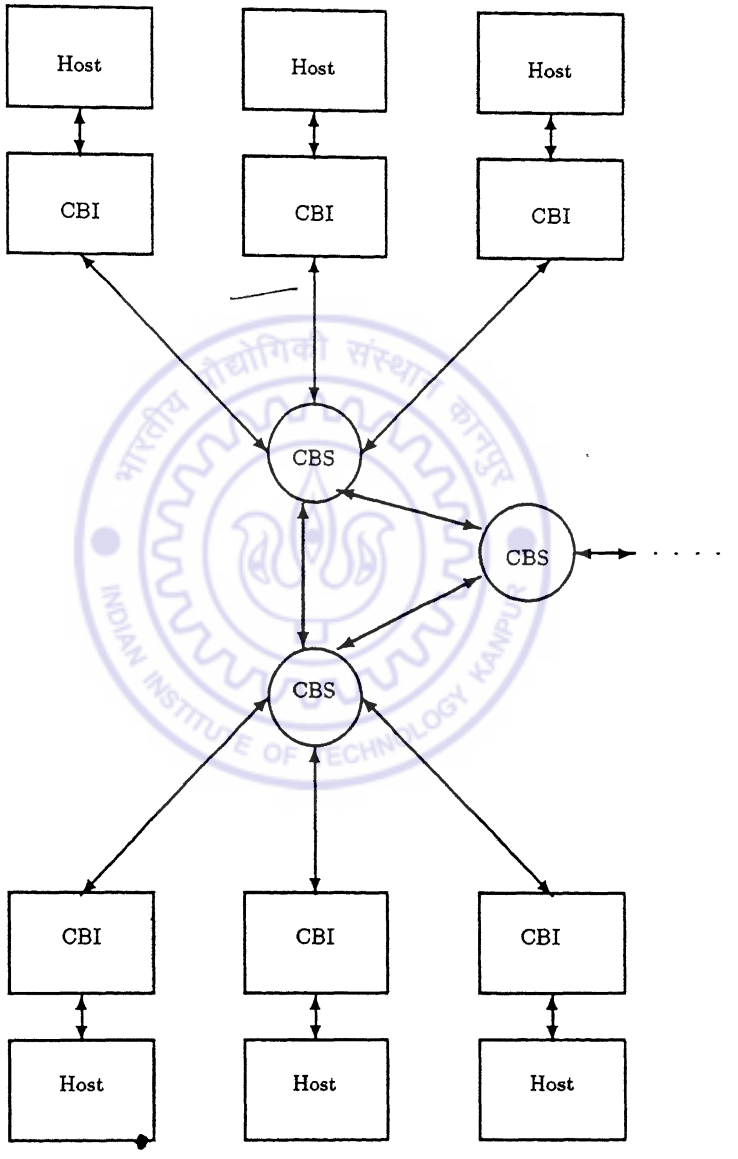


Figure 2.1: Block diagram of a Multiple Crossbar Network

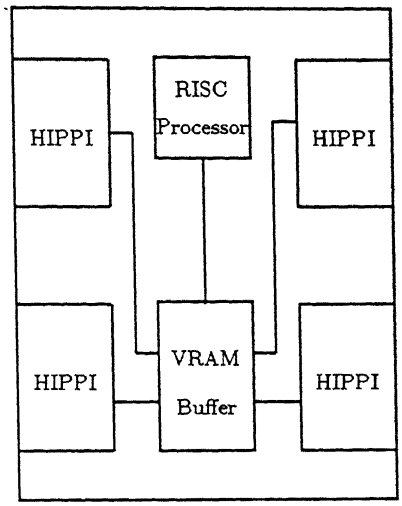


Figure 2.2: Block diagram of Crossbar Interface

2.2 Nectar

Nectar was designed with the objective of exploiting coarse-grained parallelism in programs and as a major component for building heterogeneous systems [1]. Nectar relies on high bandwidth optic-fibres as communication links to support data rates of 1.6 Gbits/s. Various communication protocols and application interfaces are provided to enable proper resource utilisation. Nectar can operate in circuit-switched or packet-switched modes.

Nectar system (refer Fig. 2.3) is centred around the Nectar-net comprising of HUBs and optic-fibre links. HUBS are basically crossbar switches with associated control circuitry and I/O ports to optic fibre links. The structure of a HUB is shown in Fig. 2.4. Computing nodes attach themselves to the net through specialised communication boards, called CABs. The number of HUBs in a Nectar-net determines the overall size of the system.

Nectar offers the user a customisable communication environment that minimises latency in communication. It maintains data-link protocols with the help of CAB–HUB commands. Transport layer protocols are necessary to manage packet transfers and involve decomposition and reassembly of messages, flow-control, retransmission and error-recovery. These are implemented in the Nectar software system.

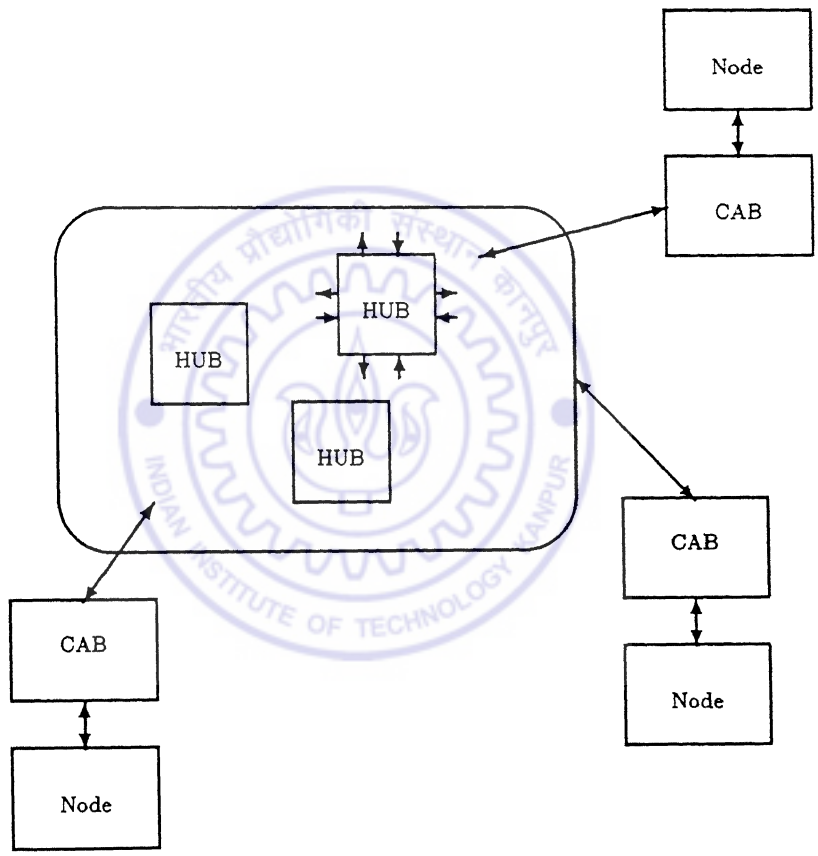


Figure 2.3: The Nectar system overview

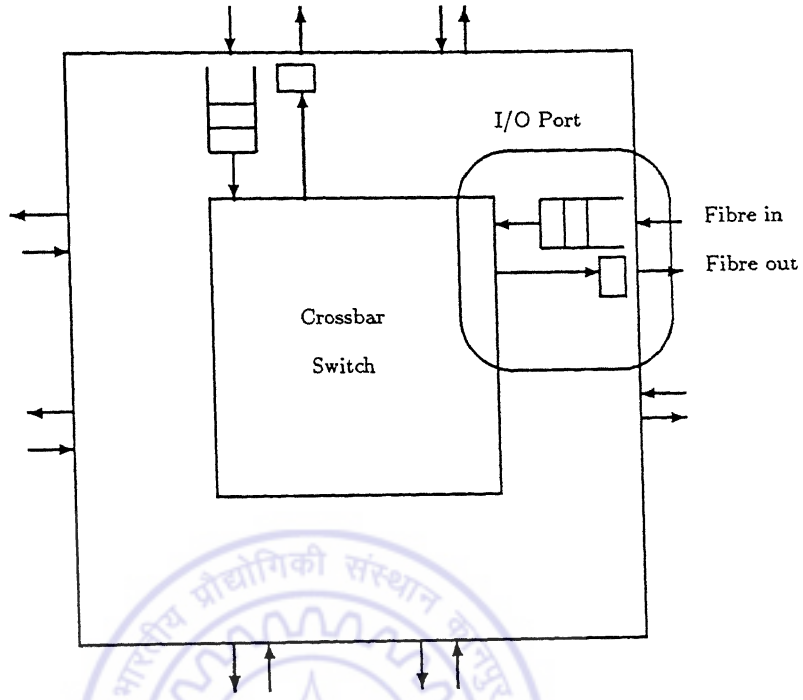


Figure 2.4: HUB overview

2.3 A few comments

The issues involved in the design of MCN and Nectar have a bearing on that of the Link Adapter. These may be summarised as:

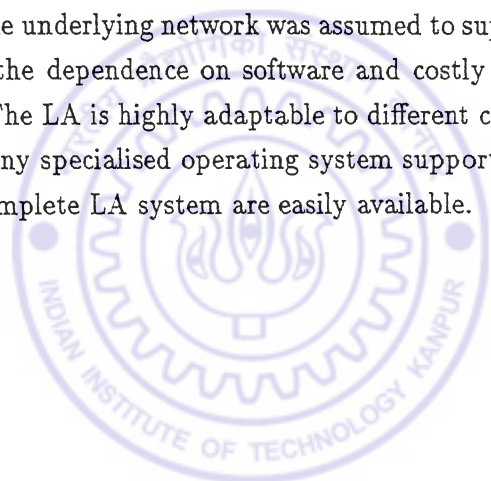
- Adaptability of design – A communication coprocessor system should not favour a particular multicomputer configuration; nor should it be tailored to any specific operating system. It should be a general purpose interconnection aid and provide a cost-effective solution to the problem of inter-computer communication.

- Hardware and software needs – The hardware requirements of the system should be satisfied by off-the-shelf components and their need should justify their cost. Programming and software support is necessary for tasks like path specification, address mapping and especially in packet-switched networks to implement many transport layer utilities and protocols.

- Packet-switched or circuit-switched – A decision has to be taken with regard to the nature of the underlying network. In packet-switching,

buffers are necessary at intermediate nodes to store packets introduced in the network. Another decision has to be taken regarding the point where packets intended for another node are to be stored. The most obvious choice would be the input port of each switch. In this case, appropriate buffering strategy has to be adopted to ensure that packets are not held up due to non-availability of path for packets preceding them in the input queue. Circuit switching, on the other hand, involves the overhead of circuit set-up before data may be transferred. This is acceptable when messages are long (typically, longer than 1KB). Repeated tries may be required to set up the circuit.

Trade-offs were made between the above mentioned points in the design of the LA. The underlying network was assumed to support circuit-switching to minimise the dependence on software and costly hardware for protocol processing. The LA is highly adaptable to different configurations and does not assume any specialised operating system support. Accessories required to build a complete LA system are easily available.



Chapter 3

Link Adapter

The Link Adapter is intended to be a general purpose interconnection module for a circuit-switched network formed by point-to-point connections of computational nodes. It has been designed as a custom VLSI circuit. This chapter presents the design and behavioural description of the Link Adapter (LA).

3.1 Overview of LA

The LA supports four ports, called *channels*, numbered 0 to 3. Each channel is connected to its counterpart on another LA or to its host via half-duplex links. *Channel0* of each LA is special in the sense that this channel is used to attach it to the host computer and the remaining are free for the interconnections necessary to build the network. A buffer connected to *Channel0* completes the LA-subsystem. This buffer stores the messages intended for the host that the subsystem is serving. A first-in-first-out memory (FIFO) with a single read port and a single write port may be used as buffer in the LA subsystem. It is the host's prerogative to use this data as and when required by it. There is no restriction on the manner in which the remaining links (*Channel1*-*Channel3*) are used. These channels are functionally identical; they differ in the priority assigned to each in case of conflicting requests (explained in Section 3.2.2). The primary components of the LA are as follows.

- The Fully Connected Switch (FCS), used to connect two or more resources on the network.

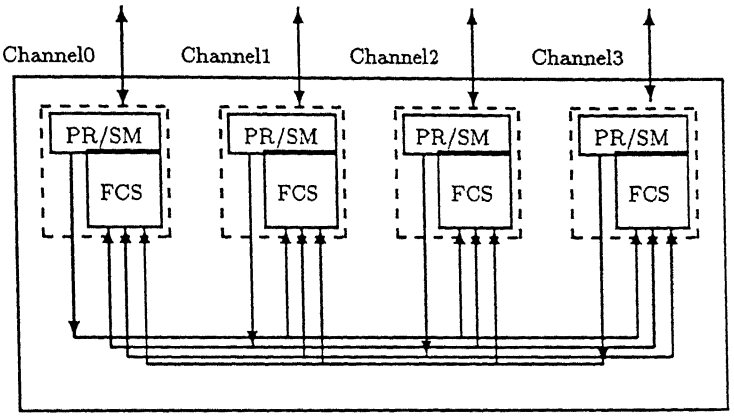


Figure 3.1: The LA overview

- The Priority Resolver (PR), required in case two different requests arrive simultaneously at a channel.
- The State Machine (SM), necessary to keep track of the state in which the channel is at present.

Fig. 3.1 gives an overview of the LA.

The following factors influenced the design of LA.

- Simplicity of circuit-switched networks. A circuit-switched approach obviates the need of buffers at each node on the path. Only the destination LA does the necessary buffering. It also reduces the dependency on elaborate software to keep track of packets and carry out protocol processing.
- The need to provide an efficient and natural way of multicasting. In multicast, a message is sent from the source to any number (selectable) of processors on the network. Broadcast is a special case of multicast in which all processors on the network other than the source are selected as destinations.
- The necessity to keep the interface with the LA as simple as possible. Since the objective was to design a general purpose interconnection module, signals used by the LA interface are the ones which can be provided by most computers. This ensures adaptability of design.

- The restriction placed by the maximum permissible number of pins for the chip. As the LA was conceived as a single chip device, data and control lines were multiplexed to keep the pin number within an acceptable limit.

3.2 Functioning of LA

LA functions can be classified under the following interactions.

1. LA–Host interaction
2. Channel–Channel (intra-LA) interaction
3. LA–LA interaction

The first is specific to *Channel0* as the host communicates over this channel. The other two apply to all the channels.

3.2.1 LA–Host Interaction

The LA–Host interaction involves initiation of circuit set-up by the source, acceptance (grant) of connection by the destination and circuit release after completion of transmission. The LA–Host interface is shown in Fig. 3.2. A Reset signal is issued by the host when it first comes up. This clears all the settings of the LA and makes it ready for use. The LA requires a circuit to be set up before any message can be sent. This is achieved through command nibbles. These nibbles are written by the source host before any data transmission. Since the host shares data lines with its associated FIFO buffer, there remains the possibility of data being put out by it when the buffer is being written to. This is prevented through the use of signals NRW (NoReadWrite) and Busy. The operations performed by the source host prior to the actual transmission of data are enumerated below.

1. Lowering of its NRW line. This causes a temporary cessation in any data transmission that may be going on over the LA–Host–FIFO link (explained in Section 3.2.2).
2. Polling of the ChBusy (ChannelBusy) line. If data is being written to the buffer this signal will be high. Only if the link is free may the host proceed with the circuit set-up bid; otherwise it reasserts NRW signal and withdraws.

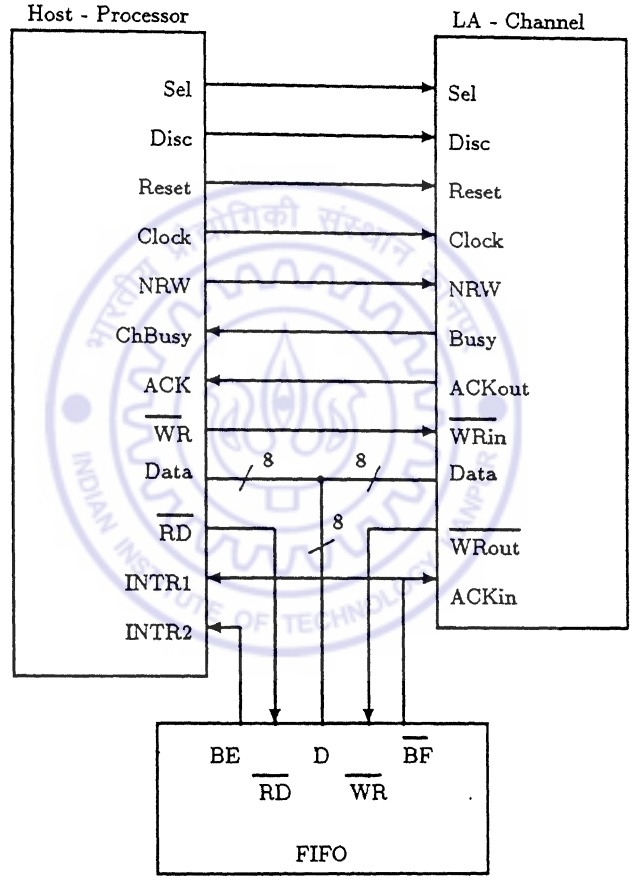


Figure 3.2: The LA–Host–FIFO Interface

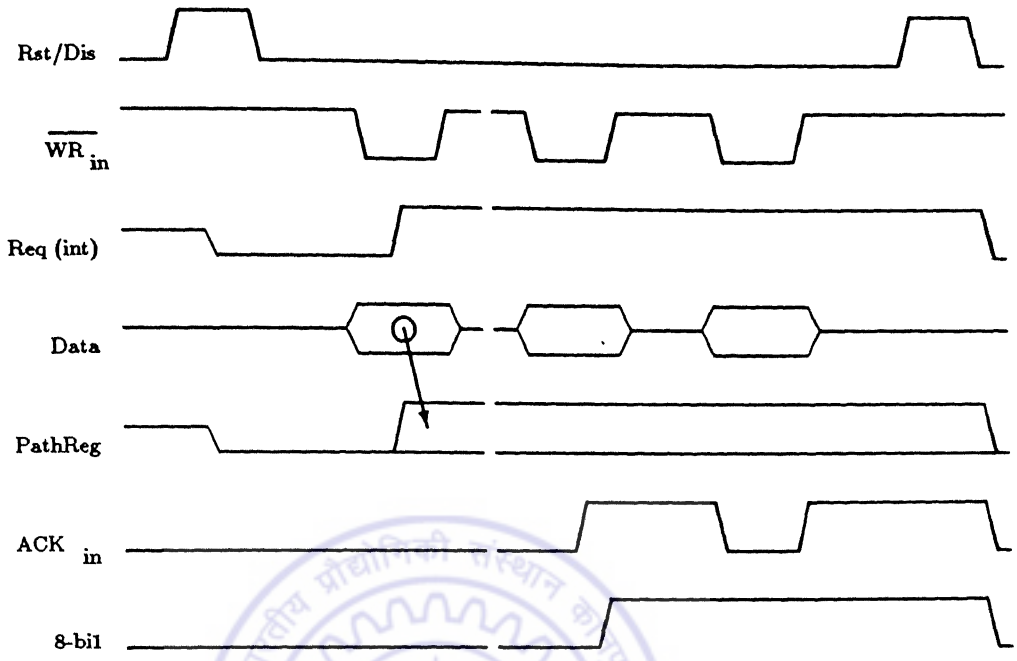


Figure 3.3: Signal transitions during Circuit Set-up

3. Writing command nibbles successively on the lower four bits of its data lines. A command nibble specifies the channel that is to act as the gateway to the remaining part of the circuit. A '1' in bit position i indicates a request for connection with channel i.

When the first pulse appears on the $\overline{WR}_{in}$ line of *Channel0*, it is interpreted as the initiation of a connection set-up. This sets the Req bit, which is internal to the channel. The first nibble is latched by the channel path-register (a 4-bit internal register) and the others are passed on to the forwarding channels. At the destination, if the LA–Host–FIFO link is not in use, *Channel0* sets its Busy bit and relays the status of $\overline{BF}$ line of its associated buffer to the source. The arrival of this signal locks the path and full 8-bit data transfer proceeds[1]. After complete data has been transmitted, source releases the circuit by issuing a Disc pulse. The whole sequence of events is described in Fig. 3.3.

[1]During circuit set-up phase only 4-bit communication takes place over the data lines.

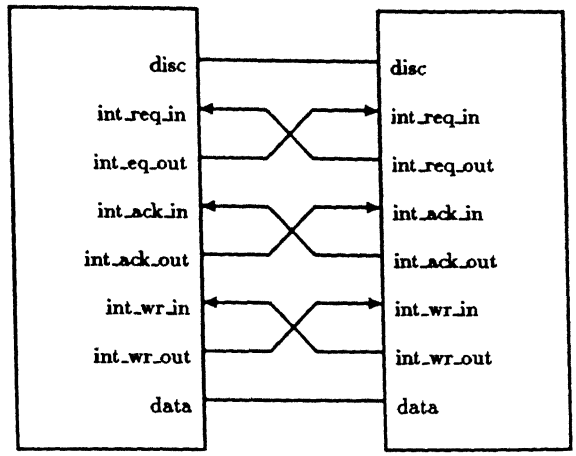


Figure 3.4: The Channel–Channel Interface

3.2.2 Channel–Channel Interaction

Intra-LA interaction is present in all transfers and forms the basis of connections. Whenever a channel receives a request on its external lines, it tries to set up connection with the channel specified in the command nibble. This is handled by Channel–Channel interface shown in Fig. 3.4.

In this section, the channel that receives an external request is referred to as the head, and the channels specified in the command nibble are referred to as the tail. The head stores the first incoming nibble in its path-register. This nibble specifies the tail channels and individual bits in the head's path-register act as internal requests to the tails. The tail may be *Channel0*, in which case it is the final destination, or any other channel. If the tail is currently not involved in any transfer, its Fully-Connected Switch (FCS) connects the data and other control lines to those of the head. The lower and upper 4-bits of the data byte are crossed-over by the tail before being put out on the external lines. If more than one internal request arrive at the same tail, the request with the highest priority gets through. Each channel has a fixed priority, with *Channel0* having the highest and *Channel3* the lowest. The succeeding nibbles are simply passed on to the tail to be forwarded to the LA adjacent to it. When the tail receives an assertive ACK_{in} signal from its adjacent LA, it locks the connection thus establishing the path between the destination and the tail channel. The ACK signal is then relayed to the head in the as int_ack_out. In case tail is the final destination, int_ack_out is generated in conjunction with ACK_{in} ($\overline{BF}$ of its FIFO) and NRW signals.

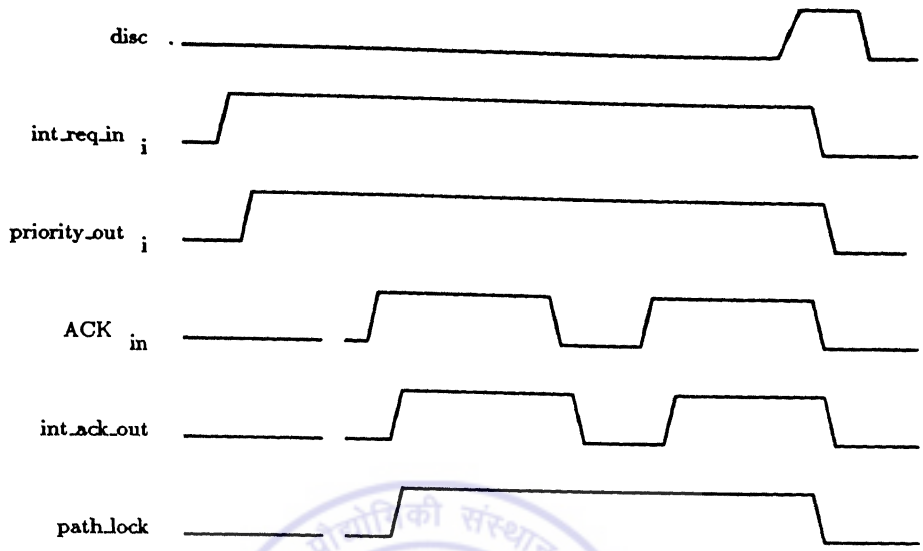


Figure 3.5: Signal transitions during Channel–Channel Interaction

This also ensures that whenever the host lowers its NRW line to ascertain the status of the LA–Host link, data transmission is temporarily halted.

A Disc pulse causes the tail FCS to release its connection. If tail FCS fails to connect a particular head owing to the tail channel being busy or due to a lower priority of the head as compared to another request, the succeeding nibbles forwarded by the head are lost. The related signal transitions are shown in Fig. 3.5.

3.2.3 LA–LA Interaction

LA–LA interaction comes into play when messages have to be sent to nodes not directly connected to the source. Fig. 3.6 shows the signals interchanged by two adjacent LAs and the manner in which data lines are connected. Every channel receives command nibbles on its lower bits and forwards them on the upper. This is necessary to prevent damage to links when two adjacent LAs try to establish connection with each other simultaneously. Hence, when LAs are linked to form a network, data lines are crossed-over[2]. Functionally all channels are identical. There is no difference in the behaviour of other channels and *Channel0*. The appearance of the first forwarded $\overline{\text{WR}}$ pulse and its corresponding command nibble causes the setting of Req bit

[2]This, however, is not the case in the LA–Host link.

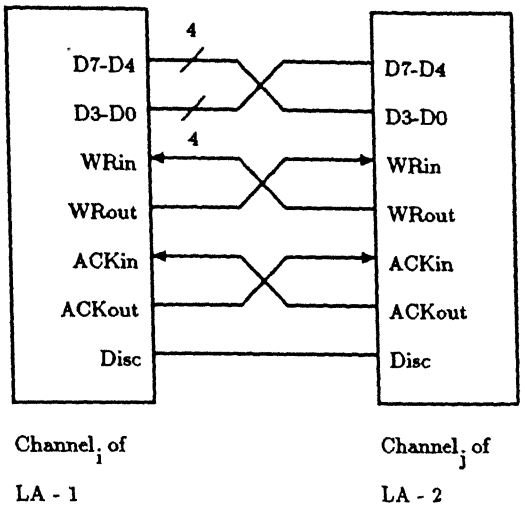


Figure 3.6: The LA–LA Interface

of the accepting LA-channel. The other transitions are similar to those explained in Section 3.2.1. A detailed schematic of each channel is presented in Fig. 3.7.

3.3 Circuit Set-up and Release – An Example

The aspects of the functionality of LA presented in the previous section are best explained through an example involving all the three interactions. Fig. 3.8 illustrates the network considered in this section. To establish a path between $Proc_1$ and $Proc_3$, the following steps have to be carried out (assuming that LA-Host link of $Proc_1$ is free):

1. $Proc_1$ lowers its NRW line and polls ChBusy. Since, FIFO of $Proc_1$ is not being written to, it proceeds with the subsequent actions.

2. $Proc_1$ writes the nibble 0100 on the link attached to *Channel0* of its LA. The '1' at bit_2 indicates a request for connection with *Channel2* which in turn is connected to *Channel1* of the LA attached to $Proc_2$.

3. $Proc_1$ then writes 1000, which is passed on in its entirety to the LA of $Proc_2$. This interprets the bits and attempts to link *Channel3* with *Channel1*.

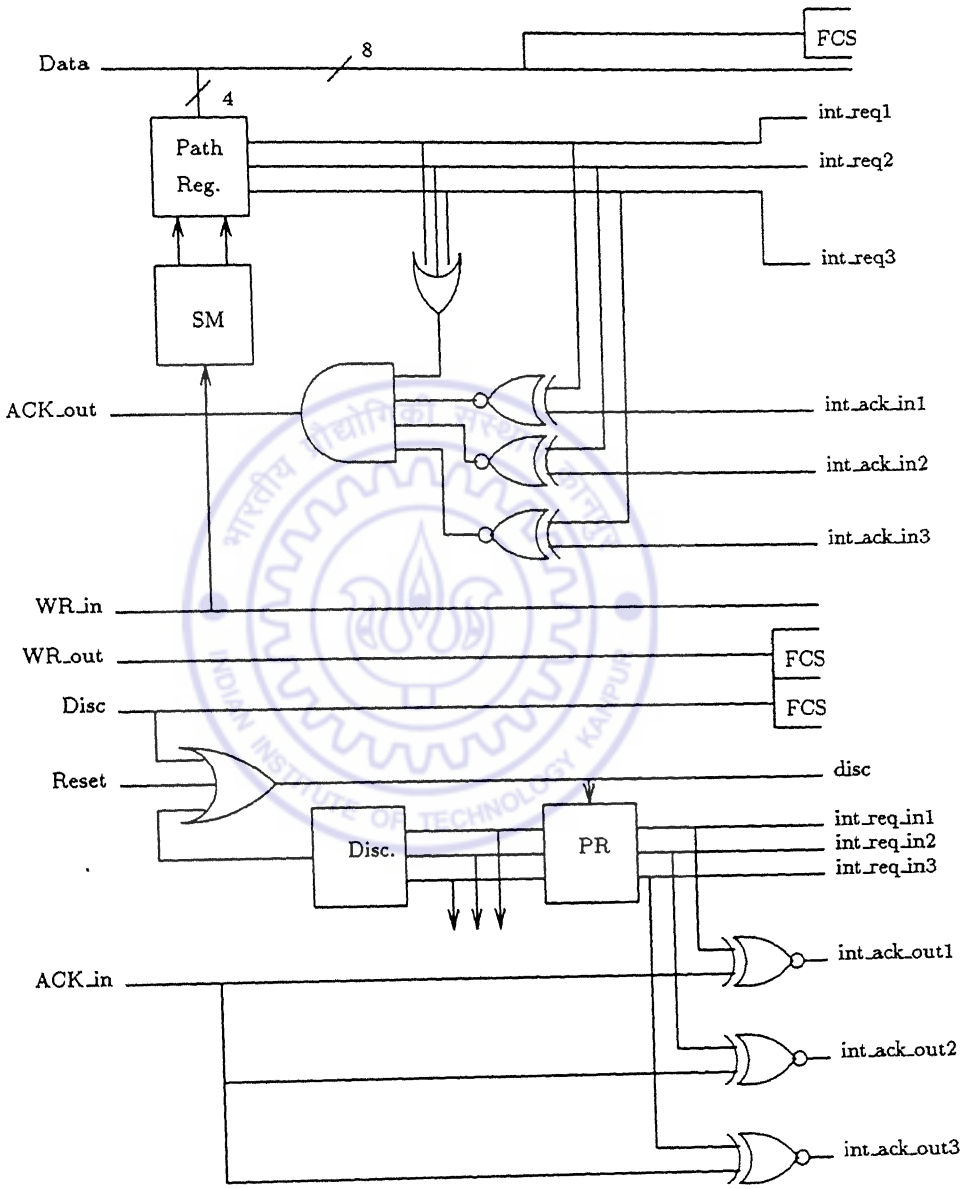


Figure 3.7: Detailed schematic of each channel of the LA

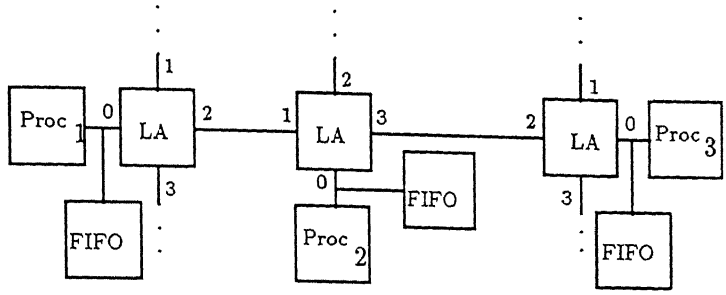


Figure 3.8: Circuit Set-up and Release

4. Finally, $Proc_1$ writes 0001 on its outgoing link. This nibble reaches the LA of $Proc_3$ where it signals a requests for connection to the processor (i.e., the FIFO buffer).

5. The status of $\overline{BF}$ signal of the FIFO is relayed back to the source LA. If the buffer is not full ($\overline{BF} = 1$), then the connection gets established and remains so till the end of transmission. Once a connection is established, a negative transition of the $\overline{BF}$ of a destination buffer causes a hold-up. The source waits till the $\overline{BF}$ signal becomes high again and resumes transmission.

6. Once the data transfer is over, a Disc (disconnect) pulse has to be generated by the source processor. This causes the path-registers in the LAs to be reset and the connection is released for use by the other LAs.

Multicast is achieved by setting the required bits in the control nibble to 1.

Due to fixed priority arbitration, a connection set-up bid might get pre-empted in case the corresponding ACK signal does not arrive in time. At such instances a disconnect signal is generated internally. Since the LA does not keep track of previous requests that had arrived, it is expected that the processor will relinquish its demand by generating a Disc pulse if, within a reasonable period of time (about two write cycles), it does not receive the ACK signal from the destination LA(s).

3.4 Conclusion

The LA fulfills most of its design objectives.

- It provides a simple interface for connection in a point-to-point connected network.
- Assumptions made about the FIFO buffer are not unrealistic. The only signals required to interface a FIFO with the LA are $\overline{BF}$ (Buffer Full), BE (Buffer Empty), $\overline{RD}$ and $\overline{WR}$. The data path is assumed to be 8-bit wide.
- Multicasting is a natural extension of the addressing technique.
- Using appropriate number of LAs, large multicomputer networks may be constructed without any penalty to the speed of computation.

Versatility of the LA is demonstrated through its application to some well-known multicomputer configurations, to be described later.

INDIAN INSTITUTE OF TECHNOLOGY KANPUR

Chapter 4

Applications

In the previous chapter, we have seen how circuit is established and messages sent between two hosts in a network realised through LAs. We have also discussed the LA–Host and LA–LA interfaces required to set up a network. In this chapter we present some examples that demonstrate the scalability and versatility of the LA.

4.1 Scaling up for bigger networks

Our design supports only three channels per LA in addition to *Channel0*. This may seem too small a number to realise a moderate sized network, wherein a host needs to be connected to more than three hosts directly. In such cases, two (or more) LAs may be connected to build more than three outgoing links per host. In this respect our design is scalable. Fig. 4.1 shows the additional connections required to interface two LAs to a single port of a host computer. Only the pins whose connection patterns get modified are shown. Other pin connections remain the same as in Fig. 3.2. Recall that Busy line remains high throughout a message transfer. This prevents the interleaving of messages from two sources at a single FIFO. Further scaling is possible by following similar connection pattern for appropriate number of LAs. In the examples to follow, this scheme is used wherever two or more LAs are required per host. In these examples, circles represent hosts and boxes represent LAs. Numbers inside circles represent processor identifiers; numbers on the links denote the channel numbers of the LAs through which connections are made between nodes.

CENTRAL LIBRARY
I.I.T. KANPUR
Acc. No. A. 115520

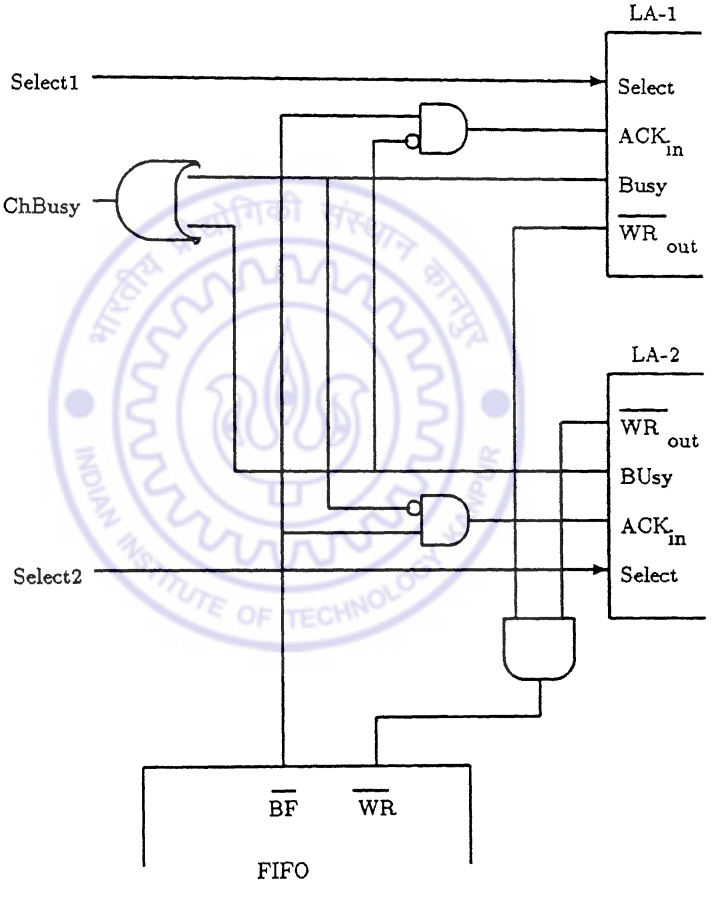


Figure 4.1: Multiple LA Interface

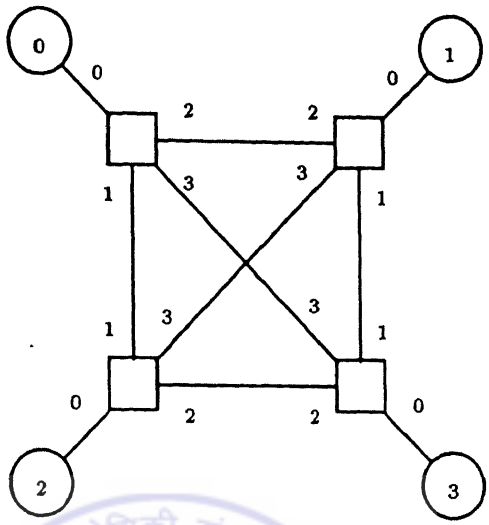


Figure 4.2: A Fully Connected Network

4.2 Fully Connected System

A fully connected system, although highly fault tolerant, does not lend itself to scaling owing to the high number of links required to form larger networks. In our first example, we present a four node fully connected network. Such a network may be realised by using just a single LA per host. Fig. 4.2 shows the network.

4.3 Binary Tree Network

The binary tree network is one of the most popular topologies. We consider a design of 7-node binary tree network. This is illustrated in Fig. 4.3. It can easily be converted into a Hypertree network [5] by connecting siblings through additional links. This would however require additional LAs at level 1.

4.4 Hypercube System

Many distributed applications have been designed for the hypercube structure. Here we demonstrate the applicability of the LA to a hypercube structure.

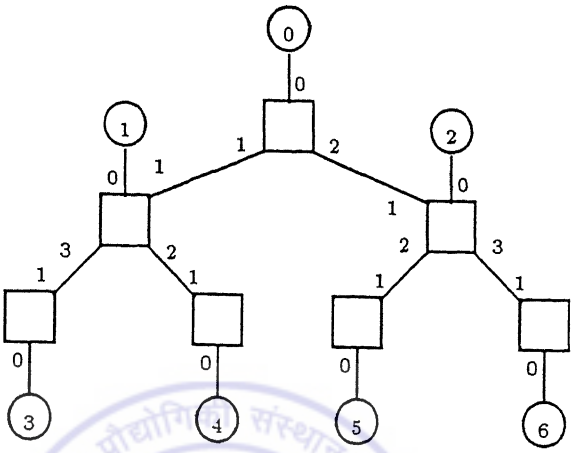


Figure 4.3: A Binary Tree Network

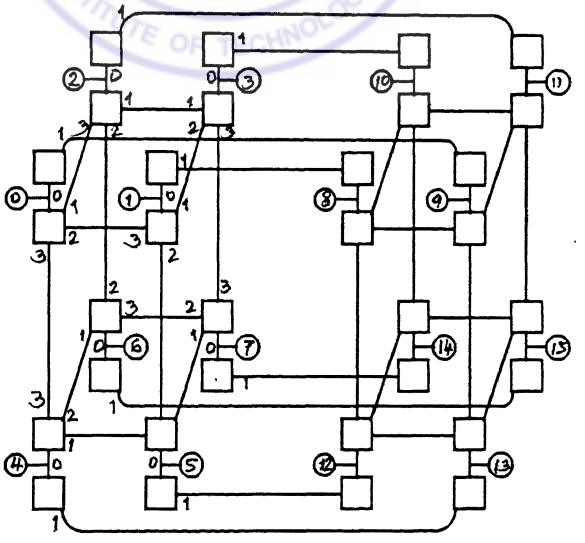


Figure 4.4: A 16-node hypercube

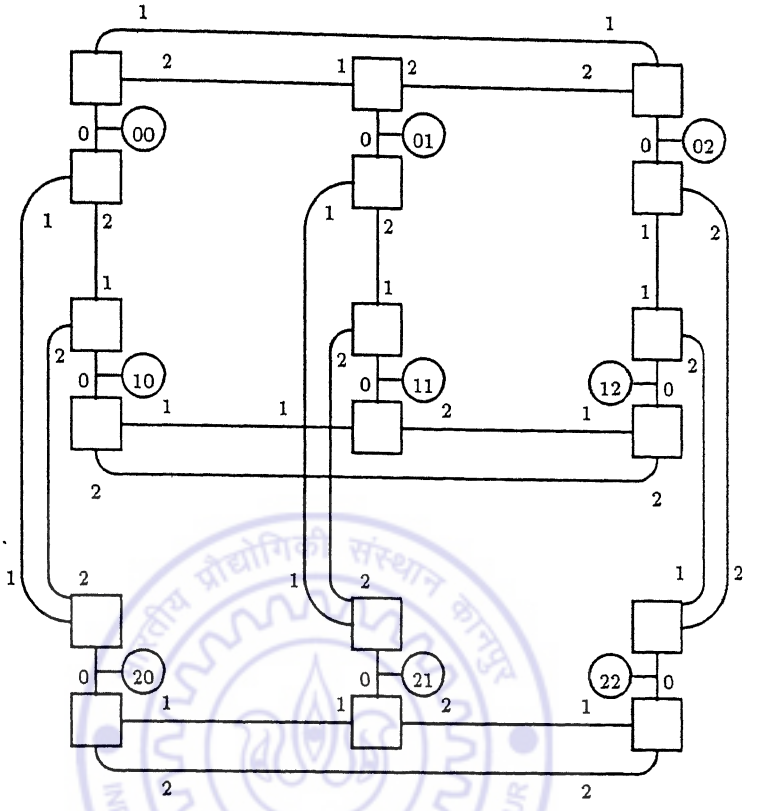


Figure 4.5: A 3-drop 2-dimension MMS Structure

A hypercube has 2^k computing nodes numbered 0 to $2^k - 1$. Two nodes are connected by a link if the binary representations of their numbers differ in a single bit. Fig. 4.4 shows a 16-node hypercube realised through LAs.

4.5 Multidimensional Multilink System

Multidimensional Multilink System (MMS) [9] was proposed as a general purpose MIMD system. It comprises of multicast networks of p computers arranged in d dimensions, requiring a total of p^d computers. Our final example illustrates the utility of LAs in constructing a 3-drop 2-dimension MMS structure in Fig. 4.5.

4.6 Conclusion

The LA is not oriented towards a particular architecture or network configuration. Almost all link-oriented multicomputer systems may be implemented

using LAs as network building blocks. The small number of channels supported by each LA (4) does not impose a limit on the size of the network as our design allows for scaling by using an appropriate number of LAs connected in a simple fashion. The LA may be termed as a truly general purpose interconnection module.

Chapter 5

Implementation Details and Simulation Results

In this chapter we present some implementational statistics of LA and results obtained from simulations performed on our design.

5.1 Real Estate

LA was designed on the Nelsis IC Design System [11] using c3tu scalable CMOS process, with the value of λ set to 0.2μ. Standard cell library for the process not being available, we took recourse to custom designing. The area occupied by the design is $5936\lambda \times 3832\lambda$ (approximately 0.9 mm^2). This excludes the space required for bonding pads. Pads and rails have not been integrated in the design. The minimum number of pads required for the LA is 57, as detailed below.

(a) 2 power pads for Vdd and Vss.

(b) 36 bidirectional pads for Data and Disc lines.

(c) 10 input pads for $\overline{WR}_{in}$, ACK_{in}, NRW and Sel lines.

(d) 9 output pads for the outbound $\overline{WR}_{out}$, ACK_{out} and Busy signals.

These requirements are met by a 68-pin die, allowing 11 additional pins for power supply lines.

Performance of an intergated circuit is, to a certain extent, influenced by I/O pads owing to their capacitive nature. Keeping this fact in mind,

we carried out simulations of our design after adding appropriate capacitances to terminals. A fixed value of 350×10^{-15}F was assumed for all pads irrespective of their types.

5.2 Performance Evaluation Strategy

Algorithms employed by popular modelling and simulation tools exhibit an $O(n^2)$ time complexity, where n is the number of circuit elements like resistors, transistors etc. To achieve a realistic estimate of the circuit performance of LA, coupling capacitances and resistances of all connections were included in the modelling. This increased the number of circuit elements and rendered simulation of the complete circuit infeasible. In order to characterise the performance of LA at an acceptable level of approximation, simulation was carried out in circuit view (refer Section 1.4). We modelled a single channel and formed a network by interconnecting two of them at the circuit level of abstraction. Figures obtained as a result of the simulations are presented in Section 4 of this chapter.

5.3 Simulation Environment

The Nelsis System provides tools for both circuit simulation and switch level simulation. Circuit simulators are the most accurate simulators since they model circuit elements as devices. Such simulators, however, are too slow to be of much use in design of circuits of real magnitude. Switch level simulators, on the other hand, can cope up with large sections of the layout. In the timing mode they can capture switching delays. Transient analyses are, however, not as accurate as those of circuit simulators. In the verification of our design, we opted for switch level analysis. After verification of individual modules under all possible input condition, circuit level description of a single channel of LA was extracted from the layout. This formed the basis of further simulation runs. Outputs produced by the simulator sls [4] appear in Appendix B of this thesis. Timing details and power dissipation are summarised in the next section.

5.4 Results

Although results produced in this section are based on single channel simulation, these are easily extended to the whole LA as all the channels are almost identical. Moreover, delays associated with switching occur mostly in the set-up phase of the circuit and do not depend on the number of channels involved in the transfer. Bus delays can be extrapolated to give a realistic hue. Major timing information is detailed below.

- t_s and t_h : Set-up and hold times for the path registers of each channel of the LA = 5ns and 7ns respectively.
- t_{dwr} : Time delay between the appearance of the first pulse on the $\overline{WR}_{in}$ line of the channel and registration of the request for connection = 8.87ns.
- 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] = 7.76ns.
- t_{dack} : Time required for the processing of the collected ACK signals to be relayed to the previous LA[2] = 5.84ns.
- t_{sw} : Switching time of the fully connected switches = 1.0ns.

The total delay introduced by each channel of the LA is in the neighbourhood of 25ns. This delay and the delay on the transmission lines determine the number of hops permitted on the way from the source to the destination. Assuming a 25MHz clock for the host processor and and a 6T machine cycle, a conservative estimate of the upper limit on the number of intermediate nodes is 5 (five). The dynamic power dissipation of each channel is 2.196mW. With such restrictions our LA can prove itself useful in moderate sized multicomputer networks.

[1] if not preempted by a higher priority request
[2] measured from the time the last ACK was received

Chapter 6

Conclusion

Our endeavour in this thesis has been to explain the need of a communication coprocessor for a multicomputer system and present our contribution towards this end. We have also tried to support our claims with application examples and performance evaluation results. While our design has achieved its major objectives, it is not without its share of shortcomings. In this, our concluding chapter, we list the capabilities of LA and discuss its drawbacks. We also suggest potential improvements over our current design.

6.1 Capabilities of LA

In view of the parameters laid out in Chapter 1, LA performs reasonably well as a communication coprocessor.

1. LA is scalable and its interfaces are simple. It is also highly adaptable as commonly available accessories like FIFO buffers and basic gates are required to build a subsystem.

2. It is easy to program; command nibbles simply state whether a connection is requested of a channel.

3. Multicasting is implemented efficiently as it is a mere extension of the addressing strategy used to communicate with individual channels.

4. Circuit set-up and release protocols are simple and ensure safety even in unnatural terminations.

5. Flow control is implicit. Statii of destination buffers are used to effect flow control.

6. Intermediate hosts are shielded from rerouting and circuit control issues, leaving them free for meaningful computation.

7. Switching times and propagation delays are low.

6.2 Drawbacks

The major drawback of LA arises from the assumption made about the underlying network. A link controller for a circuit switched network cannot guarantee low waiting times for request unless it has provisions for retries. LA does not store path information of preempted bids for future tries. This increases the source host processor's overhead during circuit set-up. Furthermore, a circuit may not get established owing to a small segment of the path being involved in another message transfer. This would again block the source host.

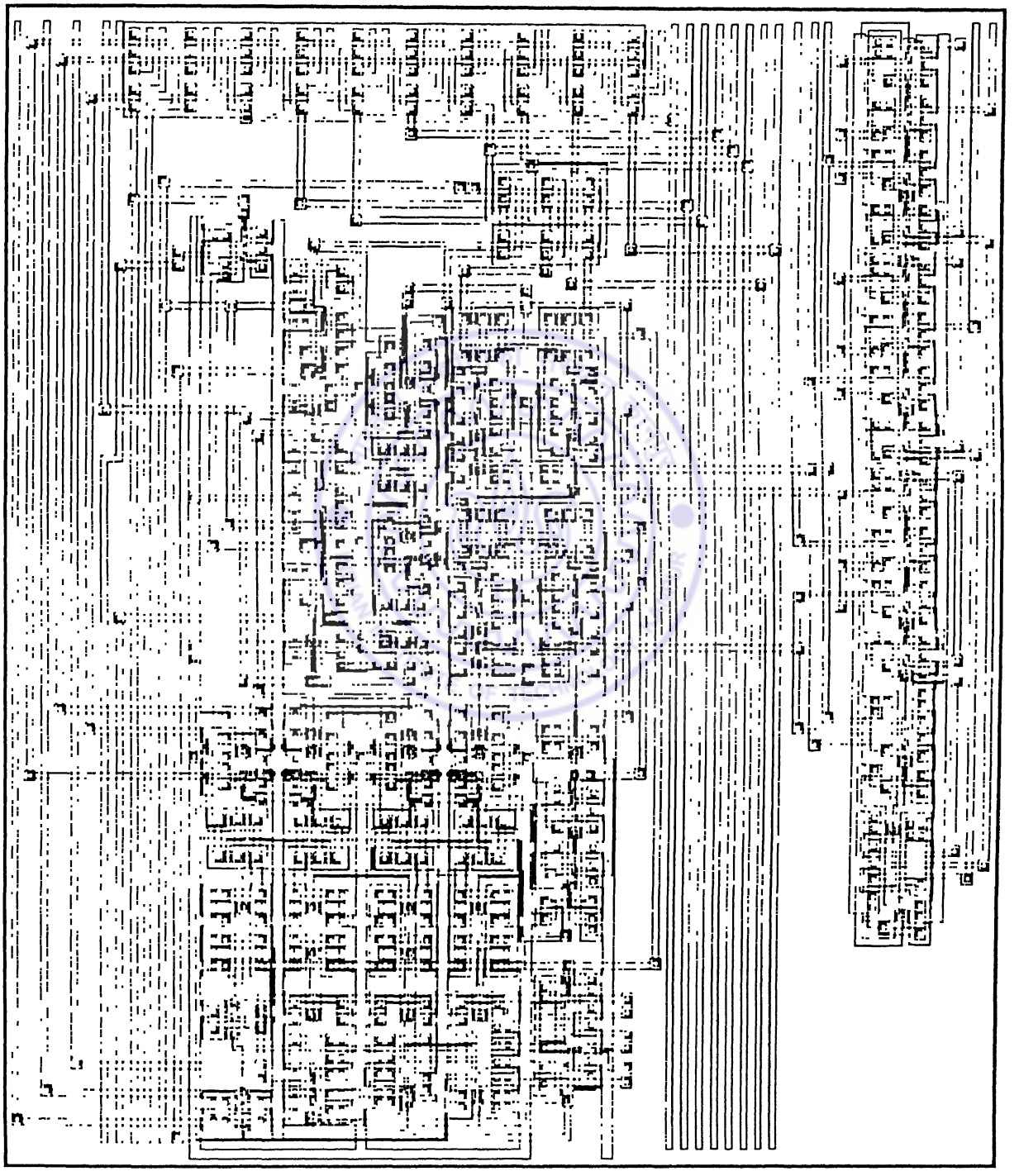
6.3 Extensions and Future Work

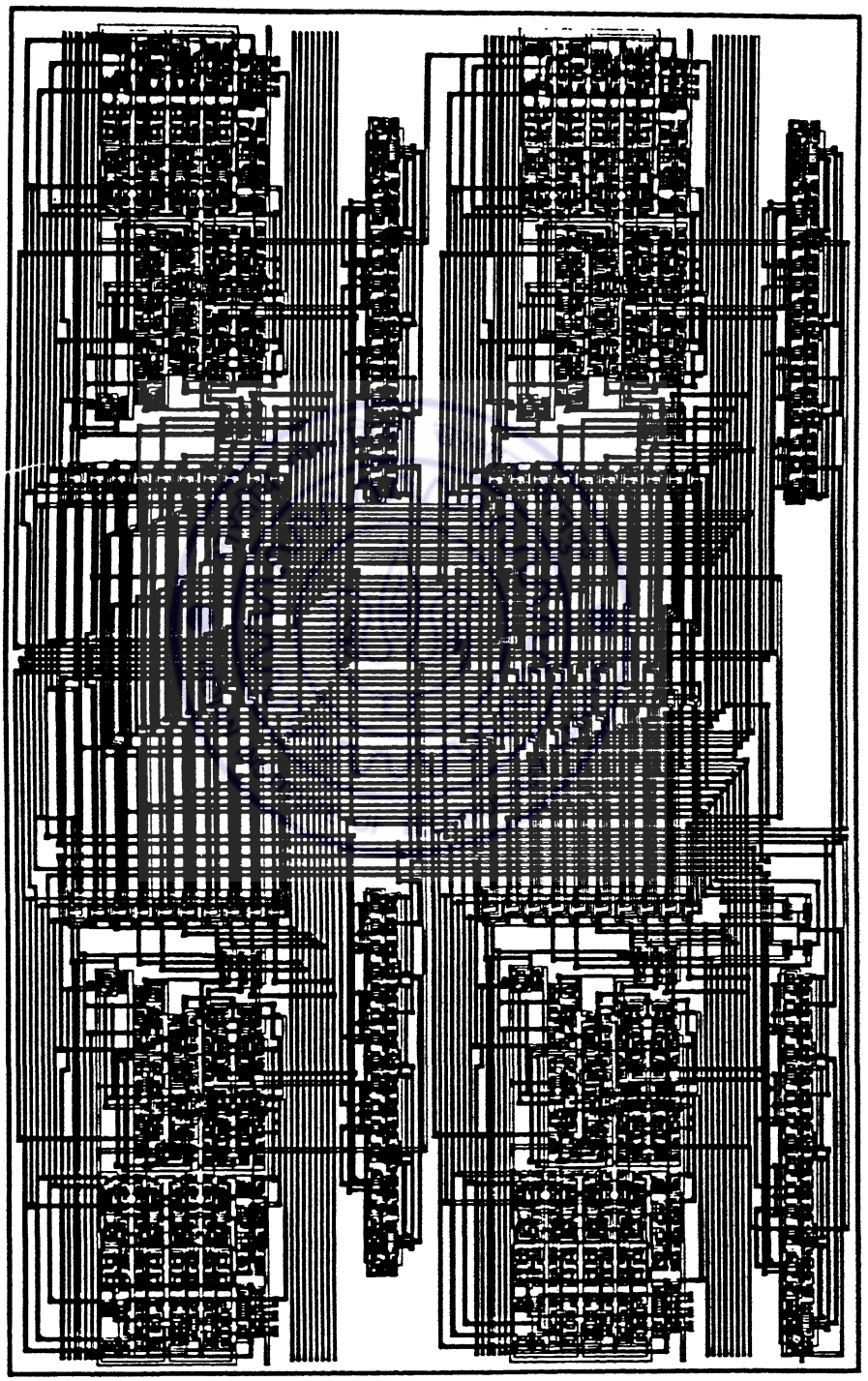
Our design may be extended to allow more flexibility to the user by providing command words for repeated connection tries. This would reduce the host's overhead. To achieve this, memory will have to be incorporated in the design of the controller to store path information. Another extension could be the facility to start transmitting if a major part of the path from source to destination has been established. This implies buffering of data at intermediate hosts and interpretation of this stored data at an appropriate point of time without the knowledge of the host. Such a facility would undoubtedly decrease the waiting time of source hosts. At the same time it would also increase the complexity of the control circuit. A first-come first-served (FCFS) priority resolver may by implemented in place of the fixed priority resolver to give the channels a fairer chance at establishing connections.

Notwithstanding its limitations, the Link Adapter serves the purpose of a generic interconnection module and may be used as an essential component in a variety of multicomputer networks.

Appendix A

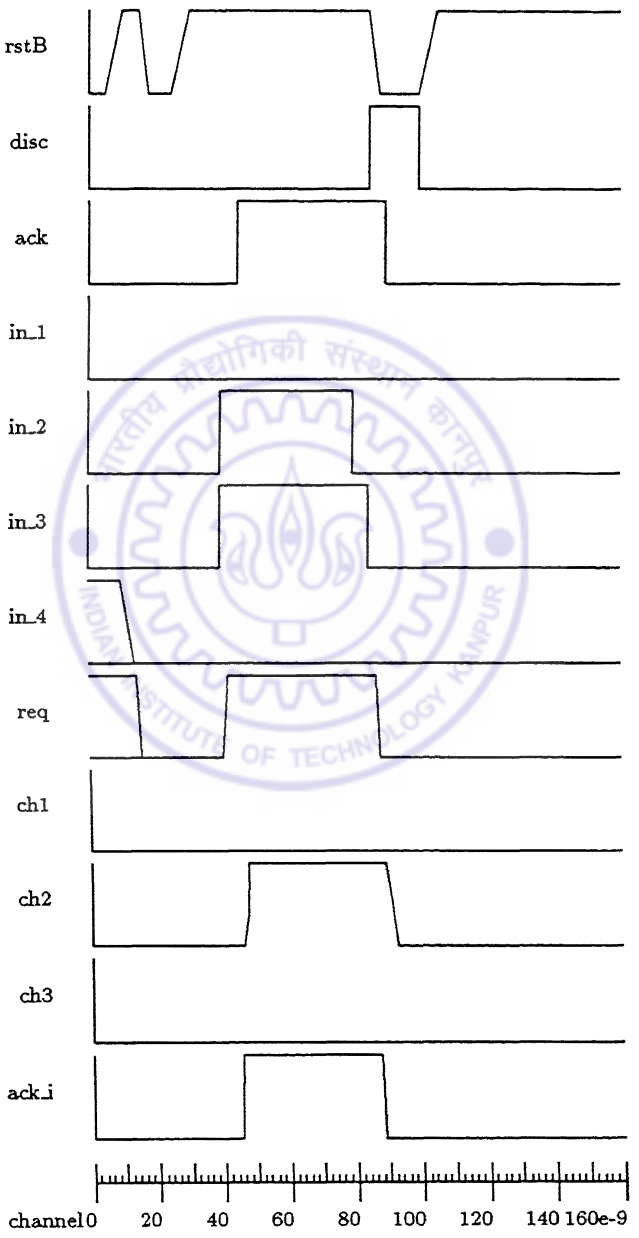
Layout of Link Adapter

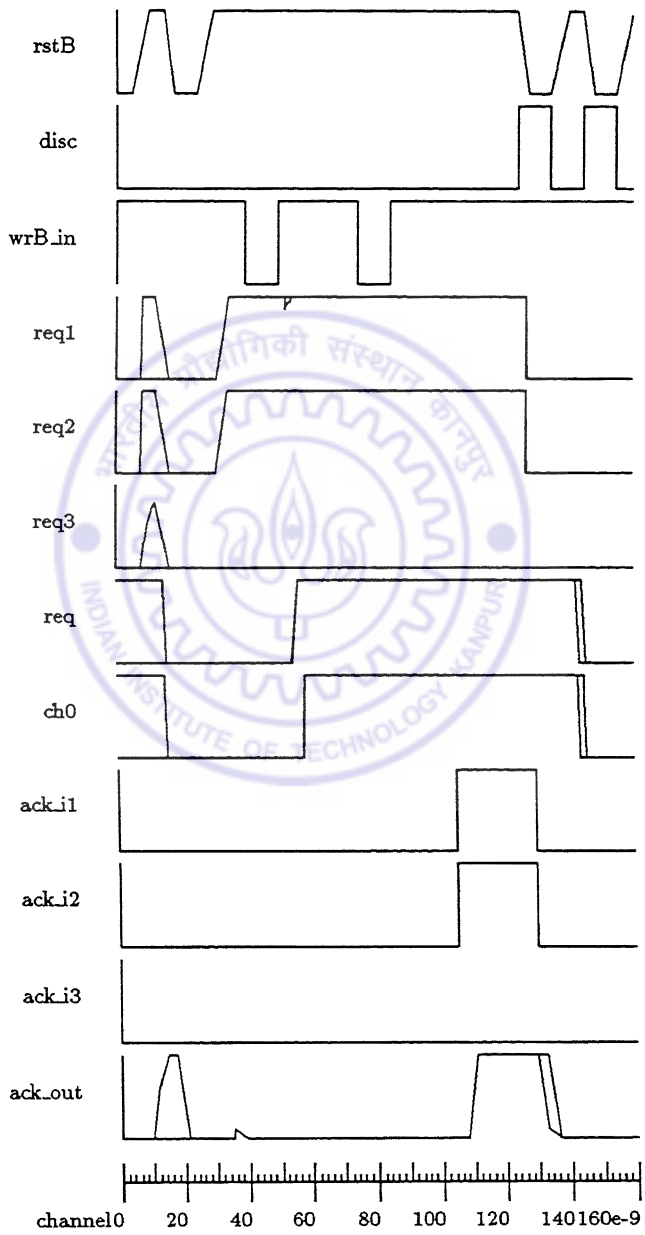



0.0
339.20
Cell: lnkadp

Appendix B

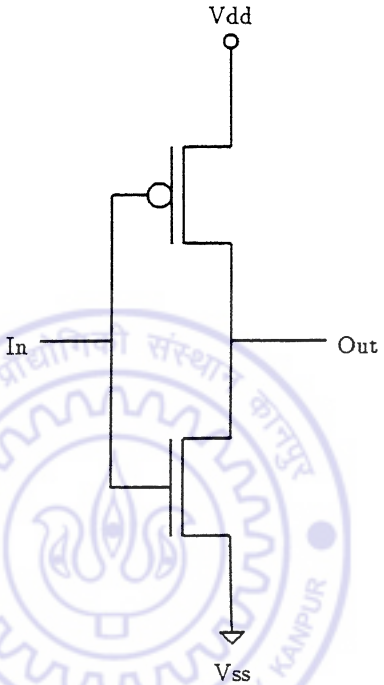
Timing Diagram of Link Adapter


rstB
disc
ack
in_1
in_2
in_3
in_4
req
ch1
ch2
ch3
ack_i
channel
0
20
40
60
80
100
120
140
160e-9

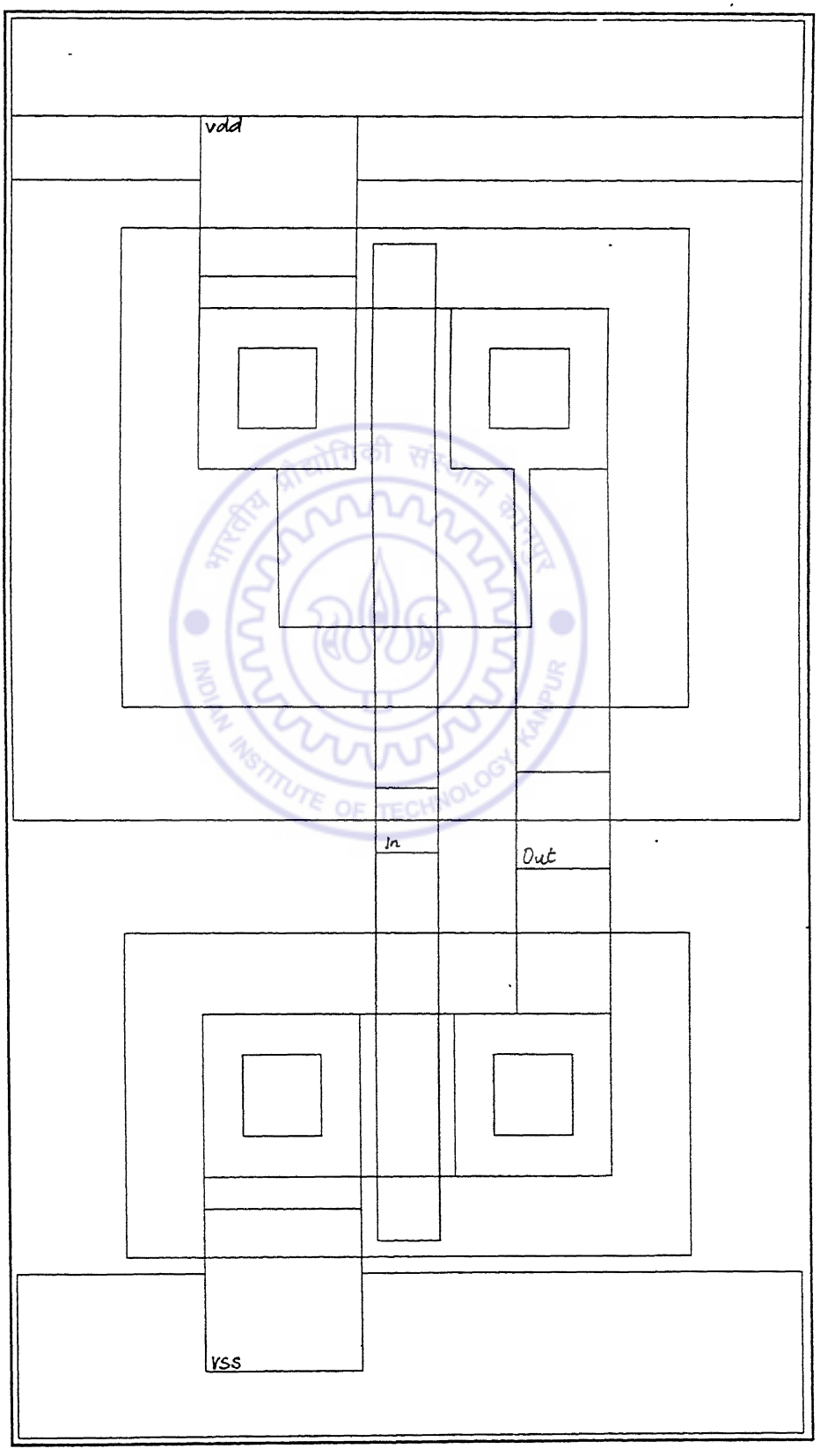

rstB
disc
wrB_in
req1
req2
req3
req
ch0
ack_i1
ack_i2
ack_i3
ack_out
channel0 20 40 60 80 100 120 140160e-9

Appendix C

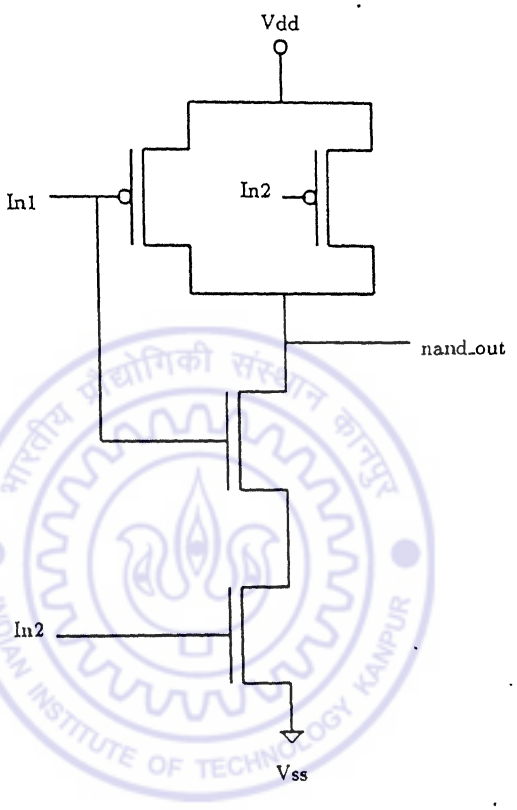
Layouts of Basic Cells



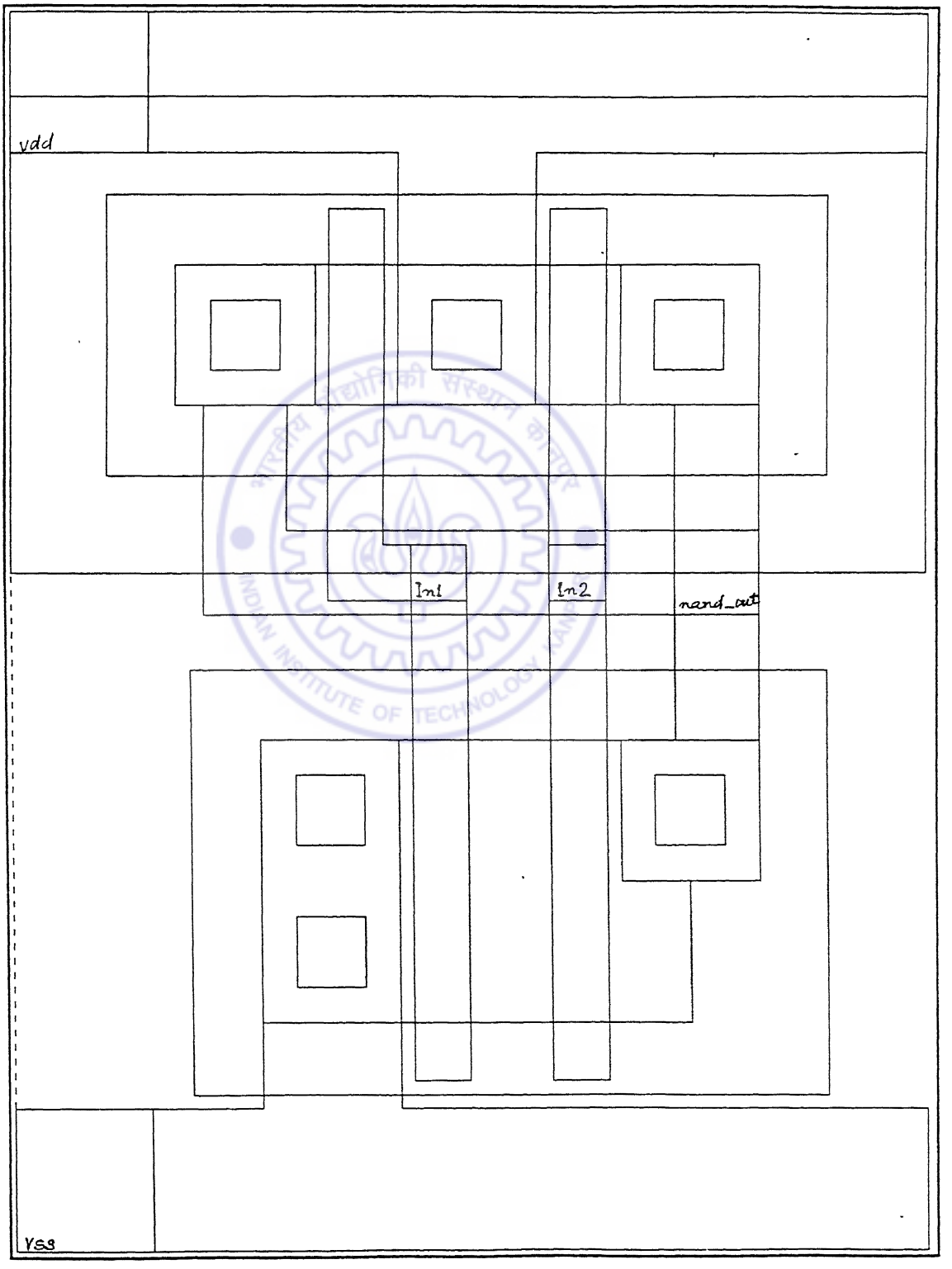
Cell	: inverter
Bounding-box	: $100\lambda \times 176\lambda$
Power-supply	: vdd, vss
Input	: in
Output	: out



Cell: inverter

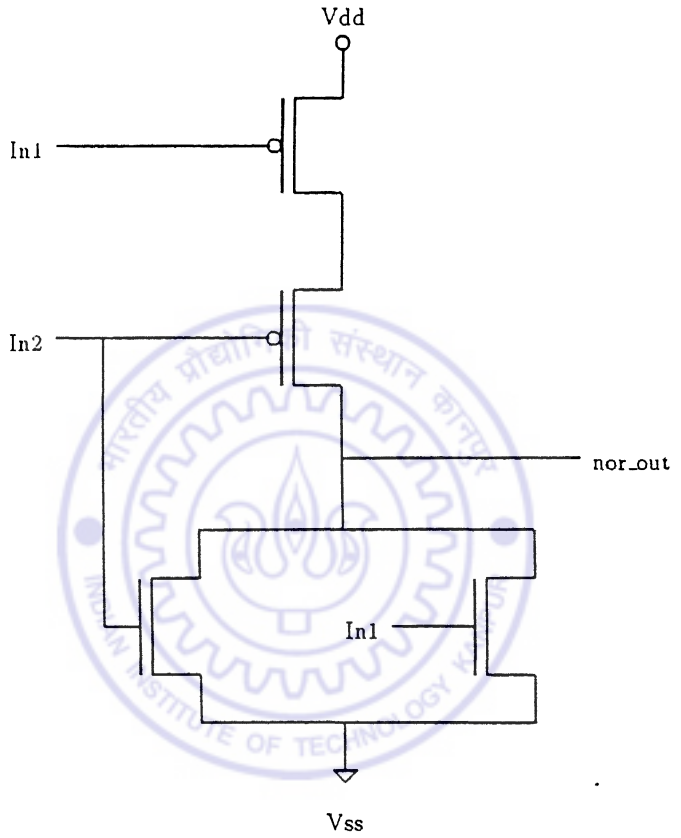


Cell	: nand
Bounding-box	: $132\lambda \times 176\lambda$
Power-supply	: vdd, vss
Inputs	: in1, in2
Output	: nand_out

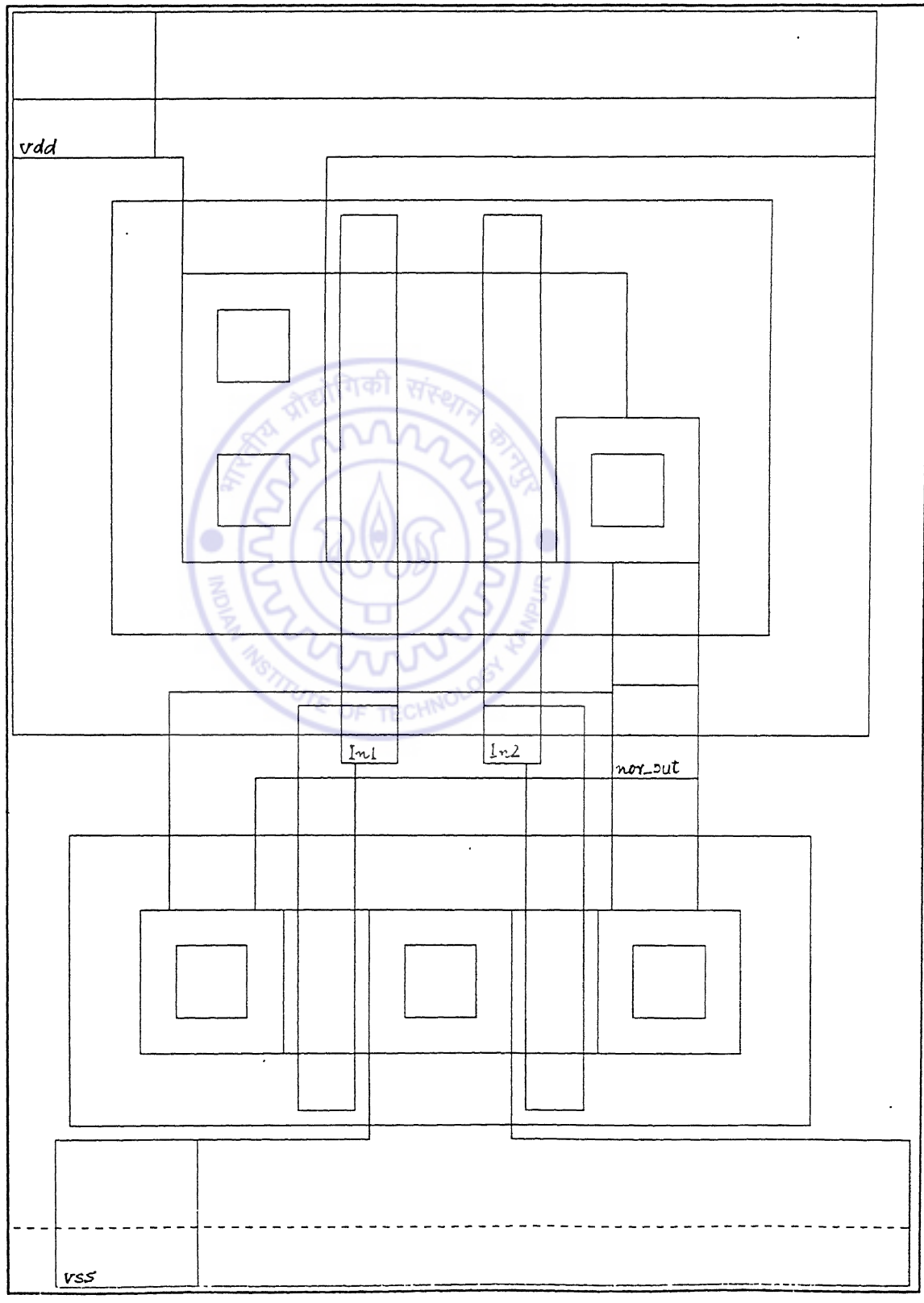


Cell: nand

0.0 7.54

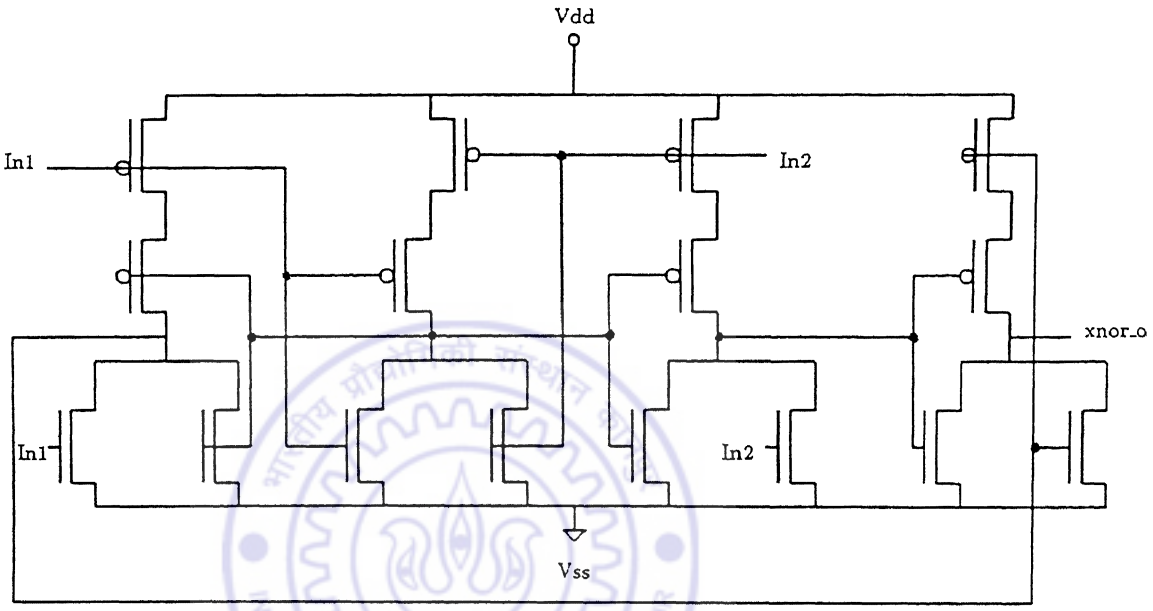


Cell	: nor
Bounding-box	: $126\lambda \times 176\lambda$
Power-supply	: vdd, vss
Inputs	: in1, in2
Output	: nor_out

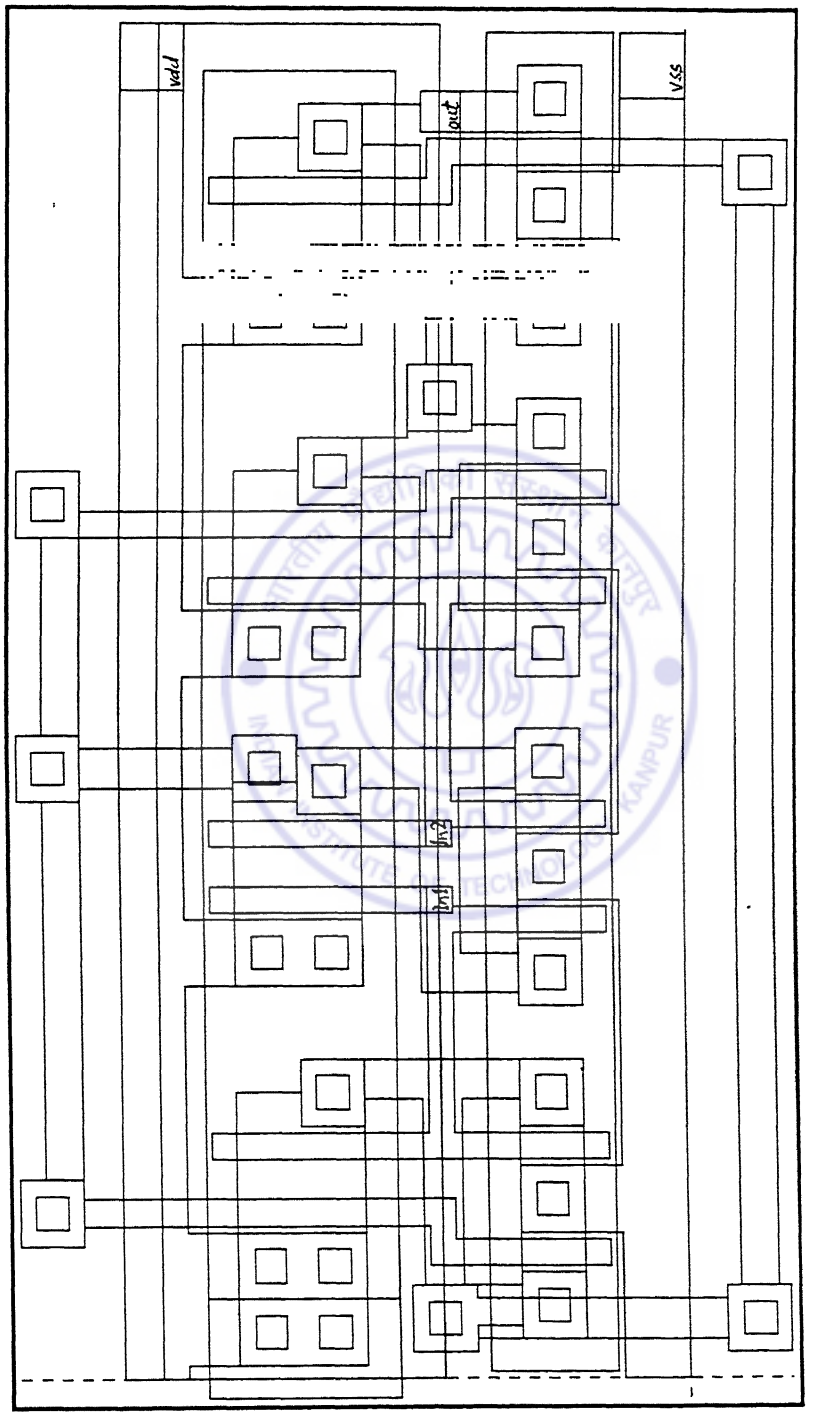


Cell: nor

0.0 7.20

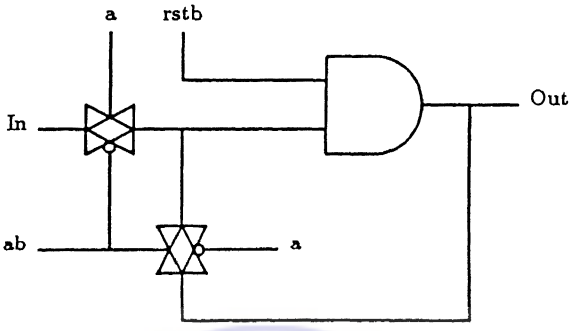


Cell : xnor
Bounding-box : $414\lambda \times 240\lambda$
Power-supply : vdd, vss
Inputs : in1, in2
Output : xnor_out

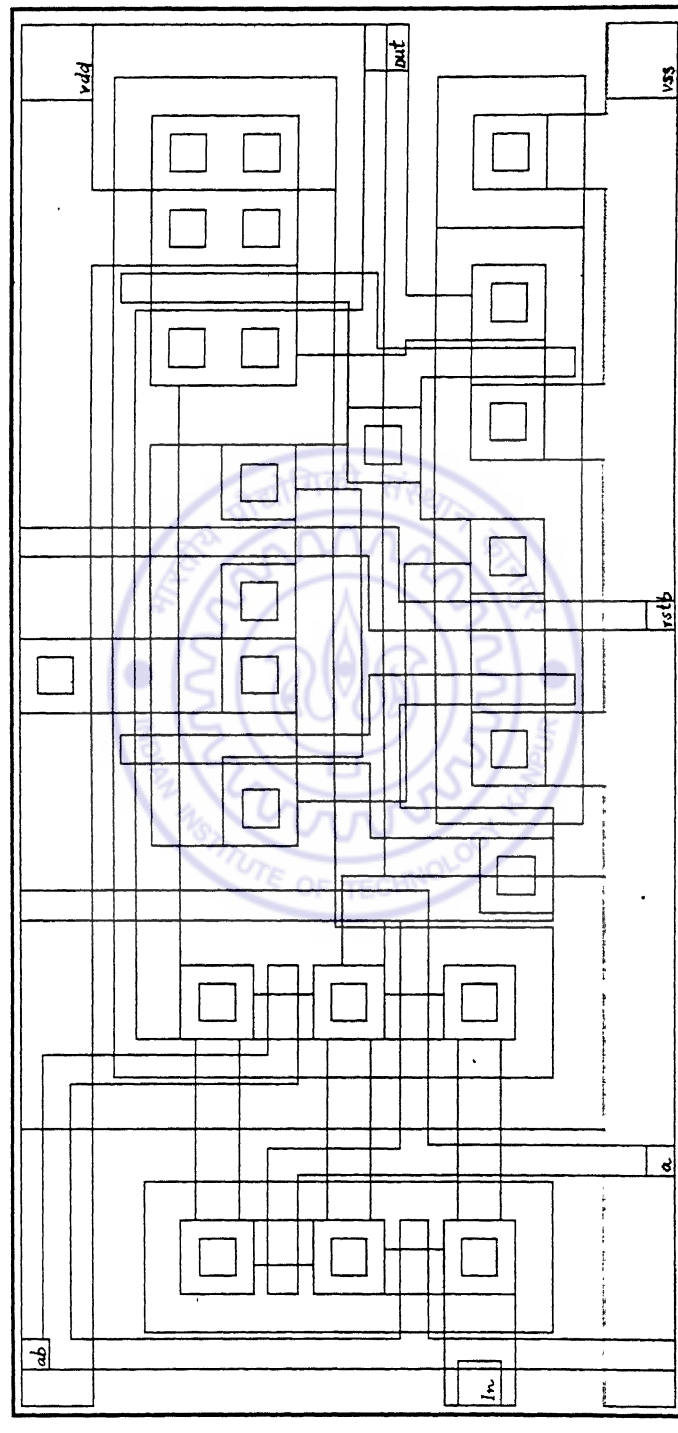


0.0 23.66

Cell: xnor

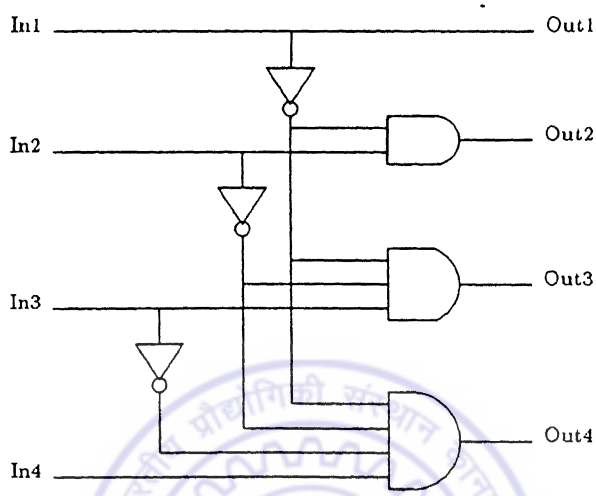


Cell : latch
Bounding-box : $270\lambda \times 172\lambda$
Power-supply : vdd, vss
Gate-control : a, ab
Inputs : rstb, in
Output : out

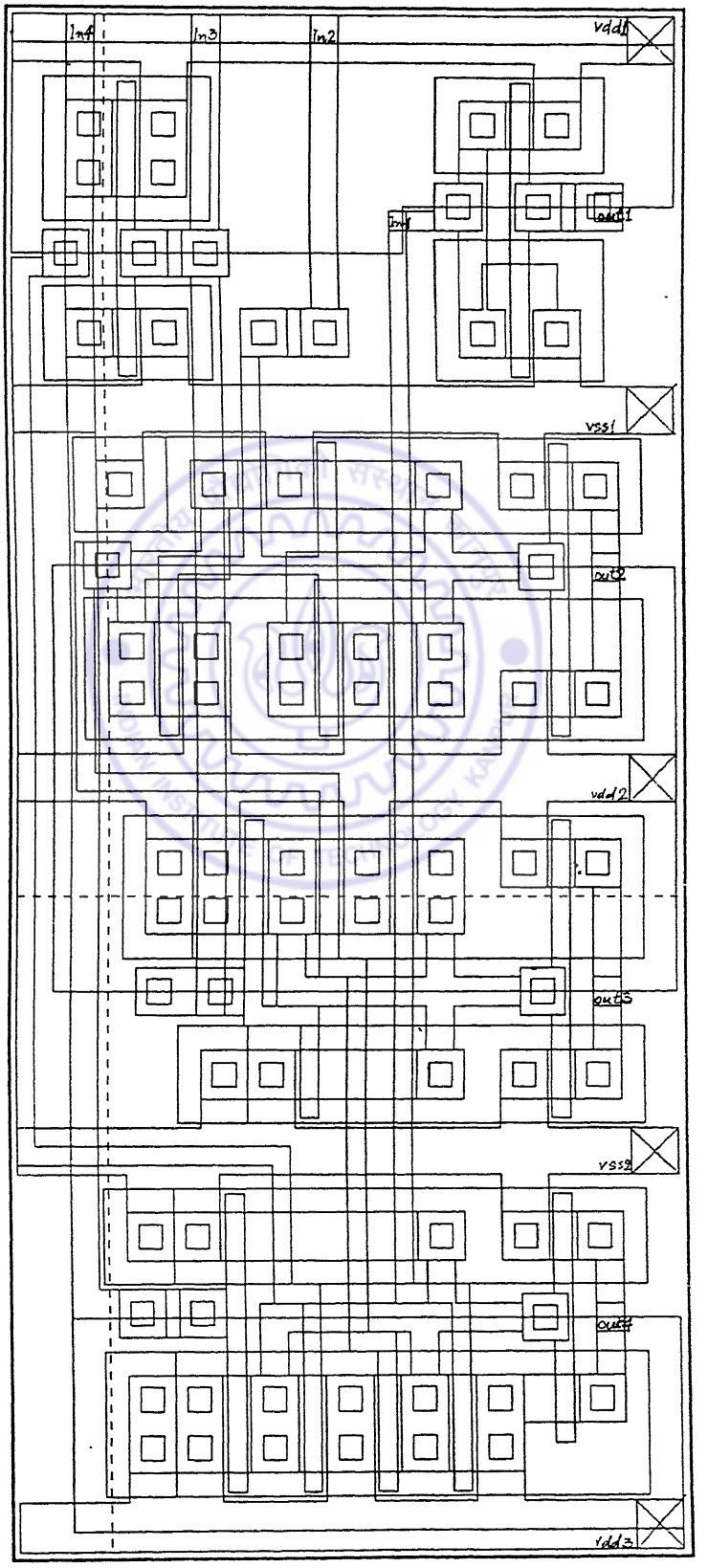


0.0 21.14

Cell: latch



Cell	: priority_resolver
Bounding-box	: $284\lambda \times 644\lambda$
Power-supply	: vdd1, vdd2, vdd3, vss1, vss2
Inputs	: in1, in2, in3, in4
Output	: out1, out2, out3, out4



Cell: priority_resolver

Bibliography

[1] Arnould, E. A., F. J. Bitz, E. C. Cooper, H. T. Kung, R. D. Sansom, A. P. Steenkiste, "The Design of Nectar: A Network Backplane for Heteregeneous Multicomputers," in *Proceedings of the Third International Conference on Architectural Support for Programming Languages and Operating Systems*, Boston, Massachussets, April 3-6, 1989, pp. 205–216.

[2] Crowther, W., J. Goodhue, R. Gurwitz, R. Rettberg and R. Thomas, "The Butterfly Parallel Processor," *IEEE Computer Architecture Newsletter*, September/December 1985, pp. 18–45.

[3] DuBois, A. J. and J. Rasure, "Design and Evaluation of a Distributed AsynchronousVLSI Crossbar Switch Controller for a Packet Switched Supercomputer Network," *SIGARCH Computer Architecture News*, 19(4), 1991, pp. 69–79.

[4] de Graaf, A. C., A. J. van Genderen, "SLS: Switch-Level Simulator," The Nelsis IC Design System Documentation, Delft University of Technology, 1988.

[5] Goodman, J. R., and C. H. Sequin, "Hypertree: A Multiprocessor Interconnection Topology," *IEEE Transaction on Computers*, C-30(12), December 1981, pp. 923-933.

[6] Gottlieb, A., R. Grishman, C. Kruskal, K. McAuliff, L. Rudolph and M. Snir, "The NYU Ultracomputer - Designing an MIMD Shared Memory Parallel Computer," *IEEE Transaction on Computers*, C-32(2), February 1983, pp. 175–189.

[7] High Performance Parallel Interface (HIPPI) Data Frame Control Requirements, ANSI X3T9/89-146 Rev 2.3.

115661

[8] High Performance Parallel Interface (HIPPI) Mechanical, Electrical and Signalling Requirements, ANSI X3T9/88-127 Rev 7.1.

[9] Moona, R., V. Rajaraman, "Multidimensional Multilink Multicomputer: A General Purpose Parallel Computer," *Journal of Indian Institute of Science*, 71(2), 1991.

[10] Seitz, C. L., "The Cosmic Cube," *Communications of the ACM*, 28(1), January 1985, pp. 22–33.

[11] "The Nelsis IC Design System Users' Manual," TU Delft Software Distribution, Delft University of Technology, April 1989.

A 115520

CSE-1993-M-SAR-DES