T
D65-87
KUL

DCMIN - A NEW INTERCONNECTION NETWORK TOPOLOGY FOR PARALLEL PROCESSING

A THESIS

submitted in fulfilment of the requirements for the award of the degree
of
DOCTOR OF PHILOSOPHY
in
ELECTRONICS AND COMMUNICATION ENGINEERING

By

KULDIP SINGH

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING
UNIVERSITY OF ROORKEE
ROORKEE-247 667 (INDIA)

April 1987

CANDIDATE'S DECLARATION

I hereby certify that the work which is being presented in this thesis entitled 'DCMIN - A New Interconnection Network Topology for Parallel Processing' in fulfilment of the requirement for the award of the Degree of Doctor of Philosophy submitted in the Department of Electronics and Communication Engineering of the University is an authentic record of my own work carried out during a period from Feb. 1984 to April 1987 under the supervision of Dr. N. K. Nanda and Dr. R. C. Joshi.

The matter embodied in this thesis has not been submitted by me for the award of any other Degree.

K Singh.
(KULDIP SINGH)

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

[illegible] 18/4/87
(N.K.NANDA)
Professor and Head,
Electronic and Comm. Engg.
Department

[illegible] 18.4.87
(R.C.JOSHI)
Reader,
Electronics and Comm. Engg
Department

The candidate has passed the Viva-Voce examination held on ________________ at ____________. The thesis is recommended for award of Ph.D. Degree.

1. ____________________
(Guide)

1. ____________________
(External Examiner)

2. ____________________
(Guide)

ABSTRACT

This dissertation addresses the problem of designing a reliable and computationally faster interconnection network with reduced complexity compared to the conventional multistage networks. The importance of 4-shuffle interconnection pattern for the problems requiring concurrent processing has been demonstrated. A 4x4 switch named as Dual Interconnection Modular Switching Element (DIMSE) has been proposed as an alternative to 2x2 switching element for use in Multistage Interconnection Networks (MINs). Design of a new MIN topology, designated as Dual Cube Multistage Interconnection Network (DCMIN), for use in parallel processing environments, has been proposed.

The performance and complexity of DCMIN has been evaluated and compared with the conventional cube type MINs. It has been established that in addition to being less complex and computationally faster, the DCMIN is also cost-effective. Conventional routing algorithms and relatively simpler fault-diagnosis techniques can be readily made applicable to DCMIN. Fault-tolerance capability of the proposed network is evaluated using multiple-paths through an extended topology of DCMIN called EDCMIN and multiple-passes techniques. A simple and elegant analytic model of DCMIN, using Kronecker product of matrices of DIMSEs, has been devised. DCMIN topology has the partitioning property, i.e., the network can be implemented in partitionable SIMD/MIMD environments as well.

ACKNOWLEDGEMENT

I wish to express my appreciation and profound gratitude for the continuous interest taken and encouragement given by Dr.N.K.Nanda and Dr.R.C.Joshi who supervised this work. Their active association through suggestions, critical comments and guidance for carrying out this research is gratefully acknowledged.

This work could not have been completed without the whole hearted help and encouragement given by Dr.P.P.Sahgal, Professor and Director, Continuing Education Department. My sincere thanks to him for providing congenial environment to carry out the work.

Special acknowledgement is due to Dr.Harpreet Singh, Professor, Department of Electrical and Computer Engineering, Wayne State University, Detroit, U.S.A. for his encouragement and fruitful discussions on several occasions.

No words can express my thanks and gratitude to Head of Electronics and Communication Engineering Department for providing me the excellent and informal working conditions in the Department to carry out this work.

I am very much grateful to my mother, wife Satpal Kaur and son Jasmeet who have been a constant source of inspiration and encouragement.

I also wish to thank my colleagues Dr.I.J.Gupta and Dr.N.Malaviya for discussions and assistance.

Finally, thanks are due to Shri Amarjeet Singh for neat typing and Shri B.R.Sharma for drawing the figures nicely.

TABLE OF CONTENTS

Page

LIST OF ABBREVIATIONS USED

ADM	Augmented Data Manipulator
CPU	Central Processing Unit
DCMIN	Dual Cube Multistage Interconnection Network
Dcube	Dual Cube
DTR	Data Transfer Register
DFA	Dynamic Full Access
DIMSE	Dual Interconnection Modular Switching Element
EDCMIN	Extra Stage Dual Cube Multistage Interconnection Network
FA	Full Access
FFT	Fast Fourier Transform
IADM	Inverse Augmented Data Manipulator
ICN	Interconnection Network
I/C	Interchange
INDRA	Interconnection Networks Designed for Reliable Architecture
LSD	Least Significant Digit
MCN	Multistage Cube Network
MIMD	Multiple Instruction Multiple Data Stream
MIN	Multistage Interconnection Network
MISD	Multiple Instruction Single Data Stream
MPP	Massively Parallel Processor
MSD	Most Significant Digit
O( )	Of the order of
P	4-Shuffle Permuter
PE	Processing Element

QR	Quaternary Representation
S	Stack
SE	Switching Element
SIMD	Single Instruction Multiple Data Stream
SISD	Single Instruction Single Data Stream
TDCMIN	Truncated Dual Cube Multistage Interconnection Network
ULM	Universal Logic Module
VLSI	Very Large Scale Integration
WSI	Wafer Scale Integration
$\oplus$	Exclusive-OR
$\otimes$	Kronecker Product
σ	General NxN Permuter

CHAPTER I

INTRODUCTION

Single processor systems, which are based on the traditional Von Neumann or microprogrammed architectures, have reached their limits in computing capability. They do not provide for parallel computation of instructions and data, which makes them inappropriate for many present day applications, requiring high computational power and speed. As a result, a great deal of architectural research was directed towards overcoming the sequential barrier and evolving systems that could perform computations in parallel [9,10,38,45,46]. This culminated in the evolution of multiprocessor systems with a wide variety of applications, where higher processing speed is achieved through the concurrent or parallel operation of a number of processors.

It is predicted that the advances in VLSI and WSI technologies will, in the near future, enable designers to integrate hundreds or even thousands of processing elements on a single wafer [155]. As the component density of integrated circuits continues to increase, computer designers will be more concerned with the interconnection of the processors than with the processor design itself. One of the prominent components, which will wield great influence on such systems, is the Interconnection Network (ICN). The performance and fault-tolerance of multiprocessor systems are subsequently dictated more by the properties of ICN than by the individual units. Hence, the design of ICN assumes a significant role in parallel architectures, such as array processor and multiprocessor systems.

The time shared bus is the simplest and earliest form of ICN. In a uniprocessor architecture, functional modules are connected to a bus and the resulting systems vary widely in complexity depending upon the number and nature of the modules. A multiprocessor environment is formed by interconnecting a number of CPUs, memory units, and controllers in addition to buses. The early multiprocessor systems consisted of only a few processors. As computer systems evolved from uniprocessor to multiprocessor systems, multiple buses or cross bar switches were used to connect processors and/or memories. The advent of low cost microprocessors and the critical need of high performance computing has triggered intensive research on various ICNs. Concepts like telephone switching system [20,144] and computer communication [9,144] were used in the design and analysis of ICNs. The current trend is to design ICNs in such a way that the multiprocessor system level functions such as resource sharing and synchronization can be easily implemented by using ICN.

1.1 INTERCONNECTION NETWORK (ICN)

An ICN basically provides communication between the sets of source and destination ports to which different units such as processors, memories etc. are connected. Functionally, under a transfer instruction, data from a specific source node is transferred to some prespecified destination or vice-versa, following a route determined by the control signals of the network. To do this, it has to be necessarily a switching network. Accordingly, an ICN may be looked upon as a system of switches, that can be opened or closed to provide a path from a set of input terminals to a set of output terminals. The smallest switch of interest, called Interchange (I/C) box, Switching Element (SE) or β-element, has two switching states corresponding to the two possible switching modes, i.e., direct or cross, depending upon the state of control signal. Usually the network is organised into several stages of switching elements, connected through links, to achieve full connectivity and is called Multistage Interconnection Network (MIN). In general, a MIN allows processor-to-processor or processor-to-memory communication.

1.2 MULTISTAGE INTERCONNECTION NETWORKS (MINs)

A multistage network consists of more than one stage of switching elements and is usually capable of connecting an arbitrary input terminal to an arbitrary output terminal. Because of their structure, MINs are capable of supporting a large number of computing units and large number of simultaneous connections using a reasonable number of SEs. With proper design a connecting network can provide connections with reasonably short delays and with a relatively simple control structure. Also, MINs have built in flexibility and redundancy which can be utilized for fault-tolerance with graceful degradation in the computational speed [117]. Multistage interconnection networks mentioned in literature can be divided into three classes:

(i) Blocking:

In blocking networks, simultaneous connections of more than one input/output terminal pair may result in conflicts in the use of network communication links.

(ii) Rearrangeable:

A network is called rearrangeable nonblocking network, if it can perform all possible connections between inputs and outputs, by rearranging its existing connections so that a connection path for a new input-output pair can always be established.

(iii) Nonblocking:

A network which can handle all possible connections without blocking is called a nonblocking network.

The Clos nonblocking network employing three stages of crossbar switches [36] was forerunner in the design hierarchy of MINs. Subsequently, Benes network [21] which is rearrangeable network, was evolved. What followed afterwards was a class of blocking networks such as Omega [79], Stran Flip [15], Indirect Binary m-cube [106], Generalized Cube [121], Generalized Shuffle [24] and Dual Cube networks [126]. In these networks, there is only one path between any source-destination pair. These networks do not realize all possible permutations of input/output combinations. However, these can perform some of the permutations which may be useful in computer communication networks. The blocking networks are advantageous because they implement simple control algorithms and employ $(N/2 \log_2 N)(2x2)$ or $(N/4 \log_4 N)(4x4)$ switches for N number of functional units.

1.3 FAULT-TOLERANCE

A fault-tolerant ICN can tolerate faults to some degree and still provide reliable communication between any input/output pair. Fault-tolerance is usually achieved by introducing redundancy in the MIN. By introducing hardware redundancy more number of alternate paths are created between source/destination pairs to augment the permutation capability of the network which makes it fault-tolerant. Hardware redundancy can be introduced in the network (i) by increasing the number of stages [3,116], (ii) by using fault-tolerant SEs [84] or (iii) by providing extra links [6,35]. These techniques have been employed in the redundant networks such as Data Manipulator [48], IADM [89, 90], Extra Stage Cube and Delta Networks [3,116], Gamma Network [103], INDRA Network [109] and Augmented C-Network [111]. An alternative approach to fault-tolerance by means of redundancy in time (multiple-passes) has been proposed in [117] and generalized in [8]. An important property of MIN is Dynamic Full Access (DFA) characteristic. A network has DFA property if each of its inputs can be connected to any one of its outputs via a finite number of passes [118].

1.4 MOTIVATION FOR THE PRESENT WORK

It has been observed that the nucleus of a modern day practical multiprocessor system is not so much the individual functional modules rather the interconnection network. In this respect, therefore, it is essential that the interconnection networks be designed as efficiently as possible and this is the prime justification for carrying out the work reported in this thesis. The important factors which influence the design of ICN are:

(i) Computational Speed:

Computational speed can be increased by reducing the propagation delay in the ICN. Since the switch delay decreases with scaling, the speed at which a circuit can operate is dominated by the interconnection delay. Thus, the computational speed can be increased by reducing the number of stages in a network.

(ii) Complexity:

Complexity of the network is directly proportional to the number of switching elements used and the number of external links in the network. Accordingly, if the size of the SE is increased the number of SEs per stage will decrease. The larger size SEs with appropriate interconnection patterns can reduce the number of stages for the same number of functional units. Reduction in the number of stages will consequently reduce the number of external links.

(iii) Reliability:

Because of the system complexity, achievement of high reliability is a significant task. The ICN should have inherent fault-tolerance properties. It should be possible to achieve fault-tolerance either by adding hardware redundancy or through multiple passes. In addition the algorithms to detect and locate faulty elements, should be simple and elegant.

(iv) Adaptability:

To suit the requirements of multiple-SIMD and partitionable SIMD/MIMD environments, it should be possible to partition the network into smaller size networks.

An NxN conventional MIN employs more than one stages of 2x2 SEs usually interconnected through perfect shuffle patterns. This gives ($\lceil(\log_2 N)\rceil$) as the lower bound on the number of stages. Accordingly, the network complexity and computational speed, become proportional to ($\lceil(\log_2 N)\rceil$). This puts the limitation on the capabilities of currently employed MINs in terms of computational time and complexity. With the increasing complexity of multiprocessor systems, due to the use of large number of functional elements, a logical question arises - is there an alternative MIN topology that allows the multiprocessing boundaries to be extended for the same amount of resources and cost ? It is this question to which the thesis addresses **itself.**

1.5 STATEMENT OF THE PROBLEM

This thesis attributes itself to the problem of designing a simple but computationally faster and cost-effective multistage interconnection network compared to the conventional MINs. The objective is to find a new topology for a class of interconnection networks, by increasing the size of conventional 2x2 switching elements. Specifically, the problems considered in this thesis can be stated as follows:

(1) To examine a 4x4 switching element and propose its realization.

(2) To investigate the link pattern to interconnect the stages of the proposed switching elements and hence evolve a new topology of a class of MIN with reduced number of stages compared to the conventional lower bound of $(\lceil \log_2 N \rceil)$.

(3) To compare the performance and complexity of the proposed network topology with the conventional topology.

(4) To evolve an analytical model of proposed MIN.

(5) To exploit the use of the proposed topology in the various parallel processing architectures.

1.6 ORGANIZATION OF THE THESIS

Applications of interconnection networks and their general classification have been reviewed in Chapter II. The terminology and definitions used in Chapter II give a general base for understanding the work carried out in the thesis.

Chapter III defines 4-shuffle permutation. It has also been established that 4-shuffle interconnection pattern has an important role in parallel processing and reduces the computational complexity to $O(\lceil \log_4 N \rceil)$.

A 4x4 switch named Dual Interconnection Modular Switching Element (DIMSE) and its implementation is presented in Chapter IV, where the design of single and multistage networks using DIMSEs and 4-shuffle patterns has been given. This new multistage network topology is designated as Dual Cube Multistage Interconnection Network (DCMIN). Finally, the complexity and performance of DCMIN have been evaluated and compared with a conventional cube type network.

Reliability aspects of the proposed DCMIN are discussed in Chapter V. A description of simple procedures for fault - diagnosis and evaluation of DFA capabilities of DCMIN under various fault conditions has been given.

Chapter VI is devoted to investigate the general properties of DCMIN such as modelling, adaptability and fault tolerance aspects.

TDCMIN, a derivative of DCMIN topology, when the size N of the network is such that $N = 2^m \neq 4^n$, has been evolved in Chapter VII. The TDCMIN topology has been evaluated from various angles.

Finally, the conclusions of the work carried out alongwith some proposals for further research in this area are given in Chapter VIII.

CHAPTER II

OVERVIEW OF INTERCONNECTION NETWORKS

An interconnection network is basically a switching network, which is designed to facilitate efficient interprocess and interprocessor communication in parallel architectures. Earlier when computer systems were confined within Von Neumann architectural boundaries and the hardware cost was a significant limiting factor, interprocess and interprocessor communication were not prominent issues. Consequently, the design of a cost-effective interconnection network was a trivial and relatively unimportant task. However, contemporary IC technology and ultra high speed computational requirements have ushered in entirely new architectural challenges. It is now economically feasible and practically desirable to construct a multiprocessor system by interconnecting hundreds or thousands of processors [155]. Such systems are called parallel systems. However, the performance and utilization of such systems, in general, are limited by the extent upto which interprocessor communication is possible. Parallel processing techniques use interconnection networks to facilitate concurrent operations.

A parallel processing system comprising multiple processors is implemented with IC chips and a network interconnecting them. The IC chips though less likely to be faulty, are replaceable and the performance of a multiprocessor system is, therefore, a function of reliable operation of ICN [117]. This chapter is devoted to the study of various aspects of ICNs such as their requirements, basic interconnection functions, topologies, reliability and applications in practical parallel systems. However, before reviewing the ICNs a brief discussion on parallel systems is undertaken. ICNs also play an important role in resource allocation and reconfiguration of the parallel systems. At the end of this chapter comparatively recent modes of parallel systems are discussed briefly.

2.1 PARALLEL SYSTEMS

Parallel processing has made possible the computational power and speed required by many present day real time tasks, where there is need to process immense data sets. These tasks include seismic data processing, weather forecasting, air traffic control, satellite collected imagery analysis, robot vision and speech understanding etc. The parallel systems support to meet the computational goals for all these applications. Parallel systems can be defined broadly as a unified system containing multiple processors capable of simultaneous operation. Depending upon the technique of introducing parallelism, parallel systems may be classified into the following four different categories [74]:

(i) Multicomputer Systems

These are assembled from N computers in such a way that any two computers may communicate with each other via an interconnection bus connected with their input/output devices.

(ii) Multiprocessor System

The interconnection bus of the multiprocessor system allows activation of a direct communication path between any two functional units contained in the system.

(iii) Array Processor

It consists of N processors each handling same instruction issued by a single control unit, on its own data.

(iv) Pipelining

It contains pipelined processors with an instruction broken into a series of simple functions to be performed in pipelined segments, i.e., the application of assembly line technique.

The degree of complexity of such systems can be gauged from the fact that hundreds or even thousands of processors are used to achieve execution rates as high as billion floating point instructions per second [49]. Such systems have been classified on the basis of the modes of operation, i.e., how the machine relates its instructions to the data being processed [55]. Four broad classifications that have emerged are Single Instruction Single Data Stream (SISD), Single Instruction Multiple Data stream (SIMD), Multiple Instruction Single Data stream (MISD) and Multiple Instruction Multiple Data stream (MIMD). Out of these, SIMD and MIMD modes of operation are of immediate interest in parallel processing systems, involving interconnection networks.

Typically, an SIMD machine is a computer system consisting of a control unit, N processors, N memory modules, and an interconnection network. The control unit broadcasts instructions to the processors, and all active processors execute the same instruction at the same time. These machines are designed to exploit the inherent parallelism of such tasks as matrix operations and such problem domains as image processing where the same operation is performed on many different matrices or image elements simultaneously. Detailed information about the SIMD systems is available in [11,64,66,136,144]. Examples of SIMD machines that have been actually implemented are Illiac IV [25], STARAN [15] and MPP [16,17]. Two common configurations of SIMD machines are processing element-to-processing element organization and processor-to-memory organization as shown in Figs.2.1 and 2.2 respectively.

The MIMD Mode of parallelism differs from the SIMD mode in that the processors in an MIMD system operate asynchronously with respect to each other, unlike the lock - step synchronous operation of all the active processors in an SIMD machine. Typically, an MIMD machine consists of N processors, N memory modules, and an interconnection network. Each of the N processors follows its own program. Thus there are multiple instruction streams. Each processor fetches its own data on which to operate, resulting in multiple data streams, as in the SIMD systems. The interconnection network provides communication among the processors and memory modules. In an MIMD system, because each processor executes its own program, inputs to the network arrive independently, i.e., asynchronously. Examples of MIMD systems that have been implemented are Cm* [69,138] and C.mmp [157]. MIMD machines can be organized in the processing element-to-processing element configuration, or in the processor-to-memory configuration as depicted in Figs.2.3 and 2.4. When using the

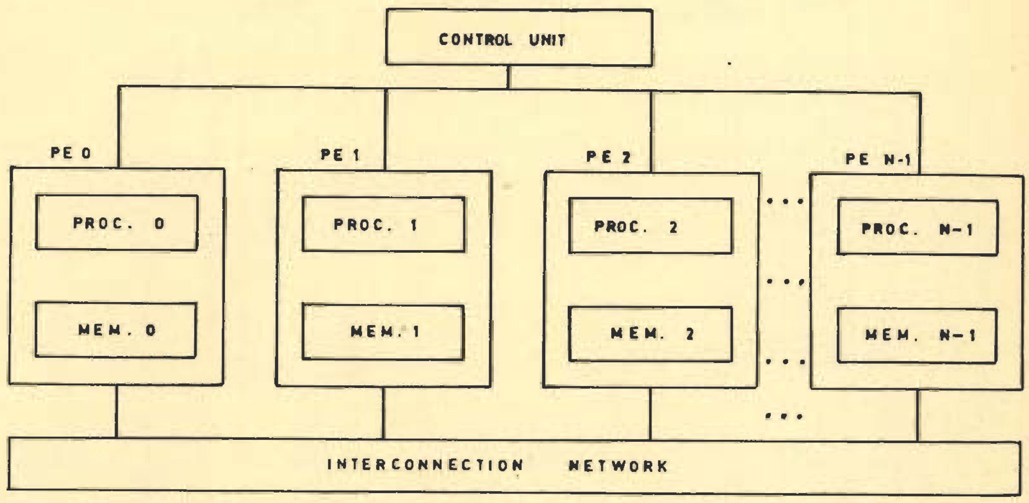


FIGURE 2.1 PROCESSING ELEMENT-TO-PROCESSING ELEMENT SIMD-MACHINE CONFIGURATION WITH N PROCESSING ELEMENTS (PEs)

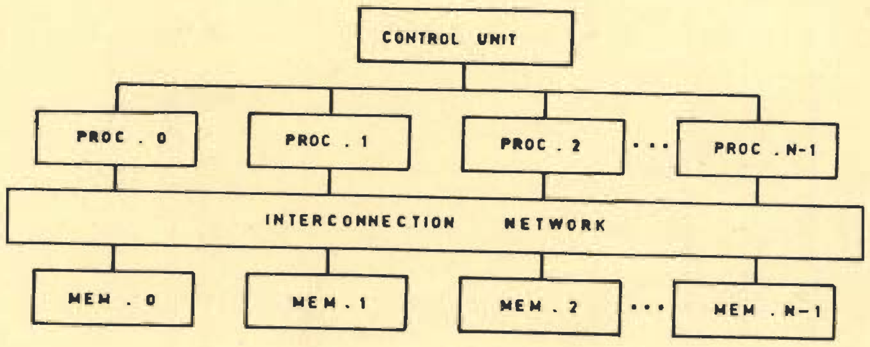


FIGURE 2.2 PROCESSOR -TO-MEMORY SIMD-MACHINE CONFIGURATION WITH N PROCESSORS AND N MEMORIES

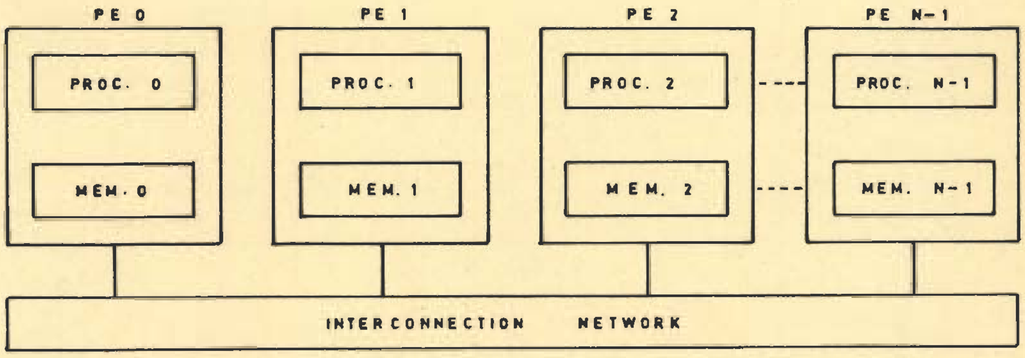


FIGURE 2.3 PROCESSING ELEMENT-TO-PROCESSING ELEMENT MIMD-MACHINE CONFIGURATION WITH N PROCESSING ELEMENTS

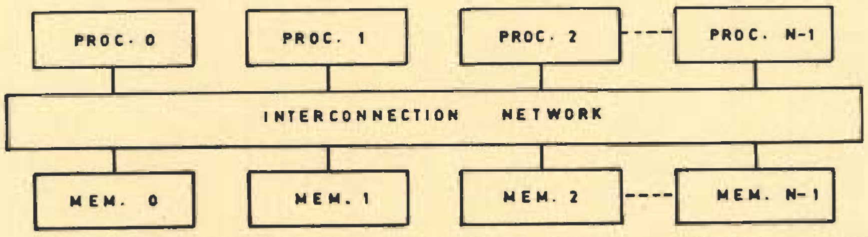


FIGURE 2.4 PROCESSOR-TO MEMORY MIMD-MACHINE CONFIGURATION WITH N PROCESSORS AND N MEMORIES

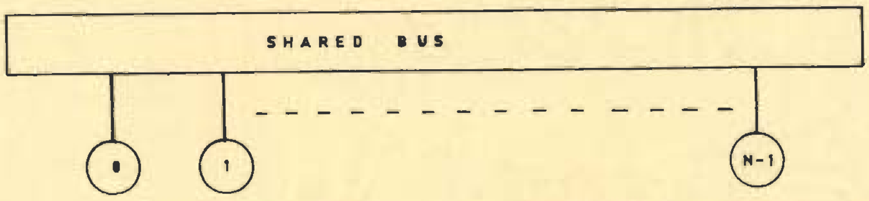


FIGURE 2.5 A SINGLE SHARED BUS PROVIDING COMMUNICATIONS FOR N DEVICES

processor-to-memory configuration, a cache may sometimes be associated with each processor, for example, as in an ultracomputer [61]. More information about MIMD systems and their use is available in [11,67,136].

Multiprocessor systems, incorporating a large number of processors, is a topic of continuous research and development [38,67,107]. Many distinct approaches have been persued to organize the large number of processors to achieve high performance. For example, a Massively Parallel Processor (MPP) [16, 17], consists of 2^{14} processing elements. A major problem in designing large-scale parallel/distributed systems is the construction of an interconnection network to provide interprocessor communication, and in some cases memory access for the processors. This has been discussed hereunder in detail.

2.2 INTERCONNECTION NETWORKS

The interconnection scheme of processing or memory modules must provide fast and flexible communication between modules at a reasonable cost. A single shared bus, as shown in Fig.2.5, is not sufficient, because it is often desirable to allow all processors to send data to other processors simultaneously. Ideally, each processor should be linked directly to every other processor via dedicated paths so that the system is completely connected as shown in Fig.2.6. Unfortunately, these configurations are highly impractical when N is large. An alternative interconnection scheme that allows all processors to communicate simultaneously is the crossbar network shown in Fig.2.7. A normal crossbar can be defined as a switching network

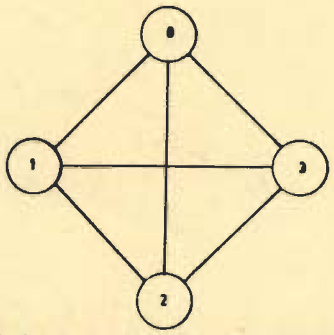


FIGURE 2.6 A COMPLETELY CONNECTED SYSTEM FOR N=4

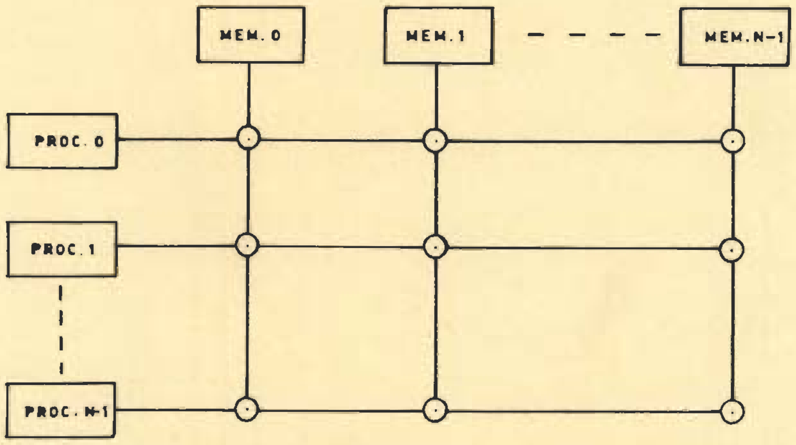


FIGURE 2.7 A CROSS BAR NETWORK CONNECTING N PROCESSORS TO N MEMORIES

that allows the arbitrary one to one interconnection of N inputs to N outputs without contention. A generalized cross-bar also allows some input to be connected to more than one outputs (broadcasting). However, the complexity and cost of such networks increases as the square of network size N. To solve the problem of providing fast and efficient communication at reasonable cost, many different networks between the two extremes of single bus and completely connected scheme have been proposed and are reviewed in numerous survey articles and books such as [9,10,28,45,49,64,67,74,75,87,121,123,143,144,155].

These are broadly classified as static and dynamic networks which are further discussed in Section 2.2.3. Various attributes of any interconnection network are:

(i) Switching methodology

(ii) Interconnection functions

(iii) Interconnection topologies

(iv) Connecting capabilities

(v) Control strategy

(vi) Reliability

These aspects of interconnection networks are described briefly hereunder.

2.2.1 Switching Methodology

The two major switching methodologies are circuit switching and packet switching. In circuit switching, a path is physically established between a source and a destination. In packet switching, data is put in a packet and routed through the interconnection network without establishing a physical connection path. In general,circuit switching is more suitable for bulk data transmission and packet switching is more efficient for short data messages.

Another option, integrated switching, includes the capabilities of both circuit switching and packet switching. Most SIMD interconnection networks assume circuit switching operations, whereas packet switched networks have been suggested mainly for MIMD machines [87,121,123,125].

2.2.2 Interconnection Functions

From analytical point of view, an ICN may be viewed as a set of interconnection functions. Each interconnection function is a bijection on the set of input/output (source/destination) addresses, integers from 0 to N-1. A bijection is a one-to-one and onto mapping. One-to-one means that a processing element receives data from exactly one PE when interconnection function is executed. Mathematically, one-to-one implies that an integer in the set of source/destination addresses is mapped to by exactly one integer in the set when an interconnection function is applied. Onto means that every PE receives data from some other PE when an interconnection function is executed. Mathematically, onto implies that every integer in the set of PE addresses is mapped to by some integer in the set when an interconnection function is applied. In the case of interconnection networks bijection is also termed as 'permutation', i.e., from a set onto the same set [122].

Let an ordered list of elements be

0,1,2,.......,(N-1)

then an interconnection function f permutes this list into

f(0), f(1),......,f(N-1).

Thus the execution of data exchange function makes a PE i to copy the contents of its Data Transfer Register (DTR) into the DTR of PE f(i) for all values of i. Mathematically,

Let i $\longrightarrow$ address of a processor connected as source to an ICN

f(i) = j $\longrightarrow$ address of a processor connected as destination to an ICN

then the data exchange function will result in

<PE i> $\longrightarrow$ < PE j>

Several interconnection functions have been defined and used for implementation of ICNs [64]. However, broadly speaking most of the proposed or existing networks incorporate one of the four types of connection functions as described below:

(i) Shuffle-Exchange

The shuffle-exchange network consists of a shuffle function 'S' and an exchange function 'E' defined as

$$S(b_{m-1}b_{m-2}\cdot\cdot b_i\cdot\cdot b_1b_0) = b_{m-2}b_{m-3}\cdot\cdot b_i\cdot\cdot b_1b_0b_{m-1} \quad ..(2.1)$$

and

$$E(b_{m-1}b_{m-2}\cdot\cdot b_i\cdot\cdot b_1b_0) = b_{m-1}b_{m-2}\cdot\cdot b_i\cdot\cdot b_1\bar{b}_0 \quad ..(2.2)$$

where each b_i is a binary bit and $(b_{m-1}b_{m-2}\cdot\cdot b_i\cdot\cdot b_1b_0)$ is the binary address of a functional unit.

Thus, both shuffle and exchange functions manipulate a binary string which is actually an address of source or destination in binary format. Features of shuffle exchange networks are discussed in [29,30,53,93, 95,108,121,122,153].

(ii) Cube

The cube network consists of m interconnection functions defined by

$$\mathrm{Cube}_i(b_{m-1}b_{m-2}\cdot\cdot b_i\cdot\cdot b_1 b_0) = b_{m-1}b_{m-2}\cdot\cdot \bar{b}_i\cdot\cdot b_1 b_0 \quad \cdot\cdot(2.3)$$

$$\text{for } 0 \leq i \leq (m-1)$$

The cube network forms the underlying structure of many multistage networks such as Banyan [60], STARAN Flip [15,18], Benes [21], Indirect Binary m-cube [106], Generalized Cube [125] and Extra State Cube [3]. Various properties of cube network are discussed in [53,93,94,95, 121,122,148].

The networks based on the cube and shuffle-exchange functions have been proposed for use in the systems such as PASM [124], PUMPS [27], Ultra Computer [61] and Data-Flow Machines [42].

(iii) Illiac

The Illiac network consists of four interconnection functions defined as follows:

$$\mathrm{Illiac}_{+1}(B) = B+1 \text{ modulo } N$$

$$\mathrm{Illiac}_{-1}(B) = B-1 \text{ modulo } N$$

$$\text{Illiac}_{+n}(B) = B+n \text{ modulo } N$$
$$\text{Illiac}_{-n}(B) = B-n \text{ modulo } N$$

where $n = \sqrt{N}$ is assumed to be an integer.

This type of network is included in the MPP [16,17] and DAP [65] SIMD systems. It is similar to the eight nearest-neighbour network used in CLIP4 [56] and BASE8 [112] machines. Various properties and capabilities of illiac network are discussed in [25,97,121, 122].

(iv) PM2I

The plus-minus 2^i(PM2I) network consists of 2m interconnection functions defined by

$$\left.\begin{aligned} PM2_{+i}(B) &= B+2^i \text{ modulo } N \\ PM2_{-i}(B) &= B-2^i \text{ modulo } N \end{aligned}\right\} \quad \text{for } 0 \leq i \leq (m-1)$$

The PM2I connection pattern forms the basis for the Data Manipulator [48], ADM [124] and Gamma [103] multistage networks. Various properties of PM2I network are discussed in [53,108,121,122,130].

In the above discussion of interconnection functions, the following notations have been used: $N = 2^m$; the binary representation of an arbitrary PE address $B = b_{m-1}b_{m-2}\cdots b_1b_0$ and $\bar{b}_i$ is the complement of b_i where 'b_i' is binary bit and 'm' is any positive integer.

2.2.3 Interconnection Network Topologies

An interconnection network can be depicted by a graph in which nodes represent switching points and edges represent communication links. The overall graph representation is called network topology. Many topologies have been proposed or used for ICNs. They tend to exhibit some regular pattern and can be grouped into two broad categories (i) static and (ii) dynamic. In the static topology, each switching point is connected to a processor, while in the dynamic case only the switching points in the input/output sides are connected to processors [9].

Several static structures with regular topology have been proposed which include Ring [85], Tree [63], Near-Neighbour Mesh [12], Systolic arrays [78], Pyramids [140] and Hyper cubes [24,114]. A comparative study of these static networks can be found in [49,151]. In these networks each processor can communicate only with a small number of processors directly and communication with others involves the transfer of information through one or more intermediate processors.

Dynamic networks allow different connections to be set up between processing elements by changing their internal states. There are three topological classes in the dynamic catagory - single stage, multistage and crossbar.

(i) Single Stage

A single stage network is composed of a stage of switching elements cascaded to a link connection pattern. The single stage network is also called a recirculating network because the data item may have to recirculate through the single stage several times before reaching their final destination. The recirculating shuffle - exchange network is an example of a single stage network [135]. Single stage networks are discussed in [121].

(ii) Multistage

A multistage network consists of more than one stage of switching elements and is usually capable of connecting an arbitrary input terminal to an arbitrary output terminal. Several multistage networks, with $N = 2^m$ inputs/outputs and $\log_2 N$ stages of 2x2 SEs, (Fig.2.8), have been studied in the literature [28,51, 155]. These include the Base line [152], Omega [79], Banyan [60] and Binary m-Cube [106] networks. In these networks each stage uses N/2 SEs and is connected to the next stage by atleast N paths. The network delay is proportional to the number of stages in a network. Fig.2.9 shows a multiprocessor system with 8 processors interconnected through an omega network. There is another class of multistage networks which does not use 2x2 SEs. These networks are called redundant-path interconnection networks or data manipulator networks. Examples are the Data Manipulator [48], Augmented Data Manipulator or ADM [88,89], the Inverse Augmented Data Manipulator or IADM [124] and the Gamma network [103, 110]. A new cell based interconnection network, [156], has been introduced as a general sorting network. The proposed cell is 4x4 switch based on the concept of

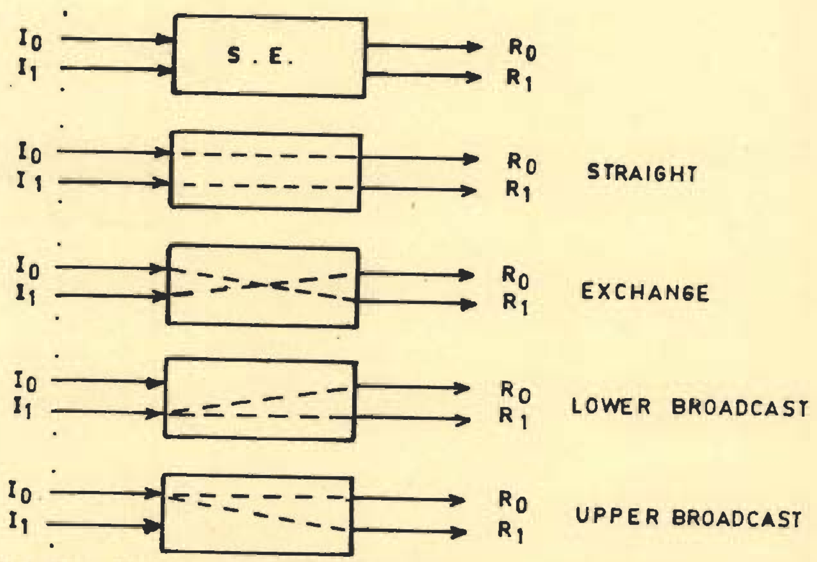


FIGURE 2.8 A TWO-BY-TWO SWITCHING ELEMENT AND ITS FOUR INTERCONNECTION STATES

26

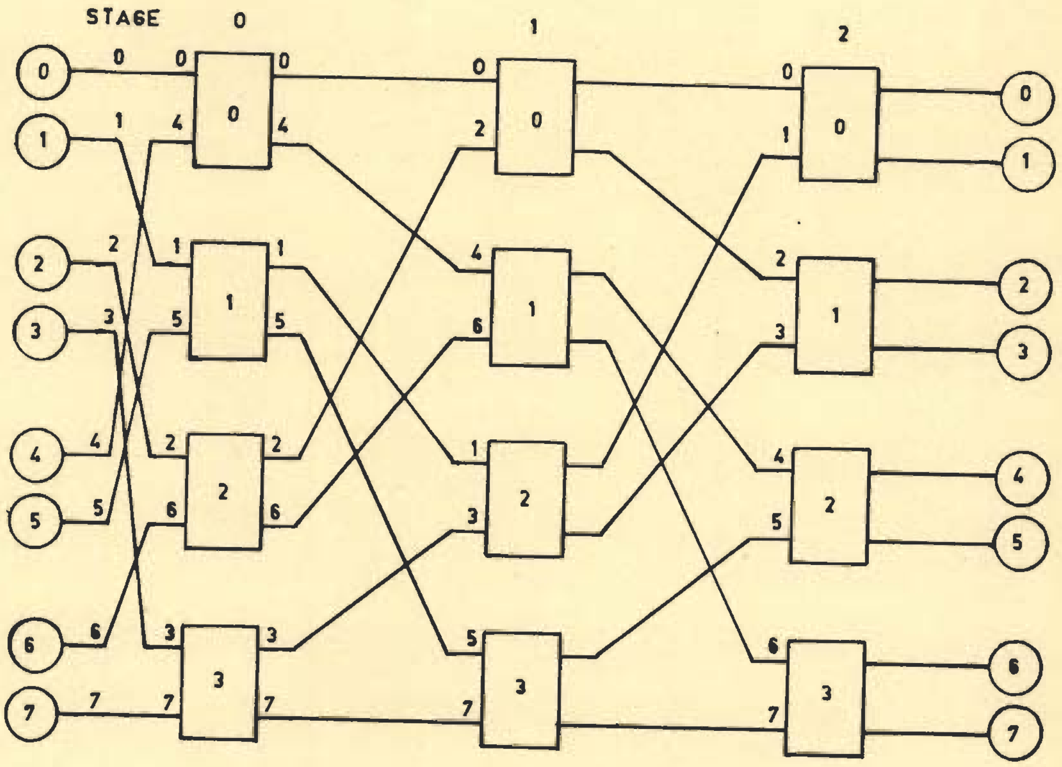


FIGURE 2.9 A MULTIPROCESSOR SYSTEM WITH
8 PROCESSORS USING OMEGA NETWORK

completely interconnected set. Multistage networks differ in the interconnection pattern between stages, the type and operation of individual SEs, and the control scheme for setting up the SEs. The topological equivalance for several of these networks has been established in [5,102,152].

(iii) Crossbar

This has already been discussed in Section 2.2 and is not suitable for use in parallel processing systems because of its high cost.

2.2.4 Connecting Capability of Multistage ICNs

From connecting capability point of view multistage networks can be divided into three broad categories - blocking, rearrangeable and nonblocking [21]. In blocking networks, simultaneous connections of more than one terminal pair may result in conflicts in the use of network communication links. Examples of a blocking network are the Data Manipulator, Omega, Binary m cube, Baseline, SW Banyan and Delta networks. A network is called a rearrangeable network if it can perform all possible connections between inputs and outputs by rearranging its existing connections so that a connection path for a new input-output pair can always be established. A well defined Benes network belongs to this class [21]. The Benes rearrangeable network topology has been extensively studied for use in synchronous data permutation [68,83,96,149], and asynchronous interprocessor communication [33,48]. A network which can handle all possible connections without blocking is called nonblocking network. Two cases have been considered in the literature. In the first case, the Clos network, a one to one connection is made between an input and an output [36]. The other case considers one-to-many connections [142]. Here, a general connection network topology is generated to pass any of the multiple mappings of inputs onto outputs. The crossbar-switch network can connect every input to a free output without blocking.

A class of connecting networks having less connecting capability than the rearrangeable networks are the full access networks [117]. A network has full access property if each of the inputs can be connected to any one of the outputs. Pease's indirect binary m-cube [106] and Lawrie's Omega network [79] are examples of full access network which are neither rearrangeable nor non-blocking. The 8x8 Omega network is illustrated in Fig.2.9. As can be observed there exists a unique route from each input to each output. An example of the network without the full access property can be obtained by deleting the third stage of switching elements in Fig.2.9. A nonblocking network is also rearrangeable, and a rearrangeable network also possesses the full access property. A key issue in the design of connecting networks is the tradeoff between connecting capability and cost, which may be measured by the number of cross points (SEs) and links used and the complexity of the control algorithm software. Greater connecting capability normally implies higher cost. Large non blocking networks are rarely used because of their high cost. Rearrangeability is a very desirable property for telephone switching networks. However, because of the complexity of their control algorithms, rearrangeable networks become undesirable for interconnected computers which utilize inter unit communication only.

2.2.5 Control Strategy

A typical interconnection network consists of a number of switching elements and interconnecting links. Interconnection functions are realised by properly setting the control of the switching elements. The control structure of a network determines how the states of the switch boxes will be set. Two distinct types of control structures have emerged. These are (i) centralized control or flip control and (ii) individual box control or distributed control. In addition sometimes partial stage control or shift control is also used. Most existing SIMD interconnection networks use the centralized control on all the switching elements. A routing tag scheme is generally used for controlling the multistage networks. The use of tags allows network control to be distributed among the processors. The routing tags for one-to-one data transfers consists of m bits. If broadcast capabilities are also to be included then 2m bits are used where m gives the number of stages used in the network. Details about the control schemes can be found in [48,66,79,83,88,89,90].

2.2.6 Fault-Diagnosis and Fault-Tolerance

In parallel processing systems, reliable operation of interconnection networks is an important issue. It is desirable that the ICN should be fault-tolerant. However, in order to achieve fault-tolerance, fault detection and location must be done before the fault-tolerance scheme could be applied. Hence the reliability issue is divided into two parts - fault-diagnosis and fault-tolerance.

(i) Fault-Diagnosis

Fault diagnosis is concerned with detection and location of faults in interconnection networks. The fault-diagnosis problem has been studied for a class of multistage interconnection networks consisting of switching elements with two valid states [51]. The problem is approached by generating suitable fault-detection and fault-location test sets for every fault in the assumed fault model. Usually single stuck-at fault model is assumed. These test sets are then trimmed to a minimal or near minimal sets. It is important to note that to conduct the fault-diagnosis, a fault model must be obtained so that one knows what faults need to be identified and located. A fault model has been introduced, for multistage full-access network constructed from 2x2 SEs [51]. Here a simple and straight forward methodology is developed which requires only four test-sequences for single fault diagnosis, independent of network size. Many variations of this method have been studied, which include, optimization of test sequences [50,141], multiple fault-detection [86], on line diagnosis [4,58,131], fault diagnosis of multistage Banyan interconnection network [72], and distributed control interconnection network [40]. The technique has also been extended for fault analysis of FFT processors [70].

(ii) Fault-Tolerance

In full access multistage networks a unique path exists from any input terminal of the network to any output terminal. The presence of single failure among the SEs or the connecting links of these networks destroys the full access property. Even though the failure rates of individual components in the network are very small, the failure rate of the entire network can be quite high in systems of large size. Therefore, fault-tolerance is a necessary attribute for the continued operation of such systems. To improve upon the reliability, several interconnection networks with redundant paths between source and destination terminals, have been proposed [3,89,90,98,103,109,111,116]. Techniques for introducing fault-tolerance include augmenting the network with extra stages, increasing the size of the switching elements, and adding redundant switching elements and links. Fault-tolerance of some of these networks is limited to one-to-one connections while some networks allow permutations of data to be performed under failures. Networks capable of fault-tolerant routing of permutations include the Merged Delta Networks, Augmented C-network [111], INDRA Network [109], and Generalized INDRA network [110].

A common technique for improving the reliability and fault-tolerance of a unique path multistage interconnection network is the addition of extra switching stages. The addition of an extra switching stage in a multistage network consisting of $\log_2 N$ stages of 2x2 SEs, introduces an extra path for routing every input-output connection through the network. The extra-stage networks provide fault-tolerance at a modest overhead. Examples include the Extra-Stage Cube network [3], Banyan networks with redundant stages [31] and Extra-Stage-Delta network [116].

An alternate approach to fault tolerance is obtained by means of redundancy in time. Thus, data may be routed in a unique-path network in the presence of faults by performing multiple passes through the network. The network is said to possess DFA capability if every processor in the system can communicate with other processor in a finite number of passes through the network, routing the data through intermediate PE's if necessary [117]. Even though the failure of a single component destroys the full-access capability, a large number of faults do not destroy the DFA capability. The general techniques for checking the DFA property of a faulty network consisting of 2x2 switching elements, with respect to a fault model, that allows only the switching elements to be stuck in one of their valid states, have been discussed in [118]. A more general fault model employing adjacency and reachability matrices of the connectivity graph of the faulty network for testing the DFA property has been used in [8]. The DFA has also been used for performance analysis of β-networks [119].

2.3 ADDITIONAL MODES OF PARALLEL ARCHITECTURE

In addition to providing communication paths in SIMD and MIMD environments, interconnection networks play an important role in other modes of parallel architecture also. These classes of architecture include multiple-SIMD and partitionable SIMD/MIMD as defined below.

2.3.1 Multiple-SIMD System

It is a parallel processing system that can be dynamically reconfigured to operate as one or more independent virtual SIMD machines of various sizes. A multiple-SIMD system consists of N processors, N memory modules, an interconnection network, and Q control units, where $Q<N$. The processors, memories, and interconnection network can be organized in different ways, as in SIMD machines. A general model of a multiple-SIMD machine, based on processing element-to-processing element structure, is shown in Fig.2.10. Each of the multiple control units can be connected to some subset of PEs. If PE i and PE j are assigned to different control units, they are no longer following the same instruction stream and will act independently. By assigning PEs to different control units, independent virtual SIMD machines of various sizes can be created [122].

If the virtual SIMD machines, that can be formed in a multiple-SIMD system, are to be independent the network these share must be partitionable into independent subnetworks. The partitioning of a network into independent subnetworks to suit the requirements of multiple-SIMD system, has been proposed in [122].

2.3.2 Partitionable SIMD/MIMD System

A partitionable SIMD/MIMD machine is a parallel processing system that can be dynamically reconfigured to operate as one or more independent virtual SIMD and/or MIMD machines of various

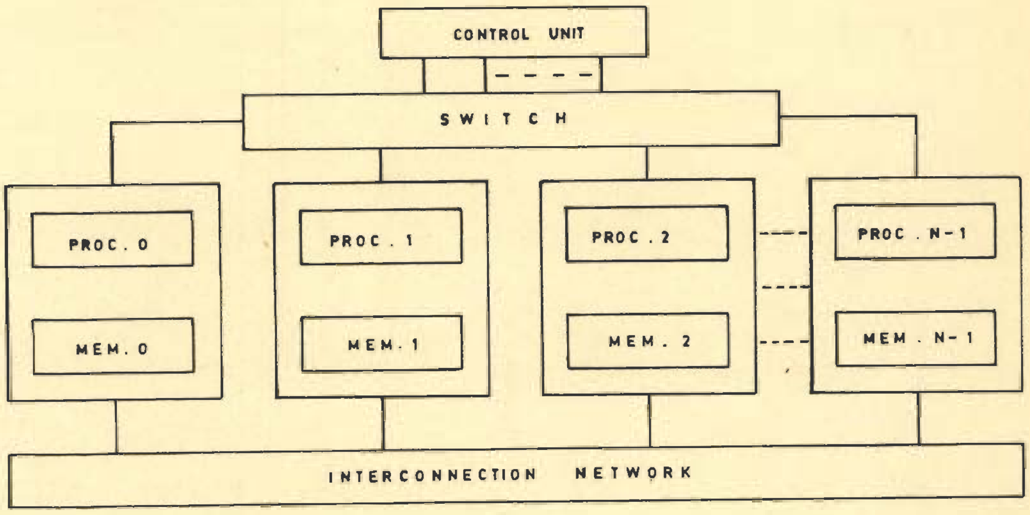


FIGURE 2.10 GENERAL MODEL OF A MULTIPLE-SIMD SYSTEM

sizes. A partitionable SIMD/MIMD machine has the same components as a multiple-SIMD machine except that each processor can follow its own instructions (MIMD operation) in addition to being capable of accepting an instruction stream from a control unit (SIMD operation). The system can be partitioned to form independent virtual machines. In this case, each virtual machine can be a SIMD machine or a MIMD machine.

The examples of multiple-SIMD and partitionable SIMD/MIMD systems are PASM [124], PM4 [26], TRAC [77,115].

2.4 CONCLUSION

In this chapter the interconnection networks have been classified from various angles. Their characteristics and desired properties have been discussed in Section 2.2 whereas the applications of the interconnection networks in parallel systems have been outlined in Sections 2.1 and 2.3. Most of the work on multistage interconnection networks is limited to employing the stages of 2x2 SEs interconnected through perfect shuffle connection pattern. The natural lower bound on the number of stages for NxN MIN is ($\lceil \log_2 N \rceil$). Accordingly, the computational speed, complexity and cost of the network becomes proportional to ($\lceil \log_2 N \rceil$). In most of the recent studies of MINs the main emphasis is on finding their equivalence and non-equivalence, comparing permutational capabilities or improving the reliability either through hardware redundancy or by decreasing the computational speed [5,6,8,72,98,100,104, 109,110,111,116,120,152]. No effort has been directed towards decreasing the number of stages thereby decreasing the computational complexity, speed and cost. In the research work undertaken, an effort has been made in this direction and a new approach has been proposed for designing an efficient and cost-effective multistage network [126]. The thesis starts with defining a 4-shuffle pattern, as an improved and better alternative to perfect-shuffle pattern for parallel processing (Chapter III). A 4x4 switch instead of 2x2 conventional switching element alongwith 4-shuffle has been used in designing the multistage interconnection network. The results obtained are derived and discussed in Chapter IV. The suitability of the proposed MIN for implementation in the practical parallel systems has been examined from various angles in Chapters V, VI and VII.

CHAPTER III

4-SHUFFLE INTERCONNECTION PATTERN IN PARALLEL PROCESSING

It has been discussed in Chapter I that an ICN plays a pivotal role in parallel processing architectures. A general review of the MINs, reveals that these employ stages of 2x2 SEs which are usually interconnected through some form of perfect shuffle pattern. Accordingly, the lower bound on the computational speed and complexity of the problems requiring concurrent processing is proportional to ($\lceil \log_2 N \rceil$). In this chapter an alternative 4-shuffle pattern has been examined for use in parallel processing to interconnect the stages of MINs. 4-shuffle has been defined from different angles and an analytical model based upon cube configuration has been developed. The potential for 4-shuffle has been demonstrated by considering examples, and its general applicability for MINs has been considered at length in succeeding chapters. Specific applications considered in this chapter are the evaluation of Fast Fourier Transform, Polynomial Evaluation and Matrix Transposition. These examples exhibit inherent parallelism property. In the development of analytical model for the 4-shuffle pattern quaternary number system has been employed. This number system will also be used for the description of MINs in subsequent chapters. Therefore, before undertaking the description of 4-shuffle it is appropriate to study the quaternary number system briefly.

3.1 QUATERNARY NUMBER SYSTEM AND ITS SET THEORETIC INTERPRETATION

Let i be an integer (element) in the vector V of length $N = 4^n$, where n is any positive integer, and

$$Q = \{0,1,2,3\}$$

then q_j represents any quaternary digit such that

$$q_j \in Q \text{ with } \# \; q_j = 1$$

Now i can be written as

$$i = q_{n-1}4^{n-1}+q_{n-2}4^{n-2}+\ldots+q_j4^j+\ldots+q_o, \quad 0 \leq j \leq (n-1) \qquad ..(3.1)$$

further, let $q_j^k \in Q$, $\forall\, 0 \leq k \leq 3$, such that $\# \; q_j^k = 1$ and

$$q_j^0 = \{0\}$$
$$q_j^1 = \{1\}$$
$$q_j^2 = \{2\}$$
$$q_j^3 = \{3\}$$

again, if $q_j^{k'} \in Q$, such that $\# \; q_j^{k'} = 3$, $\forall\, 0 \leq k' \leq 3$, i.e., $q_j^{k'} = Q \setminus q_j^k$, giving

$$q_j^{0'} = \{1,2,3\}$$
$$q_j^{1'} = \{0,2,3\}$$
$$q_j^{2'} = \{0,1,3\}$$
$$q_j^{3'} = \{0,1,2\}$$

Then $q_j^{k'}$ is relative complement of q_j^k with respect to Q.

In this thesis, $\bar{q}_j$ is designated as the relative complement of q_j which can be assigned three quaternary digits other than the one assigned to q_j. For example if $q_j = 1$, then its relative complement $\bar{q}_j$ will be assigned values $\{0,2,3\}$.

The relationship derived above has been used in this and subsequent chapters.

3.2 DEFINITION OF 4-SHUFFLE

Let V_1 be a vector of length N, where N is any positive integer. A 4-shuffle of N elements of the vector is obtained in two phases: (i) In the first phase, divide the N elements of the vector V_1 into 4 piles of elements each, such that top pile consists of first N/4 elements, the next pile consists of subsequent N/4 elements and so on. (ii) In second phase pick the i^{th} element from the i^{th} pile to constitute a new pile of N/4 elements. In this way, 4 new piles with different elements will be created. This new order of the elements represents the 4-shuffle permutation of the previous order of the vector V [104]. Fig.3.1 illustrates such a shuffle for N = 16.

Mathematically, the indices of a vector V_1, can be transformed to another vector V_2 with a new order through a permutation P such that

$$\left.\begin{array}{ll} P(i) = 4i \text{ Mod } (N-1) & 0 \leq i < (N-1) \\ P(i) = i & i = (N-1) \end{array}\right| \qquad ..(3.2)$$

The above equation gives the analytical interpretation of 4-shuffle permutation. 4-shuffle can also be seen from

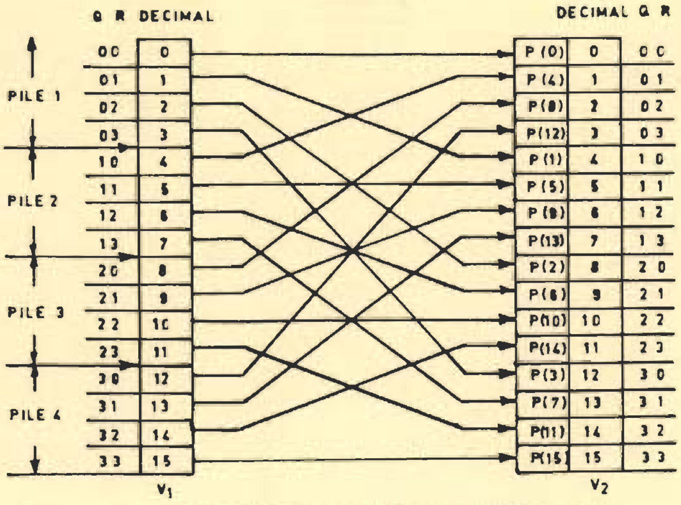


FIGURE 3.1 SHUFFLE PERMUTER
Q R→Quaternary Representation

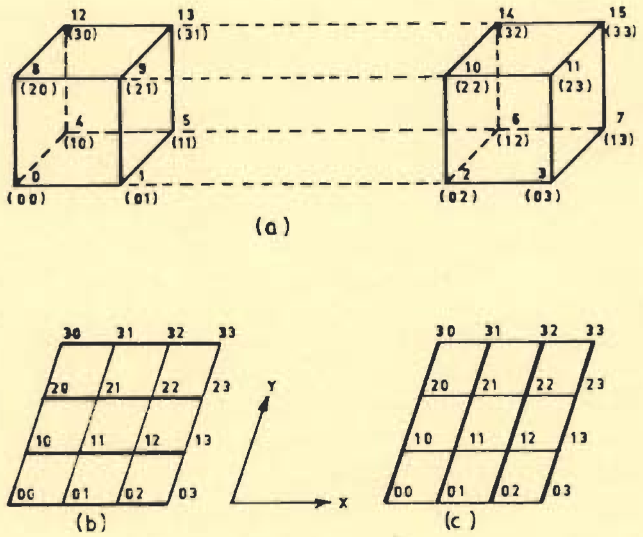


FIGURE 3.2 DUAL CUBE REPRESENTATION OF 4-SHUFFLE
(a) Dual cube with elements 0 to 15 placed on the vertices
(b) and (c) Back-spread projection of dual cube of (a). Elements which differ in single digit position are on the same line

another angle by representing the indices of the elements of the vector in quaternary number system.

Any element 'i' of the vector V_1 of length N, $N = 4^n$ for some positive integer n, can be represented by Equation (3.1) as reproduced below

$$i = q_{n-1}4^{n-1} + q_{n-2}4^{n-2} + \ldots + q_j 4^j + \ldots + q_0 \quad ..(3.1)$$

The 4-shuffle of i can be represented as

$$P(i) = q_{n-2}4^{n-1} + q_{n-3}4^{n-2} + \ldots + q_j 4^{j+1} + \ldots + q_0 4^1 + q_{n-1} \quad ..(3.3)$$

Where each q_j in (3.3) has been obtained by cyclically rotating the digits in the quaternary representation of 'i' one digit to the left.

Theorem 3.1

4-shuffle permutation represented by Equation (3.1) and (3.3) together is identical to Equation (3.2).

Proof

Equation (3.2) can be expanded as

$$\left.\begin{aligned} P(i) &= 4i & 0 &\leq i \leq (\tfrac{N}{4} - 1) \\ &= 4i + 1 - N & \tfrac{N}{4} &\leq i \leq (\tfrac{N}{2} - 1) \\ &= 4i + 2 - 2N & \tfrac{N}{2} &\leq i \leq (\tfrac{3N}{4} - 1) \\ &= 4i + 3 - 3N & \tfrac{3N}{4} &\leq i \leq (N-1) \end{aligned}\right| \quad ..(3.4)$$

In Equations (3.1) and (3.3) q_j being a quaternary digit, can have any value 0, 1, 2 or 3. Now one can calculate the value of P(i) in (3.3) in terms of 'i' of (3.2) for four possible values of q_{n-1}.

<u>Case 1</u> If $0 \leq i \leq (\frac{N}{4} - 1)$, then $q_{n-1} = 0$

from (3.2) and (3.3) if $q_{n-1} = 0$

$$P(i) = 4i \qquad ..(3.5a)$$

<u>Case 2</u> If $\frac{N}{4} \leq i \leq (\frac{N}{2} - 1)$, then $q_{n-1} = 1$

$$\text{giving } P(i) = 4i + 1 - 4^n = 4i + 1 - N \qquad ..(3.5b)$$

<u>Case 3</u> If $\frac{N}{2} \leq i \leq (\frac{3N}{4} - 1)$, then $q_{n-1} = 2$

$$\text{giving } P(i) = 4i + 2 - 2.4^n = 4i + 2 - 2N \qquad ..(3.5c)$$

<u>Case 4</u> If $\frac{3N}{4} \leq i \leq (N-1)$, then $q_{n-1} = 3$

$$\text{giving } P(i) = 4i + 3 - 3.4^n = 4i + 3 - 3N \qquad ..(3.5d)$$

Equations (3.5a-d) put together are equivalent to Equation (3.4).

$\square$

Theorem 3.2

All elements of a vector V of length $N = 4^n$ will be shuffled to their original starting positions simultaneously, after performing 4-shuffle n times on the elements of the vector.

Proof

From Equations (3.2) and (3.3) it is observed that an element 'i' has n digits in its quaternary representation. Applying the cyclic shift rule on the digits of i, each time a 4-shuffle permutation is performed one digit will be shifted to the left in circular fashion. It becomes obvious that the digits will return to their original position after performing n permutations. Hence the proof.

$\square$

3.3 CUBE REPRESENTATION OF 4-SHUFFLE PERMUTATION

In this section mathematical basis and properties of 4-shuffle have been developed and postulated. These have been achieved by extending the cube model [135] and orienting it to incorporate the 4-shuffle permutation.

For this purpose a DUAL CUBE (Dcube) consisting of two cubes, as shown in Fig.3.2(a), has been proposed as the basic building block. There are 16 vertices in the dual cube which are labelled as per the following rules:

Rule 1:

Each vertex within each cube is 1-Hamming distance apart from every adjacent vertex.

Rule 2:

The corresponding elements of the two cubes are 1-Hamming distance apart.

Considering a vector V_1 of length 16, the decimal equivalent of the elements of the vector and their quaternary representations are shown in Fig.3.2(a). Quaternary representation is simply obtained by considering base-4 number system. Using back-spread method of representing the cubes, the proposed dual cube can be projected as shown in Fig.3.2(b). The backspread dual cube has the following properties:

Property 1:

Each vertex has a two digit quaternary coordinate assigned to it. Traversal along any one dimension of the backspread representation will cause sequential variation of only one of the two digits while the other one will remain constant, i.e., horizontal (perpendicular) traversal will cause the second (first) coordinate to vary sequentially.

Property 2:

Each vertex in the backspread dual cube has 4 nearest neighbours, which are 1-quaternary distance apart from it.

Let 4 elements of a vector whose indices differ in their quaternary representation by one digit be put together to form a set of elements. Then the sets of elements varying in different positional digits will match on different coordinates of an n-dimensional backspread dual cube.

Theorem 3.3

If the sets of elements of a vector V_1, which differ in first digit of their quaternary representation, are matched in one dimension of the backspread dual cube, then the 4-shuffle will rotate these elements in the second dimension, matching the sets of elements, which differ in 2nd digit.

Proof

The sets of elements of vector V_1 differing in first digit position are matched on X-coordinate lines in Fig.3.2(b) (shown with dark lines). To obtain 4-shuffle of these elements, quaternary addresses are cyclically rotated one digit position to the left. The new addresses so obtained, are the sets of elements which are matched on Y-coordinate lines in Fig.3.2(c) (shown with dark lines). This can also be verified from Fig.3.1 where vector V_2 is the 4-shuffle of vector V_1.

$\square$

3.4 4-SHUFFLE PERMUTER (P)

The interconnection topology of N links in a network can be conveniently described by the permutation of its terminals. An NxN permuter 'σ' is defined as a network consisting of N fixed links connecting two sets of N terminals [117]. The connections realized by the permuter σ can be represented by a permutation of N elements.

Thus

$$\sigma = \begin{pmatrix} E_0 & E_1 \ldots\ldots E_i \ldots\ldots E_{N-1} \\ \sigma(E_0) & \sigma(E_1) \ldots\ldots \sigma(E_i) \ldots\ldots \sigma(E_{N-1}) \end{pmatrix}$$

Where E_i is connected to $\sigma(E_i)$ for $i = 0,1,2,\ldots,N-1$. Fig.3.3(a) depicts a general NxN permuter. In the specific case when the connections are realized using 4-shuffle permutation, the permuter is denoted as P. An example of permuter P for N = 16 is depicted in Fig.3.3(b). For the 4-shuffle Permuter P, σ implies left

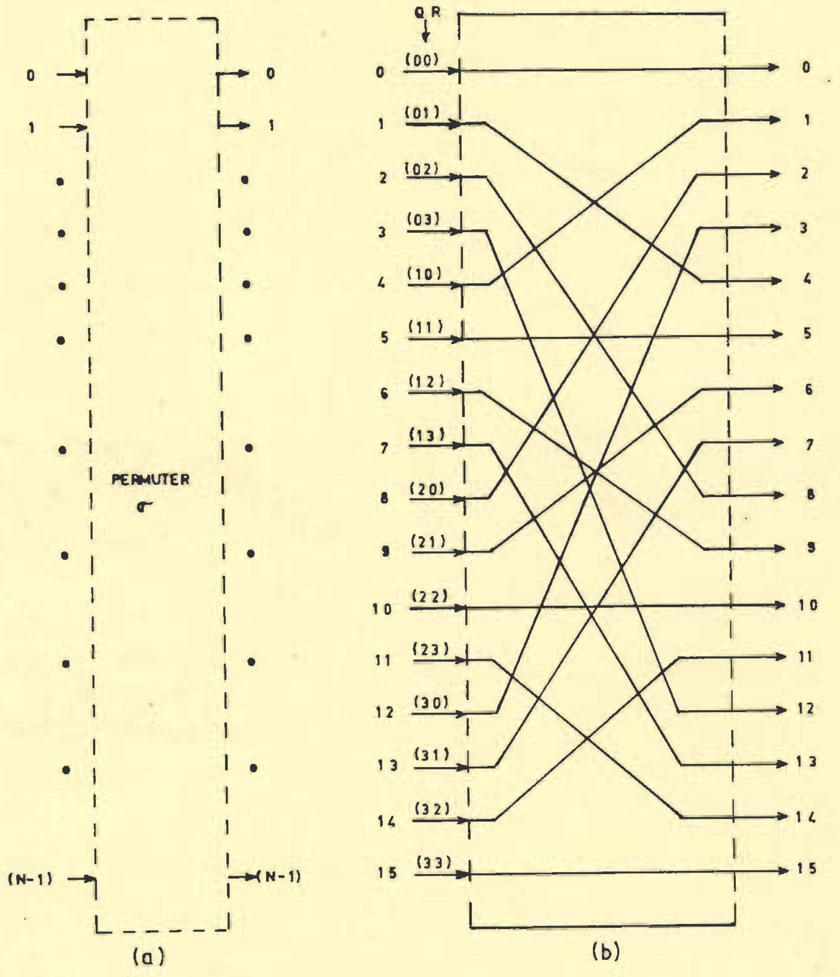


FIGURE 3.3 (a) A general N X N permuter σ
(b) The 16 X 16 4-shuffle permuter P

cyclic shift of quaternary digits.

An NxN permuter can be considered to be a passive NxN network and can be used as a subnetwork in the construction of MINs. The concept of permuters developed here has been extensively used in Chapter IV in developing single stage and multiple stage dual cube networks.

3.5 APPLICATIONS OF 4-SHUFFLE PERMUTATION IN PARALLEL PROCESSING

4-shuffle has direct application in the evaluation of various mathematical problems with built in parallelism. In this section the use of 4-shuffle as an alternative to perfect shuffle in three specific problems is demonstrated. It is also shown that the computational complexity, which is proportional to $\log_2 N$ with perfect shuffle, reduces to be proportional to $\log_4 N$ with 4-shuffle.

3.5.1 The Fast Fourier Transform (FFT)

One of the computational problems which has inherent parallelism is the evaluation of FFT [37,105]. The use of perfect shuffle interconnection pattern for executing the transform algorithm on parallel processor has been demonstrated in [135]. For N samples of a time function where $N = 2^m$, m being any positive integer, the processing repeats the sequence of the following steps 'm' times to compute the fast Fourier transform [135].

(i) Perfect Shuffling

(ii) Multiply - Add

(iii) Transfer the results back to the input of shuffle network.

This is demonstrated in Fig.3.4 for N = 16, m = 4. The network in Fig.3.4 first combines pairs of numbers whose indices differ by $2^3 = 8$ in their binary expansion. After second shuffle, the pairs combined will differ by $2^2 = 4$, after third shuffle pairs will differ by $2^1 = 2$ and finally by $2^0 = 1$.

It will now be shown, that the problem can be reconfigured for evaluation of FFT using 4-shuffle permutation and it will be required to repeat the sequence of above three steps only n times, where $N = 4^n$.

Let A(k), k = 0,1,....,N-1, be N samples of a time function, sampled at instants that are spaced equally apart. It is assumed that $N = 4^n$ where n is any positive integer. The discrete Fourier transform of A(k) is defined to be the discrete function X(j), j = 0,1,....,N-1 where

$$X(j) = \sum_{k=0}^{N-1} A(k)\ W^{jk} \qquad ..(3.6)$$

where $W = e^{2\pi i/N}$

The fast Fourier transform algorithm is derived directly by forming the quaternary expansions of the indices j and k as follows:

$$j = j_{n-1}.4^{n-1} + \ldots\ldots. + j_1 4 + j_o \qquad ..(3.7)$$

and

$$k = k_{n-1}.4^{n-1} + \ldots\ldots. + k_1 4 + k_o \qquad ..(3.8)$$

where j_i and $k_i \in \{0,1,2,3\}$ and $\# \ j_i, k_i = 1$.

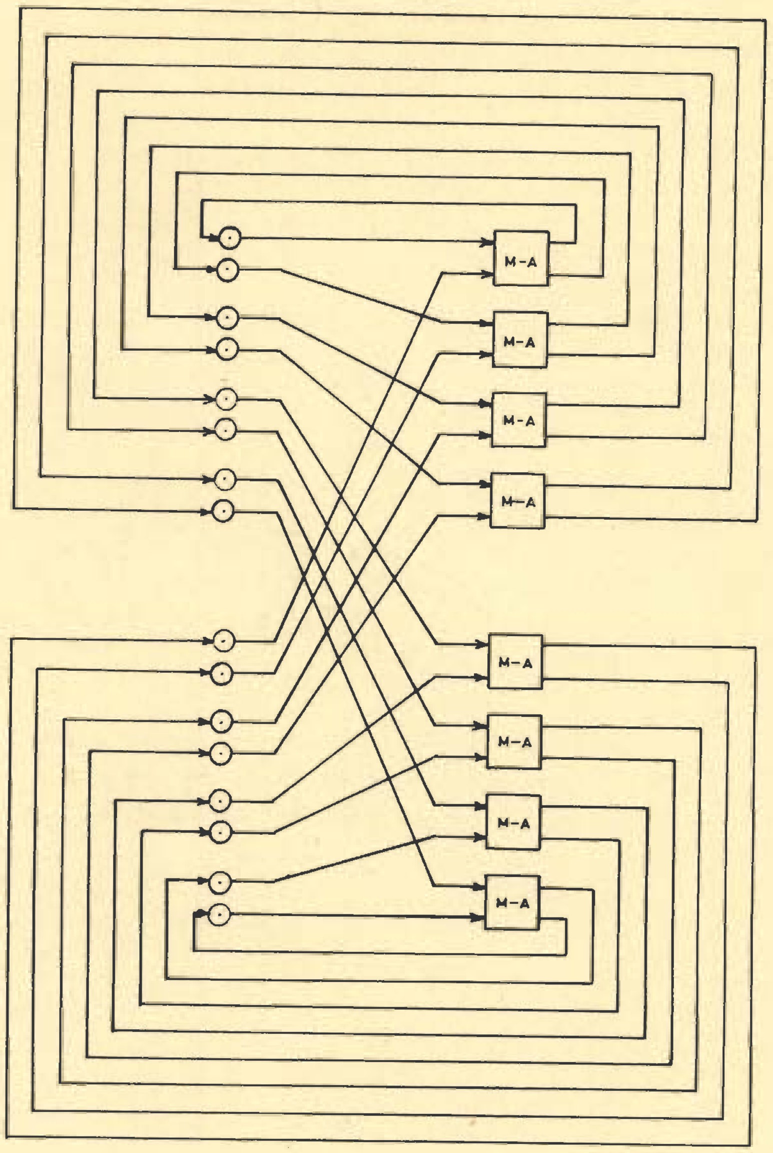


FIGURE 3.4 A PROCESSOR FOR FFT. THE 'M-A' MODULES PRODUCE WEIGHTED SUMS OF THEIR INPUTS ON THEIR OUTPUTS

Substituting the identities (3.7) and (3.8) into (3.4) one gets

$$X[j_{n-1}\cdots j_0] = \sum_{k_0} \sum_{k_1} \cdots \sum_{k_{n-1}} A(k_{n-1}\cdots k_0) W^{jk} \quad ..(3.9)$$

where each of the indices k_i are summed over the quaternary values 0,1,2, and 3.

To compute the Fourier transform of A(k), n different arrays, B_1, $B_2 \cdots B_n$ are formed where each B_i is computed from B_{i-1} for $i > 1$. The last array of this sequence, B_n has elements that are the values of X(j), but the elements are scrambled in what is known as reverse quaternary order. The arrays are defined as follows:

$$B_1(j_0, k_{n-2}, \cdots k_0) = \sum_{k_{n-1}} A(k_{n-1}\cdots k_0) W^{j_0 k_{n-1} 4^{n-1}} \quad ..(3.10)$$

$$B_s(j_0, \cdots j_{s-1}, k_{n-s-1}, \cdots k_0) =$$

$$\sum_{k_{n-s}} B_{s-1}(j_0, \cdots j_{s-2}, k_{n-s} \cdots k_0) W^{(j_{s-1}4^{s-1} + \cdots j_0) k_{n-s} 4^{n-s}} \quad ..(3.11)$$

for s = 2,3,.....n

The relation

$$W^{jk_{n-s}} = W^{(j_{s-1}4^{s-1} + \cdots j_0) k_{n-s} 4^n}$$

may be used in the summation (3.9), giving

$$B_n(j_0, j_1, \cdots j_{n-1}) = \sum_{k_0} \sum_{k_1} \cdots \sum_{k_{n-1}} A(k_{n-1}, \cdots k_s) W^{jk}$$

$$= X(j_{n-1}, j_{n-2}, \cdots j_0) \quad ..(3.12)$$

To obtain the value of X(j) when given the vector B_n, one reverses the quaternary digits in the expression of j to obtain a new index which may be called j'. Then $X(j) = B_n(j')$.

The role of 4-shuffle becomes evident when Equation (3.6) is inspected. Each element of B_i is the weighted sum of four elements of B_{i-1} (or of A). To determine which four elements of B_{i-1} are combined to form $B_i(j)$, one forms the quaternary expansion of j, and observes the coefficients of 4^{n-i}. Then the four elements of B_{i-1} that contribute to $B_i(j)$ are the elements $B_{i-1}(j)$ and $B_{i-1}(\bar{j})$ where $B_{i-1}(\bar{j})$ is the relative complement of $B_{i-1}(j)$. Figs. 3.2(b) and (c) show the setwise combinations that form the fast Fourier transform for N = 16.

A processor for FFT for N = 16 has been reconfigured in Fig.3.5, where the perfect shuffle connections of Fig.3.4 are replaced by 4-shuffle connections and 'M-A' module is capable of computing 4 weighted sums of its 4-inputs simultaneously with the results appearing on the 4-outputs.

From Theorem 3.3, it is clear that the network in Fig.3.5 first matches the sets of elements whose indices differ by $4^1 = 4$ and in second iteration the sets of elements whose indices differ by $4^0 = 1$. Accordingly, the configuration of Fig.3.5 requires two iterations, i.e., the sequence of steps are repeated only twice as compared to 4 times in configuration of Fig.3.4.

Thus, it is established that the use of 4-shuffle connection in the evaluation of FFT reduces the computational complexity to be proportional to $\log_4 N$.

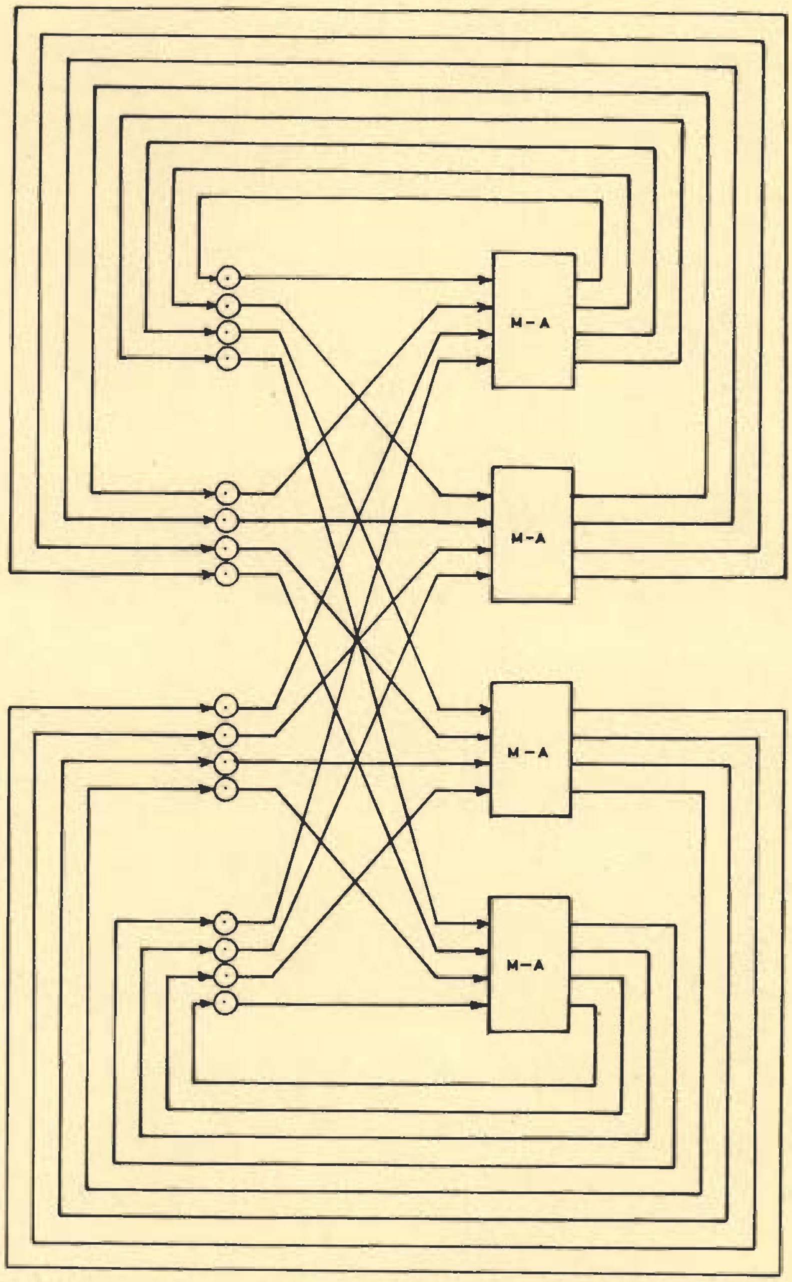


FIGURE 3.5 A PROCESSOR FOR F F T. THE "M-A" MODULES PRODUCE WEIGHTED SUMS OF THEIR INPUTS ON THEIR OUTPUTS

3.5.2 Polynomial Evaluation

In this section the problem of polynomial evaluation has been taken up to demonstrate the usefulness of 4-shuffle in parallel processing. It has been demonstrated that the algorithm to evaluate polynomial of degree N can be executed on parallel processor with either perfect shuffle interconnection pattern or 4-shuffle interconnection pattern. However, the use of 4-shuffle instead of perfect-shuffle, increases the computational speed and reduces the computational complexity. The problem to be solved is as follows:

Let $a_0, a_1 \ldots\ldots a_{n-2}$ be the coefficients of a polynomial of degree N-2. It is desired to compute $\sum_{i=0}^{N-2} a_i X^i$ on a parallel processor that can perform upto N operations simultaneously.

The solution to this problem is now discussed. The coefficients a_i, $i = 0,1,\ldots,N-2$ are arranged in consecutive locations in a block of memory and it is assumed that at each location in the block, a computation takes place. For this example the computation is multiplication. The processor operates by broadcasting a single number to all locations in the block. The numerical value of X is stored in position N-1 of the coefficient vector, the last register in the block of memory. An index register is initialized to the vector V where $V = (0,1,0,1,0,1,\ldots\ldots,0,1)$ for perfect shuffle connections and $V = (0,1,2,3,0,1,2,3,\ldots\ldots,0,1,2,3)$ for 4-shuffle connections.

At the end of each iteration the contents of the memory register are changed according to the following equation:

$$\text{New Contents} \longleftarrow (\text{Current Contents})[\text{Contents of location}(N-1)]^{V_i} \quad ..(3.13)$$

where V_i is the corresponding digit in the index register.

The evaluation of polynomial of degree 14(N = 16) is presented schematically in Fig.3.6 using perfect shuffle interconnection pattern. It is observed that it requires 4 iterations of multiplications before performing the sum of the first fifteen locations in the block to get the result

$$\sum_{i=0}^{14} a_i X^i \qquad ..(3.14)$$

The same problem has been reconfigured in Fig.3.7 with 4-shuffle interconnections. In this case only two iterations of multiplications are required before the sum is performed on the first fifteen locations in the block to get the sum of the polynomial

$$\sum_{i=0}^{14} a_i X^i \qquad ..(3.14)$$

Again it is demonstrated that the use of 4-shuffle permutation to evaluate the polynomial reduces the lower bound on the minimum time to ($\lceil \log_4 N \rceil$) from the conventional value of ($\lceil \log_2 N \rceil$) with perfect shuffle.

3.5.3 Matrix Transposition

In parallel processing sometimes it is required to rearrange the data. Two dimensional matrix calculations are particularly susceptible to these requirements. In many problems, it is necessary to have parallel access to both rows and columns of a matrix. For matrix multiplication, it is necessary to align rows of one matrix with the columns of another in order to obtain efficiency. One solution to these

Data	Index	Data	Index	Data	Index	Data	Index	Data
a_0	0	a_0	0	a_0	0	a_0	0	a_0
a_1	1	a_1X	0	a_1X	0	a_1X	0	a_1X
a_2	0	a_2	1	a_2X^2	0	a_2X^2	0	a_2X^2
a_3	1	a_3X	1	a_3X^3	0	a_3X^3	0	a_3X^3
a_4	0	a_4	0	a_4	1	a_4X^4	0	a_4X^4
a_5	1	a_5X	0	a_5X	1	a_5X^5	0	a_5X^5
a_6	0	a_6	1	a_6X^2	1	a_6X^6	0	a_6X^6
a_7	1	a_7X	1	a_7X^3	1	a_7X^7	0	a_7X^7
a_8	0	a_8	0	a_8	0	a_8	1	a_8X^8
a_9	1	a_9X	0	a_9X	0	a_9X	1	a_9X^9
a_{10}	0	a_{10}	1	$a_{10}X^2$	0	$a_{10}X^2$	1	$a_{10}X^{10}$
a_{11}	1	$a_{11}X$	1	$a_{11}X^3$	0	$a_{11}X^3$	1	$a_{11}X^{11}$
a_{12}	0	a_{12}	0	a_{12}	1	$a_{12}X^4$	1	$a_{12}X^{12}$
a_{13}	1	$a_{13}X$	0	$a_{13}X$	1	$a_{13}X^5$	1	$a_{13}X^{13}$
a_{14}	0	a_{14}	1	$a_{14}X^2$	1	$a_{14}X^6$	1	$a_{14}X^{14}$
X	1	X^2	1	X^4	1	X^8	1	X^{16}
	Ist Iteration		2nd Iteration		3rd Iteration		4th Iteration	

FIGURE 3.6 THE EVALUATION OF A POLYNOMIAL OF DEGREE 14 USING PERFECT SHUFFLE INTERCONNECTIONS.

Data	Index	Data	Index	Data
a_0	0	a_0X^0	0	a_0X^0
a_1	1	a_1X^1	0	a_1X^1
a_2	2	a_2X^2	0	a_2X^2
a_3	3	a_3X^3	0	a_3X^3
a_4	0	a_4X^0	1	a_4X^4
a_5	1	a_5X^1	1	a_5X^5
a_6	2	a_6X^2	1	a_6X^6
a_7	3	a_7X^3	1	a_7X^7
a_8	0	a_8X^0	2	a_8X^8
a_9	1	a_9X^1	2	a_9X^9
a_{10}	2	$a_{10}X^2$	2	$a_{10}X^{10}$
a_{11}	3	$a_{11}X^3$	2	$a_{11}X^{11}$
a_{12}	0	$a_{12}X^0$	3	$a_{12}X^{12}$
a_{13}	1	$a_{13}X^1$	3	$a_{13}X^{13}$
a_{14}	2	$a_{14}X^2$	3	$a_{14}X^{14}$
X	3	X^4	3	X^{12}
	1st Iteration		2nd Iteration	

FIGURE 3.7 THE EVALUATION OF A POLYNOMIAL OF DEGREE 14 USING 4-SHUFFLE INTERCONNECTIONS.

problems is to build an efficient mechanism for obtaining matrix transpose.

The transpose of an NxN matrix X can be obtained by performing $\log_2 N$ perfect shuffles of X [135]. However, the problem can be reconfigured for 4-shuffle interconnections instead of perfect shuffle and the transpose can be obtained by performing only $\log_4 N$ times 4-shuffle. In the following discussion first general concepts for obtaining transpose are explained and then a specific case of 4x4 matrix is used in both the configurations, i.e., perfect shuffle and 4-shuffle for the sake of comparison.

Let, there be a $(4^n x 4^n)$ matrix X, whose elements are stored in row major order in a computer memory, i.e., the elements of X are stored in lexicographic order by index with row index as major key and column index as minor key as shown in Fig.3.8. By inspection, it is found that the X[i,j] is displaced from X[1,1] the base of the array, by an amount given by

$$\text{displacement} = 4^n(i-1) + (j-1) \qquad ..(3.15)$$

The matrix transpose of X is obtained by interchanging i and j in the formula. To obtain the transpose of X, the X is stored in column major order, i.e., the elements of X are arranged lexicographically by index with column index as major key and row index as minor key. In this storage arrangement the displacements of X[i,j] relative to X[1,1] is given by

$$\text{displacement} = 4^n(j-1) + (i-1) \qquad ..(3.16)$$

Normal (Row Major)	Transposed (Column Major)
X[1,1]	X[1,1]
X[1,2]	X[2,1]
X[1,3]	$\vdots$
$\vdots$	
X[1,4^n]	X[4^n,1]
X[2,1]	X[1,2]
X[2,2]	X[2,2]
$\vdots$	$\vdots$
X[2,4^n]	X[4^n,2]
$\vdots$	$\vdots$
X[4^n,1]	X[1,4^n]
X[4^n,2]	X[2,4^n]
$\vdots$	$\vdots$
X[4^n,4^n]	X[4^n,4^n]

FIGURE 3.8 THE STORAGE ARRANGEMENT FOR A MATRIX IN NORMAL ORDERING AND IN TRANSPOSE ORDERING

Now the definition of 4-shuffle given in Equations (3.2) and (3.3) can be used to show that the matrix transpose of X can be obtained by performing $n = \log_4 N$ times the 4-shuffle of X.

Let there be a shift register having 2n storage locations. The high order n stages of the shift register hold the quaternary representation of (i-1) in Equation (3.15) and the low order n-stages hold the quaternary representation of (j-1). The shift register thus holds the representation of the right hand side of Equation (3.15), i.e., of $4^n(i-1) + (j-1)$ which is the displacement of X[i,j] relative to X[1,1]. This is shown in Fig.3.9(a). Now, from Equations (3.2) and (3.3), the 4-shuffle of the indices of a vector can be obtained by cyclically rotating the quaternary digits of the element, one digit position to the left. After performing n shuffles on the digits stored in shift register of Fig.3.9(a), the data will occupy the new position in the shift register as shown in Fig.3.9(b). This is nothing else but the representation of Equation (3.16).

Thus, it is established that the element that is displaced from X[1,1] by $4^n(i-1) + (j-1)$ before the shuffle is displaced from X[1,1] by $4^n(j-1) + (i-1)$ after performing n shuffles.

In Fig.3.10 the transpose of 4x4 matrix is obtained using perfect shuffle permutation. It can be seen that the transpose is achieved by performing perfect shuffle twice, i.e., ($\log_2 N$) times. The same problem is reconfigured in Fig.3.11 with 4-shuffle permuter. In this case matrix transpose is achieved by performing 4-shuffle only once, i.e., ($\log_4 N$) times.

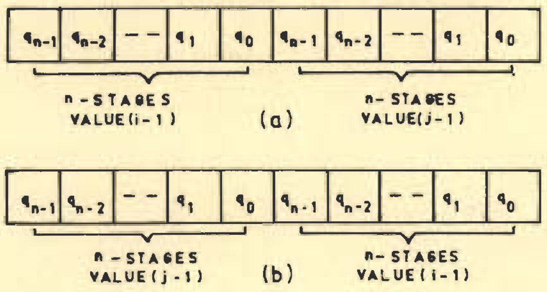


FIGURE 3.9 A SHIFT REGISTER ANALOGY THAT n SEQUENCES OF FOUR-SHUFFLE TRANSPOSE A 4^n X 4^n MATRIX

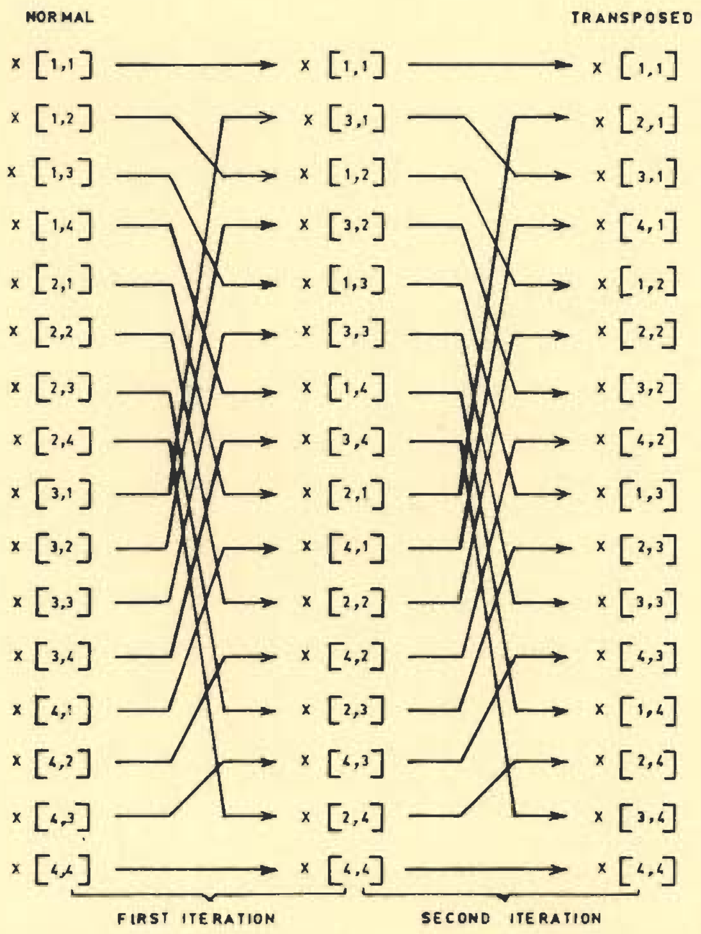


FIGURE 3.10 THE TRANSPOSE OF 4 X 4 MATRIX OBTAINED BY PERFORMING PERFECT SHUFFLE TWICE

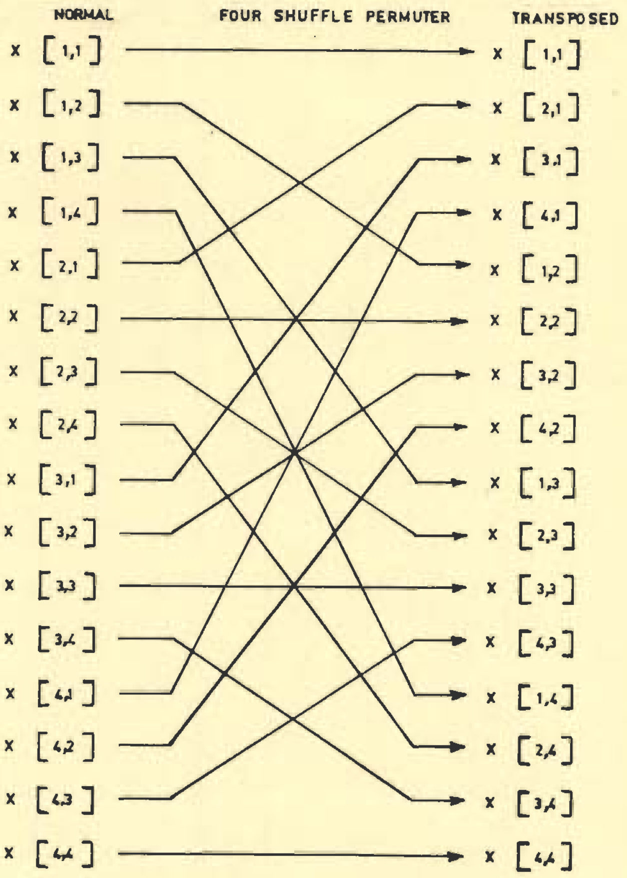


FIGURE 3.11 TRANSPOSE OF A 4 X 4 MATRIX IS OBTAINED BY FOUR-SHUFFLE PERMUTER. AS $n = 1$, ONLY ONCE FOUR-SHUFFLE IS PERFORMED TO OBTAIN TRANSPOSE

If X is any square matrix of size less than $4^n x 4^n$, then it can be transposed by storing the elements in an upper left submatrix of a $4^n x 4^n$ square matrix. Rectangular matrices cannot be transposed by this technique.

3.6 CONCLUSION

In this chapter 4-shuffle permutation has been proposed as an alternative to perfect shuffle scheme for use in parallel processing. This reduces the computational complexity and time for algorithms requiring concurrent processing. From a general point of view, some of the algorithms may match elements whose indices differ in single positional digit of their quaternary representation. This is equivalent to matching the sets of nodes on the lines in one coordinate of backspread dual cube. For these algorithms the processor can be constructed to match single dimension of the backspread dual cube. Then the 4-shuffle can be used to 'rotate' other dimension into computational position. Fig.3.5 suggests the possibility of using 4-shuffle with 4x4 SEs in ICNs. This concept has been exploited in greater detail in Chapter IV for developing the dual cube ICN. Figs.3.2 (a) and (b) give the two dimensional concept of dual cube topology. Its extension to other dimensions has also been discussed in Chapter IV while developing the topology of dual cube multistage interconnection networks.

An appropriate point of view is that the 4-shuffle interconnection scheme deserves to be exploited for implementation in parallel processing. Whether this interconnection pattern should be used instead of, or in addition to, the existing schemes depends very much on the size and intended application of the parallel processing environment.

CHAPTER IV

DESIGN OF DUAL CUBE MULTISTAGE INTERCONNECTION NETWORK

In a parallel processing system the speed of computation can be enhanced by reducing the switching time and propagation delays in intercommunication networks. Invariably, ICNs are used between processors and/or memories in a parallel system. Because of the advances in contemporary IC technology, the component density of integrated circuits continues to increase. As the switching delay decreases with scaling, the speed at which an ICN can operate is dominated by interconnection delays [92,136]. In Chapter I it was suggested that the propagation delay in multistage interconnection networks can be reduced by reducing the number of stages. The reduction in number of stages and modularization of switching elements makes the MIN inherently more reliable. In this chapter an effort has been made to design a MIN with reduced number of stages compared to the conventional interconnection networks.

A VLSI implementation of four I/C boxes of 2x2 switching elements on a single chip for the cube network has been proposed in [125]. It was also envisaged that instead of conventional 2x2 SE, a 4x4 switch will be a better candidate for an NxN MIN [23]. A modular alignment network based on modular implementation of multistage cube interconnection network suitable for VLSI environment has been proposed in [100,101]. However, this module has been used only as a redundant switch for fault-tolerance.

In Chapter III, the use of 4-shuffle with 4x4 switching elements was suggested. It has been established in Chapter III that the use of 4-shuffle in parallel processing reduces the number of iterations to $\log_4 N$. The number of iterations are linked with number of passes in an ICN or number of stages in a multistage interconnection network.

In this chapter a 4x4 switching element called the Dual Interconnection Modular Switching Element (DIMSE) has been proposed as an alternative to conventional 2x2 SEs. Its realization and input/output mappings for various control combinations are discussed. A new multistage interconnection network topology using 4-shuffle link pattern to interconnect the stages of DIMSEs has been designed and developed. The proposed MIN topology is designated as Dual Cube Multistage Interconnection Network (DCMIN). It has been further shown that the conventional routing algorithms used for MINs, are applicable to DCMIN as well. The advantages of DCMIN in terms of performance and complexity as compared to conventional cube network topology have been spelled out in detail.

The proposed DCMIN is basically a blocking network which does not allow all possible permutations. A conflict arises, when two or more processors need the same link between two successive stages in reaching their destinations. However, DCMIN has full-access property which guarantees connectivity between all the input and output terminals.

4.1 DUAL INTERCONNECTION MODULAR SWITCHING ELEMENT (DIMSE)

In this section a 4x4 modular switching element is proposed. Fig.4.1 gives the black box representation of such a switch, where I indicates the input to the box and R represents the output response. The subscripted I or R indicates the input/output terminal number in question. The switch has two control lines C_1 and C_2. The combinations of control signals are used to obtain 4 different input/output mappings of the switch.

Figs.4.2(a-d) give these input/output mappings for different control settings. These are also summarised in Table 4.1.

Control	Inputs	Input/Output mapping	Mode of operation	General form of the I/O mapping
C_2	C_1			for $0 \leq k \leq 3$
0	0	Fig.4.2(a)	0	$R_k = I_k$
1	0	Fig.4.2(b)	1	$R_k = I_{(k+2)\bmod 4}$
0	1	Fig.4.2(c)	2	$R_k = I_{(k\underline{+}1)}$ With '+' sign when I_k is even and '−' sign when I_k is odd
1	1	Fig.4.2(d)	3	$R_k = I_{(3-k)}$

TABLE 4.1 INPUT/OUTPUT MAPPINGS OF DIMSE FOR DIFFERENT CONTROL SETTINGS.

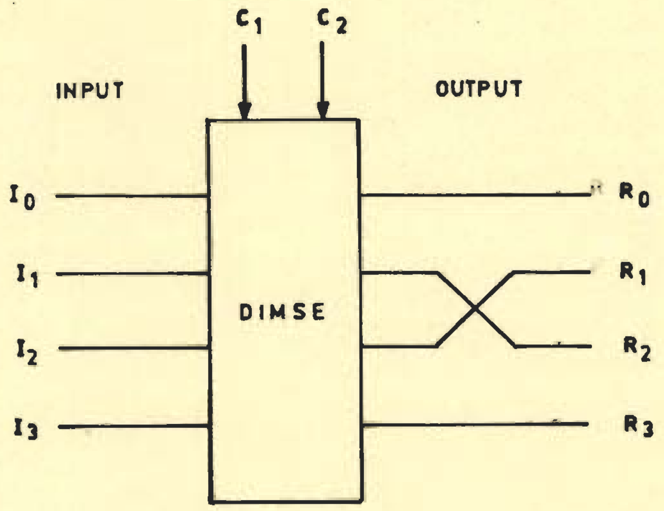
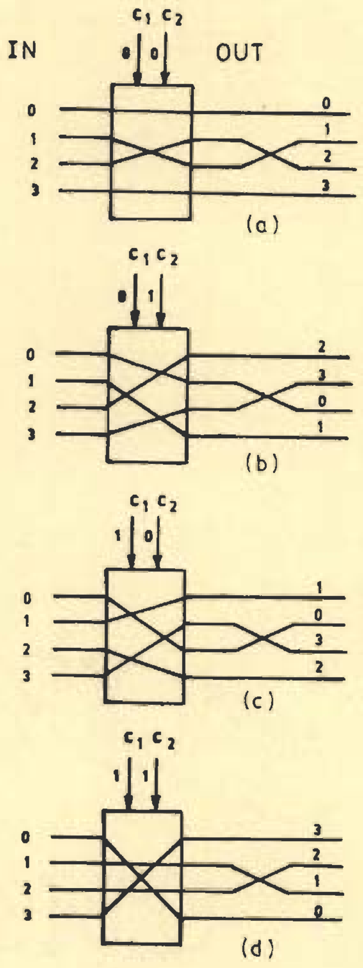


FIGURE 4.1 SYMBOL OF DUAL INTERCONNECTION MODULAR SWITCHING ELEMENT



MAPPING MATRIX CONTROL MODE

$$\begin{array}{cc} \text{IN} \\ \text{OUT} \end{array} \begin{pmatrix} 0 & 1 & 2 & 3 \\ 0 & 1 & 2 & 3 \end{pmatrix}_{00} \quad 0$$

$$\begin{array}{cc} \text{IN} \\ \text{OUT} \end{array} \begin{pmatrix} 0 & 1 & 2 & 3 \\ 2 & 3 & 0 & 1 \end{pmatrix}_{01} \quad 1$$

$$\begin{array}{cc} \text{IN} \\ \text{OUT} \end{array} \begin{pmatrix} 0 & 1 & 2 & 3 \\ 1 & 0 & 3 & 2 \end{pmatrix}_{10} \quad 2$$

$$\begin{array}{cc} \text{IN} \\ \text{OUT} \end{array} \begin{pmatrix} 0 & 1 & 2 & 3 \\ 3 & 2 & 1 & 0 \end{pmatrix}_{11} \quad 3$$

FIGURE 4.2 4 X 4 MODULAR SWITCHING ELEMENT WITH FOUR DIFFERENT CONTROL SETTINGS.

When C_1 and C_2 control lines are connected together, i.e., if the same input is given to both the control terminals, the switch is transformed into two separate 2x2 switching elements as shown in Fig.4.3. Terminals (0,3) and (1,2) act as two separate switching elements which are straight connected when $C_1 = C_2 = C' = 0$ and are in exchange state when $C_1 = C_2 = C' = 1$. Accordingly, the name Dual Interconnection Modular Switching Element (DIMSE) has been proposed for this 4x4 modular switching element. However, the use of DIMSE as two units of 2x2 SEs is neither advisable nor advocated as this will increase the number of stages required, and thus the advantages of increase in computational speed and reliability will be masked as discussed subsequently in Section 4.7. It is, therefore, recommended not to use DIMSE as a 2(2x2 SE) unless special circumstances warrant its use for fault-tolerance or for some other purposes.

One possible gate-level realization of DIMSE is outlined in Fig.4.4. Because of the circuit symmetry and regular structure it can be readily represented in terms of multiplexers/ULMs etc.

The following are the features of DIMSE:

i. The circuits connecting gates are repetitive in nature.

ii. The conventional 2x2 switching element has two levels of gates whereas DIMSE has three levels. To achieve 4x4 connections using conventional 2x2 SEs one will require 4 levels of gates. However, due to the increased component density of integrated circuits the individual gate delay becomes an insignificant fraction of

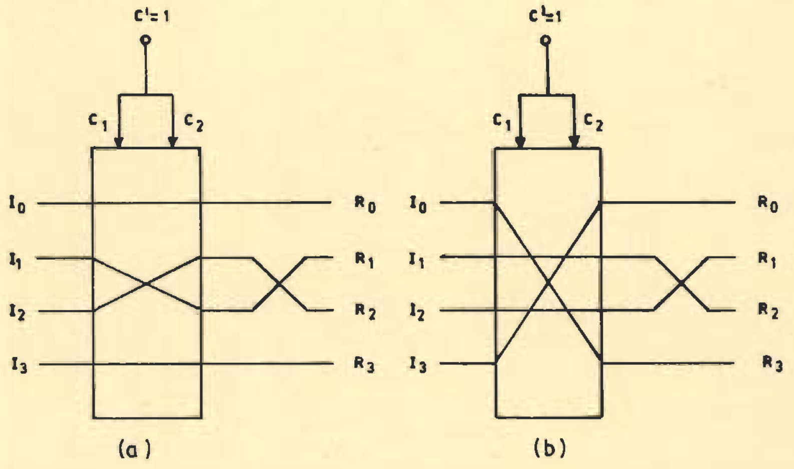


FIGURE 4.3 DIMSE IS DIVIDED INTO 2 UNITS OF 2 X 2 SWITCHES
(a) Terminals (0 ,3) and (1 ,2) straight connected
(b) Terminals (0 , 3) and (1 ,2) cross connected

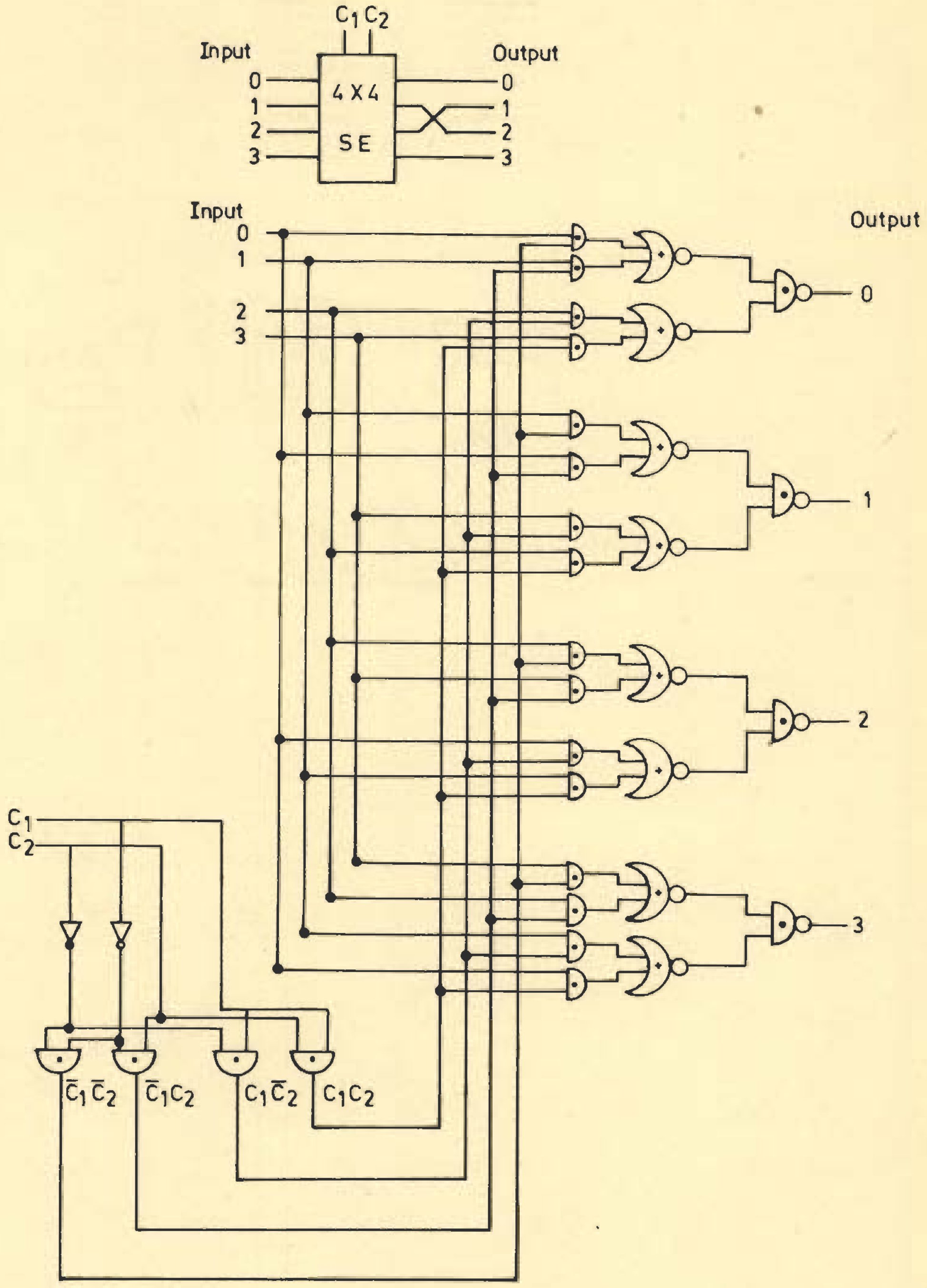


FIGURE 4.4 LOGIC CIRCUIT FOR 4 X 4 SWITCH.

the total module-switching delay. Thus, in general module-switching delay is considered as a single parameter to model the delay of the switching element.

iii. Any arbitrary input can be connected to any arbitrary output by changing the control signal combinations of C_1C_2, i.e., by changing the control signals to appropriate mode.

iv. The use of DIMSEs in the conventional interconnection networks will reduce the total number of modules required. Also, as envisaged in Chapter III, incorporation of 4-shuffle permutation with DIMSEs will also reduce the number of passes (stages) and external links for network of any size N.

Thus, DIMSE is a potential alternative to replace 2x2 SEs from the existing interconnection networks with improved reliability and reduced complexity.

4.2 DEVELOPMENT OF SINGLE STAGE NETWORK USING DIMSEs

An NxN, n-stage interconnection network is the network containing n stages of switching elements where each stage constitutes an NxN stack of switching elements. A 1-stage network is commonly called a single stage network. A general review of literature on single stage networks has been presented in Section 2.2.3. Single stage networks are recirculating networks, i.e., for establishing a desired mapping it requires recirculation(s) through the stage.

In this section the ICN has been developed using DIMSEs as proposed in preceding section. The single stage network has been developed as an NxN interconnection network using a stack of N/4 DIMSEs and a permuter σ (Section 3.4). Fig.4.5 represents two possible alternatives for NxN single stage network. In Fig. 4.5(a) the stack of DIMSEs is followed by a permuter, whereas, in Fig.4.5(b) permuter precedes the DIMSE stack. Since the permuter and the DIMSE stack are in tandem, the overall connection pattern between the inputs (sources) and the outputs (destinations) is identical. Such an ICN is a single stage DIMSE network (S_N). For a 4-shuffle permuter as defined in Section 3.4, the 1-stage network is represented in Fig.4.6.

The interconnection scheme between inputs and outputs as developed above consists of 4-shuffle function 'P' (of permuter) and exchange function 'E' (of DIMSE). Together these are termed as Shuffle-Exchange Function,'F'. F basically represents mapping between the inputs (I) and outputs (R) such as

$$R = F(I) \qquad ..(4.1)$$

F could be derived either as a shuffle-exchange function or as an exchange-shuffle function. In both the cases mapping will hold good. A more appropriate representation of F would be in terms of individual shuffle and exchange functions as

$$R = F(P(I)) \quad \text{or} \qquad ..(4.2)$$

$$R = P(E(I)) \qquad ..(4.3)$$

The 4-shuffle function has already been defined in Section 3.2 by Equations (3.2) and (3.3) and is reproduced below as

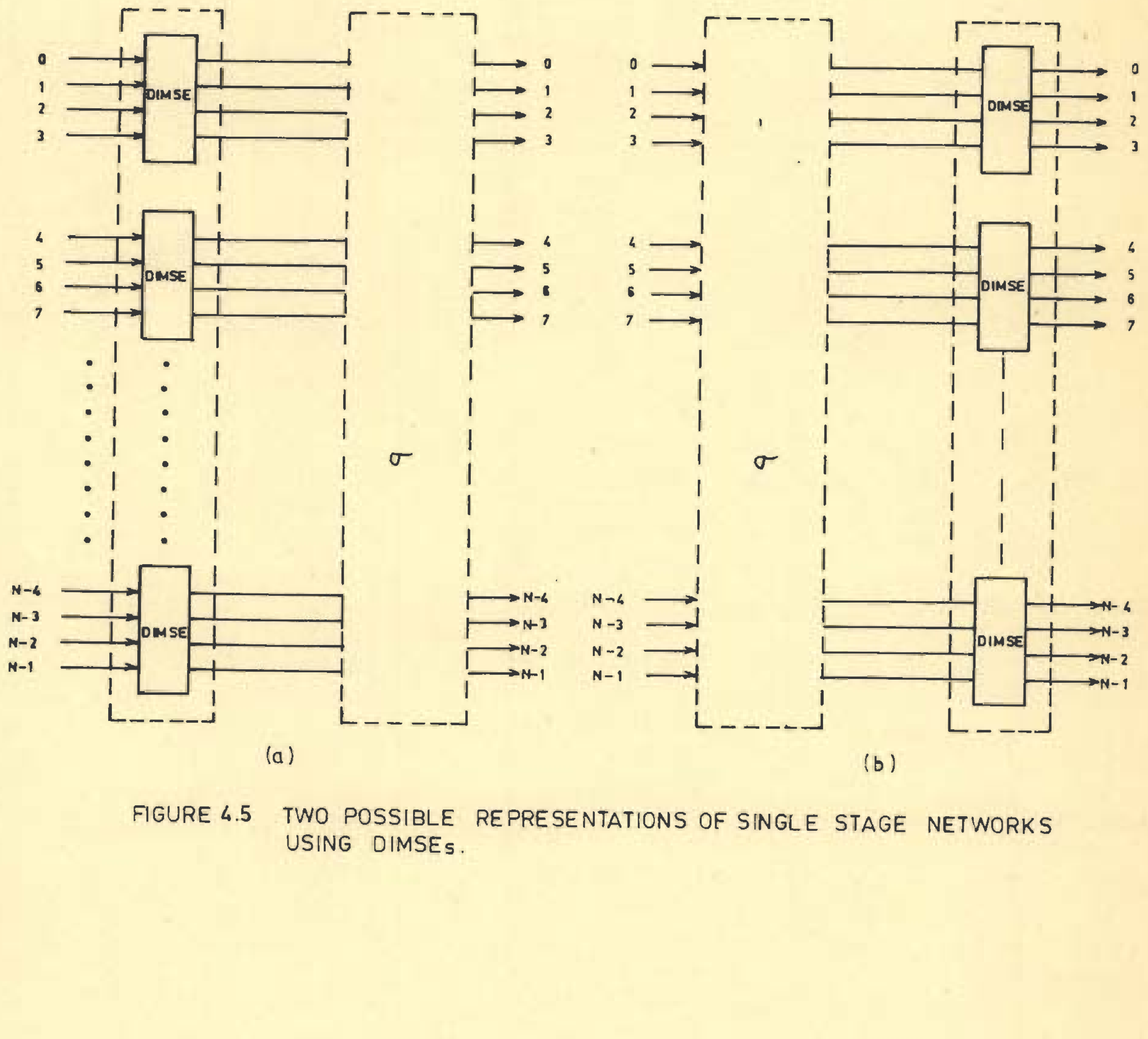


FIGURE 4.5 TWO POSSIBLE REPRESENTATIONS OF SINGLE STAGE NETWORKS USING DIMSEs.

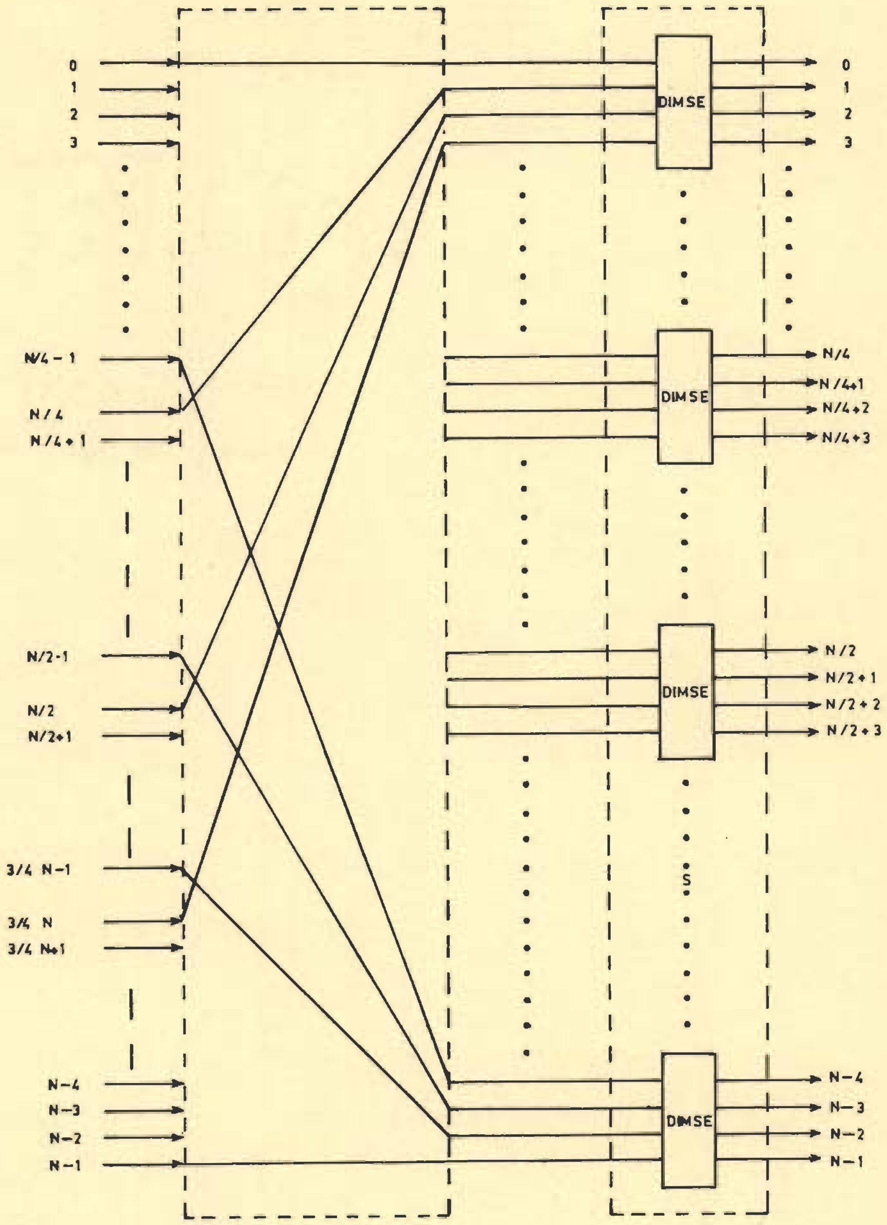


FIGURE 4.6 SINGLE STAGE NETWORK $S_N = P \cdot S$
$S \longrightarrow$ STACK OF N/4 DIMSES

Equation (4.4).

$$P(q_{n-1}q_{n-2}\cdots q_1q_0) = q_{n-2}q_{n-3}\cdots q_1q_0q_{n-1} \qquad ..(4.4)$$

From Fig.4.2, it is observed that DIMSE allows each input address to exchange data with output whose address differs from it in only low order quaternary digit position. The exchange function is thus defined as

$$E(q_{n-1}q_{n-2}\cdots q_1q_0) = q_{n-1}q_{n-2}\cdots q_1\bar{q}_0 \qquad ..(4.5)$$

where $\bar{q}_0$ is relative complement of q_0 as defined in Section 3.1. Fig.4.7 depicts a single stage shuffle - exchange network for n = 16. The shuffle functions are shown by dashed lines and exchange functions by solid lines.

4.3 MULTISTAGE INTERCONNECTION NETWORKS USING DIMSEs

Single stage networks are topologically less complex but perform a limited number of permutations directly. Data items may have to recirculate through the single stage several times before reaching their final destinations. Because of the number of settings required each time for recirculation, the larger switching time and difficult control routing schemes, such networks are usually replaced by the more convenient multistage networks. A multistage interconnection network has more than one stages of switching elements and links, which allow arbitrary permutations. Literature is available as discussed in Section 2.2.3, dealing with how the ICNs can be designed to establish the communication from input to output terminals. These approaches use various topologies to achieve the objective.

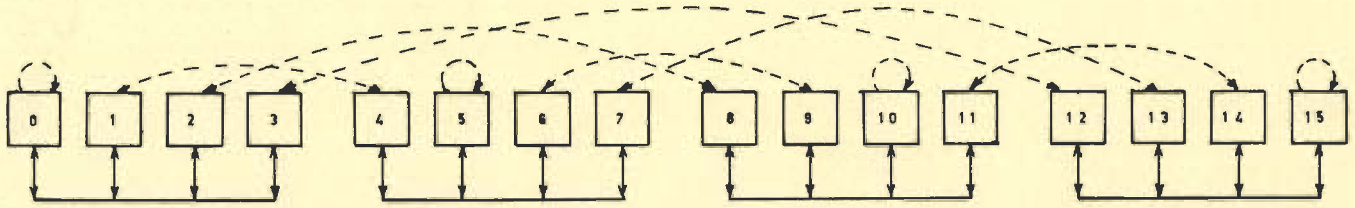


FIGURE 4.7 4-SHUFFLE-EXCHANGE NETWORK FOR N-16 USING 4-SHUFFLE PERMUTER AND DIMSEs. DASHED LINE INDICATE THE SHUFFLE FUNCTION AND SOLID LINES EXCHANGE FUNCTION.

The objective of this section is to propose a new multistage interconnection network topology using DIMSEs and 4-shuffle link pattern. Further, it will be demonstrated that this topology falls in the domain of general cube family. The proposed topology is based on the following criteria:

(i) Dual interconnection modular switches will be employed instead of conventional 2x2 SEs.

(ii) To match the above switches, 4-shuffle permuters (P) as discussed in Section 3.4, and the analogy of 4-shuffle with dual cube as has been described by Theorem 3.3 will be used.

(iii) The ICN is of size NxN, $N = 4^n$, with full access capability.

4.3.1 Development of DCMIN - Dual Cube Multistage Interconnection Network

Consider a network with N = 16. It is desired to develop an ICN for this using DIMSEs. It is obvious that 4-unit stack of DIMSEs will be required to match 16 inputs. Let each of the inputs be designated in quaternary number system. Each input node of a DIMSE will have the input addresses as $(ij)_{QR}$ with i remaining constant and j varying from 0 to 3. If now the output nodes of the stack are connected to a 4-shuffle permuter, then the output nodes of the permuters will have the sets of addresses as $(ji)_{QR}$, each set consisting of 4 nodes, with i remaining constant for each set and j varying from 0 to 3. Mathematically

$$P(ij)_{QR} \longrightarrow (ji)_{QR} \qquad 0 \leq i \leq 3 \qquad ..(4.6)$$
$$0 \leq j \leq 3$$

This is illustrated in Fig.4.8. It can be seen from the figure that to maintain full-access capability, i.e., for any arbitrary input/output connections, it may require two passes. For example, consider the mapping of input address '00' on output address 33. In the first pass '00' will map on '03' and after feed back through the terminal '30' in second pass it will map on '33'. This is shown by dark lines in Fig.4.8.

A more appropriate representation of 16x16 network is given in Fig.4.9, where two identical stacks of DIMSEs have been used rather than feedback. This has been achieved by transforming the network from time-mode to spatial-mode. The ICN is transformed into iterative blocks of DIMSEs connected through the 4-shuffle permuters. The problem of feedback for each pass is then reduced to adding of an individual DIMSE stack. This time mode to spatial mode transformation thus transfers the concept of n-passes into n-stages of switching elements. The ICN thus developed is a multistage interconnection network. Fig.4.9 is, therefore, a 2-stage network.

4.3.2 Dual Cube Analogy

An inspection of Fig.4.9 reveals that the input addresses of stage-1 and stage-2 are basically the sets of addresses on 'X' and 'Y' coordinate lines of backspread dual cube respectively, as shown in Fig.3.2(b). Since the network exhibits the properties of dual cube, it is termed as dual cube two stage interconnection network or in general 2-stage Dual Cube Multistage Interconnection Network (DCMIN). Thus, Fig.3.2(b) or

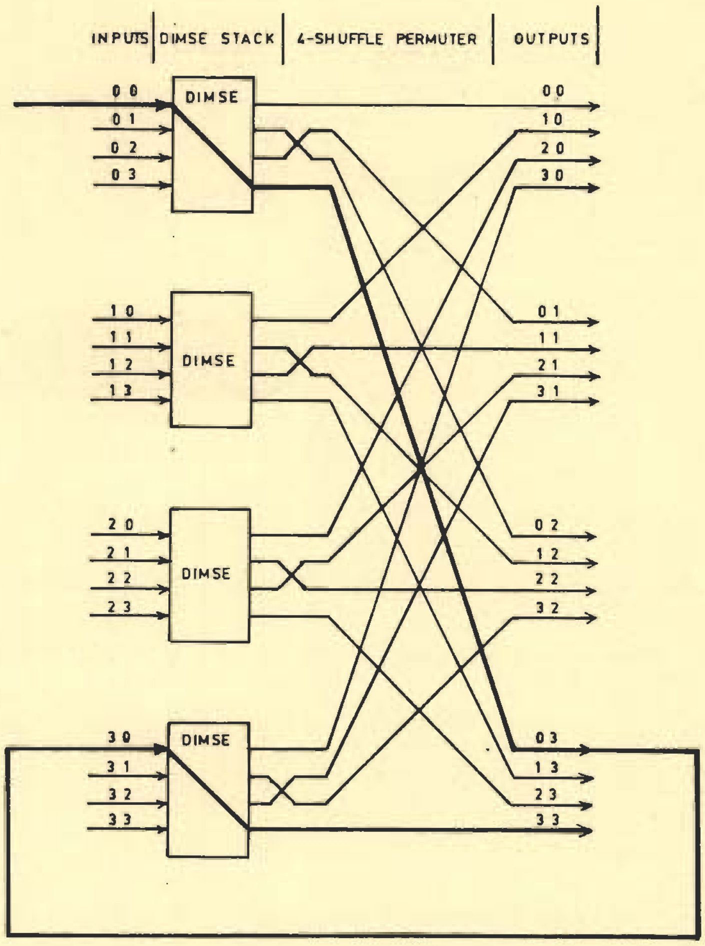


FIGURE 4.8 A 16 X 16 ICN USING DIMSEs AND 4 – SHUFFLE PERMUTER

Fig.3.2(c) are the dual-cube representations of 16x16 DCMIN.

4.3.3 Generalization of DCMIN

To generalize the analysis of Section 4.3.1, the following symbols and notations are used.

(i) Basic switching element (DIMSE) being single stage network is represented by M_1 and NxN, $N = 4^n$, interconnection network being n stage network is represented by M_n.

(ii) S denotes the stack of DIMSEs.

(iii) '+' sign is used to represent stack and '.' for cascade in mathematical representation.

(iv) The stages are numbered as 1,2,3....n from source links to destination links.

In order to design DCMIN of higher sizes, Fig.4.9 depicting M_2 network is analysed.

(i) In the figure, $N = 16 = 4^2$, gives $n = 2$. Also the number of stages are two. Hence, it is postulated that the network of size N will have n number of stages given by

$$n = \log_4 N$$

(ii) Each stage has $\frac{N}{4}$ DIMSEs.

(iii) DIMSE being a 4x4 network, following steps are involved in developing 16x16 network from 4x4 network:

(a) Four number of single stage networks are stacked one over the other.

(b) In addition (4^{2-1}) number of DIMSEs are stacked one over the other and connected as an additional stage.

(c) Two stacks, i.e., the stack of step (i) and the stack of step (ii) are connected through a four shuffle permuter.

The above observations of Fig.4.9 can be readily used to develop a network of size $N = 4^3 = 64$ in the following steps.

(i) $N = 64 = 4^3$ suggests that the network should have 3 stages and be denoted by M_3.

(ii) Each stage should have $\frac{N}{4} = \frac{64}{4} = 16$ DIMSEs.

(iii) 4-Units of M_2 be stacked one over the other.

(iv) 16 DIMSEs should be stacked to form the last stage.

(v) Stacks of step (iii) and (iv) be connected through 4-shuffle permuter of size 64x64.

A DCMIN of size 64x64 with 3 stages, i.e., M_3 is shown in Fig.4.10.

Theorem 4.1

A DCMIN of size NxN is a hierarchical structure and will always consists of 4-units of M_{n-1} subnetworks each having 4^{n-1} inputs for $n \geq 2$.

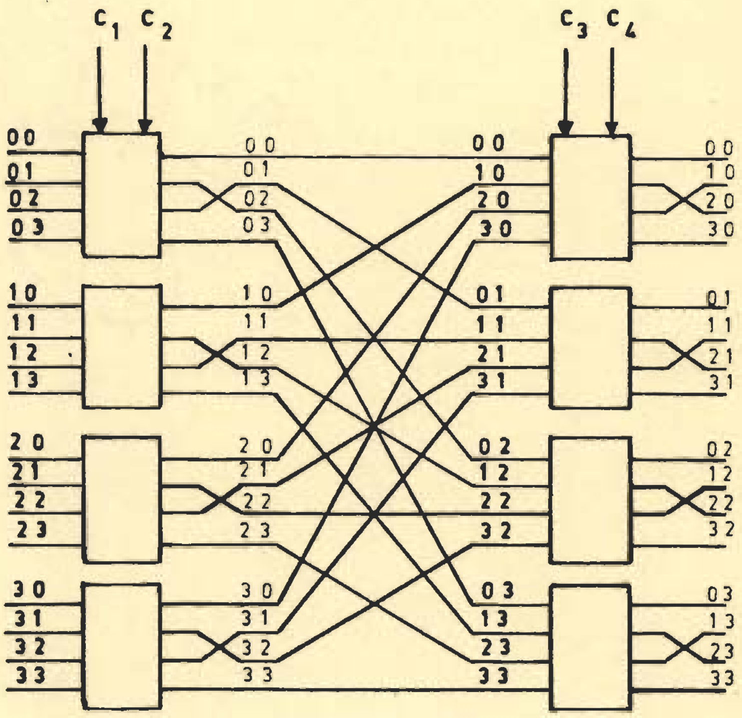


FIGURE 4.9 16 X 16 DCMIN. IT USES 8 DIMSEs AND 16 EXTERNAL LINKS.

Proof

The proof of this theorem is by induction. For n = 2, the size of the network is 16x16 as shown in Fig.4.9. It consists of a stack of 4-units of M_1 connected through 4-shuffle permuter with a stack of $4^{2-1} = 4$ units of DIMSEs.

For n = 3, accordingly the network size is 64x64 as shown in Fig.4.10. Again, this consists of a stack of 4-identical units of M_2 cascaded with a stack of $4^{3-1} = 4^2 = 16$ DIMSEs.

Each of the subnetworks M_1 has 4 input/output terminals and that of M_2, 16 input/output terminals.

Proceeding in this way for networks of higher size, it can be said that a DCMIN of size NxN is hierarchical structure and will always consist of 4-units of M_{n-1} subnetworks each having 4^{n-1} inputs for $n \geq 2$.

▱

Theorem 4.2

A DCMIN of size NxN, $N = 4^n$, will require minimum n-stages of N/4 DIMSEs to achieve full access property.

Proof

Theorem 4.1 states that a DCMIN of size M_n will always consist of a stack of 4-units of DCMIN of size M_{n-1} cascaded with an additional stage of DIMSEs.

A DIMSE is a basic building block of DCMIN of size 4x4. In Section 4.2 it was stated that DIMSE can perform any arbitrary permutation on its input/output terminals and thus

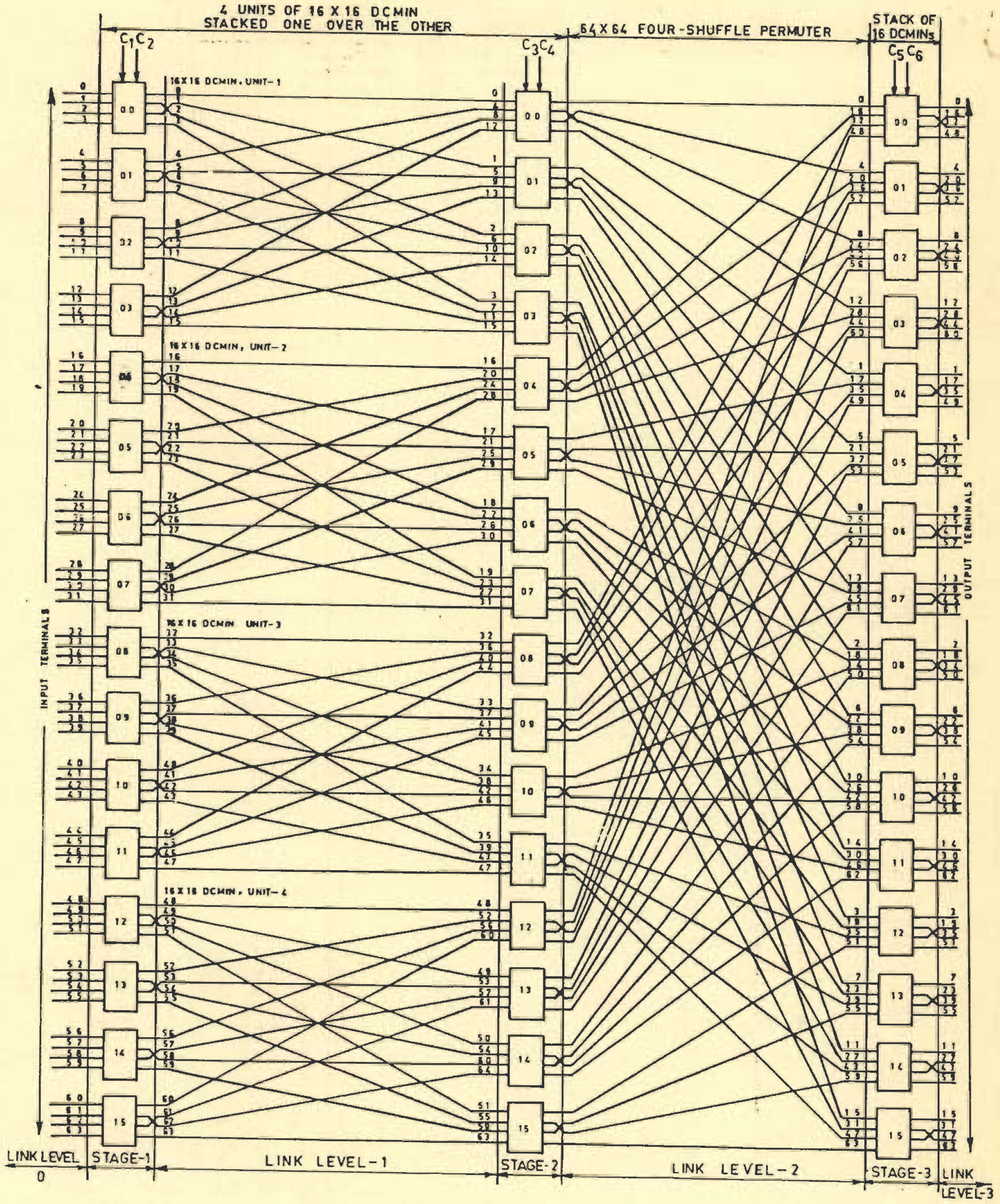


FIGURE 4.10 64 X 64 DCMIN USES 4 UNITS OF 16 X 16 DCMIN AND A STACK OF 16 DIMSEs CONNECTED BY FOUR-SHUFFLE PERMUTER. FOR CLARITY CONTROL LINES OF INDIVIDUAL DIMSEs ARE OMITTED

posses FA property. In Section 4.3.1 it has been demonstrated that a 16x16 network requires minimum two stages to maintain FA. In Fig.4.10, depicting 64x64 DCMINs, a stack of 4 identical units of 16x16 DCMINs constitutes first two stages. If the last stage (stage-3) is removed then each of the 4 units will become independent subnetworks of size 16x16. PEs connected to one subnetwork will not be able to communicate with any of the PEs connected to other three subnetworks. Thus the DCMIN of 64x64 will loose its FA property. However, the addition of third stage will restore its FA property. Hence, it is concluded that 16x16 DCMIN needs minimum two and 64x64 DCMIN needs atleast three stages to maintain FA. Further, 16x16 DCMIN has 4 DIMSEs in each stage and 64x64 DCMIN has 16 DIMSEs in each stage. These observations can be extended for a network of any size NxN by the statements that a DCMIN will always require minimum n-stages employing N/4 DIMSEs in each stage to maintain FA.

$\square$

Corollary 4.1

A DCMIN of size N can be developed from a DCMIN of lower size N/4 according to the following equation:

$$M_n = (M_{n-1} + M_{n-1} + M_{n-1} + M_{n-1}).P.S \qquad ..(4.7)$$

Proof

The proof is obvious from Theorem 4.1 and 4.2.

$\square$

Equation 4.7 is depicted in Fig.4.11 giving the general structure of a DCMIN (M_n).

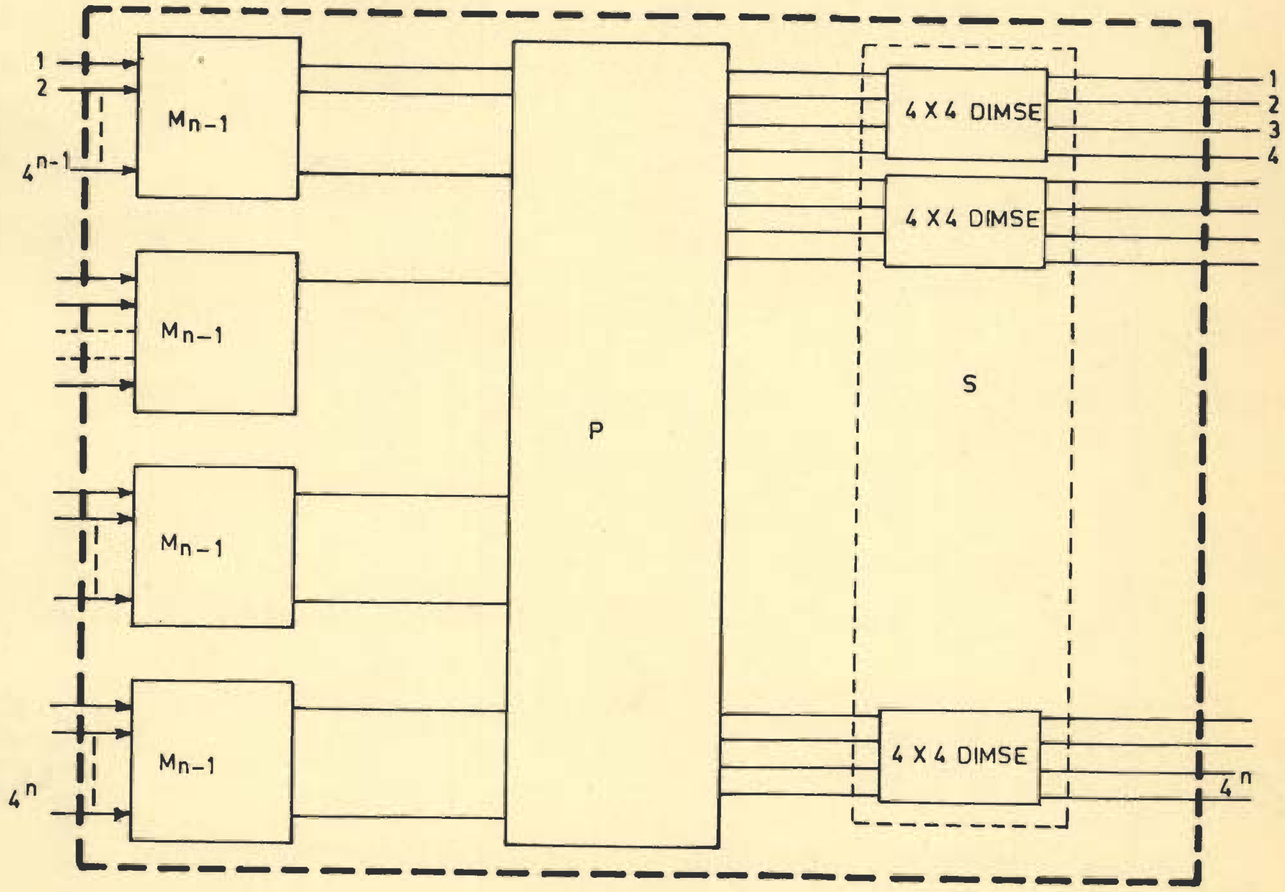


FIGURE 4.11 THE GENERAL STRUCTURE OF N X N DCMIN (M_n) WHERE $N = 4^n$

Figure 4.12 is same as Fig.4.10 but with input/output addresses represented in quaternary system. Observing the addresses of each stage from top to bottom in Fig.4.12, if an input address at stage-1 is represented as

$$(ijk)_{\text{Ist stage}} \qquad 0 \leq i,j,k, \leq 3 \qquad ..(4.8)$$

then the corresponding address in the stage-2 will have the representation as

$$(ikj)_{\text{2nd stage}} = i(P(jk)) \qquad ..(4.9)$$

Similarly in the stage-3, it will be represented as

$$(kji)_{\text{3rd stage}} = P(i\ P(jk)) = P(ikj) \qquad ..(4.10)$$

Example 4.2

An address at a distance of 31 position from the top address (000) in Fig.4.12 is represented as (132). The corresponding address in stage-2 (that is, 31-positions away from top address (000)) is represented by (123) and in stage-3 as (231).

4.3.4 Dual Cube Transformation of Three Stage DCMIN

In Section 4.3.2 it has been demonstrated that Fig.3.2(b) is the dual cube representation of 16x16 DCMIN. From Theorem 4.1 DCMIN of size N will always have 4-units of DCMIN of size N/4. Applying Theorem 4.1 to develop the dual cube analogy for a network of size N = 64, it requires that 4-units of backspread dual cube of Fig.3.2 (a) or (b) be stacked one over the other. The resulting topology will have three dimensions as shown in Fig.4.13. With reference to Fig.4.12, the sets of elements

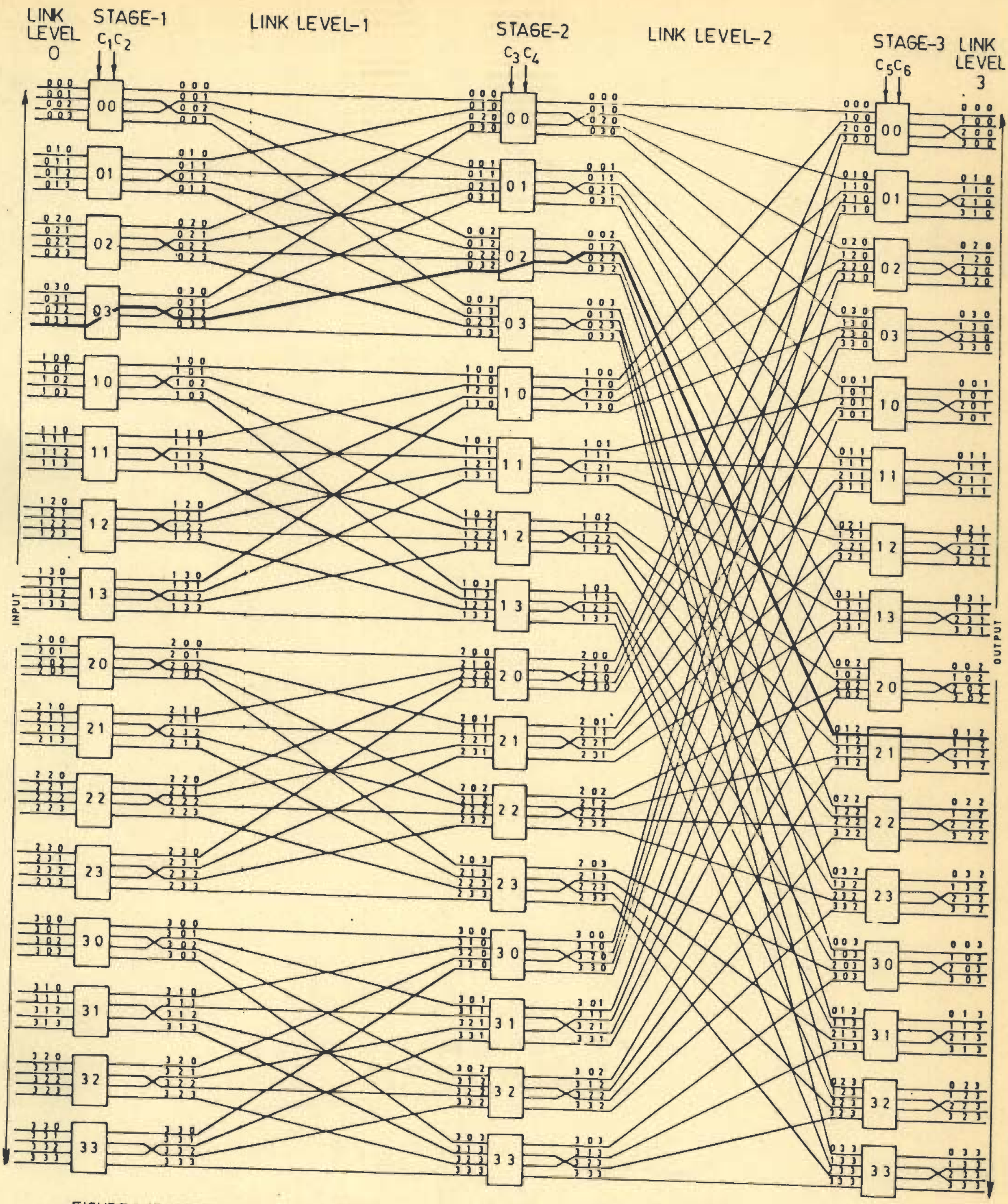


FIGURE 4.12 DESTINATION-TAG ALGORITHM IS USED TO CONNECT SOURCE ADDRESS (033) TO DESTINATION ADDRESS (012)

which are on X-coordinate lines will match in the first stage of 3-stage network, the sets of elements on Y-coordinate lines will match in stage-2, and the sets of elements which are on Z-coordinate lines will match in stage-3 of the three stage network.

4.4 INTERCONNECTION FUNCTION OF DCMIN

An interconnection network can be described by a set of interconnection functions. In Section 2.2.2 the commonly used interconnection functions were discussed. In this section the interconnection function representing DCMIN is defined and is abbreviated as 'Dcube' function.

In Section 3.3 the cube representation of 4-shuffle permutation was discussed. From Theorem 3.3 if the sets of elements (addresses of PEs) which are matched in one dimension (Ist stage) differ in their lowest digit, then after 4-shuffle the sets of elements which will be matched in the other dimension (2nd stage) will differ in their next lower digit. The above statement can be generalized to extend to other dimensions also. Each stage of DCMIN can be visualized as one dimension of multi-dimensional backspread dual cube. ~~Thus in general n-dimensional backspread dual cube.~~ Thus, in general n-dimensional backspread dual cube can be visualized. For example 3-dimensional-back-spread dual cube is shown in Fig.4.13 for N = 64.

Theorem 4.3

The DCMIN of size N consists of n interconnection functions, $n = \log_4 N$, defined by

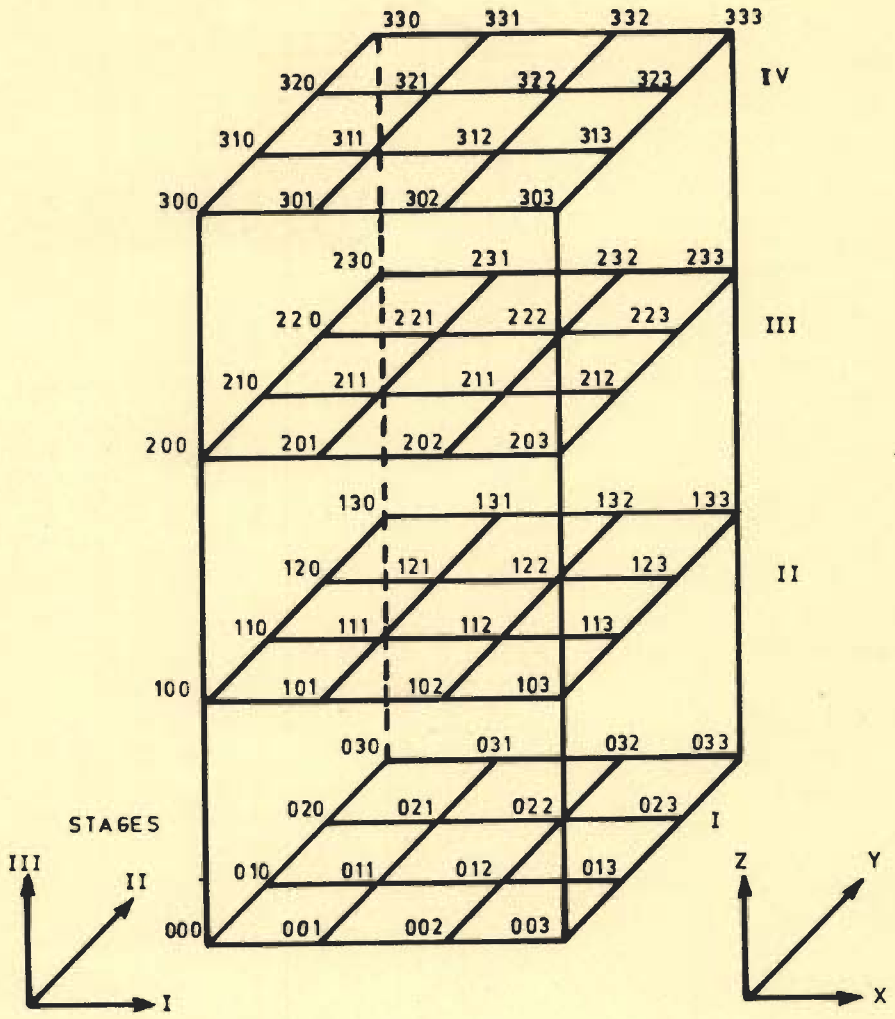


FIGURE 4.13 4 UNITS OF BACK—SPREAD DUAL CUBE ARE STACKED ONE—OVER THE OTHER TO DEVELOP THIRD-DIMENSION.

$$\text{Dcube}_i\ (q_{n-1}q_{n-2}\cdots q_i\cdots q_0) = q_{n-1}q_{n-2}\cdots \bar{q}_i\cdots q_0 \quad ..(4.10)$$

where $0 \leq i \leq (n-1)$ and $\bar{q}_i$ is relative complement of q_i as defined in Section 3.1.

Proof

From Fig.4.12, it can be seen that at stage-1, the PEs whose quaternary addresses differ in the lowest digit position only, can exchange their data items, and at stage-2, the PEs differing in the first positional digit of their addresses can exchange the data. Similarly, at stage-3, the PEs differing in second positional digit of their addresses are able to exchange data. Mathematically the above statement is equivalent to taking the relative complement of the 1st digit at stage-1, 2nd digit at stage-2 and 3rd digit at stage-3. Thus, in terms of mapping source addresses to destinations, the Dcube interconnection function Dcube_i complements the i^{th} positional digit of the PE addresses (in terms of relative complement). This proves the validity of Equation (4.10).

$\square$

Theorem 4.4

The Dcube network allows each PE to communicate with any of the n-sets of PEs whose addresses differ from it in any one quaternary digit position.

Proof

The Dcube network function is a generalization of the exchange function defined in Section 4.2 in that the exchange allows the complementing of the 0th digit position and Dcube allows the complementing of any of the (n-1) digit positions. In NxN, $N = 4^n$, DCMIN there are n stages of DIMSE stacks. In Theorem 4.3 it has been shown that at each (i+1)th stage the PEs whose addresses differ in ith position can communicate with each other. Thus, the Dcube network allows each PE to communicate with any of the n-sets of PEs whose addresses differ from it in any one digit position.

$\square$

Figure 4.14 depicts the dual cube interconnection functions Dcube_0 and Dcube_1 for N = 16. Also, Fig.4.7 is the representation of shuffle - exchange functions for a network of size N = 16. Comparison of Figs.4.7 and 4.14(a) shows that Dcube_0 and exchange functions are identical.

4.5 LABELING AND ADDRESSING OF LINKS AND DIMSEs IN DCMIN

In Fig.4.10 the stages are numbered from 1 to n, the stage connecting input terminals is named as stage-1 and the one connecting the output terminals as stage-n. Link levels are numbered from 0 to n, input terminals are in link level-0 and output terminals in link level-n, where $n = \log_4 N$. The links (terminals) in link level-0 and link level-n are directly connected to PEs. Interstage terminals are not directly connected to PEs. Hence, the interstage links are called external links. Sometimes to emphasise that links in link level-0 and n are not external links they are referred to as terminal links.

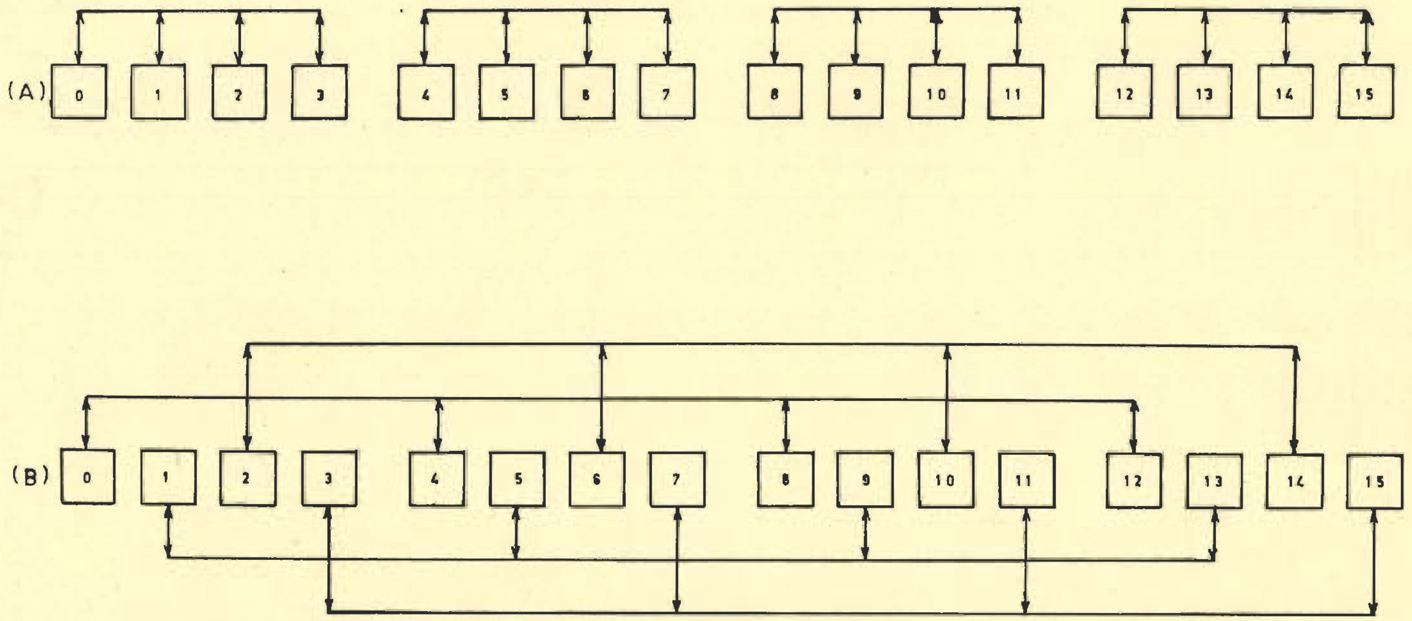


FIGURE 4.14 DCUBE NETWORK FOR N 16. (A) Dcube$_0$ connections. (B) Dcube$_1$ connections.

The input terminals (links) are addressed according to their physical locations - the top link is addressed as 0 and the bottom link as (N-1). Each input terminal is connected to its own processing element, i.e., the addresses of the PEs and input terminals coincide. In subsequent stages (link levels), when all the DIMSEs are put to mode '0' condition, the links bringing the particular data item are labeled by the same addresses. This is the addressing method under which DCMIN can perform identity permutation. DIMSEs are also addressed in each stage according to their physical location in the stage - the top most DIMSE is addressed 0 and the bottom one as $(\frac{N}{4} - 1)$.

In Fig.4.12, same nomenclature is used as in Fig.4.10 except that the links and DIMSEs are addressed in equivalent quaternary number system. Link-addresses are represented by n-digits and the DIMSE-addresses by (n-1) digits. In each stage, the addresses of the links connected to each DIMSE vary only in the lowest positional digit. Thus, (n-1) higher digits of any link address will represent the address of the DIMSE to which it is connected in the corresponding stage. In Fig.4.15 the terminals (link) are addressed in binary number system. In binary each link-address will have 2n bits because $N = 4^n = 2^{2n}$.

For fault diagnosis purpose, it is required to know the physical locations of the links and DIMSEs in each stage (link-level) to repair or replace the faulty item. Accordingly, in Fig.4.16, the links in each link level are addressed according to their physical location at outputs side, assuming links in level-0 are the outputs of PEs. However, DIMSEs are assigned addresses according to their physical location in a particular

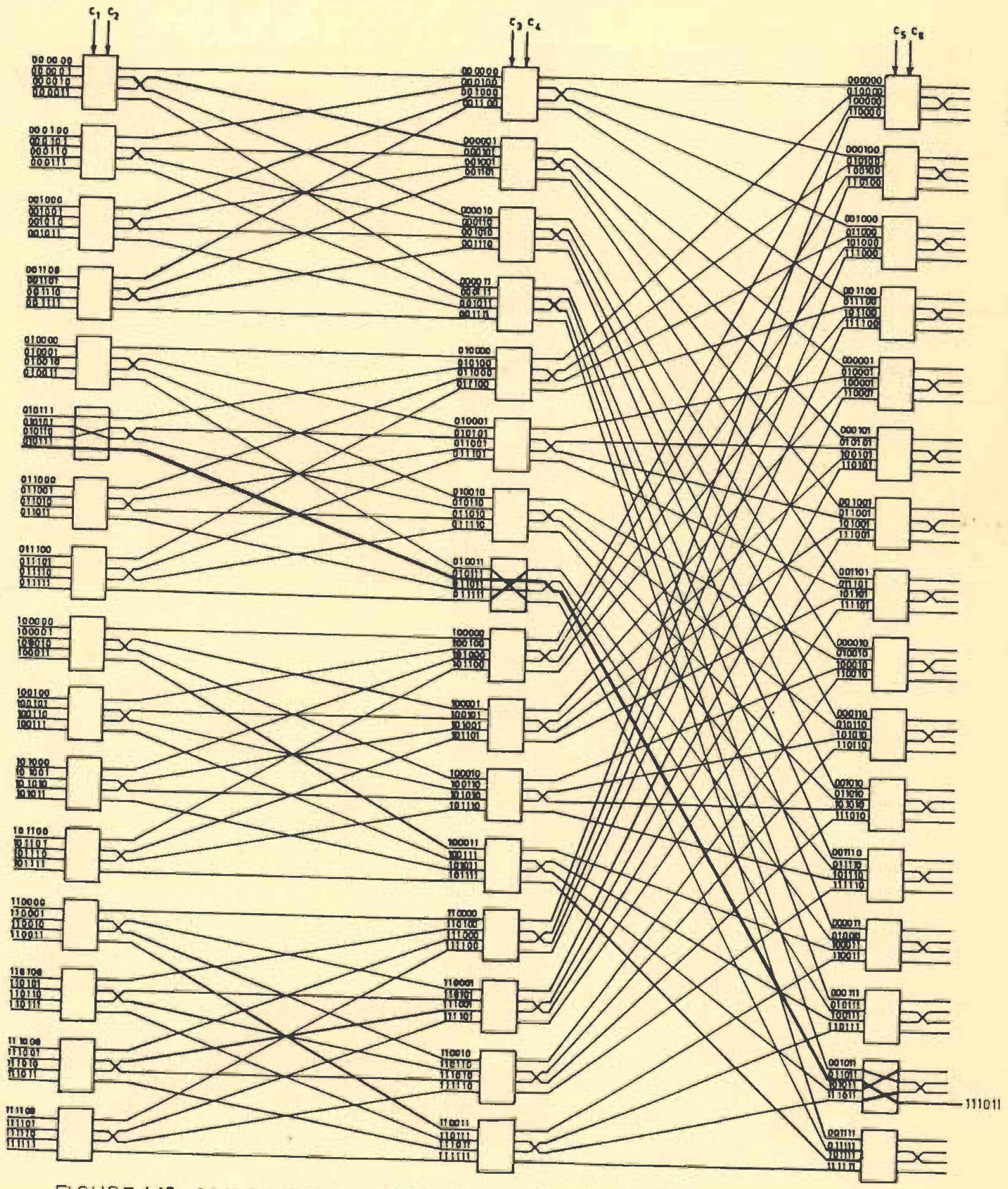


FIGURE 4.15 64 X 64 DCMIN USING 4 X 4 – DIMSEs DARK LINE INDICATES ROUTE FROM SOURCE (010111) TO DESTINATION (111011) AND CORRESPONDING SWITCH CONTROL SETTINGS.

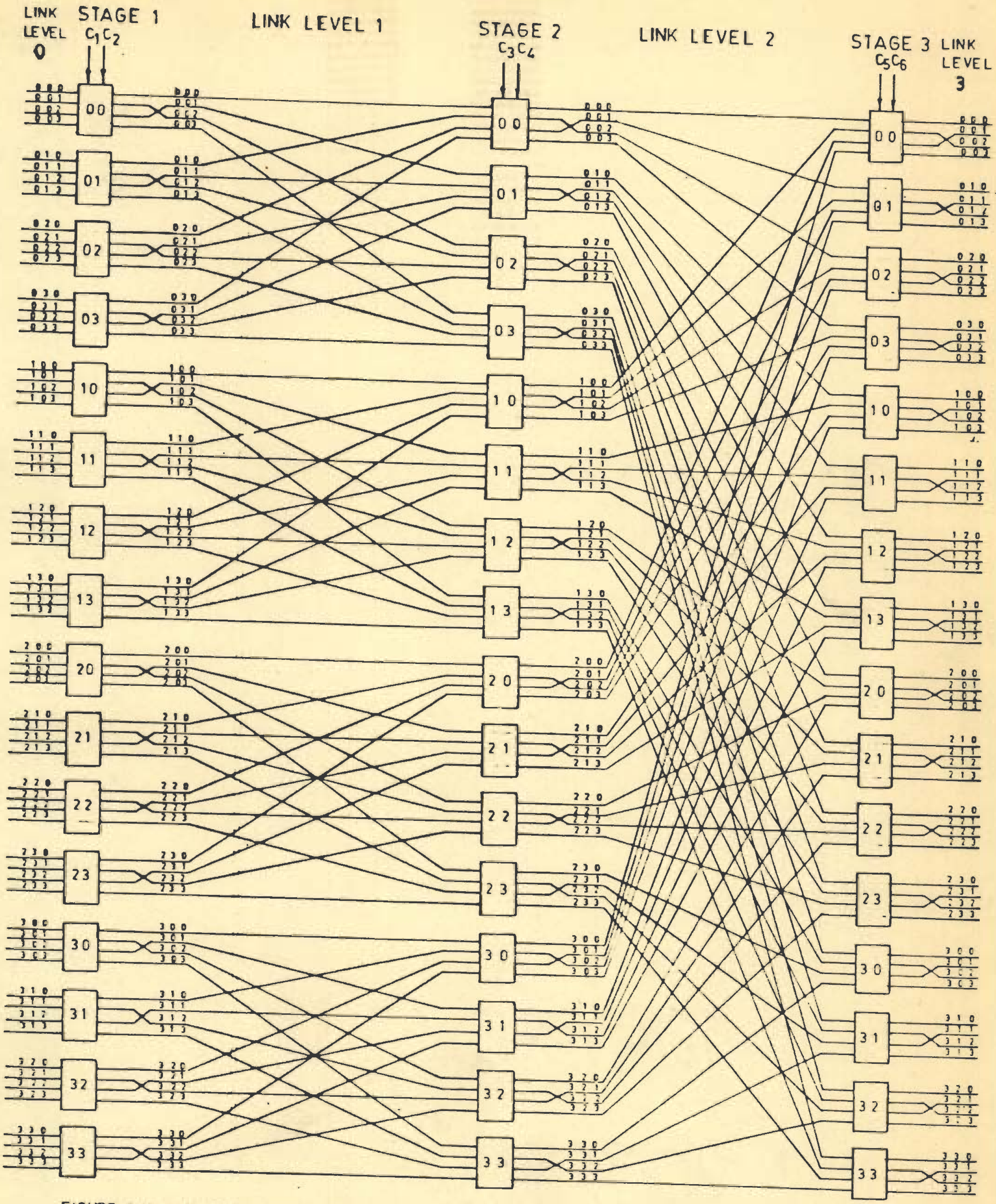


FIGURE 4.16 A 3-STAGE DCMIN. THE LINKS IN EACH LEVEL ARE ADDRESSED ACCORDING TO THEIR PHYSICAL LOCATION IN THE LEVEL

stage, as has been done above.

4.5.1 Locating the Fault-Source From the Known Address of the Output Terminal Reflecting the Fault

A data while travelling from a source to destination passes through intermediate stages and various external links. The fault can occur in any of the DIMSE or external link. However, the faults are identified only at the output link level (level-n). To determine the fault-source, usually back tracking is done [154] which becomes too involved and cumbersome for large sized networks. Here a method has been devised to locate the fault-source without back tracking when the physical address of the output terminal reflecting the fault is known.

Theorem 4.5

If a particular data is carried by an output terminal in link level-n having the physical location 'K_n', $0 \leq K_n \leq (N-1)$ with quaternary address of K_n given as

$$K_n = [q_{n-1}\, q_{n-2} \cdots q_i \cdots q_1 q_0] \qquad ..(4.11)$$

then the same data item in link level $-(n-i)$, $0 < i < n$, will be carried by the link location (K_{n-i}), with quaternary address given as

$$K_{n-i} = [q_0 q_1 \cdots q_{i-1} q_{n-1} q_{n-2} \cdots q_i] \qquad ..(4.12)$$

With the condition that all the stages are set to mode '0' operation.

Proof

Under mode '0' operation the identity permutation (as described in Section 4.1) is performed on the DIMSE. Hence in tracking back from the DIMSE output to DIMSE input there is no variation of addresses. However, in traversing back from the stage-i to the preceding stage, the reverse 4-shuffling is performed by cyclically rotating the address digits one digit to the right.

As per Corollary 4.1, since DCMIN is recursive in nature, i.e., each network M_n contains 4-units of network M_{n-1}, each M_{n-1} contains 4 units of M_{n-2} and so on. Accordingly, in traversing from last stage to $(n-1)^{th}$ stage the reverse shuffle is performed on n digits address, i.e., from q_0 to q_{n-1}. In traversing from stage-$(n-1)$ to stage-$(n-2)$, the positional digits from 0^{th} position to $(n-2)^{th}$ position are shifted and similarly in traversing from $(n-i+1)^{th}$ stage to $(n-i)^{th}$ stage the positional digits from 0^{th} position to $(n-i)^{th}$ position are rotated.

Thus in traversing back i stages from the n^{th} stage the link address given by Equation 4.11 will change to the new address given by Equation 4.12.

$\square$

Corollary 4.2

In Theorem 4.5, if all the stages are in mode '3' operation of DIMSE then Equation (4.12) will be replaced by Equation (4.1
as given below:

$$K_{n-1} = [(3 - q_0)(3 - q_1)\ldots.(3 - q_{i-1})q_{n-1}q_{n-2}\ldots.q_1] \quad ..(4.13)$$

Proof

From Table 4.1, if the DIMSE is put to mode '3' operation, the input/output mapping is defined as

$$R_k = I_{(3-k)} \quad ..(4.14)$$

From Equation (4.14) it is observed that the LSDs in the addresses of input and output are related by

$$(q_i)_{output} = (3 - q_i)_{input} \quad ..(4.15)$$

Hence, if the DIMSEs of all the stages are put to mode '3' operation, in addition to reverse 4-shuffle, the additional exchange function on the lowest digit will be performed as given by Equation 4.15. Thus the Equation 4.12 will be modified as Equation 4.13.

$\square$

4.6 ROUTING SCHEMES

A routing algorithm is used to control the route of message in DCMIN. The routing tags for one to one data transfers consist of n-bits if quaternary representation is used or it consists of 2n-bits in case binary representation is used to address the terminal links. When a message is transmitted through a network, a routing tag is added as first item in the message. Each switching element (DIMSE), in which the message enters, examines the routing tag to determine the mode of operation of the DIMSE and then sets itself accordingly. The data follows the routing tag through the network. Both destination tag algorithm [80] or Exclusive-OR algorithm [125] can be used for DCMIN. From Figs.4.9 and 4.12 it can be seen that there is one and only one path between any given input/output pair. Since there is only one path from a given source to a given destination, a destination tag algorithm and an Exclusive-OR algorithm generate the same unique paths.

4.6.1 Destination-Tag Routing

In this scheme destination $D = d_n, d_{n-1} \dots d_i \dots d_1$ is taken as a routing tag. A switching element in the network at stage i need only examine the digit d_i. If $d_i = 0$, the output terminal number '0' of corresponding DIMSE is taken, if d_i has value 1,2, or 3 then output terminal number 1,2 or 3 is taken respectively. As an example, consider the path from source (033) to destination (012) as shown in Fig.4.12. The output at terminal number 2 is taken at stage 1, at terminal number 1 at stage 2, and at terminal number 0 at stage 3. This same destination tag (D = 012) would be used to route any input to output 012. This scheme works because four output terminals of a DIMSE in stage i differ in their i^{th} digit position, i.e., terminal 0 has i^{th} digit as 0, terminal 1 has i^{th} digit as 1 and so on. Thus taking any one of the four terminals implies that the network output reached will have corresponding value in the i^{th} digit position.

4.6.2 Exclusive-OR Routing

In this method, first input/output addresses are changed to binary number system as shown in Fig.4.15. Thus each address will have 2n bits and routing tag will also have 2n bits corresponding to 2n control signals. The routing tag is calculated as follows:

If $S = s_{2n}s_{2n-1}\dots s_i\dots s_2s_1$ be the source address in binary and

$D = d_{2n}d_{2n-1}\dots d_i\dots d_2d_1$ be the destination address in binary then the routing tag

$R = r_{2n}r_{2n-1}\dots r_i\dots r_2r_1$ where $r_i = s_i \oplus d_i$

$\oplus$ denotes Exclusive-OR operation and $r_{2n-1}\ r_{2n}$ specify the values (00,01,10 or 11) of the control signals of nth stage modules. As an example, the dark line in Fig.4.15 establishes a route from 010111 to 111011 by setting control signals of stage 1 to (00), stage 2 to (11) and stage 3 to (01).

4.7 COMPLEXITY AND PERFORMANCE EVALUATION OF DCMIN

Complexity of a system is directly proportional to the number of switching elements in a network and number of external links. Also the reliability of a given size of network is directly related to the number of external links or number of stages required (number of SE modules enroute). For evaluating the complexity and performance of DCMIN as has been proposed in the earlier section, certain parameters deciding these factors are evaluated first and then compared with a generalized cube topology using 2x2 switching elements.

(a) Number of Switching Element Modules Required (M)

Each DIMSE can accomodate four terminal links. Thus the number of modules per stage becomes N/4. Further, the number of stages required are $n = \log_4 N$. So the total number of modules required in a network are given by M_{min}.

$$M_{min} = \frac{N}{4}(\log_4 N)$$

(b) Number of External Links (L) (Excluding Terminal Links)

Since there are N links between any two stages and there are (n-1) interstage locations, the number of external links is given as L_{min}

$$L_{min} = N[(\log_4 N)-1]$$

(c) Number of Modules Enroute, i.e., **Passes** (P)

For a fault free network, the information is routed through a single module per stage. Thus the number of passes (modules enroute) becomes equal to the number of stages and is given as

$$P_{min} = (\log_4 N)$$

(d) Cost (C)

As a rough estimate the cost of an RxR module is expected to be approximately equal to (R.R) units [24]. Thus the cost of each DIMSE becomes 16 units and the total cost of NxN DCMIN is given as

$$C_{min} = 16.M_{min} = 4N\log_4 N \text{ units.}$$

However, if the cost of external links is also taken into consideration, the network becomes cost effective.

(e) Network Switching Delay (D)

If Dm is the average delay introduced by the module and Dim as the delay in the intermodular path; presuming no conflicts.

$$D_{min} = D_m(\log_4 N) + D_{im}((\log_4 N)-1)$$

Table 4.2 gives the comparison of proposed DCMIN with the conventional topology of Multistage Cube Network (MCN) [125]. Subscript min is used for DCMIN and mcn for MCN.

From the Table 4.2 it is obvious that the proposed DCMIN, when compared with conventional MCN of same size, has the following advantages:

(i) Number of total switching element modules being one fourth of the conventional networks, the proposed network is obviously less complex.

(ii) The number of external links are reduced to less than half the value for conventional networks, making the proposed DCMIN inherently more reliable.

(iii) As the number of passes are reduced in establishing a desired route, network delay is significantly reduced, enhancing the speed of computation.

(iv) Using the four shuffle permuters the computational complexity also becomes proportional to ($\lceil \log_4 N \rceil$) as compared to the conventional lower bound of ($\lceil \log_2 N \rceil$).

Parameter	Network Size							
	16x16		64x64		256x256		1024x1024	
	MCN	MIN	MCN	MIN	MCN	MIN	MCN	MIN
Number of Modules (M) $M_{min} = \frac{N}{4}.(\log_4 N)$ $M_{mcn} = \frac{N}{2}.(\log_2 N)$	32	8	192	48	1024	256	5120	1280
Number of External Links (L) $L_{min} = N.((\log_4 N) - 1)$ $L_{mcn} = N.((\log_2 N) - 1)$	48	16	320	128	1792	768	9216	4096
Number of Modules Enroute i.e. passes (P) $P_{min} = \log_4 N$ $P_{mcn} = \log_2 N$	4	2	6	3	8	4	10	5
Network Switching Delay (D) $D_{min} = D_n(\log_4 N)+D_{im}((\log_4 N)-1)$ $D_{mcn} = D_n(\log_4 N)+D_{im}((\log_2 N)-1)$	$4D_n+3D_{im}$	$2D_n+3D_{im}$	$6D_n+5D_{im}$	$3D_n+2D_{im}$	$8D_n+7D_{im}$	$4D_n+3D_{im}$	$10D_n+9D_{im}$	$5D_n+4D_{im}$
Cost (C) $C_{min} = 4N(\log_4 N)$ units $C_{mcn} = 2N(\log_2 N)$ units	128	128	768	768	4096	4096	20480	20480

TABLE 4.2 COMPARISON OF DCMIN AND CONVENTIONAL CUBE NETWORK TOPOLOGIES.

(v) From the general observations, based upon guidelines given in [24], it can be readily determined that the cost remains the same whether one uses 4x4 DIMSEs or 2x2 SEs. A fairly accurate estimate should however include the cost of intermodular links also. If the cost of external links is included, the proposed topology becomes much cheaper as compared to the conventional networks.

4.8 CONCLUSION

In this chapter a new topology of MIN designated as DCMIN for parallel architectures has been suggested. A DCMIN is a connecting network composed of 4x4 switches and 4-shuffle permuters. A 4-shuffle permuter has been discussed in Chapter III and the concept of 4x4 switch named DIMSE has been developed in this chapter.

An analysis of DCMIN topology and its comparison with conventional cube type network has been presented. It is observed that the DCMIN topology enhances the computational speed, offers more reliability and is cost-effective. DCMIN has a unique path for any source-destination pair and can perform any one to one connection. However, it is limited in the ways it can permute data. It is shown that a larger size network can be developed recursively from the smaller size networks. For any MIN to be functional, it should have some additional properties such as fault-tolerance and simple methods of fault diagnosis, partitionability to smaller size networks and an easy elegant mathematical representation. These properties of DCMIN are evaluated in subsequent chapters.

CHAPTER V

FAULT-DIAGNOSIS AND FAULT-TOLERANCE OF DCMIN

In parallel processing systems, reliable processing results to a great extent rely on the integrity of the data communication paths. Since a large multistage interconnection network is also prone to the occurance of faults, fault-diagnosis and fault-tolerance issues assume significance. It has been discussed in Section 2.2.3 and 4.3 that a MIN consists of stages of switching elements having p-inputs/p-outputs, interconnected through a definite link pattern. Therefore, in order to consider a MIN in totality, its fault-diagnosis and fault-tolerance aspects must be investigated. Fault-diagnosis techniques for multistage interconnection networks using 2x2 SEs have been discussed in [4,31,72,154]. The network should be implemented in such a way that a non-catastrophic fault may not force a complete shut-off and the system should rather continue working with reduced capacity. This is an essential property of the ICNs used in parallel processing and is described as fault-tolerance property [117]. A measure of fault-tolerance has been described by a property called dynamic full access property of a given ICN [118,119]. A network is said to have DFA capability if under a given fault set, each input of the network could still be connected to any one of the network outputs, and parallel system continues to function with gracefully degraded communication.

To study the fault-diagnostic and fault-tolerance aspects of DCMIN, Stuck At Fault model has been adopted. Further, it is assumed that only non overlapping single stuck type faults can occur. These faults could be either on the control lines or in any input/output lines of DIMSEs. The chapter is divided into two parts one dealing with the fault-diagnosis (Section 5.1) and the other dealing with fault-tolerance (Section 5.2).

5.1 FAULT-DIAGNOSIS OF DCMIN

It has been discussed in Section 4.3 that DCMIN employs stages of 4x4 switching elements interconnected through 4-shuffle link pattern. The techniques of fault diagnosis developed for MINs using 2x2 SEs [4,47] are not directly applicable to DCMIN topology. Due to idiosyncracies of the DCMIN, if the test vectors are not selected carefully, some of the faults will be masked and may pass off undetected. The fault-diagnosis problem consists of two steps - fault-detection and fault-location. The presence of a fault is detected first by applying a set of known stimuli at the input terminals and comparing the actual outputs with expected values. Any mismatch indicates the presence of a fault. Once the fault is detected then additional tests may be necessary to locate it.

5.1.1 Fault Model of Dual Interconnection Modular Switch

Faults occur more frequently at an IC's terminal than within the logic circuitry. It is assumed that the faults occuring within the IC manifest themselves as logical faults on the pins of the switching elements [150]. Logical faults on the pins are known stuck-at-0 or stuck-at-1 faults abbreviated as S-a-α with $\alpha \in 0, 1$. These faults cause malfunctioning of the switch and preclude changes in the logic level of the faulty terminal. A single stuck type terminal fault can be any one of the following faults:

(i) Either of the input lines becomes S-a-α, $\alpha \in (0,1)$

(ii) Either of the output lines becomes S-a-α, $\alpha \in (0,1)$

(iii) Either of the control lines becomes S-a-α, $\alpha \in (0,1)$

Faults of type (i) and (ii) above are illustrated in Figs.5.1 and 5.2,whereas, control line faults will constrain the DIMSEs to operate in one of the four allowed modes as already shown in Figs.4.2(a-d). It is assumed that any switching element will not have stuck-at-fault at more than one terminal at any time. This is a reasonable assumption because the diagnostic programmes are run at regular intervals and the possibility of multiple faults occuring on the same switching element becomes remote except under catastrophic conditions.

As the fault model for link faults is different from the fault model of control line faults, different methods are employed for their diagnosis.

5.1.2 Diagnosis of Fault at an External Link

In the fault diagnosis of DCMIN, it is assumed that the terminal links are fault-free. This is a logical assumption and is justified because if a terminal link becomes faulty the corresponding processing unit cannot be connected to the network and the line cannot be used for any data transmission. Additional stages will be of no help. The fault diagnosis of external links is done in two phases:

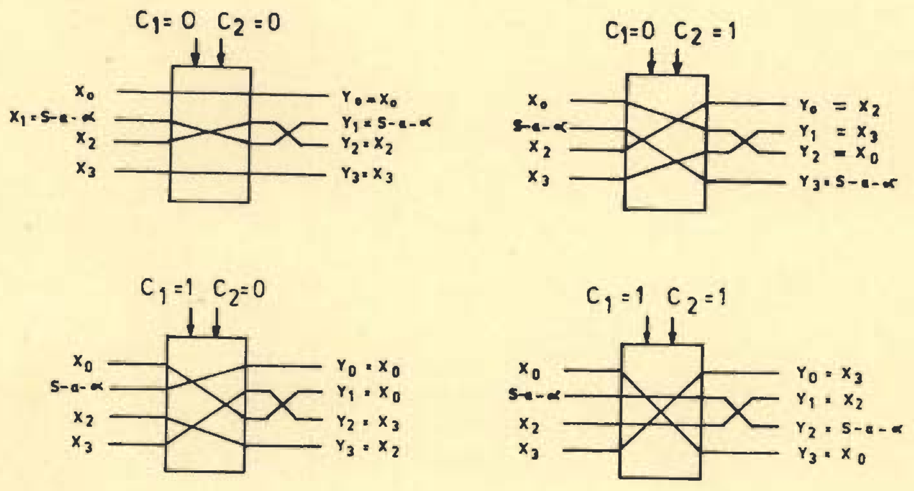


FIGURE 5.1 DIMSE WITH INPUT LINE $X_1 = S-a-\alpha$ ($\alpha \in o,1$)

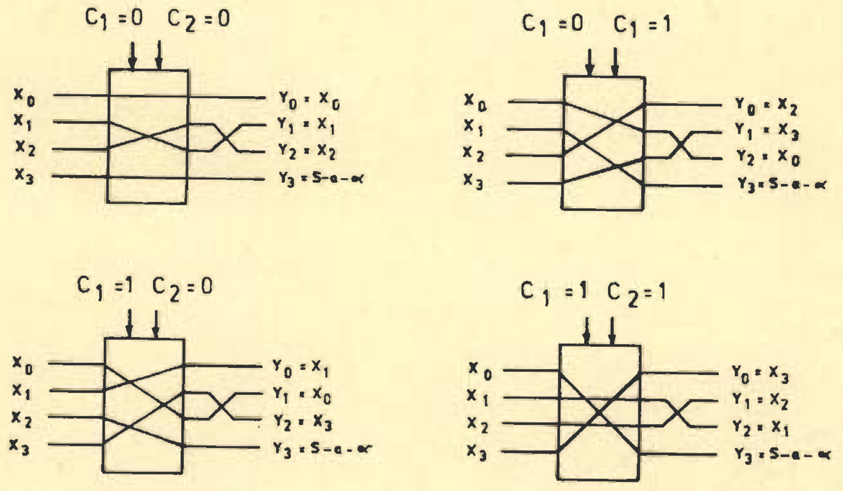


FIGURE 5.2 DIMSE WITH OUTPUT LINE $Y = S-a-\alpha$ ($\alpha \in o,1$)

(i) In the first phase an input vector 'V' with all its elements at 'O' logical value is applied keeping the network in mode 'O' condition. If the network is fault free or link fault is S-a-O type, all the output terminals should get a 'O' logical value. In the second stage of this test complementary logical values are applied as the input vector. Again if the network is fault free or link fault is S-a-1, all output terminals will receive a '1' logical value.

(ii) In phase two both the above tests are repeated while the network is changed to mode '3' condition. The stepwise procedure is described as under:

PHASE I

Step 1: Set all the DIMSEs to mode 'O' condition.

Step 2: Apply a test-vector such that all the input terminals are assigned a 'O' logical value. Observe the response at the output terminals.

Step 3: Apply complementary test-vector, i.e., logical '1' value at all the input terminals. Observe the response at output terminals again.

Step 4: Record the physical addresses of output terminals having the identical output values in Step 2 and Step 3. These are the output terminals having faulty response.

Step 5: From the physical addresses recorded in Step 4, write down the addresses of links in all levels from level (n-1) to 1, that bring data to the recorded addresses of the terminals. The determination of such addresses has already been explained in Section 4.5.

PHASE II

Step 6: Change the control signals so that all DIMSEs are set to mode '3' condition.

Step 7: Repeat Step 2 to 5 of Phase I above.

Step 8: Addresses of the links in different link-levels found in Phase I and Phase II are compared and common addresses are listed separately. Then this list of common addresses will be the list of faulty links.

Example 5.1

Consider the following set of non overlapping arbitrary faults as shown in Figs.5.3 and 5.4.

Link-Level	Addresses of faulty links	Type of fault
1	000	S - a - 0
	110	S - a - 1
	213	S - a - 0
	333	S - a - 1
2	001	S - a - 1
	103	S - a - 0
	223	S - a - 1

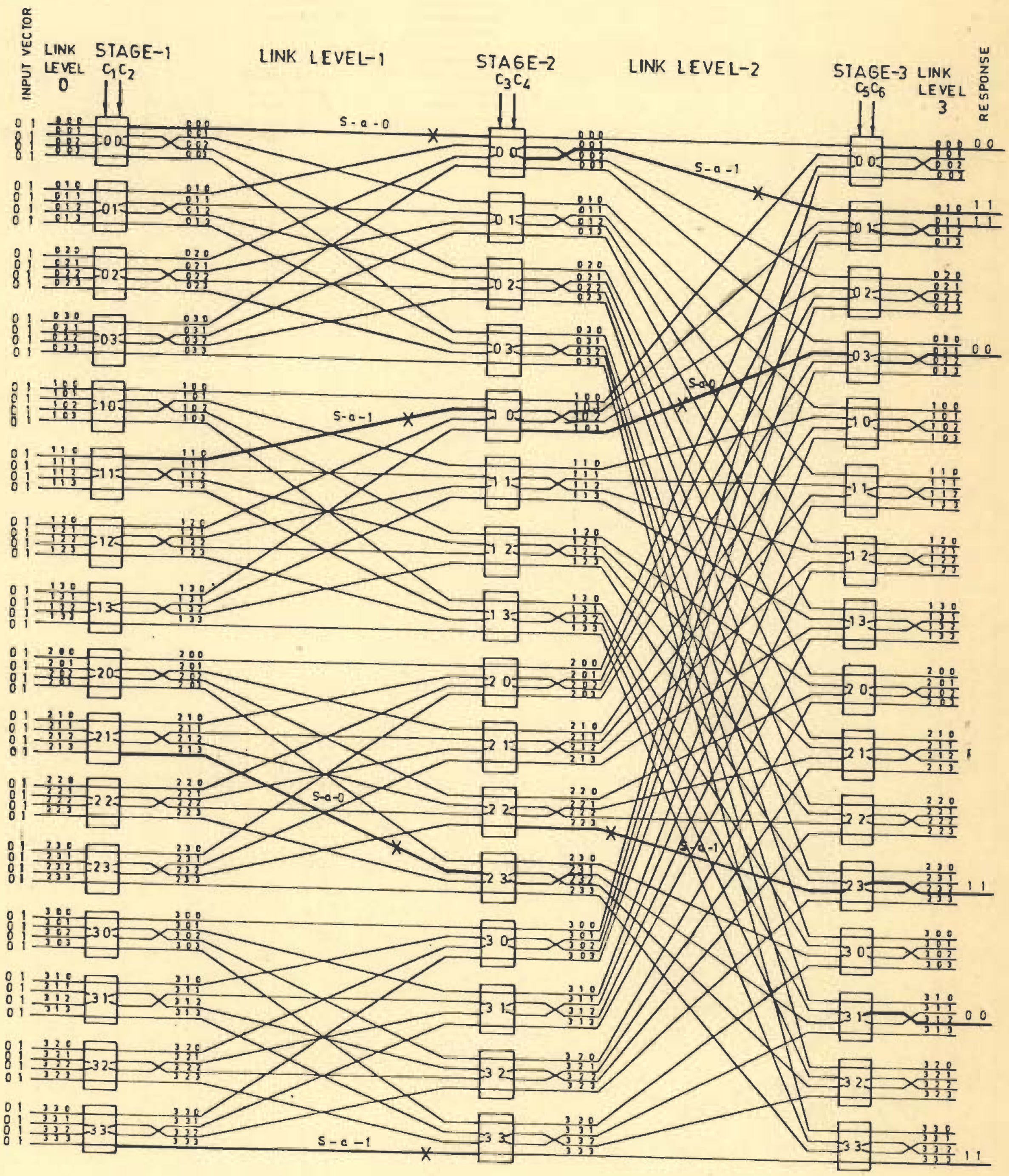


FIGURE 5.3 THREE STAGE DCMIN WITH ALL DIMSES IN MODE 0. FAULTY LINKS ARE SHOWN DARK. ALSO OUTPUT TERMINALS HAVING FAULTY RESPONSE ARE MARKED DARK.

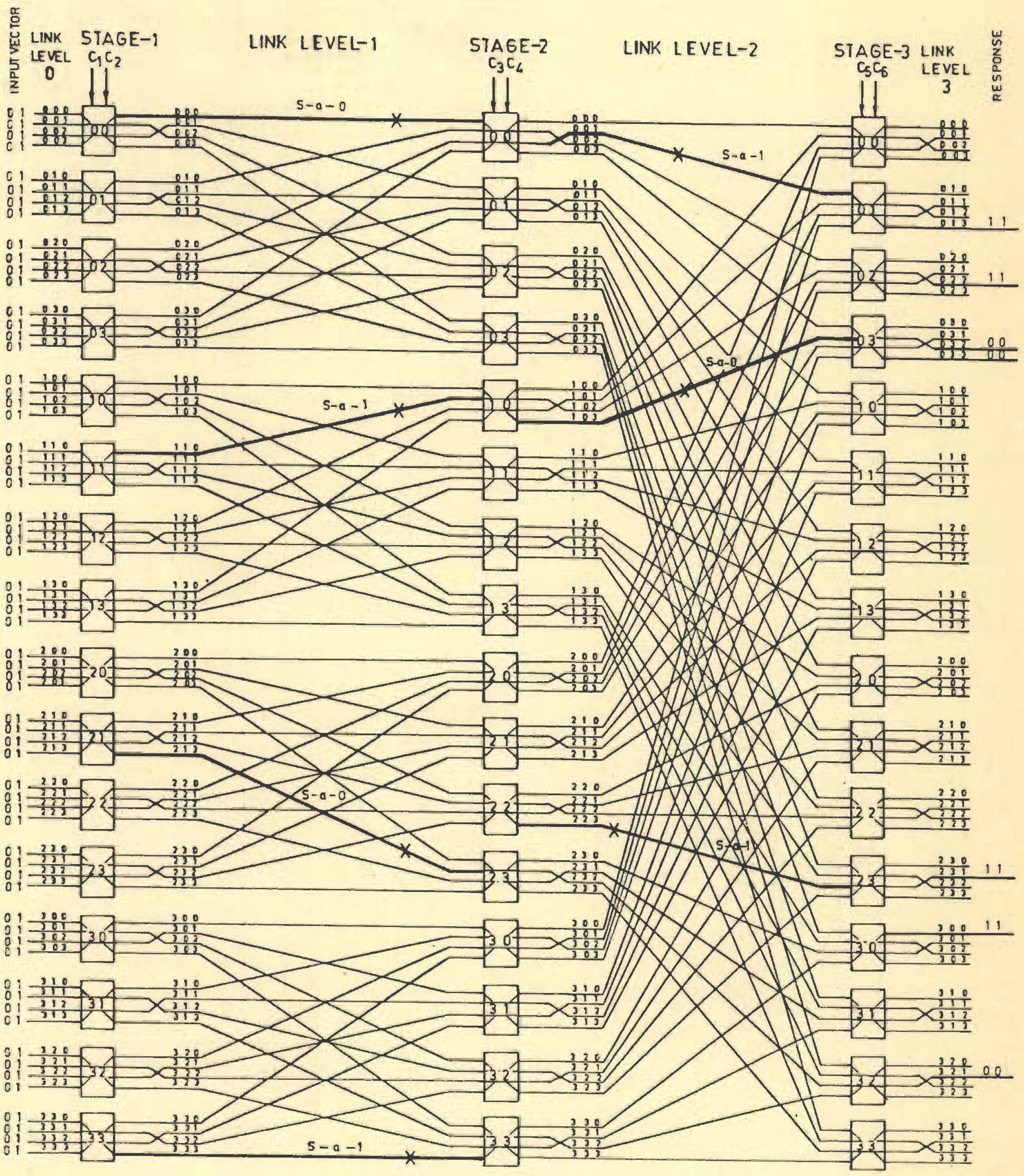


FIGURE 5.4 THREE STAGE DCMIN WITH ALL DIMSES IN MODE 3. FAULTY LINKS ARE SHOWN DARK. OUTPUT TERMINALS HAVING FAULTY RESPONSE ARE ENLARGED.

(i) Set all DIMSEs to mode 'O' operation as shown in Fig.5.3.

(ii) Apply the input vector 'V' such that a logical value 'O' is applied at all the input terminals.

In this case S-a-O faults will pass-off undetected but S-a-1 faults will be reflected on output terminals. For example S-a-1 fault of address OO1 in link level-2 is reflected at output address O1O in link level-3. Observe the response at all output terminals. Now complement the input vector. In this case S-a-O faults will be reflected at the output terminals. Again observe the response at output terminals. The addresses of the output terminals having identical response in both the cases, i.e., logical values 'OO' or '11' are recorded and the corresponding link addresses in link level-2 and 1 are derived as given below:

Output terminal addresses reflecting faults	Type of fault	Corresponding link addresses in	
		link level-2	link level-1
000	S - a - 0	$(000)_0$	$(000)_0$
010	S - a - 1	$(001)_1$	$(010)_1$
011	S - a - 1	$(101)_1$	$(110)_1$
031	S - a - 0	$(103)_0$	$(130)_0$
232	S - a - 1	$(223)_1$	$(232)_1$
312	S - a - 0	$(231)_0$	$(213)_0$
333	S - a - 1	$(333)_1$	$(333)_1$

(iii) In Phase II, the DIMSEs are set to mode '3' operation as shown in Fig.5.4.

(iv) Again two test vectors as above are applied sequentially. The addresses of the output terminals having identical response in both the cases are recorded and corresponding link addresses in link levels-2 and 1 are derived as given below:

Output terminal addresses reflecting faults	Type of fault	Corresponding addresses in link level-2	Corresponding addresses in link level-1
013	S - a - 1	$(001)_1$	$(021)_1$
022	S - a - 1	$(102)_1$	$(110)_1$
032	S - a - 0	$(103)_0$	$(100)_0$
033	S - a - 0	$(003)_0$	$(000)_0$
231	S - a - 1	$(223)_1$	$(202)_1$
300	S - a - 1	$(330)_1$	$(333)_1$
321	S - a - 0	$(232)_0$	$(213)_0$

(v) Addresses of links in the above two phases are compared and the common links and their fault types are listed as follows:

Addresses in link level-2	Type of fault
(001)	S - a - 1
(103)	S - a - 0
(223)	S - a - 1

Addresses in link level-1	Type of Fault
(000)	S - a - 0
(110)	S - a - 1
(213)	S - a - 0
(333)	S - a - 1

(vi) The above results so obtained tally with the assumed arbitrary faults in three stage DCMIN of Fig.5.3.

5.1.3 Diagnosis of Fault at Control Lines

A fault in one of the control lines of a DIMSE will constrain it to work in only two of the four valid modes. As two valid states are still operative, the detection of fault becomes complicated and there cannot be a general procedure to detect exact location of the fault. However, the faults will be either S-a-0 or S-a-1 type. The problem is simplified if instead of locating the control line, the corresponding DIMSE is located. For locating DIMSE having one of the control lines faulty, both the control lines of all the DIMSEs are short circuited. By doing so, under all circumstances, the DIMSEs will work either in mode '0' or mode '3' operation only.

Again, as stipulated previously two phases of the test procedure will be required. In Phase-I, all the DIMSEs are set to mode '0' condition to detect and locate stuck-at-1 fault in the control line of a DIMSE. In Phase-II, all the DIMSEs are set to mode '3' condition to detect and locate stuck-at-0 fault in the control line of a DIMSE. Thus, each phase will independently detect and locate the faulty element. Any single stuck-at-fault, if it is detected, will be reflected at four output terminals. The physical distance, in terms of number of terminals, (also called terminal distance) between the output terminals reflecting faulty outputs, can be directly used for locating the stage of the faulty element as described below.

Theorem 5.1

If the fault is reflected on the output terminals which are displaced at an interval of $N/4^i$ terminals distance, $1 \leq i \leq n$, then the faulty switching element (DIMSE) is located in the i^{th} stage.

Proof

Consider an n stage network. It consists of 4-units of (n-1)-stage networks connected through 4-shuffle permuter to n^{th} stage as was discussed in Section 4.3.

Any fault in a DIMSE of last (n^{th}) stage will be reflected in 4 output terminals which are displaced at 1 terminal ($= N/4^n$) distance apart.

Stage-(n-1) and stage-n are connected through 4-shuffle permuter. A fault in one of the DIMSEs in stage-(n-1) will thus be reflected in 4 output terminals of stage-n which will be 4 terminals ($= N/4^{n-1}$) distance apart.

Now each of the (n-1) stage network consists of 4 units of (n-2) stage networks and $(n-1)^{th}$ stage as the last stage. The fault in any DIMSE at stage-(n-2) will be reflccted in four terminals at the output of $(n-1)^{th}$ stage which will be 4 terminals distance apart. If these are mapped on the output of n^{th} stage, the fault will be reflected by the terminals which are displaced $16(= N/4^{n-2})$ terminals distance apart. Because of the recursive nature of the network it can be said that in general a fault in a DIMSE of any stage-i will be reflected in 4 terminals at the output of n^{th} stage which are displaced at an interval of $N/4^{i}$ terminals distance.

$\square$

Theorem 5.1 is an important conclusion for the situations where stages are integrated on a single chip and the presence of any fault needs the replacement of the chip as a whole.

The stepwise procedure for diagnosis of single fault in control lines is as follows:

<u>PHASE I (For S-a-1 type faults)</u>

Step 1: Set all DIMSEs to mode '0' condition.

Step 2: Apply the input test vector V = (0101, 1010,... ..., 0101) such that the four input terminals of each alternative DIMSE in the network get inputs as

v = (0101) or v' = (1010).

Step 3: Compare the actual response at output terminals with the expected values. Record the physical addresses of the output terminals having faulty (complementary) outputs.

Step 4: Find the distance 'A' amongst the addresses recorded in Step 3. A is to be found in terms of number of terminals. From this calculate the stage in which the faulty DIMSE is located (Theorem 5.1) as:

$$\text{Stage number 'i'} = \log_4(N/A).$$

Step 5: Derive the physical location of the faulty DIMSE in stage-i from any one of the four physical addresses recorded in Step 3, as described in Section 4.5.

PHASE II (For S-a-0 type faults)

Step 6: Set all DIMSEs to mode '3' condition.

Step 7: Repeat Steps 2 to 5.

Example 5.2

Fig.5.5 represents a 3-stage DCMIN. Let one of the control lines of the DIMSE at physical location (13) in stage-2 be S-a-1.

Since the fault assumed is S-a-1 type, it can be detected and located in Phase I. Phase II will not detect this fault.

Now all the DIMSEs in Fig.5.5 are set to operate in mode '0' condition. However, the DIMSE (13) of stage-2 will remain in mode '3'.

Applying the input vector as shown in Fig.5.5, gives the faulty response in output terminals at physical locations

(301), (311), (321), (331).

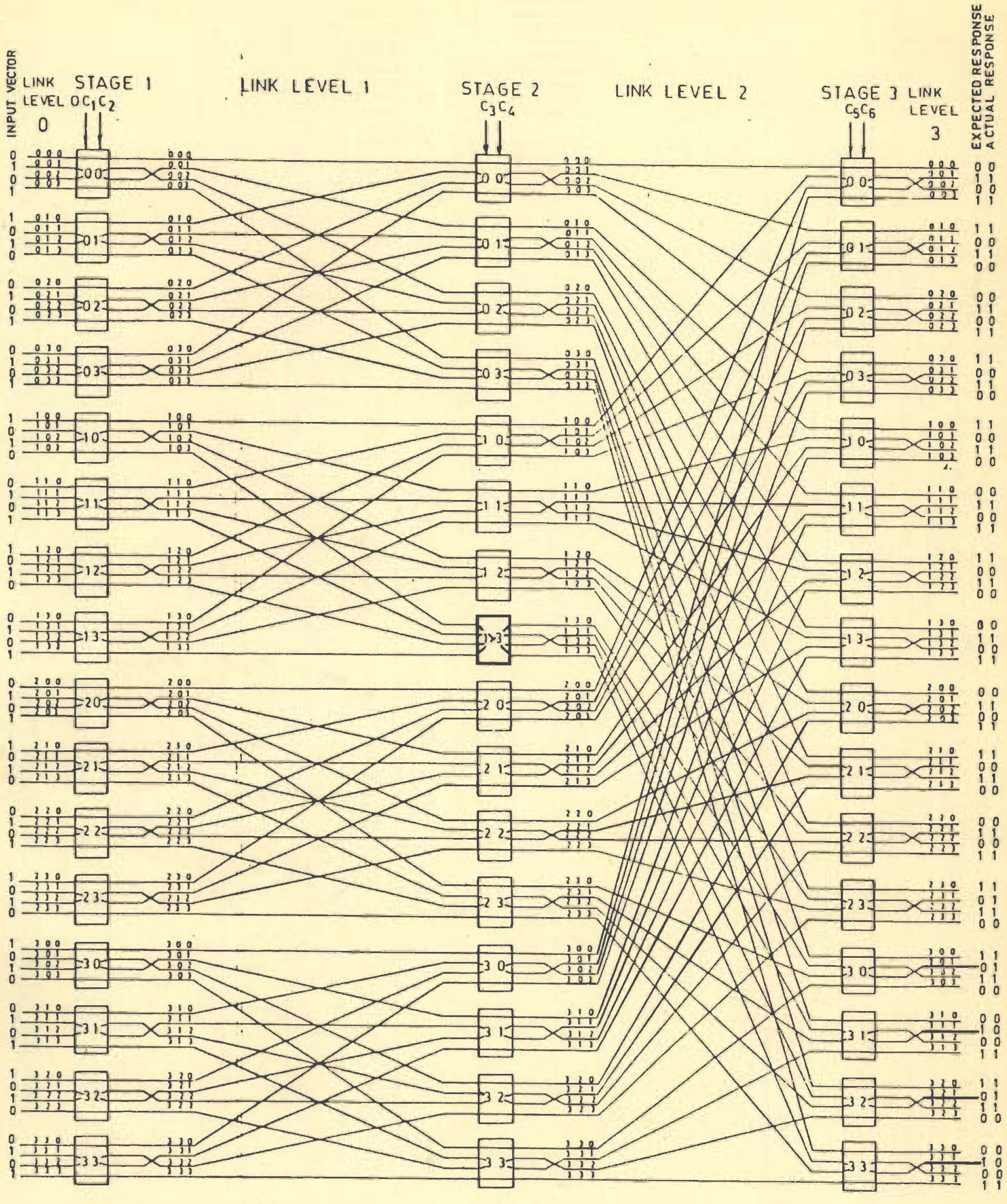


FIG. 5.5 THREE STAGE DCMIN WITH ALL DIMSES SET IN MODE 0. DIMSE (13) OF STAGE-2 HAS ITS CONTROL LINE FAULTY (S-a-1). TERMINAL HAVING FAULTY RESPONSE ARE ENLARGED.

Clearly,these four terminals are separated by a distance A = 4 terminals. Hence the stage 'i' in which faulty DIMSE is located is given as

$$i = \log_4(\frac{N}{A}) = \log_4(\frac{64}{4}) = \log_4 16 = 2$$

Again to find the address of the faulty DIMSE in this stage, the address of the link carrying the data corresponding to any one of the four output terminals, reflecting the fault, is calculated. For example, link address in stage-2, corresponding to the output terminal(301) is (130) as discussed in Section 4.5.

If the Least Significant Digit (LSD) is dropped then the DIMSE address in stage-2 will be (13).

5.1.4 Multiple Faults

Multiple faults, as long as they do not mask each other (non overlapping), can be detected by the procedures given earlier. However, to achieve conclusive results, two testing phases as outlined in the preceding sections are not adequate. For overlapping multiple faults each stage is tested separately. In stagewise testing procedure, the stages are tested in sequence starting from either stage-1 or stage-n. The input test-vector is selected such that while testing stage-i, the faults in stages-(i+1) to n, if sequence starts from stage-1 (the faults in stages-1 to (i-1), if the sequence starts from stage-n) become transparent and do not affect testing of stage-i. A method for diagnosis of multiple control line stuck at faults has been proposed. In the proposed method, testing sequence starts from stage-n. The procedure to test any stage 'i', $1 \leq i \leq n$, is described in the following steps:

Step 1: Divide the input terminals into blocks containing (4^{i-1}) terminals in each block and number these blocks as 0,1,2,...etc. from top to bottom.

Step 2: Divide the output terminals into groups containing (4^{n-i}) terminals in each group, where n is the number of stages given by $n = \log_4 N$. Index these groups as (0,1,2,3) from top to bottom using modulo-4 arithmatic.

PHASE I (For S-a-1 faults)

Step 3: Set all DIMSEs to mode '0' condition.

Step 4: Apply input test vector V = (0,0,0,0;1,1,1,1; 0,0,0,0;...;1,1,1,1) such that each terminal in even numbered input-block gets a '0' logical value and each terminal in odd numbered input-block gets a '1' logical value.

Step 5: Compare the actual response with the expected values and record the physical addresses of the output terminals having faulty outputs, in groups having index number 0 only.

Step 6: Derive the physical location of the faulty DIMSEs in stage-i from the physical addresses of the output terminals recorded in Step 5.

<u>PHASE II (For S-a-O faults)</u>

Step 7: Set all DIMSEs in mode '3' condition

Step 8: Repeat Steps 4 to 6 of Phase I above.

Example 5.3

Fig.5.6 is a 3-stage DCMIN of size N = 64. Let the DIMSEs at physical locations (01), (10), (12), (22) and (33) in stage-2 have their control lines S-a-1.

This example, for N = 64, n = 3 and i = 2 is carried out in Table 5.1 as explained below:

(i) The input terminals are divided into $(64/4^{i-1}) = 16$ blocks where each block contains 4 consecutive terminals. The blocks are numbered from 0 to 15 as shown in column 3 of the Table 5.1.

(ii) The 64 output terminals are divided into $(64/4^{n-i}) = 16$ groups, each group containing 4 consecutive terminals. The groups are indexed as 0,1,2,3 using modulo 4 arithmatic as is depicted in column 4 of the Table 5.1.

(iii) Now all the DIMSEs are set to mode '0' condition and the input vector is applied as shown in Fig.5.6. The response is compared with the expected values for terminals falling in groups with index number 0. This is shown in columns 6 and 7 of the Table 5.1.

(iv) The terminals having faulty response are recorded in column 8. Column 9 of the table gives the physical location of the faulty DIMSEs in stage-2 of Fig.5.6.

1	2	3	4	5	6	7	8	9
SERIAL NUMBER OF THE INPUT/OUTPUT TERMINALS	PHYSICAL ADDRESSES OF INPUT/OUTPUT TERMINALS	STEP 1 (BLOCK NUMBERS)	STEP 2 INDEX NUMBERS OF OUTPUT GROUPS	INPUT VECTOR TO BE APPLIED AT INPUT TERMINALS	EXPECTED VALUES OF THE OUTPUTS	ACTUAL OUTPUTS ON OUTPUT TERMINALS HAVING INDEX VALUE 0	ADDRESSES OF THE TERMINALS HAVING FAULTY OUTPUTS	PHYSICAL LOCATION OF THE FAULTY DIMSE IN STAGE-2
0	0 0 0	0 (E)	0	0	0	0		
1	0 0 1			0	0	1	0 0 1	1 0
2	0 0 2			0	0	0		
3	0 0 3			0	0	0		
4	0 1 0	1 (O)	1	1	1			
5	0 1 1			1	1			
6	0 1 2			1	1			
7	0 1 3			1	1			
8	0 2 0	2 (E)	2	0	0			
9	0 2 1			0	0			
10	0 2 2			0	0			
11	0 2 3			0	0			
12	0 3 0	3 (O)	3	1	1			
13	0 3 1			1	1			
14	0 3 2			1	1			
15	0 3 3			1	1			
16	1 0 0	4 (E)	0	0	0	1	1 0 0	0 1
17	1 0 1			0	0	0		
18	1 0 2			0	0	0		
19	1 0 3			0	0	0		
20	1 1 0	5 (O)	1	1	1			
21	1 1 1			1	1			
22	1 1 2			1	1			
23	1 1 3			1	1			
24	1 2 0	6 (E)	2	0	0			
25	1 2 1			0	0			
26	1 2 2			0	0			
27	1 2 3			0	0			
28	1 3 0	7 (O)	3	1	1			
29	1 3 1			1	1			
30	1 3 2			1	1			
31	1 3 3			1	1			
32	2 0 0	8 (E)	0	0	0	0		
33	2 0 1			0	0	1	2 0 1	1 2
34	2 0 2			0	0	1	2 0 2	2 2
35	2 0 3			0	0	0		
36	2 1 0	9 (O)	1	1	1			
37	2 1 1			1	1			
38	2 1 2			1	1			
39	2 1 3			1	1			
40	2 2 0	10 (E)	2	0	0			
41	2 2 1			0	0			
42	2 2 2			0	0			
43	2 2 3			0	0			
44	2 3 0	11 (O)	3	1	1			
45	2 3 1			1	1			
46	2 3 2			1	1			
47	2 3 3			1	1			
48	3 0 0	12 E	0	0	0	0		
49	3 0 1			0	0	0		
50	3 0 2			0	0	0		
51	3 0 3			0	0	1	3 0 3	3 3
52	3 1 0	13 O	1	1	1			
53	3 1 1			1	1			
54	3 1 2			1	1			
55	3 1 3			1	1			
56	3 2 0	14 E	2	0	0			
57	3 2 1			0	0			
58	3 2 2			0	0			
59	3 2 3			0	0			
60	3 3 0	15 O	3	1	1			
61	3 3 1			1	1			
62	3 3 2			1	1			
63	3 3 3			1	1			

TABLE 5.1 CALCULATION FOR FINDING FAULTY DIMSEs IN EXAMPLE 5.3

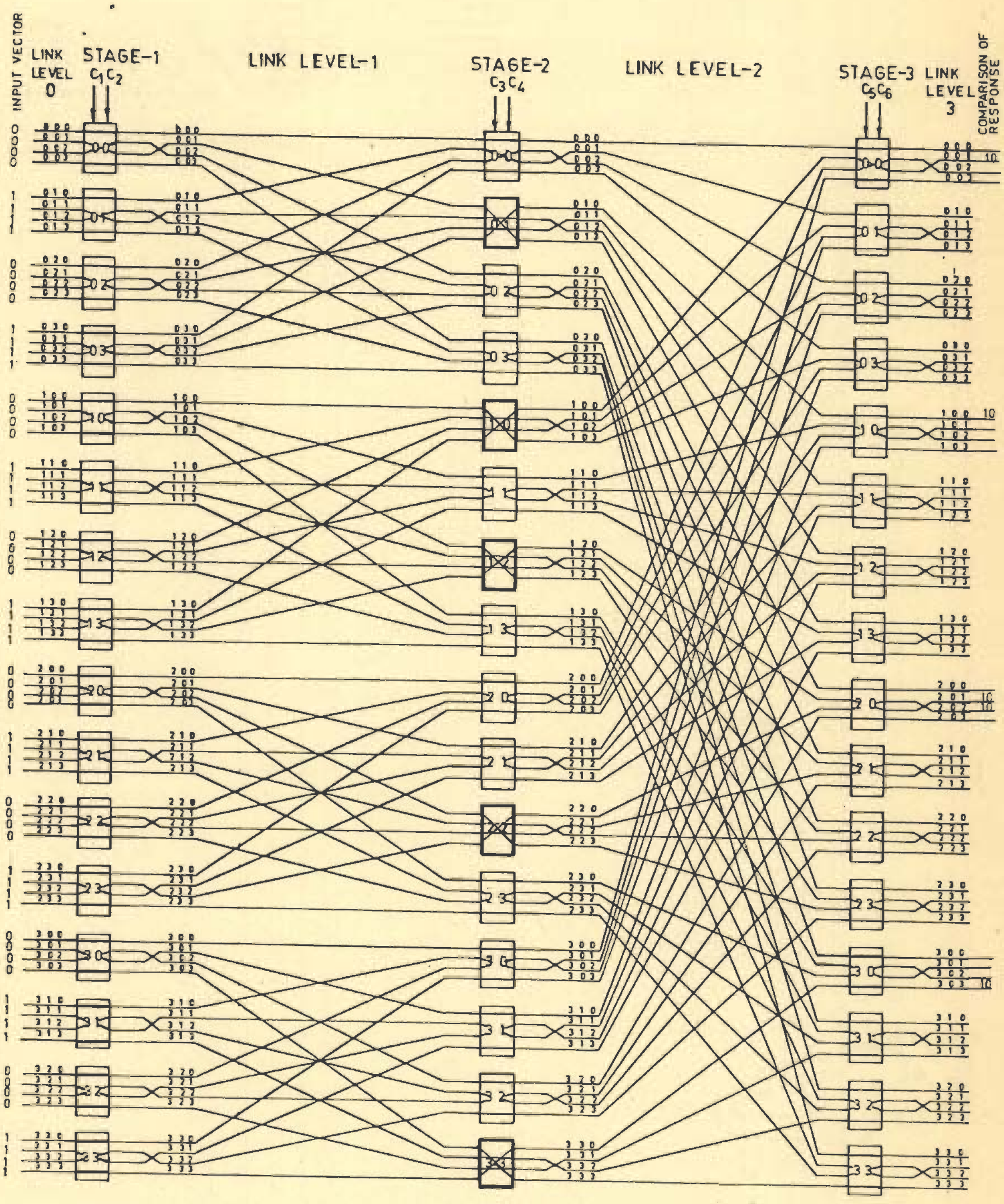


FIGURE 5.6 THREE STAGE DCMIN IN MODE 0. DIMSEs SHOWN DARK HAVE S-a-1 FAULT IN THEIR CONTROL LINES. OUTPUT TERMINALS HAVING INDEX 0 ARE ENLARGED. TERMINALS HAVING FAULTY OUTPUTS ARE MARKED WITH COMPELMENTARY RESPONSE.

5.2 FAULT-TOLERANCE OF DCMIN

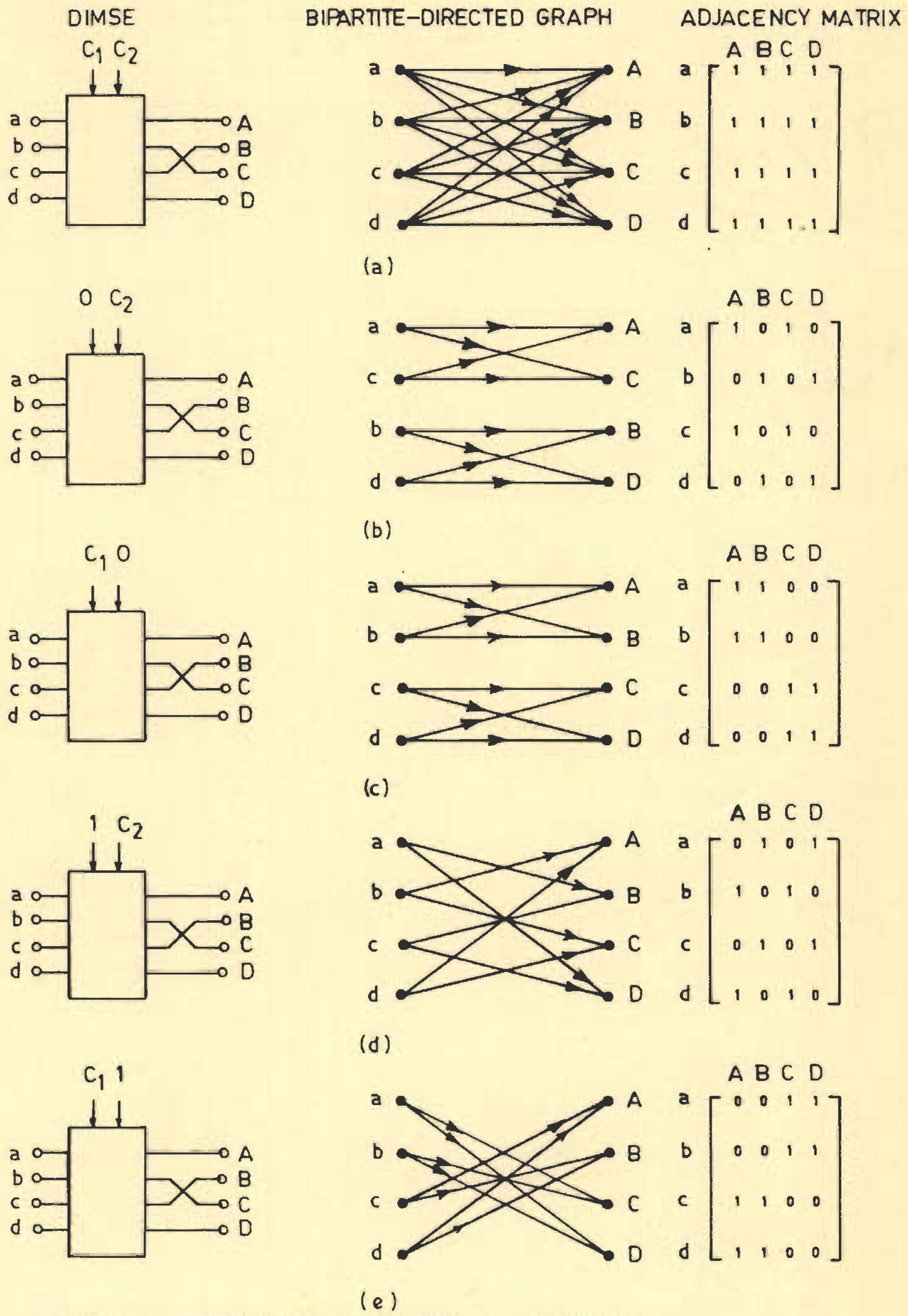
The full access property of DCMIN has been illustrated in Section 4.4. In this section, it is discussed as to how the dynamic full access capability can be incorporated into DCMIN. For this purpose a graph model has been developed. The graph model, based upon the connectivity concepts [43] is used for the analysis purpose.

A DCMIN essentially consists of various stages of DIMSEs interconnected through 4-shuffle link patterns. Therefore, it is appropriate to develop the graph model of DIMSEs as the basic block and then use it for evaluating the DFA property of DCMIN. Additionally, it has been shown as to how fault-tolerance can be achieved for the purpose of DFA through multiple passes.

5.2.1 Graph Model of DIMSE

A general graph model of DIMSE based on the connectivity property is depicted in Fig.5.7. Presuming that faults that can occur on the control lines are of stuck type, the modified graph models for various faults on control lines are shown in Figs.5.7(b-e). However, if both the control lines C_1 and C_2 have simultaneous faults, then the graph models will be modified as shown in Figs.5.8(a-d). The adjacency matrices, as will be defined in Section 5.2.2, of each graph model of Figs.5.7 and 5.8 are also shown alongside the graphs.

The graph models for stuck-at-faults on input/output lines of DIMSEs are shown in Fig.5.9. The modelling in these cases is on a different pattern. Here, if a link becomes faulty



(a)
$$\begin{array}{c|cccc} & A & B & C & D \\ \hline a & 1 & 1 & 1 & 1 \\ b & 1 & 1 & 1 & 1 \\ c & 1 & 1 & 1 & 1 \\ d & 1 & 1 & 1 & 1 \end{array}$$

(b)
$$\begin{array}{c|cccc} & A & B & C & D \\ \hline a & 1 & 0 & 1 & 0 \\ b & 0 & 1 & 0 & 1 \\ c & 1 & 0 & 1 & 0 \\ d & 0 & 1 & 0 & 1 \end{array}$$

(c)
$$\begin{array}{c|cccc} & A & B & C & D \\ \hline a & 1 & 1 & 0 & 0 \\ b & 1 & 1 & 0 & 0 \\ c & 0 & 0 & 1 & 1 \\ d & 0 & 0 & 1 & 1 \end{array}$$

(d)
$$\begin{array}{c|cccc} & A & B & C & D \\ \hline a & 0 & 1 & 0 & 1 \\ b & 1 & 0 & 1 & 0 \\ c & 0 & 1 & 0 & 1 \\ d & 1 & 0 & 1 & 0 \end{array}$$

(e)
$$\begin{array}{c|cccc} & A & B & C & D \\ \hline a & 0 & 0 & 1 & 1 \\ b & 0 & 0 & 1 & 1 \\ c & 1 & 1 & 0 & 0 \\ d & 1 & 1 & 0 & 0 \end{array}$$

FIGURE 5.7 GRAPH MODEL OF DIMSE (a) Fault free (b) C_1 S-a-o, (c) C_2 S-a-o, (d) C_1 S-a-1 (e) C_2 S-a-1

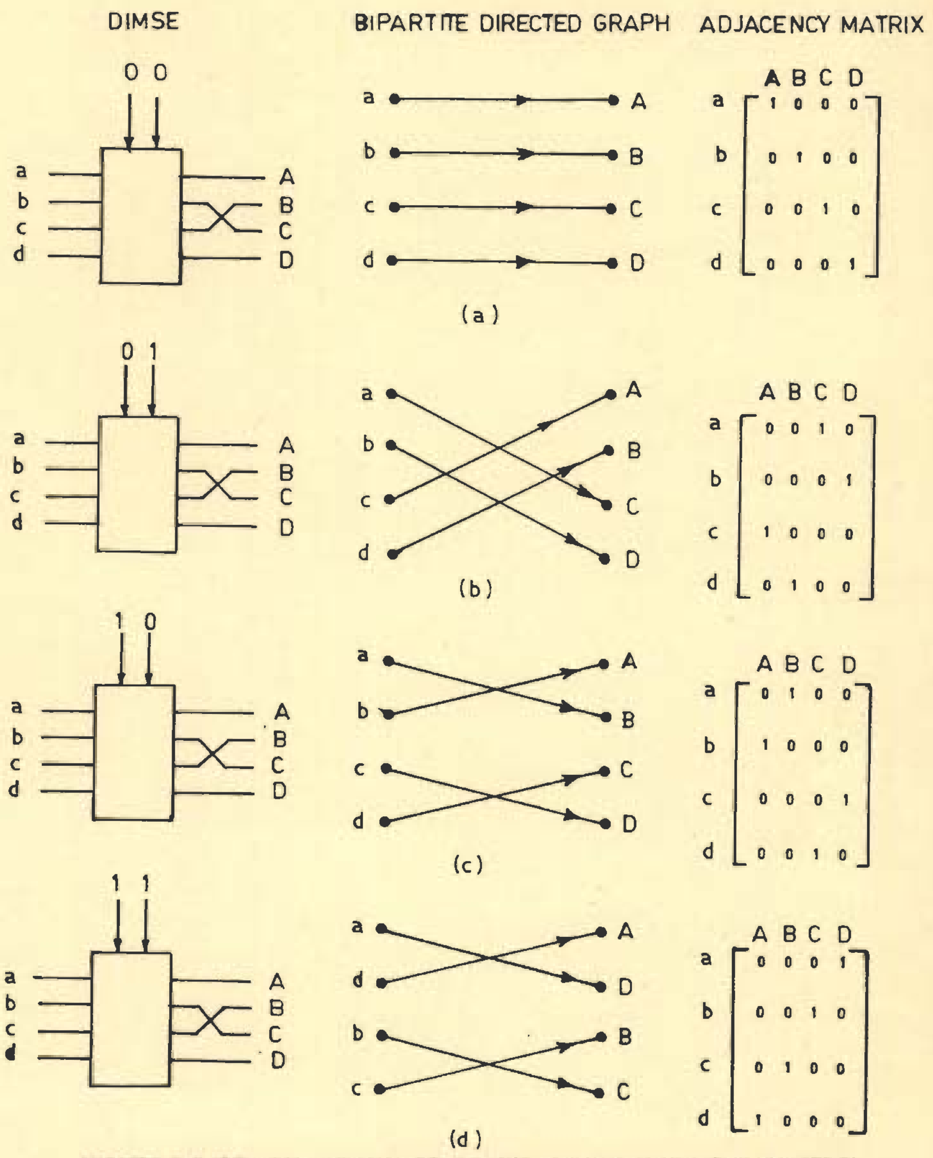
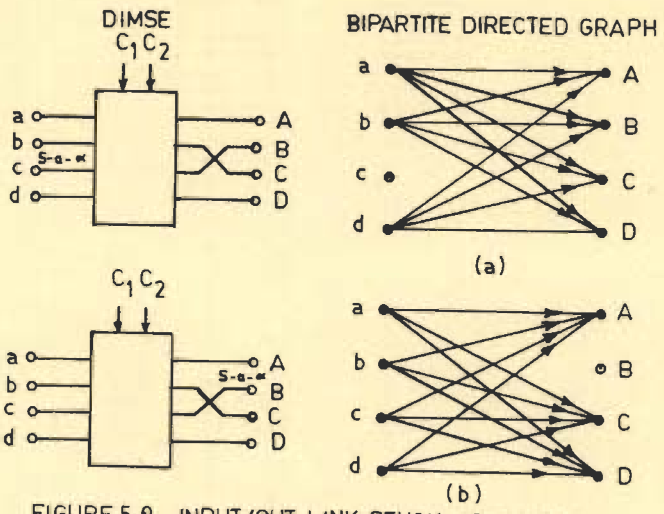


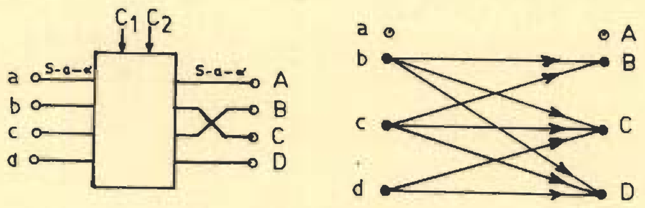
FIGURE 5.8 GRAPH MODEL OF DIMSE. WHEN BOTH THE CONTROL LINES ARE STUCK-AT-ALPHA FAULTS
(a) C_1,C_2 S-a-o, (b) C_1 S-a-o, C_2 S-a-1
(c) C_1 S-a-1, C_2 S-a-o (d) C_1 S-a-1, C_2 S-a-1



$$\begin{array}{c|cccc} & A & B & C & D \\ \hline a & 1 & 1 & 1 & 1 \\ b & 1 & 1 & 1 & 1 \\ c & 0 & 0 & 0 & 0 \\ d & 1 & 1 & 1 & 1 \end{array} \qquad (a)$$

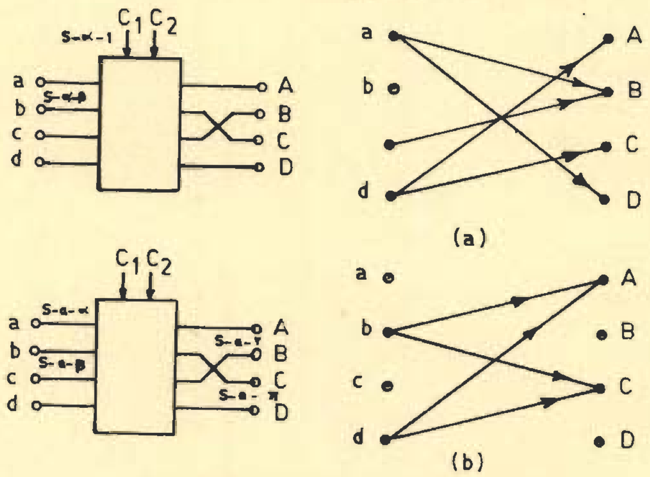
$$\begin{array}{c|cccc} & A & B & C & D \\ \hline a & 1 & 0 & 1 & 1 \\ b & 1 & 0 & 1 & 1 \\ c & 1 & 0 & 1 & 1 \\ d & 1 & 0 & 1 & 1 \end{array} \qquad (b)$$

FIGURE 5.9 INPUT/OUT LINK STUCK-AT-ALPHA
(a) Input link C S-a-α (b) Output link B S-a-α



$$\begin{array}{c|cccc} & A & B & C & D \\ \hline a & 0 & 0 & 0 & 0 \\ b & 0 & 1 & 1 & 1 \\ c & 0 & 1 & 1 & 1 \\ d & 0 & 1 & 1 & 1 \end{array}$$

FIGURE 5.10 MULTIPLE LINK FAULTS



$$\begin{array}{c|cccc} & A & B & C & D \\ \hline a & 0 & 1 & 0 & 1 \\ b & 0 & 0 & 0 & 0 \\ c & 0 & 1 & 0 & 1 \\ d & 1 & 0 & 1 & 0 \end{array} \qquad (a)$$

$$\begin{array}{c|cccc} & A & B & C & D \\ \hline a & 0 & 0 & 0 & 0 \\ b & 1 & 0 & 1 & 0 \\ c & 0 & 0 & 0 & 0 \\ d & 1 & 0 & 1 & 0 \end{array} \qquad (b)$$

FIGURE 5.11 MIXED MULTIPLE FAULTS
(a) Control C_1 S-a-1 and input b S-a-β
(b) Input links a S-a-α, C S-a-β and Output links B S-a-γ, D S-a-π

(S-a-O or S-a-1) then it becomes a stuck link and cannot be used for transmitting any data. This is reflected in the graph model through the removal of the corresponding node and hence eliminating all the edges adjacent to the node. For example in Fig.5.9(a), input link 'C' is stuck at a fault. Accordingly, all paths emanating from it, and through DIMSE have been removed as shown in the corresponding graph model.

In case multiple faults are also considered, they can occur in a variety of combinations, i.e., either on the inputs, on the outputs, on the control lines or a combination of the above. Each multiple fault situation will present a different picture and accordingly modify the graph. For the sake of illustration, Figs.5.9-5.11 depict occurrances of multiple faults and their respective graph models.

5.2.2 Graph Model of DCMIN and Its DFA Capabilities

The graph models of DIMSEs under various fault conditions form the basis for evaluating the DFA property of DCMIN. For achieving this, adjacency and reachability matrix concepts are required. These matrices are derived from the graph model and are described in brief hereunder to maintain the continuity and readability:

(i) Adjacency Matrix

The adjacency matrix 'A' of a graph is the NxN matrix $[a_{pr}]$ with $a_{pr} = 1$ if there is an edge from node p to node r in the graph, otherwise $a_{pr} = 0$.

The adjacency matrices of DIMSE under various fault conditions are shown alongside the graphs in Figs.5.7-5.11. Fig.5.7(a) gives the adjacency matrix of fault-free DIMSE. It is seen that all elements of the adjacency matrix have the logical value '1' indicating that the fault-free DIMSE is a full-access network. Adjacency matrices in Figs.5.7(b)-5.11(b) do not have all their elements at logical value '1' indicating that the full-access property cannot be achieved in a single pass.

For DCMIN a stage will require large number of DIMSEs depending upon the size N of the network. In an n-stage DCMIN, let A_i be the adjacency matrix of the bipartite graph of the i^{th} stage $(1 \leq i \leq n)$ representing its connectivity, then R, the Reachability matrix from input nodes to output nodes of the DCMIN is defined as

$$R = \prod_{i=1}^{n} A_i$$

Observation of the reachability matrix then reveals, whether the network has full-access property or not. For full access all the elements of R should have logical value '1'. If R does not show the full-access capability of the network, then R is tested for DFA capability through multiple passes.

If multiple passes are allowed then reachability in m-passes could be given as

$$R_m = (R)^m$$

The DCMIN will have DFA in m-passes, if all the elements of R_m are at logical value '1'.

For a single stage network, the reachability matrix R is same as adjacency matrix A. In all the figures representing adjacency or reachability matrices, rows represent inputs and columns the outputs.

Following assumptions are made for the study of DFA property for the proposed DCMIN:

(i) Terminal links are fault free. This is a valid assumption, because the failure of a terminal link implies that the associated processor cannot be connected to the network either to receive or to transmit data.

(ii) Multiple faults are allowed on different DIMSEs only and any single DIMSE cannot have multiple faults. This is also a valid assumption because the diagnostic programmes in multiprocessor systems involving MINs are run at regular intervals to detect the occurrance of faults. Accordingly, the following types of faults are the valid faults in any MIN environment and hence for DCMIN as well.

(a) One of the input links is stuck at α, $\alpha \in (0,1)$

(b) One of the output links is stuck at α, $\alpha \in (0,1)$

(c) One of the control lines is stuck at α, $\alpha \in (0,1)$

Theorem 5.2

In a fault-free DCMIN, connecting N inputs to N outputs, any of the outputs can be accessed from any one of the input terminals in a single pass only.

Proof

The proof is by induction. It has been shown in Fig.5.7(a) that the adjacency matrix of a fault-free DIMSE has all its elements at logical value '1'. DIMSE is a single stage 4x4 network M_1. Hence, its adjacency and reachability matrices are identical.

The DCMIN M_2 of size 16x16 was shown in Fig.4.9. It has 4 DIMSEs in each of the two stages. 4 inputs of each of the DIMSE of second stage connect 4 different DIMSEs of first stage. Thus, under fault-free conditions any one of the outputs can be accessed by any one of the inputs in a single pass through the DIMSEs of the first and second stages.

Again consider a three stage DCMIN M_3 as shown in Fig.4.10 First two stages are 4 independent blocks of 16x16 (M_2) networks with full-access property under fault-free conditions as explained above. Each DIMSE of third stage has 4 inputs. Each of the 4 inputs to any DIMSE in third stage is connected to a different M_2 block. Since there are only 4 M_2 blocks, with full-access property, any of the input node can access the 4 outputs of the DIMSE of third stage to which it is connected. Since, 4 inputs of each DIMSE of third stage connect the 4 different M_2 blocks, hence any arbitrary input node can access any arbitrary output node in a single pass.

In corollary 4.1, it has been demonstrated that the topology of DCMIN is recursive in nature. Thus the above analogy can be extended to a DCMIN of any size NxN.

Hence, it is proved that in a fault-free DCMIN of size NxN, any of the outputs can be accessed from any one of the input terminal in a single pass only.

$\square$

Theorem 5.3

If a control line of DIMSE is S-a-α, then each input node can have access to at least two of the output nodes.

Proof

If a DIMSE is fault free, any input node can access any one of the 4 output nodes by changing the control modes to O(OO), 1(O1), 2(1O) or 3 (11). If one of the control signals is stuck at α, the DIMSE can have only two working control modes and hence an input can access the two output nodes.

$\square$

Theorem 5.4

To maintain DFA capability in DCMIN at least one of the stages (either input or output) should be free from control line faults.

Proof

From Figs.5.7(b) and 5.7(c), it can be seen that a stuck-at-O fault on a control line divides the bipartite directed graph into two unconnected subgraphs. Once a set of nodes becomes unconnected it is not possible to attain the connectivity property under any conditions. If a network is having all the switches stuck at control line faults, there is possibility that the network may divide itself into two or more unconnected subnetworks. However, if one of the stages is fault-free, then the graph cannot be divided into unconnected subgraphs which is the necessary condition for maintaining the DFA property.

▱

Example 5.3

Figure 5.12 shows two stage DCMIN where one of the control lines of each DIMSE is stuck-at-0. Fig.5.13 gives its graph model, which is redrawn in Fig.5.14 to show that the graph is divided into three unconnected subgraphs.

Theorem 5.5

In a two stage DCMIN if some or all DIMSEs of any one stage have a single stuck type control line fault then DFA is achieved in a maximum of two passes.

Proof

The graph models and adjacency matrices of a DIMSE with one of the control lines stuck-at-α are shown in Fig.5.7. From adjacency matrices it can be observed that any input node of faulty DIMSE can access at least two of its four output nodes.

Let the faulty DIMSEs be in stage-1. Each of the output nodes of any DIMSE in stage-1 is connected to a different DIMSE in stage-2, and can access all the four output nodes of the

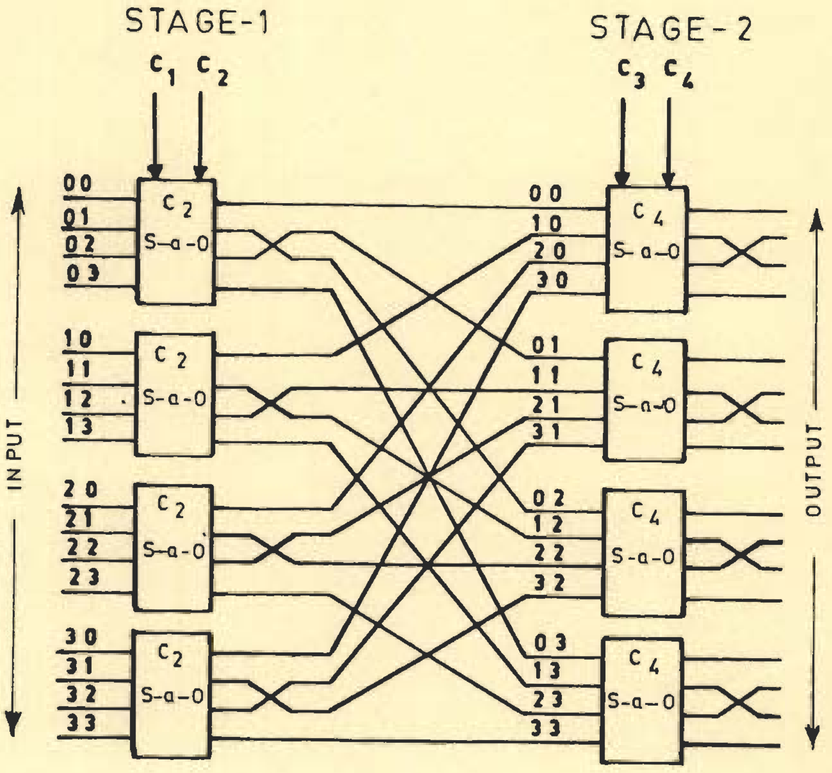
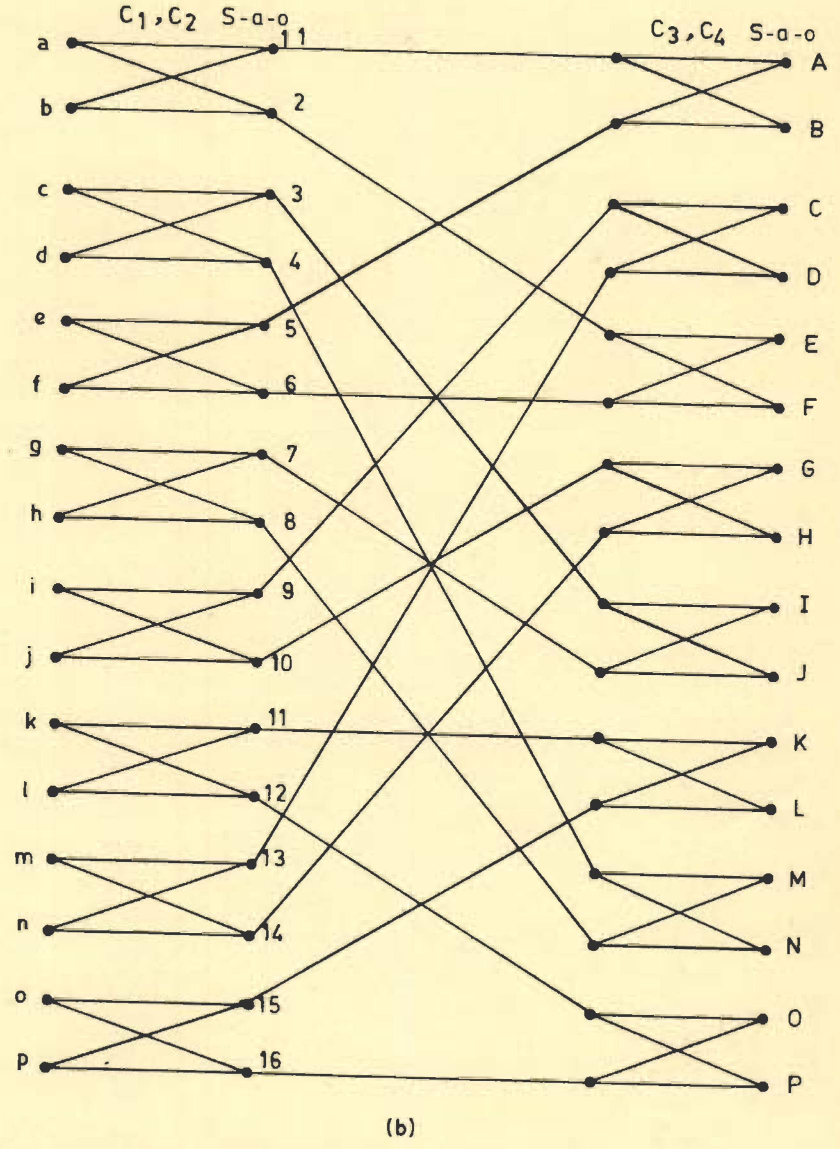


FIGURE 5.12 TWO STAGE DCMIN WITH EACH DIMSE HAVING ONE OF ITS CONTROL LINE STUCK-AT-0



(b)

FIGURE 5.13 GRAPH MODEL OF 16 X 16 I C N WHEN ALL THE ELEMENTS HAVE SINGLE STUCK AT CONTROL LINE FAULT

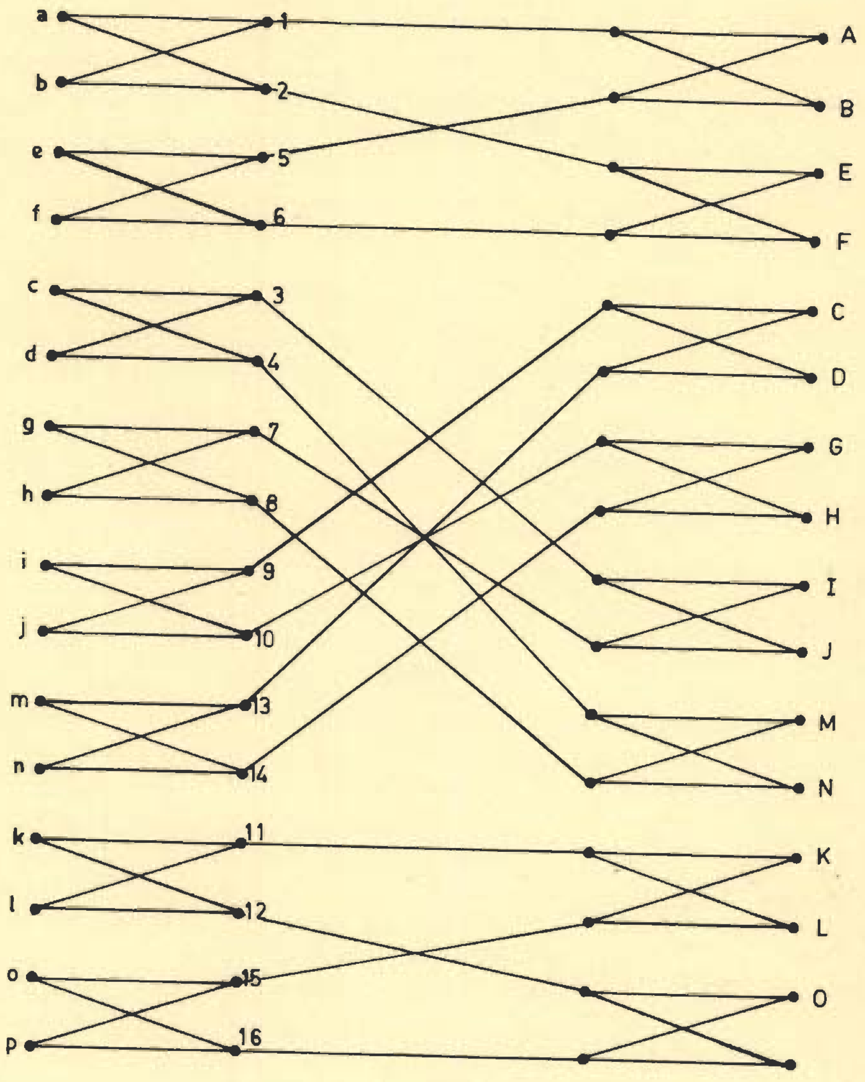


FIGURE 5.14 FIG. 5.13 REDRAWN INDICATING THAT ICN HAS LOST THE CONNECTIVITY PROPERTY. GRAPH IS SUB-DIVIDED INTO THREE UNCONNECTED SUB-GRAPHS.

corresponding DIMSE. Thus any input node of stage-1 can access 8 of the output nodes of stage-2. Hence, it becomes obvious that in the second pass, it can access all the 16 output nodes of stage-2.

Similarly,let the faulty DIMSEs be in stage-2. Now any input node can access all the 4 DIMSEs of stage-2. Since the faulty DIMSEs are in stage-2, at least two of its output nodes can be accessed by any of its input nodes. Thus, 8 of the output nodes of stage-2 can be accessed by any input node of stage-1. In second pass it can access all the 16 output nodes. Hence the DFA is achieved in two passes only.

Example 5.4

Figure 5.15 depicts a two stage DCMIN with one of the control lines of each DIMSE in stage-1 having S-a-α fault. Figs.* F1,F2 give the adjacency matrices of stage-1 and stage-2. F3 gives its reachability matrix. Since all elements of matrix A3 are not at logical value '1', multiple passes will be required to achieve DFA.

Figure F4 depicts matrix $R_2 = R^2$ with all its elements at logical value '1'. Hence the DCMIN of Fig.5.15, with indicated faults, will achieve DFA in a maximum of two passes.

Theorem 5.6

In an NxN DCMIN, where $N = 4^n$, for $n \geq 3$, if some or all DIMSEs (except those of last stage or first stage) have single

* Figure numbers starting with letter F are enclosed as appendix.

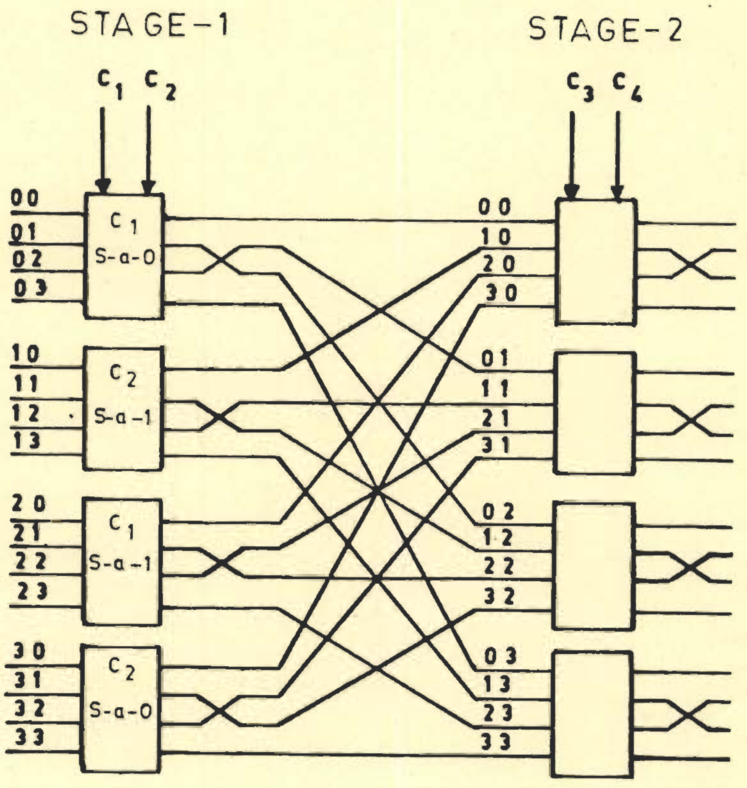


FIGURE 5.15 TWO STAGE DCMIN WITH CONTROL LINE FAULTS AS INDICATED ON DIMSEs OF STAGE-1

stuck-at-α faults in their control lines, then the network will retain its DFA property in a maximum of three passes.

Proof

Assume that the last stage is fault-free. Any input node i, $0 \leq i \leq (N-1)$, can access at least 2 links in the first level through the faulty switch of first stage. These two links will be able to access two DIMSEs (2^{2-1}) of the second stage, which can further access 4 DIMSEs (2^{3-1}) of the third stage and so on. Thus 2^{n-1} DIMSEs of the last fault-free stage i.e. n^{th} stage can be accessed in the first pass.

Now outputs of these 2^{n-1} DIMSEs of the last stage are fed back for second pass. Thus in the second pass 2^{n-1} DIMSEs of the first stage become accessible, which can further access at least 2^{n-1} DIMSEs of the second stage. Now proceeding as in the first pass, 2.2^{n-1} DIMSEs of third stage, $2^2.2^{n-1}$ of fourth stage etc. will become accessible. Thus in the second pass $(2^{n-2}).(2^{n-1})$ DIMSEs can be accessed.

Similarly, at the end of the third pass one can access $(2^{n-2}).(2^{n-2}).(2^{n-1})$ DIMSEs of the last stage.

Now $\quad 2^{n-2}.2^{n-2}.2^{n-1} = 2^{n-3}.4^{n-1} \geq 4^{n-1}$ for $n \geq 3$.

4^{n-1} is the number of DIMSEs in any one stage. Since the last stage is assumed to be fault free, all the 4^n output nodes are accessible in a maximum of three passes.

$\square$

Example 5.5

A three stage DCMIN, where each DIMSE of stage-1 and stage-2 has one of its control lines s-a-α fault, is shown in Fig.5.16. The adjacency matrices A_1, A_2 and A_3 of first, second and third stages are given in Figs.F5, F6 and F7 respectively. Figs.F8, F9 and F10 are reachability matrices in first, second and third pass respectively. A close observation of Fig.F9 reveals that the majority of nodes can be accessed to in only two passes. However, for DFA it requires maximum of three passes as is shown in Fig.F10. It is to be mentioned here that by assuming faults in all DIMSEs except those of the last stage, the worst possible case has been considered.

Theorem 5.7

In an NxN DCMIN where $N = 4^n$ and $n \geq 3$, if some or all DIMSEs have a single stuck-at-link (except the terminal links) type faults, then the network will retain its DFA property in a maximum of two passes.

Proof

The graph model of DIMSE with stuck type link faults is shown in Fig.5.9. Under single stuck-at-link type faults any input node i, $0 \leq i \leq (N-1)$, can access three output links of its own DIMSE in the link level-1. Through these three links, three different DIMSEs of second stage or 9 links from the link level-2 can be accessed. Proceeding in this way 3^{n-1} DIMSEs of the last stage can be accessed in the first pass.

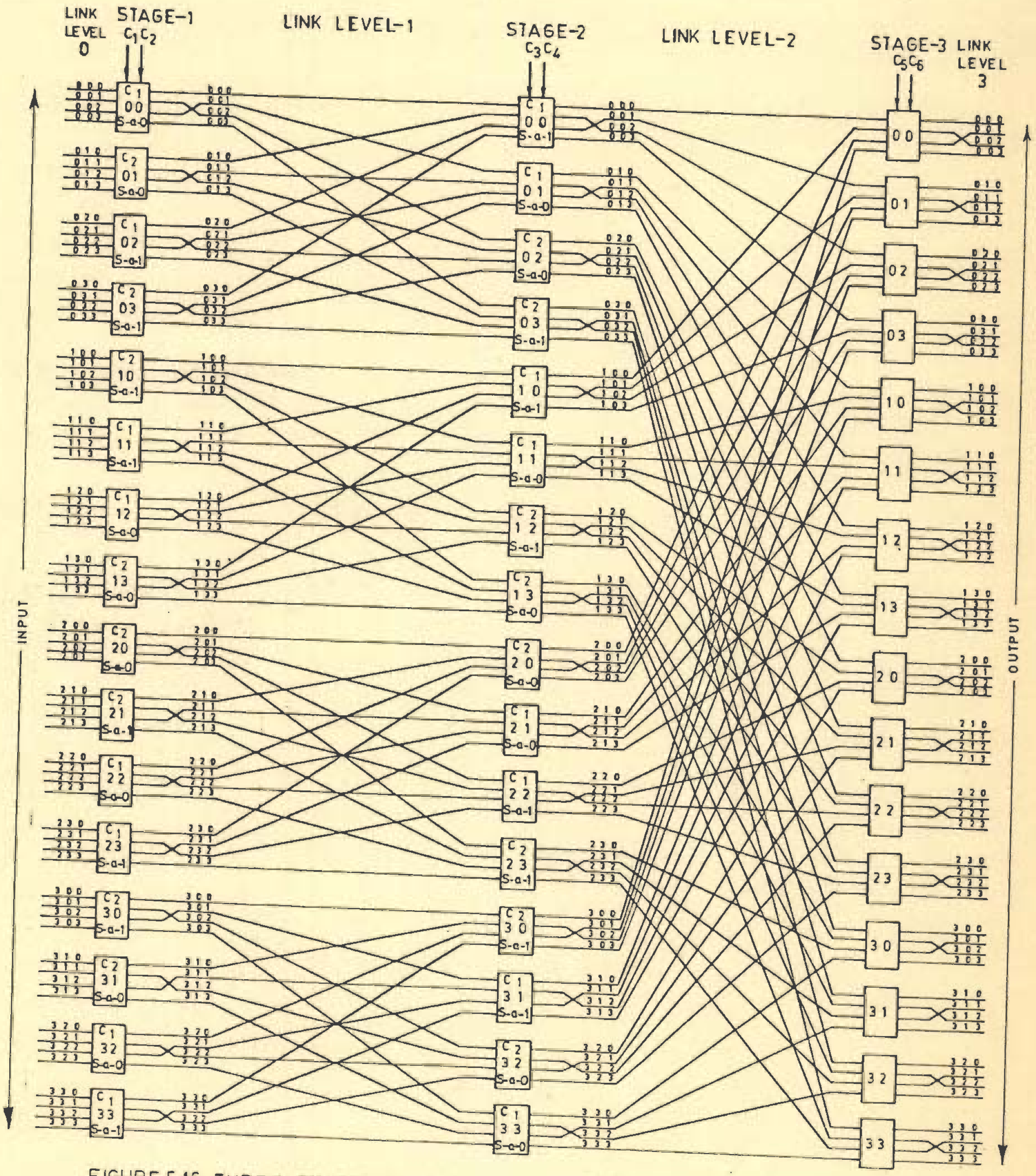


FIGURE 5.16 THREE STAGE DCMIN. ALL ELEMENTS OF STAGE-1 AND 2 HAVE CONTROL LINE FAULTS AS INDICATE ON THE ELEMENTS.

In the second pass, the outputs of 3^{n-1} DIMSEs will be fed back to 3^{n-1} DIMSEs of the first stage, which in turn may access at least 3^{n-1} DIMSEs of the second stage, 3.3^{n-1} DIMSEs of third stage, $3^2.3^{n-1}$ DIMSEs of fourth stage and $(3^{n-2}.3^{n-1})$ DIMSEs of the last stage, i.e., nth stage. Again

$$3^{n-2}.3^{n-1} = 9^{n-1}.3^{-1} > 4^{n-1}$$

for $n \geq 3$.

Thus, all the output nodes can be accessed to in a maximum of two passes maintaining DFA characteristic.

$\square$

Example 5.6

Figure 5.17 shows a three stage DCMIN having maximum possible single stuck-at-link faults. Figs. F11, F12 and F13 give the adjacency matrices of first, second and third stages respectively. Fig. F14 represents the reachability matrix in single pass. It can be seen that all the elements of matrix R of Fig.F14 are not at logical value '1'. $R_2 = R.R$, the reachability matrix in two passes, is shown in Fig.F15. All the elements of R_2 are at logical value '1'.

Thus, in the above case with maximum possible link faults the DFA is maintained in a maximum of two passes.

Theorem 5.8

In an NxN DCMIN where $N = 4^n$ and $n \geq 3$, if some or all DIMSEs have single stuck-type fault (link or control line) with the following assumptions:

(a) Terminal links are fault-free.

(b) All DIMSEs of either first stage or last stage are free from control line faults.

then the network will retain its DFA property in a maximum of three passes.

Proof

The proof of this theorem is obvious and can be readily derived using the approach adapted for Theorems 5.6 and 5.7.

▱

Example 5.7

A three stage DCMIN, in which all DIMSEs have single stuck-type mixed (link or control line) faults subject to the restrictions imposed by Theorem 5.8, is shown in Fig.5.18.

Figures F16, F17 and F18 give the adjacency matrices of first, second and third stages respectively. Fig.F19 gives the reachability matrix R in single pass and Fig.F20 the reachability matrix in two passes (R_2). It is observed that the matrix R_2 has all its elements at logical value '1'. Thus,the DFA is maintained in maximum two passes under the assumed fault conditions of Fig.5.18.

Now to compare the DFA capabilities of DCMIN with conventional cube networks two examples of binary 4-cube array are taken as follows:

In the second pass, the outputs of 3^{n-1} DIMSEs will be fed back to 3^{n-1} DIMSEs of the first stage, which in turn may access at least 3^{n-1} DIMSEs of the second stage, 3.3^{n-1} DIMSEs of third stage, $3^2.3^{n-1}$ DIMSEs of fourth stage and $(3^{n-2}.3^{n-1})$ DIMSEs of the last stage, i.e., nth stage. Again

$$3^{n-2}.3^{n-1} = 9^{n-1}.3^{-1} > 4^{n-1}$$

$$\text{for } n \geq 3.$$

Thus, all the output nodes can be accessed to in a maximum of two passes maintaining DFA characteristic.

$\square$

Example 5.6

Figure 5.17 shows a three stage DCMIN having maximum possible single stuck-at-link faults. Figs. F11, F12 and F13 give the adjacency matrices of first, second and third stages respectively. Fig. F14 represents the reachability matrix in single pass. It can be seen that all the elements of matrix R of Fig. F14 are not at logical value '1'. $R_2 = R.R$, the reachability matrix in two passes, is shown in Fig. F15. All the elements of R_2 are at logical value '1'.

Thus, in the above case with maximum possible link faults the DFA is maintained in a maximum of two passes.

Theorem 5.8

In an NxN DCMIN where $N = 4^n$ and $n \geq 3$, if some or all DIMSEs have single stuck-type fault (link or control line) with the following assumptions:

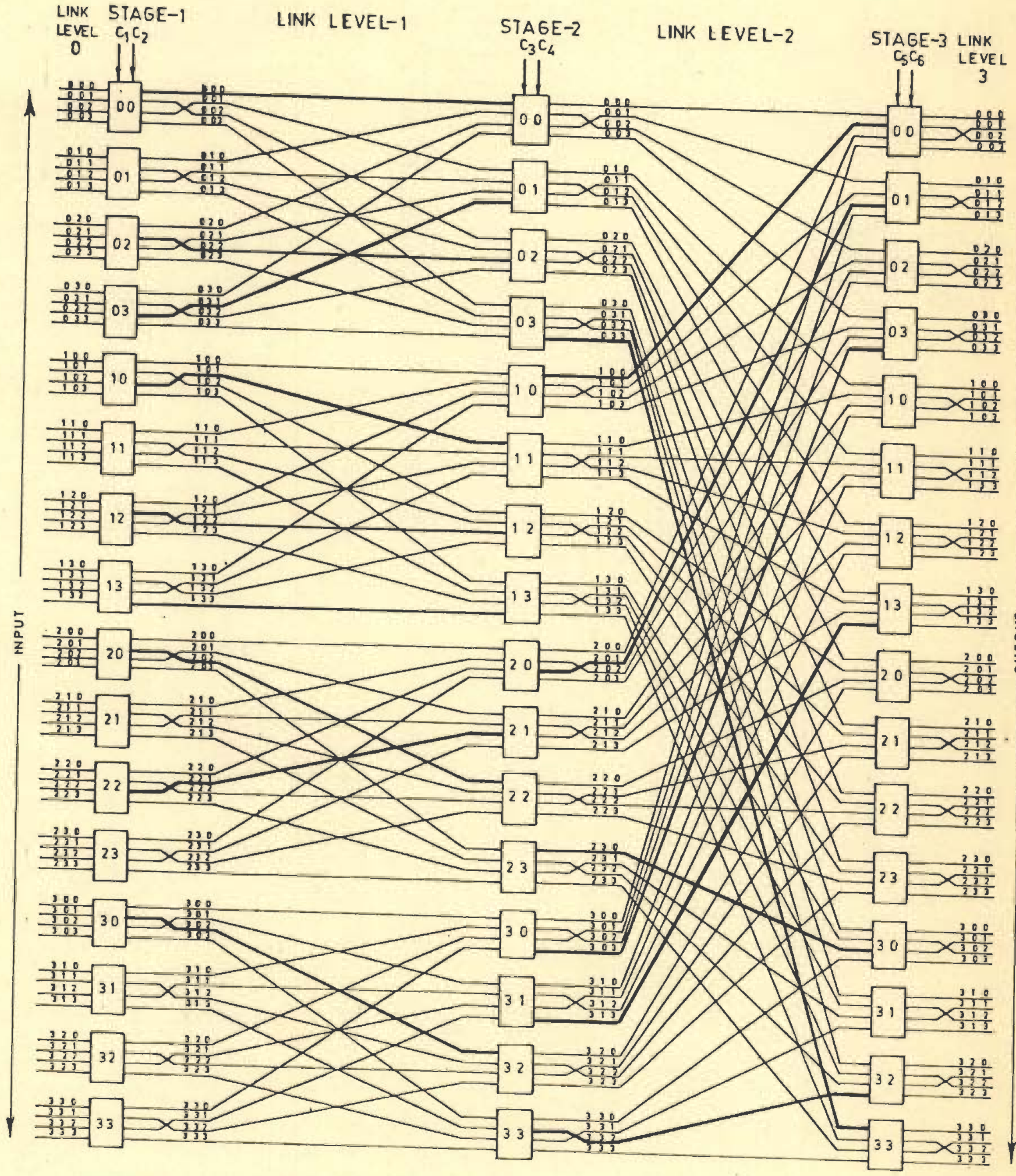


FIGURE 5.17 THREE STAGE DCMIN WITH MAXIMUM POSSIBLE SINGLE-STUCK-AT LINK FAULTS. FAULTY LINKS ARE SHOWN DARK

(a) Terminal links are fault-free.

(b) All DIMSEs of either first stage or last stage are free from control line faults.

then the network will retain its DFA property in a maximum of three passes.

Proof

The proof of this theorem is obvious and can be readily derived using the approach adapted for Theorems 5.6 and 5.7.

$\square$

Example 5.7

A three stage DCMIN, in which all DIMSEs have single stuck-type mixed (link or control line) faults subject to the restrictions imposed by Theorem 5.8, is shown in Fig.5.18.

Figures F16, F17 and F18 give the adjacency matrices of first, second and third stages respectively. Fig.F19 gives the reachability matrix R in single pass and Fig.F20 the reachability matrix in two passes (R_2). It is observed that the matrix R_2 has all its elements at logical value '1'. Thus,the DFA is maintained in maximum two passes under the assumed fault conditions of Fig.5.18.

Now to compare the DFA capabilities of DCMIN with conventional cube networks two examples of binary 4-cube array are taken as follows:

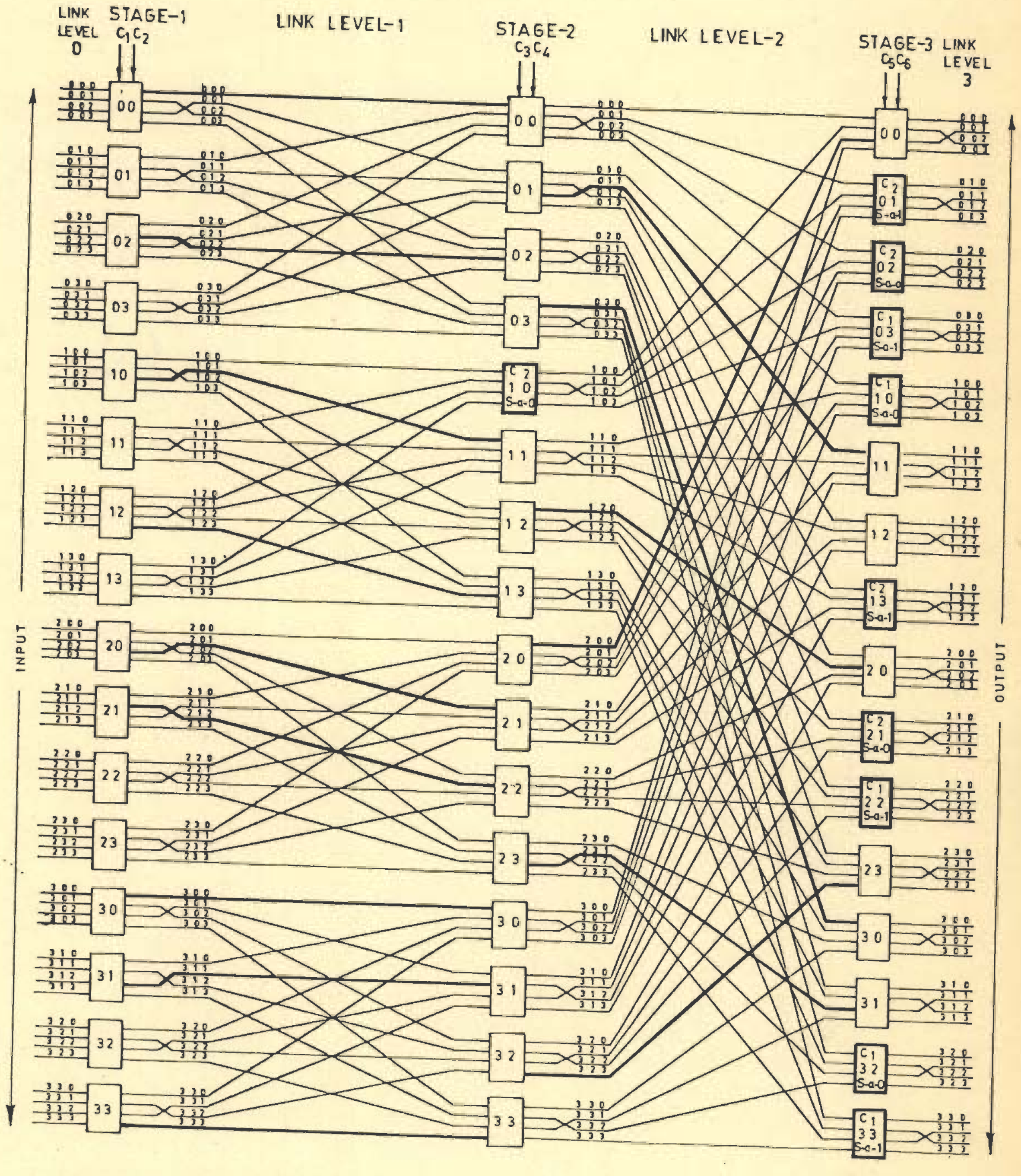


FIGURE 5.18 THREE STAGE DCMIN HAVING MIXED FAULTS. FAULTY LINKS (S-a-α, $\alpha \in$ 0,1) AND ELEMENTS ARE SHOWN DARK.

Example 5.8

Figure 5.19 gives indirect binary 4-cube array with maximum possible single stuck-at-link faults. Figs. F21-F24 give the adjacency matrices of stages 1 to 4 respectively. Fig. F25 is the reachability matrix in single pass which does not demonstrate the full-access property. Figs. F26-F30 give R_2, R_3, R_4, R_5 and R_6 matrices. It is observed that matrix R_6 has all its elements at logical value '1'. This demonstrates that the network requires maximum of six passes to maintain DFA under assumed fault conditions of Fig. 5.19.

Example 5.9

Figure 5.20 depicts the same cube network in which all elements of stage-2 have their control lines faulty. Figs. F31-F34 represent the adjacency matrices of stages 1 to 4 respectively. Fig. F35 is the reachability matrix in single pass which does not demonstrate full-access property. Fig. F36 is R_2 matrix with all its elements at logical value '1'. Thus, the network of Fig. 5.20, with assumed control line faults, needs maximum of two passes to maintain DFA.

If an input can access a desired output in single pass then the path length is called a unity path length and is denoted by L. From reachability matrices it can be seen that all the desired input/output connections do not require the maximum number of passes required for DFA. For DCMIN with the above assumed fault conditions some of the input/output connections require only single path length L, whereas others

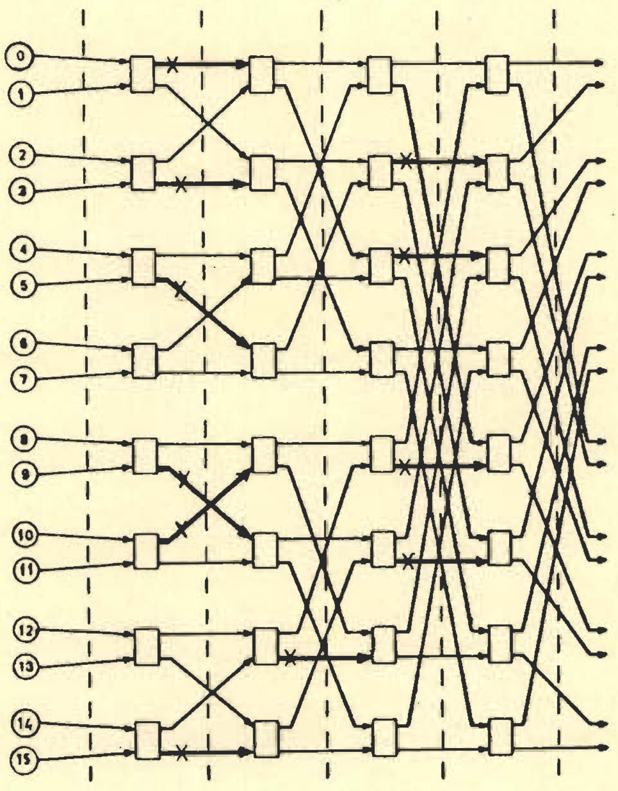


FIGURE 5.19 16 X 16 INDIRECT BINARY 4-CUBE NETWORK WITH SINGLE STUCK-AT-ALPHA FAULTS IN THE LINKS. FAULTY LINKS ARE SHOWN DARK.

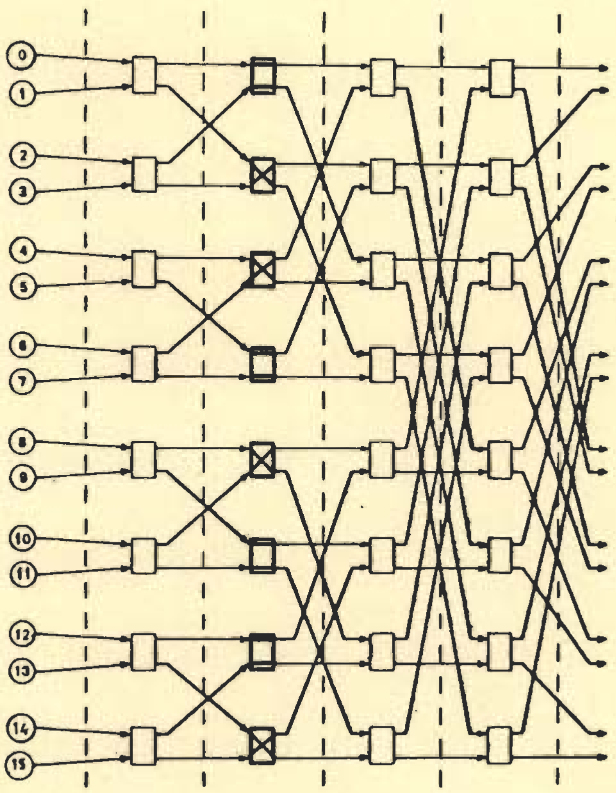


FIGURE 5.20 16 X 16 INDIRECT BINARY 4-CUBE NETWORK. ALL SWITCHING ELEMENTS OF STAGE-2 ARE HAVING STUCK-AT-ALPHA FAULTS AT THEIR CONTROL LINES AS INDICATED IN THE FIGURE.

may require path lengths of 2L or 3L. Thus, it will be desirable to find out the average of these path lengths for establishing an arbitrary input/output connection. This average value then may be used to examine the suitability of the network for practical implementation. The next section is devoted to calculate the average path lengths for the examples taken in this section.

5.3 DISTANCE MATRIX AND AVERAGE PATH LENGTH OF DCMIN

Let the number of passes required for a request to reach from an arbitrary input node p to output node r be m. Then the Distance matrix D [62] of a DCMIN is defined as an NxN matrix $[d_{pr}]$ with entries

$$d_{pr} = 0 \qquad \text{for } p = r$$
$$d_{pr} = m \qquad \text{otherwise.}$$

If no conflict in data path is assumed for the random requests, then the average of the path lengths, called the static average and represented by SAV, can be computed from the distance matrix D as follows [8]:

$$SAV = \frac{L}{N^2} \sum_{p=1}^{N} \sum_{r=1}^{N} d_{pr}$$

where L is the path length in each pass.

The average path length (SAV) for each of the examples taken in Section 5.2 has been calculated for the assumed faults. These are compared in Table 5.2 where L_{dc} represents path length of dual cube and L_{bc} that of binary cube.

Example Number	Type of network	Size of network	No. of Stages in single pass	Type of Faults	No. of passes for DFA	SAV
5.4	DCMIN	16x16	2	Each DIMSE of stage one having a control line S-a-α	2	1.41 L_{dc}
5.5	DCMIN	64x64	3	Each DIMSE in first two stages having a control line S-a-α	3	1.54 L_{dc}
5.6	DCMIN	64x64	3	Maximum possible single stuck type link faults	2	1.32 L_{dc}
5.7	DCMIN	64x64	3	Maximum possible mixed faults in all the DIMSEs	2	1.47 L_{dc}
5.8	Binary 4-cube	16x16	4	Maximum possible single stuck type link faults	6	2.28 L_{bc}
5.9	Binary 4-cube	16x16	4	Control line of each SE in stage-2 is S-a-α	2	1.41 L_{bc}

TABLE 5.2 COMPARISON OF AVERAGE PATH LENGTHS OF DCMIN AND BINARY CUBE NETWORKS UNDER VARIOUS FAULT CONDITIONS.

It has already been shown in Section 4.7 that a multi-stage network of size N in DCMIN topology requires half the number of stages compared to conventional cube topology. Accordingly, in Table 5.2

$$L_{dc} \approx 0.5\ L_{bc}$$

From the Table 5.2, for 16x16 MIN, if all the DIMSEs of one out of two stages of DCMIN have single control line faults then SAV becomes 1.41 L_{dc}. Also, if all SEs of one out of 4 stages of conventional cube topology have their control lines S-a-α, then SAV is 1.41 L_{bc}.

In Example 5.6 and 5.8, the maximum possible single stuck-at-link faults are assumed for 64x64 DCMIN topology and 16x16 cube topology respectively. The SAV for conventional 16x16 cube topology becomes 2.28 L_{bc} whereas it is 1.32 L_{dc} for 64x64 DCMIN.

Further, for worst case faults assumed for DCMIN in above examples, average path length never exceeds 1.5 times the path length in a single pass.

From the above analysis of Table 5.2, it is clearly revealed that in all cases DCMIN topology has better fault-tolerance capabilities when compared to conventional cube topology using 2x2 SEs.

5.4 CONCLUSION

In Chapter IV DCMIN topology was proposed as an alternative topology for use in parallel architectures. Its performance and complexity were evaluated and compared with conventional cube topology. In this chapter fault-diagnosis and fault-tolerance aspects of DCMIN have been examined. Considering the logical faults on the pins of DIMSEs, procedures have been discussed to detect and locate the faults. Regular test-vectors for this purpose have been proposed. It has been demonstrated that non overlapping multiple faults can be diagnosed using the proposed methods. However, for overlapping multiple faults each stage is required to be tested separately. A method to derive test-vector for testing each stage separately, for overlapping multiple faults in control lines, has been proposed and demonstrated.

The fault-tolerance of DCMIN has been evaluated by allowing multiple passes. For this purpose the graph model of DCMIN based on connectivity concept has been used to find the adjacency matrix for each stage of DCMIN. Examples with maximum allowable faults have been used to demonstrate that DCMIN has better fault-tolerance capabilities, compared to the conventional cube networks using 2x2 SEs. In the next chapter the fault-tolerance aspect of DCMIN using multiple paths will be evaluated.

CHAPTER VI

GENERAL PROPERTIES OF DCMIN

The topology of DCMIN and its various characteristics which include modes of operation, interconnection function, control strategy, performance and complexity have been discussed in Chapter IV. Further, in Chapter V, the fault-diagnosis and fault-tolerance aspects were considered. Specifically, the fault-tolerance of DCMIN using multiple-passes technique was investigated. In this chapter yet another approach to achieve fault-tolerance of DCMIN using multiple-paths will be discussed. However, before discussing the alternative approach for fault tolerance, two other important aspects which need to be investigated are - analytical model of DCMIN and its partitionability to lower size DCMINs. These characteristics are essential for any MIN if it has to be used and implemented in large parallel processing systems [91,122].

6.1 ANALYTICAL MODEL OF DCMIN

In general, the analysis of MINs available at present is based on the mathematical treatment involving permutation functions and graph theory. Recently, matrix representation has been proposed as an alternative and convenient model for analysis of MINs [92,133] because of its ease in implementing on computer. MINs perform certain specific interconnection functions on the sets of input/output addresses through control signals. These functions can be conveniently expressed by using a switching matrix for the network. This matrix representation is elegant and can be readily generalized for larger networks. Let the matrix defining the input/output relationship of a MIN be called mapping matrix. It can be derived from the elementary matrices of the basic switching elements of the MIN.

In this section the mapping matrix of DIMSE is derived first. Subsequently a procedure has been evolved to obtain the mapping matrix of DCMIN, of any desired size, using DIMSE matrix as the basic building block. If the mapping matrix is available, it can be used to develop the corresponding network representing this matrix. Network interpretation of the matrix has been discussed and demonstrated in Section 6.1.3.

6.1.1 Matrix Representation of DIMSE

In this section matrix response of the DIMSE of Fig.6.1, is derived. The switching element has four inputs I_0-I_3 and four outputs R_{10}-R_{13}. The input/output connections are controlled by two control signals C_1 and C_2. The four working modes of DIMSE for different combinations of the control signals C_1, C_2 have already been explained in Fig.4.2. The response of the DIMSE in terms of various control settings and input signals can be written as:

$$\begin{aligned}
R_{10} &= \bar{C}_1\bar{C}_2I_0 + C_1\bar{C}_2I_1 + \bar{C}_1C_2I_2 + C_1C_2I_3 \\
R_{11} &= C_1\bar{C}_2I_0 + \bar{C}_1\bar{C}_2I_1 + C_1C_2I_2 + \bar{C}_1C_2I_3 \\
R_{12} &= \bar{C}_1C_2I_0 + C_1C_2I_1 + \bar{C}_1\bar{C}_2I_2 + C_1\bar{C}_2I_3 \qquad ..(6.1) \\
R_{13} &= C_1C_2I_0 + \bar{C}_1C_2I_1 + C_1\bar{C}_2I_2 + \bar{C}_1\bar{C}_2I_3
\end{aligned}$$

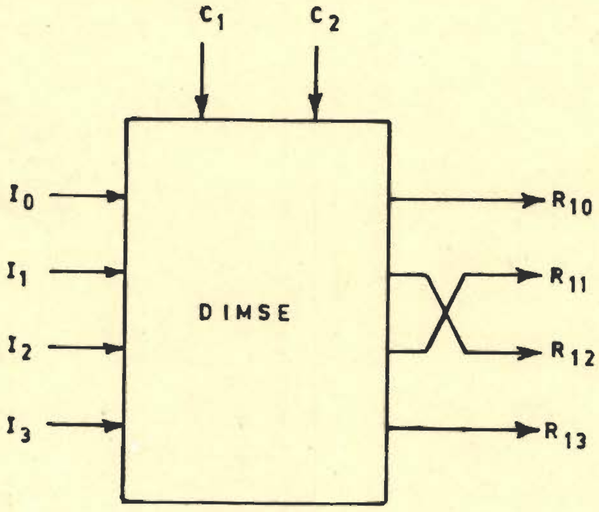


FIGURE 6.1 DUAL - INTERCONNECTION MODULAR SWITCHING ELEMENT. I IS INPUT VECTOR, R-OUTPUT VECTOR AND C_1, C_2 ARE CONTROL SIGNALS.

A matrix representation of (6.1) is

$$\begin{bmatrix} R_{10} \\ R_{11} \\ R_{12} \\ R_{13} \end{bmatrix} = \begin{bmatrix} \bar{C}_1\bar{C}_2 & C_1\bar{C}_2 & \bar{C}_1C_2 & C_1C_2 \\ C_1\bar{C}_2 & \bar{C}_1\bar{C}_2 & C_1C_2 & \bar{C}_1C_2 \\ \bar{C}_1C_2 & C_1C_2 & \bar{C}_1\bar{C}_2 & C_1\bar{C}_2 \\ C_1C_2 & \bar{C}_1C_2 & C_1\bar{C}_2 & \bar{C}_1\bar{C}_2 \end{bmatrix} \begin{bmatrix} I_0 \\ I_1 \\ I_2 \\ I_3 \end{bmatrix} \quad ..(6.2)$$

or $\quad [R] \quad = \quad [M]_{C_1C_2} \quad [I]$

where [R] and [I] represent the response and the input matrices respectively. $[M]_{C_1C_2}$ which relates inputs to outputs is the mapping matrix. The order of $[M]_{C_1C_2}$ is same as that of the network. Thus, a square network of size NxN will have matrix [M] of the order NxN. The size of the square mapping matrix may also be represented by putting a subscript. Accordingly, the mapping matrix of 4x4 switch, in Equation (6.2), can be written as $[M_4]$ or $[M]_{C_1C_2}$ as may be convenient in writing.

Using the Kronecker product, $\otimes$, of matrices [41], the mapping matrix $[M_4]$ of Equation (6.2) can be further simplified as

$$[M_4] = \begin{bmatrix} \bar{C}_1\bar{C}_2 & C_1\bar{C}_2 & \bar{C}_1C_2 & C_1C_2 \\ C_1\bar{C}_2 & \bar{C}_1\bar{C}_2 & C_1C_2 & \bar{C}_1C_2 \\ \bar{C}_1C_2 & C_1C_2 & \bar{C}_1\bar{C}_2 & C_1\bar{C}_2 \\ C_1C_2 & \bar{C}_1C_2 & C_1\bar{C}_2 & \bar{C}_1\bar{C}_2 \end{bmatrix} = \begin{bmatrix} \bar{C}_2 & C_2 \\ C_2 & \bar{C}_2 \end{bmatrix} \otimes \begin{bmatrix} \bar{C}_1 & C_1 \\ C_1 & \bar{C}_1 \end{bmatrix}$$

or

$$[M_4] = [M_2]_{C_2} \otimes [M_2]_{C_1} \quad ..(6.3)$$

$[M_2]$ is the fundamental mapping matrix which is indivisible and can also be written as $[M_e]$. Since each control signal can have two values, i.e., logical '0' or logical '1', $[M_e]$ will always be a 2x2 matrix. Thus Equation (6.3) can be rewritten as

$$[M_4] = [M_e]_{C_2} \otimes [M_e]_{C_1} \qquad ..(6.4)$$

The above concept developed for DIMSE will now be applied to develop analytical model of DCMIN.

6.1.2 Matrix Representation of DCMIN

A two stage DCMIN with appropriate labeling of inputs and outputs is shown in Fig.6.2. Subscripts on input I indicate the physical location of the input terminals. The response R has three subscripts and can be written as R_{ipk}, where

i - represents the stage number and

pk - are two decimal digits representing the input terminal number for which this terminal carries data under mode '0' condition of all DIMSEs

The different settings of the stage control signals give the different permutations for data transfer from input to output. Table 6.1 gives the mapping matrices for some sample settings of the control signals. In Fig.6.2, the outputs of the switches of the first stage act as inputs to the switches of the second stage. The outputs of the second stage can be readily derived in terms of the inputs of the first stage by proper substitution. This process can be repeated for n-number of stages. As an illustration, using the terminology given in Fig.6.2, response of the network at the output terminal R_{201}

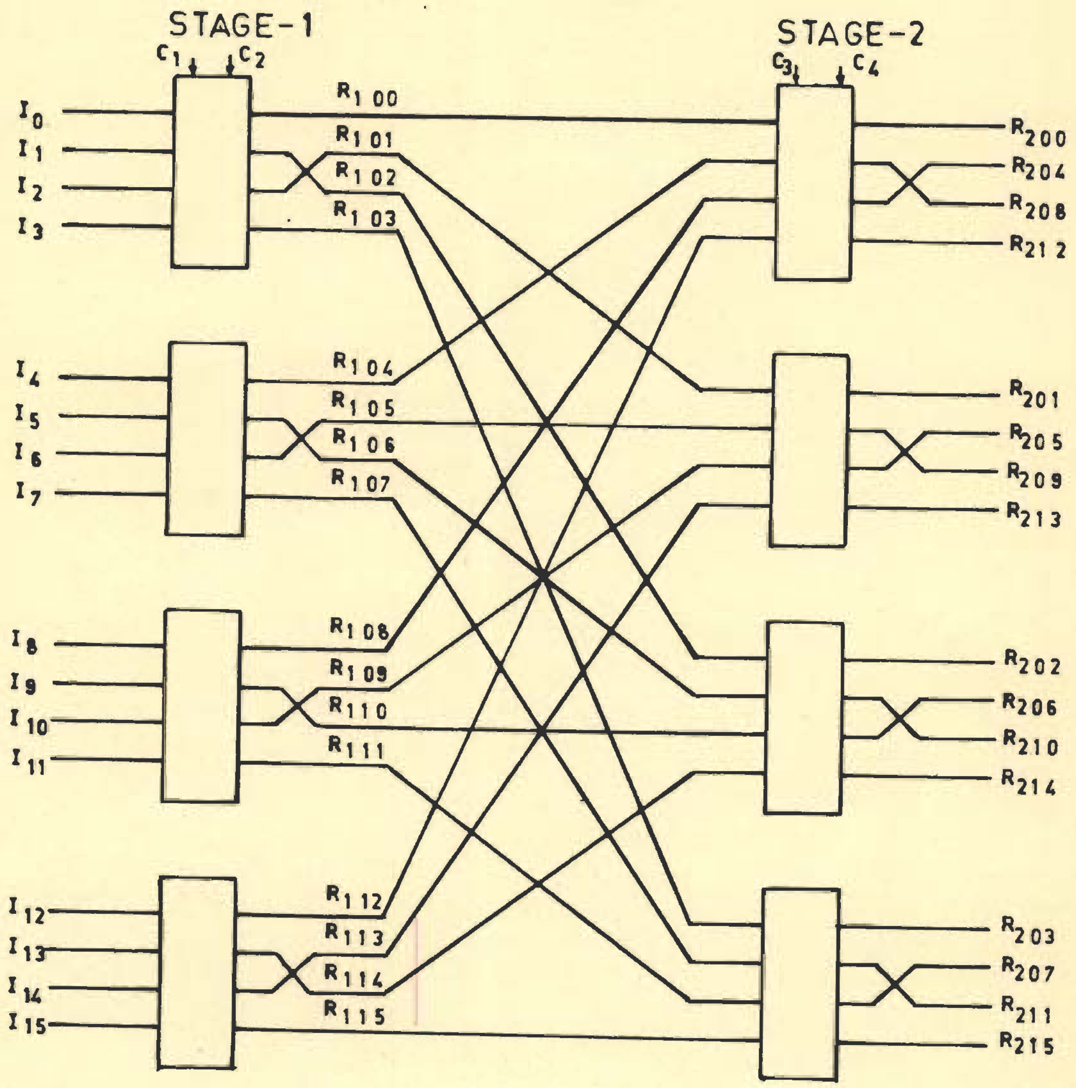


FIGURE 6.2 A TWO-STAGE DCMIN. I INDICATES THE INPUT AND R THE RESPONSE.

$$\begin{array}{c} \text{IN} \\ \text{OUT} \end{array} \left(\begin{array}{cccccccccccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 \end{array}\right) \begin{array}{c} \\ C_1C_2C_3C_4 \\ 0\ 0\ 0\ 0 \end{array}$$

(a)

$$\begin{array}{c} \text{IN} \\ \text{OUT} \end{array} \left(\begin{array}{cccccccccccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 \\ 15 & 14 & 13 & 12 & 11 & 10 & 9 & 8 & 7 & 6 & 5 & 4 & 3 & 2 & 1 & 0 \end{array}\right) \begin{array}{c} \\ 1\ 1\ 1\ 1 \end{array}$$

(b)

$$\begin{array}{c} \text{IN} \\ \text{OUT} \end{array} \left(\begin{array}{cccccccccccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 \\ 12 & 13 & 14 & 15 & 8 & 9 & 10 & 11 & 4 & 5 & 6 & 7 & 0 & 1 & 2 & 3 \end{array}\right) \begin{array}{c} \\ 0\ 0\ 1\ 1 \end{array}$$

(c)

$$\begin{array}{c} \text{IN} \\ \text{OUT} \end{array} \left(\begin{array}{cccccccccccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 \\ 3 & 2 & 1 & 0 & 7 & 6 & 5 & 4 & 11 & 10 & 9 & 8 & 15 & 14 & 13 & 12 \end{array}\right) \begin{array}{c} \\ 1\ 1\ 0\ 0 \end{array}$$

(d)

TABLE 6.1 INPUT/OUTPUT MAPPING FOR FOUR DIFFERENT SETTINGS OF THE CONTROL SIGNALS
(a) $C_1=C_2=C_3=C_4=0$ (b) $C_1=C_2=C_3=C_4=1$
(c) $C_1=C_2=0$, $C_3=C_4=1$ (d) $C_1=C_2=1$, $C_3=C_4=0$.

can be written as

$$R_{201} = \bar{C}_3\bar{C}_4R_{101} + C_3\bar{C}_4R_{105} + \bar{C}_3C_4R_{109} + C_3C_4R_{113} \quad ..(6.5)$$

Substituting the values of R_{101}, R_{105}, R_{109} and R_{113} (using the concept of the mapping matrix of DIMSE as given by Equation (6.2))

$$R_{201} = \begin{array}{l} \bar{C}_3\bar{C}_4\,[C_1\bar{C}_2I_0 + \bar{C}_1\bar{C}_2I_1 + C_1C_2I_2 + \bar{C}_1C_2I_3] + \\ C_3\bar{C}_4\,[C_1\bar{C}_2I_4 + \bar{C}_1\bar{C}_2I_5 + C_1C_2I_6 + \bar{C}_1C_2I_7] + \\ \bar{C}_3C_4\,[C_1\bar{C}_2I_8 + \bar{C}_1\bar{C}_2I_9 + C_1C_2I_{10} + \bar{C}_1C_2I_{11}] + \\ C_3C_4[C_1\bar{C}_2I_{12} + \bar{C}_1\bar{C}_2I_{13} + C_1C_2I_{14} + \bar{C}_1C_2I_{15}] \end{array} \quad ..(6.6)$$

The other outputs can also be written on similar lines. Proceeding this way, the mapping matrix of 2-stage DCMIN, where N = 16, can be written as

$$[M_{16}] = \begin{bmatrix} \bar{C}_3\bar{C}_4[M]_{C_1C_2} & C_3\bar{C}_4[M]_{C_1C_2} & \bar{C}_3C_4[M]_{C_1C_2} & C_3C_4[M]_{C_1C_2} \\ C_3\bar{C}_4[M]_{C_1C_2} & \bar{C}_3\bar{C}_4[M]_{C_1C_2} & C_3C_4[M]_{C_1C_2} & \bar{C}_3C_4[M]_{C_1C_2} \\ \bar{C}_3C_4[M]_{C_1C_2} & C_3C_4[M]_{C_1C_2} & \bar{C}_3\bar{C}_4[M]_{C_1C_2} & C_3\bar{C}_4[M]_{C_1C_2} \\ C_3C_4[M]_{C_1C_2} & \bar{C}_3C_4[M]_{C_1C_2} & C_3\bar{C}_4[M]_{C_1C_2} & \bar{C}_3\bar{C}_4[M]_{C_1C_2} \end{bmatrix} \quad ..(6.7)$$

$$= \begin{bmatrix} \bar{C}_3\bar{C}_4 & C_3\bar{C}_4 & \bar{C}_3C_4 & C_3C_4 \\ C_3\bar{C}_4 & \bar{C}_3\bar{C}_4 & C_3C_4 & \bar{C}_3C_4 \\ \bar{C}_3C_4 & C_3C_4 & \bar{C}_3\bar{C}_4 & C_3\bar{C}_4 \\ C_3C_4 & \bar{C}_3C_4 & C_3\bar{C}_4 & \bar{C}_3\bar{C}_4 \end{bmatrix} \otimes \begin{bmatrix} \bar{C}_1\bar{C}_2 & C_1\bar{C}_2 & \bar{C}_1C_2 & C_1C_2 \\ C_1\bar{C}_2 & \bar{C}_1\bar{C}_2 & C_1C_2 & \bar{C}_1C_2 \\ \bar{C}_1C_2 & C_1C_2 & \bar{C}_1\bar{C}_2 & C_1\bar{C}_2 \\ C_1C_2 & \bar{C}_1C_2 & C_1\bar{C}_2 & \bar{C}_1\bar{C}_2 \end{bmatrix} \quad ..(6.8)$$

$$= [M]_{C_3C_4} \otimes [M]_{C_1C_2} \quad ..(6.9)$$

$[M]_{C_3C_4}$ represents the mapping matrix of stage-2, independently. Further the equation (6.8) can be rewritten as

$$[M]_{C_1C_2C_3C_4} = [M_e]_{C_4} \otimes [M_e]_{C_3} \otimes [M_e]_{C_2} \otimes [M_e]_{C_1} \quad ..(6.10)$$

Because of the recursive nature of the proposed topology of DCMIN, this concept can be extended to a network of any size $N = 4^n$ as

$$[M]_{C_1C_2..C_{2n-1}C_{2n}} = [M]_{C_{2n-1}C_{2n}} \otimes [M]_{C_{2n-3}C_{2n-2}} \otimes .. \\ ... \otimes [M]_{C_3C_4} \otimes [M]_{C_1C_2} \quad ..(6.11)$$

$$= [M_e]_{C_{2n}} \otimes [M_e]_{C_{2n-1}} \otimes [M_e]_{C_{2n-2}} \\ \otimes [M_e]_{C_{2n-3}} \otimes \ldots \otimes [M_e]_{C_4} \\ \otimes [M_e]_{C_3} \otimes [M_e]_{C_2} \otimes [M_e]_{C_1} \quad ..(6.12)$$

$$= \bigotimes_{K=2n}^{1} [M_e]_{C_K} \quad ..(6.13)$$

$$= \bigotimes_{K=n}^{1} [M]_{C_{2K-1}C_{2K}} \quad ..(6.14)$$

In view of the regular structure offered by Kronecker product, it becomes a simple matter to derive the mapping matrix of any larger sized DCMIN just from the mapping matrix of a DIMSE. In the next section the development of DCMIN network from the given mapping matrix will be demonstrated. In the rest of the text symbol $[M_K]$ or M_K will be used interchangeably to mean KxK matrix.

6.1.3 Development of DCMIN From a Given Mapping Matrix

It has been shown in Equation (6.13) that in the transformation $R = MI$ of n-stage DCMIN, M is a Kronecker product of mapping matrices of DIMSE with appropriate labeling of the control signals. The transformation $R = MI$ can be interpreted in network terms: an operator M, seen as a black box, acts upon an input vector I to produce an output vector R. Thus, if the mapping matrix of a DCMIN is given, the corresponding network topology can be derived by expressing the given mapping matrix $M = \bigotimes M_k$. The unique black box M splits up in sets of smaller black boxes of type M_k. In DCMIN topology $K = 4$ (as explained in Section 4.3.3).

Corollary 4.1 demonstrated that an n-stage DCMIN can always be developed from two sets of networks:

(i) A stack of $\frac{N}{4}$ DIMSEs where $N = 4^n$. This is shown in Fig.6.3 for $n = 2$, and

(ii) A stack of $\frac{N}{4}$ DIMSEs preceded by a 4-shuffle permuter as shown in Fig.6.4 for $n = 2$.

It is a simple matter to represent the networks of Figs.6.3 and 6.4 by the transformation Equations (6.15) and (6.16) respectively

$$R = (U_4 \otimes M_4)I \qquad ..(6.15)$$

$$R = (M_4 \otimes U_4)I \qquad ..(6.16)$$

where U represents a unit matrix and the subscript represents the order of the matrix. Thus U_4 represents a unit matrix of order 4.

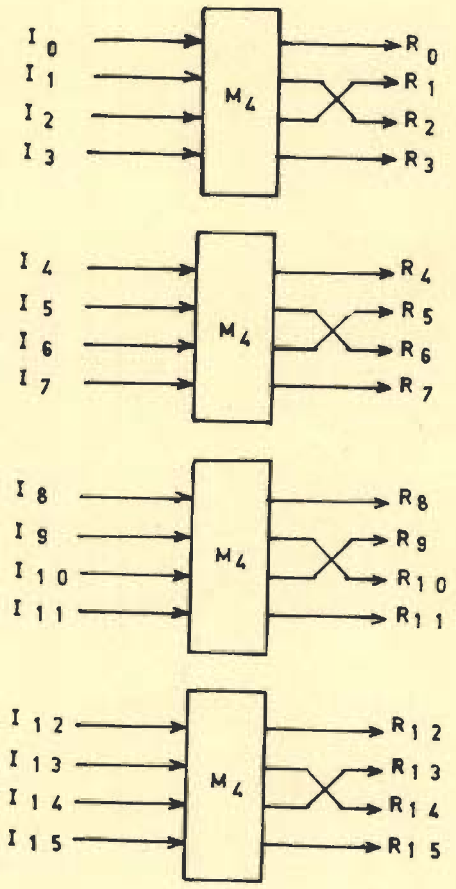


FIGURE 6.3 THE STACK OF 4 DIMSEs HAVING MATHEMATICAL REPRESENTATION $M = (U_4 \otimes M_4)$

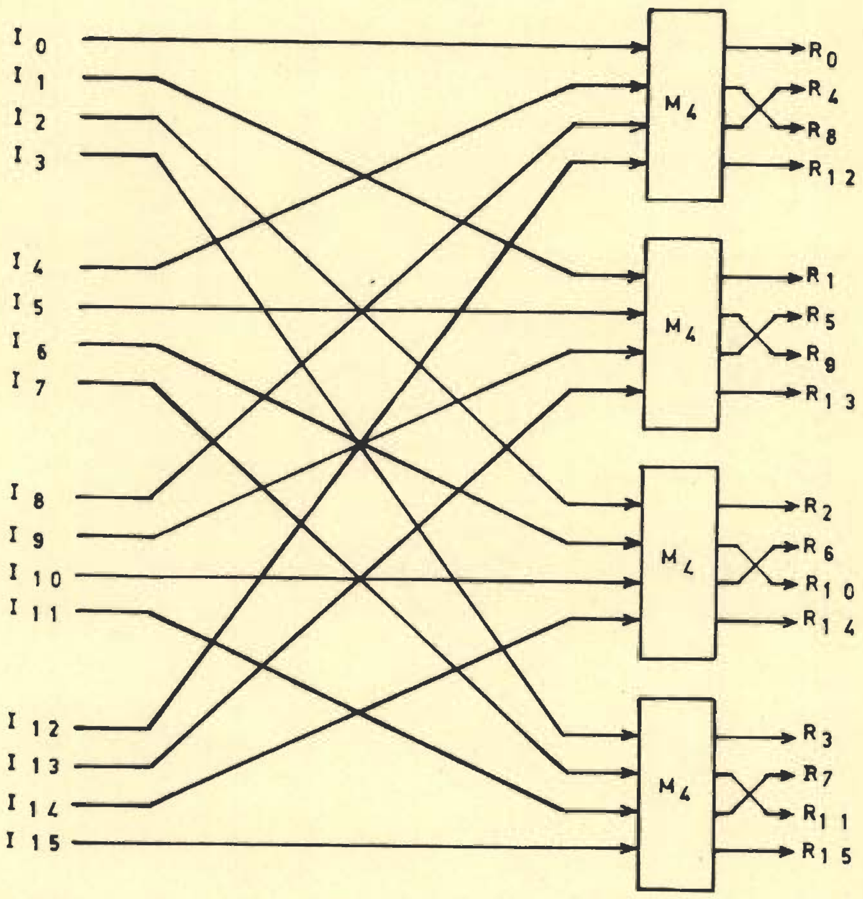


FIGURE 6.4 THE STACK OF 4 DIMSEs FOLLOWED BY 4-SHUFFLE PERMUTER HAVING MATHEMATICAL REPRESENTATION $M = (M_4 \otimes U_4)$

It is well known [41] that if M_a and M_b are two matrices of order ra x ca and rb x cb respectively, then

$$M_a \otimes M_b = (U_{ra} \otimes M_b)(M_a \otimes U_{cb}) \qquad ..(6.17)$$

$$= (M_a \otimes U_{rb})(U_{ca} \otimes M_b) \qquad ..(6.18)$$

The DCMIN topology for any given square mapping matrix can now be derived as explained hereunder:

(i) Decompose the mapping matrix to smaller matrices of size M_4 using Kronecker product of matrices, resulting in an equation similar to Equation (6.11).

(ii) Starting from the matrix with control signal labeling of stage-1, convert these matrices in the form of Equation (6.17) or (6.18) sequentially and derive the network.

Example 6.1

Let M_{64} be given. This can be decomposed using Kronecker product of matrices as:

$$[M_{64}] = [M_4]_{C_5C_6} \otimes [M_4]_{C_3C_4} \otimes [M_4]_{C_1C_2}$$

Using (6.17)

$$= [M_4]_{C_5C_6} \otimes [(U_4 \otimes M_4)(M_4 \otimes U_4)]$$

$= [M_4]_{C_5C_6} \otimes$ [(Stack of 4-units of DIMSEs) •(Stack of 4-units of DIMSEs preceded by 4-Shuffle permuter)]

$= [M_4]_{C_5C_6} \otimes$ [2-Stage DCMIN]

$= [M_4]_{C_5C_6} \otimes [M_{16}]_{C_1C_2C_3C_4}$

Again using Equation (6.17)

$= (U_4 \otimes M_{16})(M_4 \otimes U_{16})$

= [(Stack of 4-units of 2-stage DCMIN)

•(Stack of 16-units of DIMSEs preceded by 4-shuffle permuter)]

= 3-stage DCMIN

Thus, it is shown that a given DCMIN topology can be represented mathematically by simple and regular form of Kronecker product of matrices. Also, if the transformation matrix is known, the DCMIN topology can be derived easily by decomposing the operator M into smaller operators of the type $\otimes\ M_k$.

6.1.4 Properties of Mapping Matrix of DCMIN

From the analysis in Section 6.1.2 the following conclusions can be drawn about the mapping matrix M_k:

(i) The mapping matrix is symmetrical about its principal diagonal. Accordingly, each element $r_{ij} = r_{ji}$.

(ii) The elements of matrices are product of specific combinations of $\bar{C}_1$, C_1; $\bar{C}_2$, C_2; etc. and follows a unique repetitive pattern. $\bar{C}_1$ and C_1 appear alternatively in the elements in any row or column. $\bar{C}_2$ appears in the elements of two consectuve row/column positions and C_2 then appears in the next two consecutive row/column positions. In general $\bar{C}_j$

appears in elements of 2^{j-1} consecutive row/column positions and C_j in the next 2^{j-1} consecutive row/column positions. This repetitive pattern is maintained in the mapping matrix of any order.

(iii) A mapping matrix readily provides the possible interconnection for a given control signal combination. Also, for a given mapping, it is possible to measure the control conditions of the network.

To summarize, in Section 6.1 it has been postulated as to how the mapping matrix of 4x4 DIMSE can be used to represent a DCMIN of any size analytically. Further, it is interesting to note that the properties of the mapping matrix of DCMIN and multistage cube interconnection network [92] are identical thereby demonstrating that DCMIN belongs to the cube family of ICNs.

6.2 PARTITIONING OF DUAL CUBE NETWORKS

In order to use the parallel processing power of a large-scale multiprocessor system efficiently it is desirable that it should be a partitionable system. A partitionable machine is one which can be dynamically reconfigured to operate as one or more independent virtual machines of various sizes. If the virtual subsystems obtained from the partitioning of a parallel system are to be independent, the network they share must also be partitionable into independent subnetworks. Thus, for any ICN to be functional in parallel architectures, it should be partitionable.

The partitionability of an ICN is the ability to divide the network into independent subnetworks of different sizes. Each subnetwork will then behave as an independent network of size $N' < N$.

In this section the partitionability aspect of DCMIN is investigated. However, before formally evaluating the partitioning property of DCMIN, the Dcube function (introduced in Section 4.4) is defined in terms of its permutation-cycle structure.

6.2.1 Dual Cube Network Cyclic-Structure

The Dcube_i interconnection function for $0 \leq i \leq (n-1)$ can be expressed uniquely as a product of $(N/4)$ disjoint sets of length 4 each as

$$\prod_{\substack{p=0 \\ i^{th} \text{ bit of } p = 0}}^{N-1} (p \ \text{Dcube}_i(p))$$

where p is the PE address in quaternary system

This product will produce different groups of N/4 disjoint sets for each value of i. The PEs in each set will communicate with PEs of their own set only.

Example 6.2

For $N = 16$, $n = 2$, there will be two groups of PEs each containing 4 disjoint sets of length 4 each. With addresses of PEs in quaternary system, these are

$\text{Dcube}_0 = (00,01,02,03)(10,11,12,13)(20,21,22,23)(30,31,32,33)$

$\text{Dcube}_1 = (00,10,20,30)(01,11,21,31)(02,12,22,32)(03,13,23,33)$

In this example if $Dcube_1$ function is not used, the subgroups formed by $Dcube_0$ become independent and cannot communicate amongst themselves. Similarly, if $Dcube_0$ function is not used, the subgroups formed by $Dcube_1$ become independent. This is also demonstrated in Fig.4.14.

Thus, each $Dcube_i$ function, $0 \leq i \leq (n-i)$, divides the PE addresses (in their quaternary representation) into four subgroups as follows:

Subgroup I : PEs with addresses having i^{th} positional digit 0
Subgroup II : PEs with addresses having i^{th} positional digit 1
Subgroup III: PEs with addresses having i^{th} positional digit 2
Subgroup IV : PEs with addresses having i^{th} positional digit 3

The concepts developed above will now be used for partitioning of DCMIN.

6.2.2 Partitioning of DCMIN

In a DCMIN of size 4^n each stage $(i+1)^{th}$, $0 \leq i \leq (n-1)$, performs $Dcube_i$ function. It has been shown in the preceding section that each $Dcube_i$ function yields (N/4) disjoint subgroups of PEs which can communicate with the PEs of their own group only. For full access property of DCMIN, each of the $(i+1)^{th}$ stage should perform its own $Dcube_i$ function. If any of these $Dcube_i$ functions is not performed, and the stage is bypassed, then the network will divide itself into 4 independent subnetworks of size 4^{n-1}. Each of these subnetworks will behave as a DCMIN of size 4^{n-1} and PEs of one subnetwork will not be able to communicate with the PEs of other subnetworks unless $Dcube_i$ function is performed.

Thus, to partition any n stage DCMIN, there are n choices of selecting the PE addresses in each group. Each choice is based on a different positional digit of input/output addresses represented in quaternary system.

In a DCMIN if any Dcube_i function is not to be used then the corresponding $(i+1)^{th}$ stage is disabled either by using multiplexers and demultiplexers or by forcing the stage to mode 'O' condition.

Theorem 6.1

In a given DCMIN of size $N = 4^n$, with physical addresses of the input terminals in quaternary number system, if any stage $(i+1)^{th} \leq n$, is forced to mode 'O' condition, the remaining network will always be partitioned into four independent DCMINs of size $N' = 4^{n-1}$.

Proof

In a DCMIN any $(i+1)^{th}$ stage can be disabled from performing the Dcube_i function by forcing it to mode 'O' condition. By disabling the $(i+1)^{th}$ stage the terminal addresses will be divided into four groups

Group 1 - Contains terminals with i^{th} positional digit as 0
Group 2 - Contains terminals with i^{th} positional digit as 1
Group 3 - Contains terminals with i^{th} positional digit as 2
Group 4 - Contains terminals with i^{th} positional digit as 3

These groups cannot communicate amongst themselves and become independent. Now if the i^{th} positional digit is dropped from the addresses of each group formed above then the addresses will have (n-1) digits and each subnetwork can be addressed as a DCMIN of size 4^{n-1}. The new network can again perform Dcube_i functions where $0 \leq i \leq (n-2)$.

Thus, the network of size $N = 4^n$ is divided into 4 independent subnetworks of size $N' = 4^{n-1}$.

$\square$

Example 6.3

Consider a three stage DCMIN depicted in Fig.6.5-6.7.

In Fig.6.5 the network is divided into four subnetworks (A,B,C and D) of size 16x16 each by forcing the third stage into mode 'O' condition. It can be seen that the addresses in each group have their second positional digit common and it is different for different groups. Similarly in Figs.6.6 and 6.7 the second and third stages respectively are forced to mode 'O' operation. Table 6.2 gives the addresses of PEs in each subnetwork when a 3-stage DCMIN is partitioned by forcing the different stages to mode 'O'.

Since each subnetwork has the properties of DCMIN, it can be further subdivided into 4 independent DCMINs of lower size. This process of dividing subnetworks into independent compartments can be applied recursively to create any size of subnetwork till the lowest value of the network of size 4^1(4x4) is reached. If a large scale parallel system has a DCMIN of size 4^n then the subnetworks obtained by repeated divisions will have the following properties:

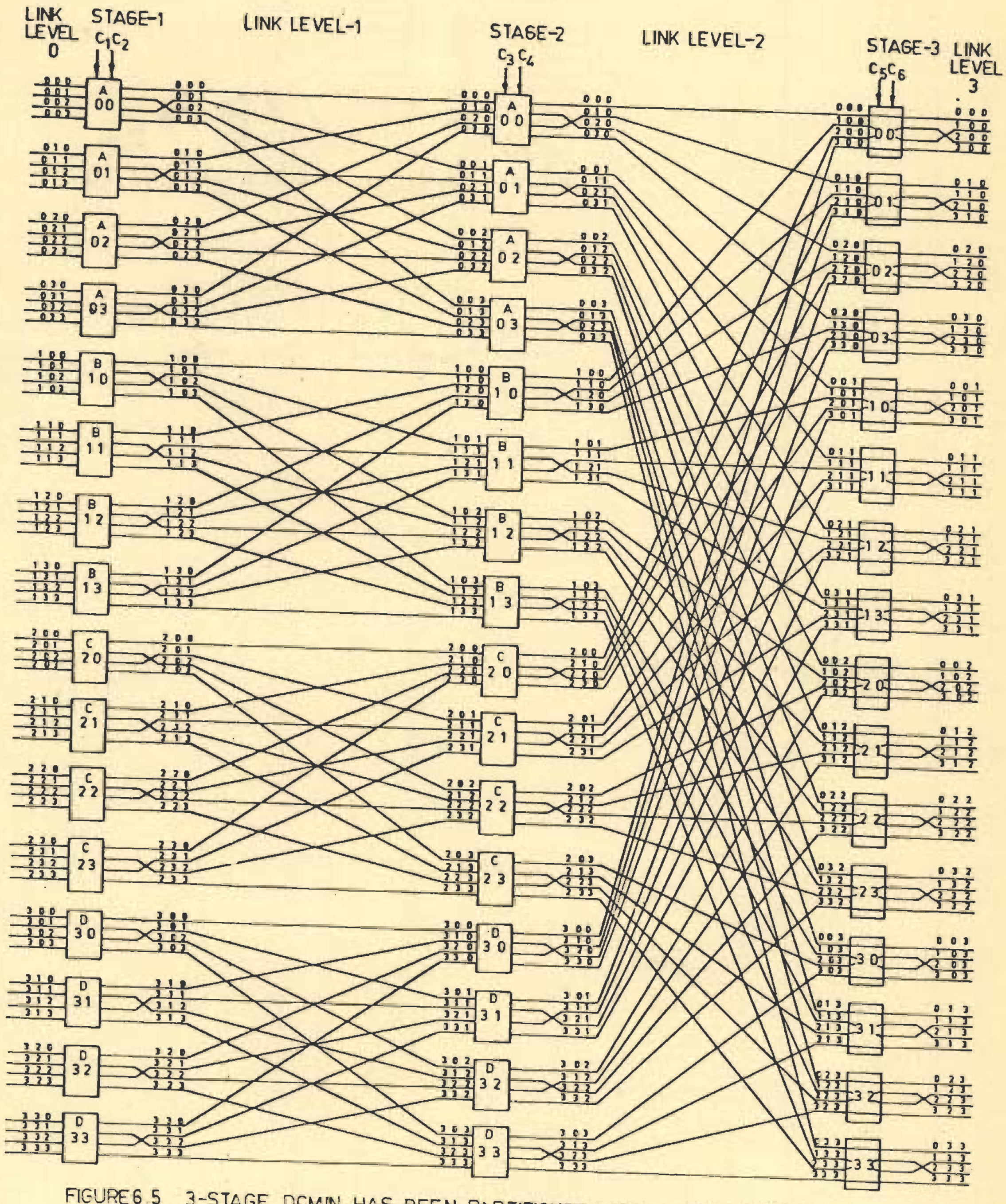


FIGURE 6.5 3-STAGE DCMIN HAS BEEN PARTITIONED INTO 4-INDEPENDENT 2-STAGE DCMINs, ADDRESSES OF PEs IN EACH PARTITIONED NETWORK AGREE IN THEIR HIGHER BIT POSITION. DIMSEs MARKED A, B, C AND D INDICATE THE DIMSEs USED BY 4 PARTITIONED NETWORKS RESPECTIVELY BY FORCING STAGE-3 TO MODE 0 CONDITION.

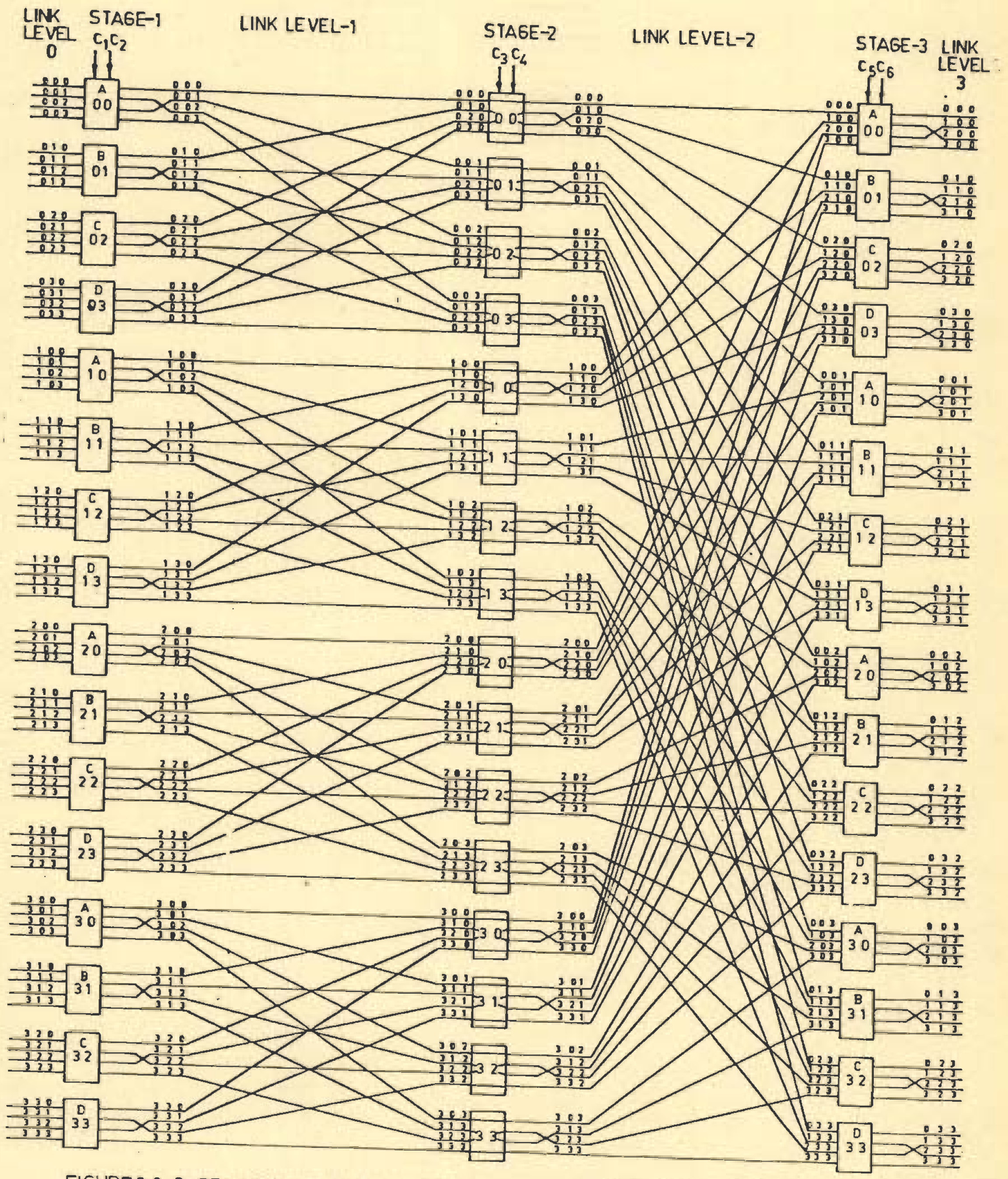


FIGURE 6.6 3-STAGE DCMIN HAS BEEN PARTITIONED INTO 4-INDEPENDENT 2-STAGE DCMINs. ADDRESSES OF PEs IN EACH PARTITIONED NETWORK AGREE IN THEIR MIDDLE BIT POSITION. DIMSEs MARKED A, B, C AND D INDICATE THE DIMSEs USED BY 4 PARTITIONED NETWORKS BY FORCING STAGE-2 TO MODE 0 CONDITION.

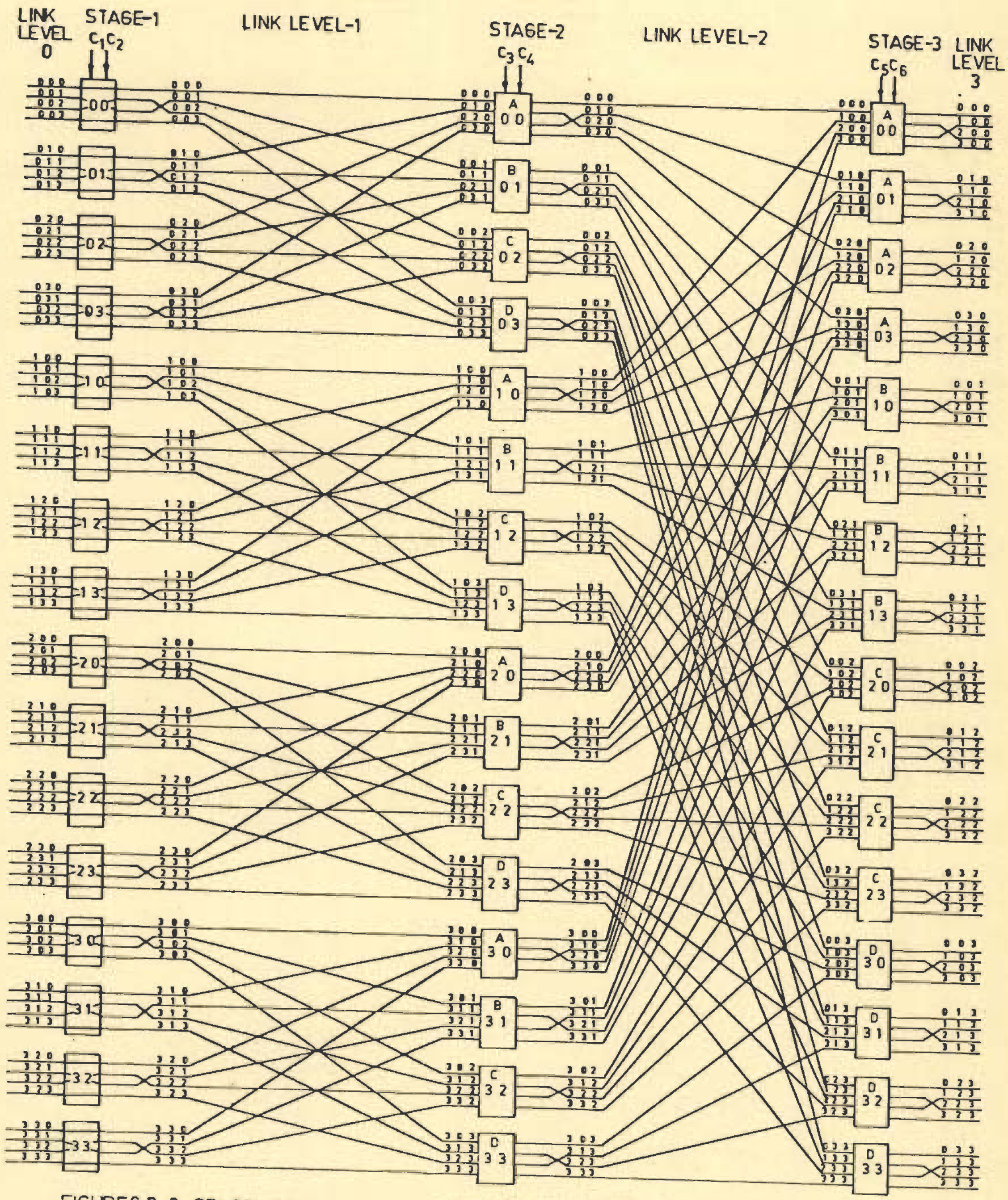


FIGURE 6.7 3-STAGE DCMIN HAS BEEN PARTITIONED INTO 4-INDEPENDENT 2-STAGE DCMINs. ADDRESSES OF PEs IN EACH PARTITIONED NETWORK AGREE IN THEIR LOWER BIT POSITION. DIMSEs MARKED A, B, C AND D INDICATE DIMSEs USED BY 4 PARTITIONED NETWORKS RESPECTIVELY BY FORCING STAGE-1 TO MODE 0 CONDITION.

STAGE-3 IN MODE 'O'				STAGE-2 IN MODE 'O'				STAGE-1 IN MODE 'O'			
ADDRESSES IN SEBNETWORKS				ADDRESSES IN SUBNETWORKS				ADDRESSES IN SUBNETWORKS			
A	B	C	D	A	B	C	D	A	B	C	D
000	100	200	300	000	010	020	030	000	001	002	003
001	101	201	301	001	011	021	031	010	011	012	013
002	102	202	302	002	012	022	032	020	021	022	023
003	103	203	303	003	013	023	033	030	031	032	033
010	110	210	310	100	110	120	130	100	101	102	103
011	111	211	311	101	111	121	131	110	111	112	113
012	112	212	312	102	112	122	132	120	121	122	123
013	113	213	313	103	113	123	133	130	131	132	133
020	120	220	320	200	210	220	230	200	201	202	203
021	121	221	321	201	211	221	231	210	211	212	213
022	122	222	322	202	212	222	232	220	221	222	223
023	123	223	323	203	213	223	233	230	231	232	233
030	130	230	330	300	310	320	330	300	301	302	303
031	131	231	331	301	311	321	331	310	321	312	313
032	132	232	332	302	312	322	332	320	331	322	323
033	133	233	333	303	313	323	333	330	331	332	333

TABLE 6.2 PHYSICAL ADDRESSES OF PEs IN EACH SUBNETWORK BY FORCING DIFFERENT STAGES OF 3-STAGE DCMIN TO MODE 'O' OPERATION.

1. The size of each subnetwork must be a power of 4.
2. The physical addresses (in quaternary system) of input terminals of a subnetwork of size 4^s must have the identical digits in any fixed set of (n-s) digit positions.

The partitioning property of DCMIN has been demonstrated in this section. The ability of DCMIN to partition to smaller size DCMINs makes it suitable for use in large parallel systems where partitionability is an important aspect.

6.3 THE EXTRA STAGE DCMIN (EDCMIN)

In Section 5.2 the fault-tolerance aspect of DCMIN through redundancy in time was discussed. In this section, an alternative approach through the hardware redundancy is proposed. The DCMIN is a unique path network, i.e., between any input/output pair there is only one unique path. This unique path is called the primary path. In case there is a fault on this unique path, either due to link or switch failure, the communication between the corresponding input/output pair is disrupted.

In this section it is established that if an additional stage of DIMSEs is added as $(n+1)^{th}$ stage, three additional redundant paths between every input/output pair will be generated which will make the network inherently fault-tolerant.

Let this fault-tolerant topology be called Extra Stage DCMIN (EDCMIN). The fault-tolerance properties of this topology vis-a-vis DCMIN are discussed hereunder.

6.3.1 Topology of EDCMIN

EDCMIN is formed from the DCMIN by adding an additional (n+1)th stage at the output of the network. This is shown in Fig.6.8 for a three stage DCMIN. Under fault-free conditions the extra stage will always implement Dcube_0 interconnection function. Thus, with this constraint the EDCMIN will have its first and (n+1)th stages implementing Dcube_0 function. Each of these stages can be activated independently. Thus, a topology consisting of (n+1) stages is evolved which can function either as n-stage DCMIN or as a (n+1) stage fault-tolerant EDCMIN. Under fault-free conditions any n adjacent stages need be enabled. As discussed earlier in Section 6.2, the enabling and disabling of a stage can be accomplished in two ways.

(i) By forcing a stage into mode '0' condition, or

(ii) by providing multiplexers and demultiplexers with each DIMSE.

It is obvious that to have n-stage DCMIN, either the first stage or the last stage of EDCMIN need be disabled. Figs.6.9 and 6.10 show a 4-stage EDCMIN with last and the first stage respectively disabled to achieve a three stage conventional DCMIN.

For both, the conventional n stage DCMIN and (n+1) stage EDCMIN, any (i+1)th stage $0 \leq i \leq (n-1)$ determines the ith positional digit of the address of the output port to which data are sent. In conventional DCMIN, the first stage determines the 0th positional digit of the output addresses. However, in EDCMIN both, the first and (n+1)th stages determine the 0th

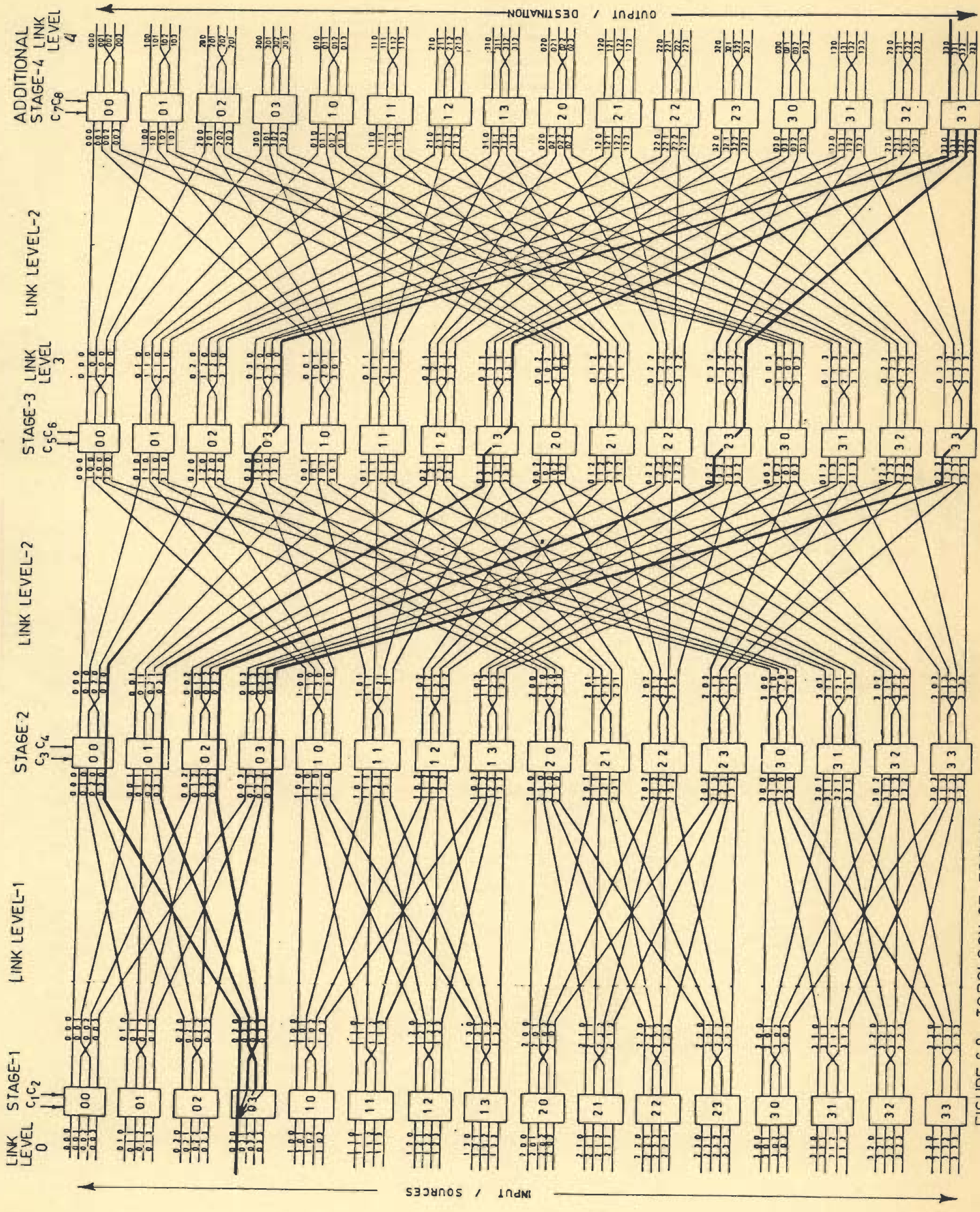


FIGURE 6.8 TOPOLOGY OF EDCMIN

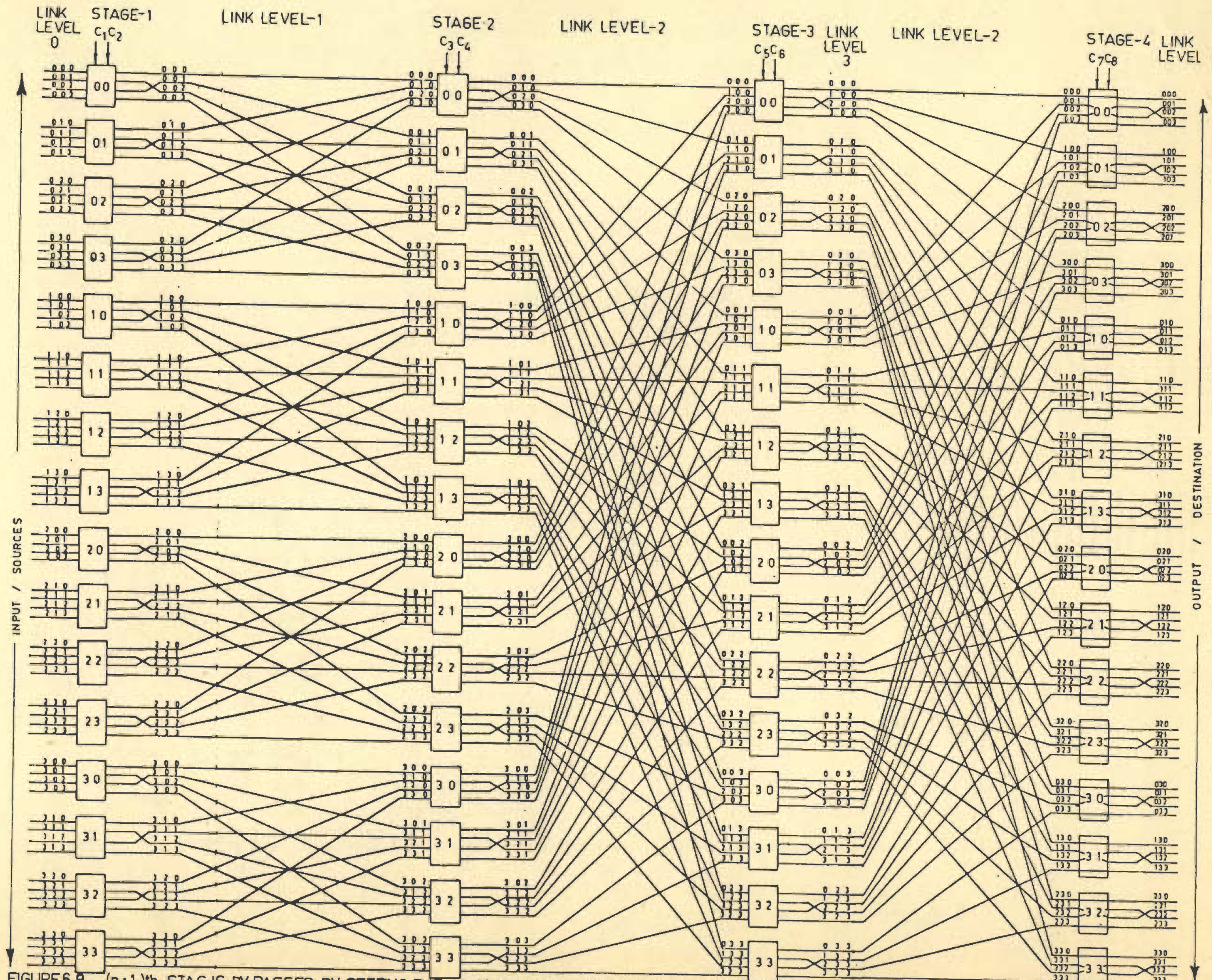


FIGURE 6.9 (n+1)th. STAG IS BY PASSED BY SETTING THE CONTROL MODE TO ZERO. THIS CAN BE BY PASSED BY USE OF MULTIPLEXERS/ DEMULTIPLEXERS ALSO.

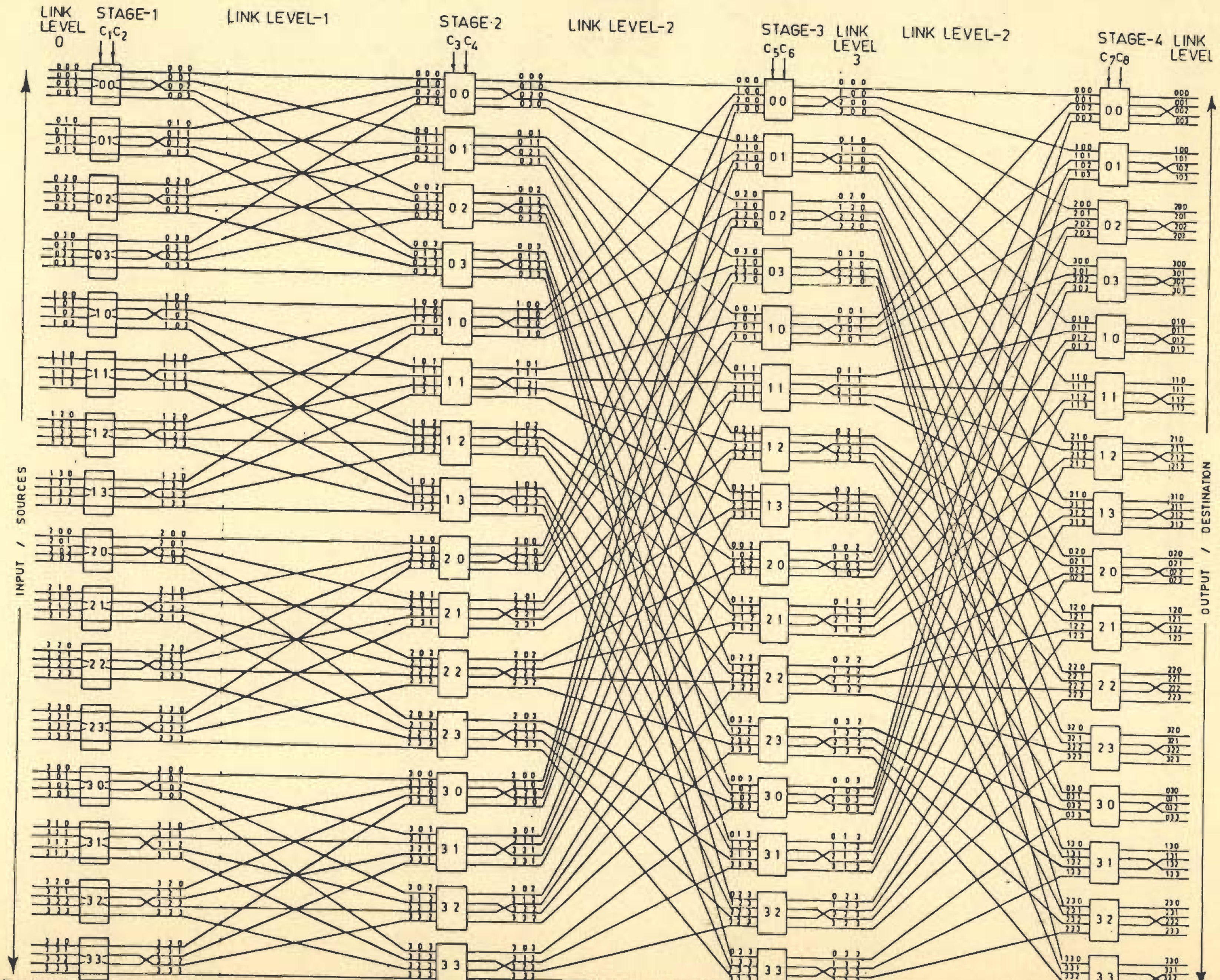


FIGURE 6.10 STAGE-1 IS DISABLED BY PUTTING THE CONTROL TO MODE 0. IT CAN ALSO BE BY PASSED BY USING MULTIPLEXERS AND DEMULTIPLEXERS.

positional digit. Forcing (n+1)th stage into mode '0', the EDCMIN topology behaves as a conventional unique path DCMIN. However, by setting the (n+1)th stage in the remaining three mode-settings i.e. 1,2, or 3, three other alternative routes not present in the conventional DCMIN are made available.

6.3.2 Fault-Tolerance of DCMIN

In the fault model to be used, failures may occur in the DIMSEs or links connecting the stages of DIMSEs. However, the input/output links (terminal links) are always assumed to be fault free. If the input/output links are faulty the associated device will have no access to the network. Such faults will not yield fault-tolerance and are not considered.

The existance of at least two paths between any source destination pair is a necessary condition for fault-tolerance. Redundant paths allow continued communication between a given source and destination if after a fault occurs at least one path remains fully functional. This means that the redundant paths between 2nd stage and nth stage should be disjoint, i.e., there should not be any link or DIMSE in common.

Theorem 6.2

An EDCMIN, with all its (n+1) stages functional, provides four alternate paths between a source-destination pair having [122]

(i) No link in common and

(ii) No DIMSE in common from stage-2 to nth stage.

Proof

Proof of the theorem is given in three parts. Part (I) proves that there are four distinct paths. Part (II) proves that there are no common links and part (III) proves that no DIMSEs are common from stage-2 to (n)th stage.

(I) Stage-1 of EDCMIN can perform Dcube$_0$ function. Hence, it allows any input terminal 'S' to access four distinct inputs of second stage, S and Dcube$_0$(S). Since all the stages of EDCMIN are functional, stage-2 to (n+1) form a conventional n-stage DCMIN. Since DCMIN is a unique path network, the four inputs to stage-2 will provide four alternate paths from any source S to destination D.

(II) A source S can connect to stage-2 inputs S or Dcube$_0$ (S). These four inputs differ in 0th digit position. Other than stage-1, only stage-(n+1) can cause a source to be mapped to a destination which differs from the source in the 0th order digit position. Therefore, the path from any source S through stage-2 input S to a destination D contains only links with addresses which agree with S in the 0th order digit position. Similarly, the path through stage-2 input Dcube$_0$(S) contains only links with addresses agreeing with Dcube$_0$(S) in the 0th order position. Thus no link is part of both paths.

(III) Since the four paths have the same source-destination pair, they will pass through the same DIMSE in stage-1 and stage (n+1). No DIMSE in 2nd to n^{th} stage has input addresses which differ in the 0^{th} order digit position. One path from S to D, out of the four generated by EDCMIN contains only links with addresses agreeing with S in the 0^{th} order digit position. The other three paths have links with addresses which are relative complement (defined in Section 3.1) of S in the 0^{th}-order digit position. Therefore, no DIMSE from stage-2 to stage-n is common in the 4 distinct paths.

Thus, the theorem is proved.

$\square$

Example 6.4

Consider Fig.6.8 for N = 64, source (030) can enter stage-2 at link address (030) or addresses $\mathrm{Dcube}_0(030)$ i.e. (031), (032) or (033). Thus, for any arbitrary source-destination pair (030) to (330) there are four distinct paths. The primary path enters the stage-2 at link address 030 and follows the same path as it should have followed in the natural course with stage-4 disabled. However, if the stage-4 is enabled the data may also follow any of the three auxiliary paths by entering at stage-2 through the addresses $\mathrm{Dcube}_0(033)$ as shown in the figure. The primary path from source (030) to destination (330) enters stage-2 at link address (030). Between stages-2 and 3 it goes through link (030) and between stages-3 and 4 through link (330). Both link addresses (030) and (330) agree with S (030) in 0^{th} digit position. Similarly, the link addresses of other three paths agree in respective 0^{th} digit positions.

It can be seen from the figure that the 4 paths are independent paths. No link is common and no DIMSE in second and third stages is common.

6.3.3 Routing Schemes For EDCMIN

In EDCMIN, under fault free conditions, the $(n+1)^{th}$ stage is disabled. Accordingly, the routing tag can be found as in the conventional DCMIN, using either destination tag algorithm or Exclusive-OR algorithm. This has been discussed in Section 4.6. If after fault diagnosis, primary path is found faulty one of the secondary paths is selected. The primary and secondary paths are routed through the Ist stage output links which differ in the lower order positional-digit only. Accordingly, the routing tag from stage-2 to stage-n will remain the same as for the primary path. Destination tag is not affected by the source address. Hence for destination tag algorithm, the routing tag digit provided for stage-1, will be used by (n+1)th stage. However, in Exclusive-OR routing scheme, source address pays an important role in measuring the tag bit. Since, in the secondary path two low-order bits (in binary) in the address of the link feeding the second stage are changed, the routing tag bits for (n+1)th stage will also be different from the bits used by stage-1. The routing tag bits for (n+1)th stage can be determined as follows:

Let L_1 and L_0 be two lower binary bits of the link selected to feed the stage-2 and D_1 and D_0 be the two lower binary bits of the destination address. Then the routing tag bits for (n+1)th stage will be

$$C_{2n+2} = L_1 \oplus D_1$$

$$C_{2n+1} = L_0 \oplus D_0$$

In Section 6.3 it has been demonstrated that DCMIN can be made fault - tolerant by adding an additional stage. The additional stage generates three additional paths for any source/destination pair. The EDCMIN will maintain continuous communication till at least one of the paths is fully functional.

6.4 CONCLUSION

In this chapter the three important aspects of the DCMIN - mathematical modelling, partitionability and fault-tolerance through multiple paths have been investigated. It has been established that the mapping matrix of DCMIN has a simple and regular structure. Also a procedure has been evolved to interpret the network from any square mapping matrix of order 4^n, where n is any positive integer. Partitioning property of DCMIN as demonstrated in this chapter establishes that the network can be implemented in any large sized multi-processor system. Fault-tolerance of DCMIN through multiple passes was demonstrated in Chapter V. In this chapter fault-tolerance through multiple paths has been established.

It is worth mentioning that these and other properties of DCMIN discussed in preceding chapters, categorize it in the general class of cube networks, which have been implemented and used in many practical systems.

CHAPTER VII

TRUNCATED DUAL CUBE MULTISTAGE INTERCONNECTION NETWORK

The DCMIN as an interconnection network of size 4^n has been proposed in Section 4.3. In parallel processing environments practical situations may sometimes arise where the PEs need to shuffle data through a network of size 2^{2n-1} for any positive integer $n \geq 2$. In such cases use of topology described in Section 4.3 will result in unnecessary hardware wastage. One possibility would be to reconfigure DIMSEs as two units of 2x2 switching elements. However, this will negate the very philosophy of DIMSEs to reduce the number of stages. In this chapter it has been illustrated that DCMIN can be configured to a size of N/2 while still retaining all its other properties. An N/2 DCMIN will always yield a network of size 2^{2n-1}. The proposed topology of N/2 DCMIN is designated as Truncated DCMIN or TDCMIN. In this chapter the properties of TDCMIN in relation to those of DCMIN will be discussed and variations, wherever occuring will be highlighted.

7.1 TOPOLOGY OF TDCMIN

Let $M = N/2 = 2^{2n-1}$ for $n \geq 2$, giving the lowest value of M as 8. A network using DIMSEs with full access property would need at least two stages. Referring Fig.3.2(a) once again, the eight PEs can be represented as the lower eight vertices of a dual cube. The lower half of the dual cube is redrawn as Fig.7.1(a) to accomodate eight nodes, i.e., eight PEs only. Fig.7.1(b) depicts the stack of the four units of lower-half of the dual cube to accomodate 32 PEs.

Let a vector V be of length 8. If the sets of adjacent elements of this vector are placed in horizontal rectangles as shown in Fig.7.1(a) then performing the perfect shuffle on the elements of the vector will rotate them in sets as shown in the vertical rectangles.

For an 8x8 TDCMIN, the two stages can be interconnected through 4-shuffle pattern as in DCMIN. However, the use of 4-shuffle will disturb the symmetry and dual cube analogy of the network. Fig.7.1 depicts that the use of perfect-shuffle, instead of 4-shuffle interconnection pattern, between the first and second stages will maintain both, the symmetry and dual cube analogy. In addition, by using the perfect-shuffle between the first two stages of TDCMIN topology, only three control signals will be required to activate these two stages. This aspect is further explained in Section 7.3. Thus, Fig.7.2 depicts a two stage TDCMIN.

The dual cube analogy of 32x32 TDCMIN is demonstrated in Fig.7.1(b). It is observed that the stages-2 and 3 are interconnected through 4-shuffle pattern. Accordingly, Fig.7.3 depicts a 32x32 TDCMIN.

The above analogy can be extended to a TDCMIN of any size $M = 2^{2n-1}$ for $n \geq 2$. Also, TDCMIN will have n number of stages. Thus, the TDCMIN topology for $n \geq 3$ becomes recursive in nature and the Corollary 4.1 can be modified and made applicable to TDCMIN, as stated below:

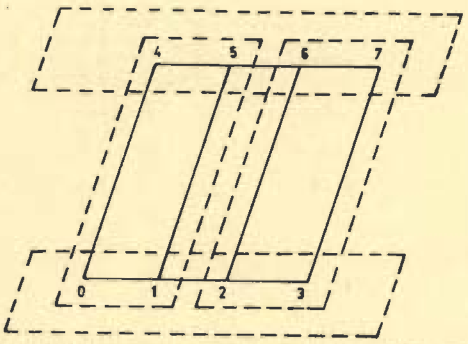


FIGURE 7.1 (a) Lower half vertices of dual cube. Dashed horizontal blocks contain the set of PE_s in the Ist stage and perpendicular blocks contain sets of PE_s in the IInd stage.

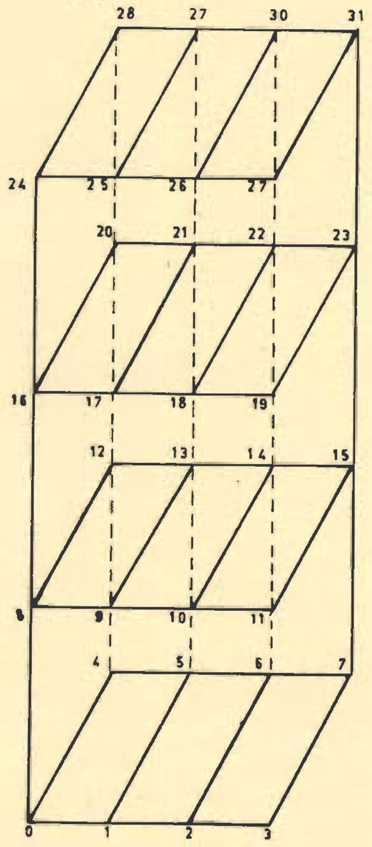


FIGURE 7.1 (b) 32 PE_s placed on the vertices of 4 – lower half of the dual cube units placed in a stack.

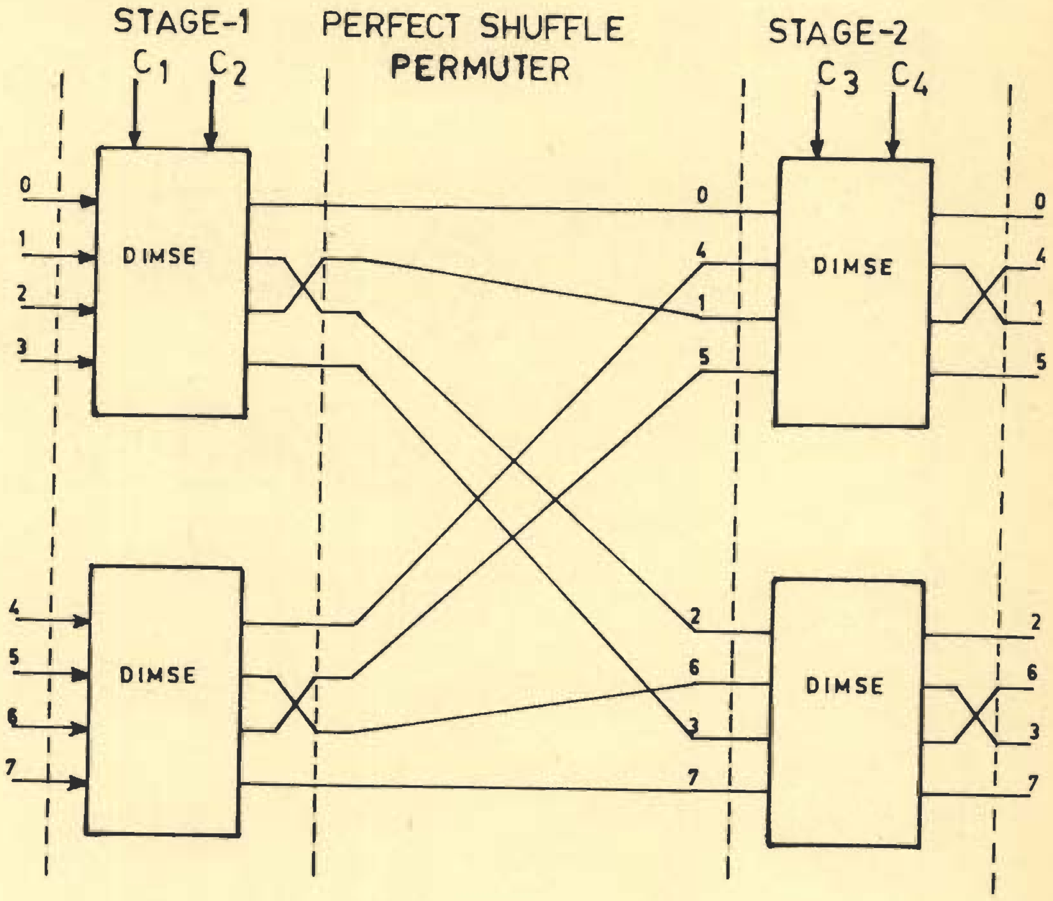


FIGURE 7.2 8 X 8 TDCMIN USING DIMSEs

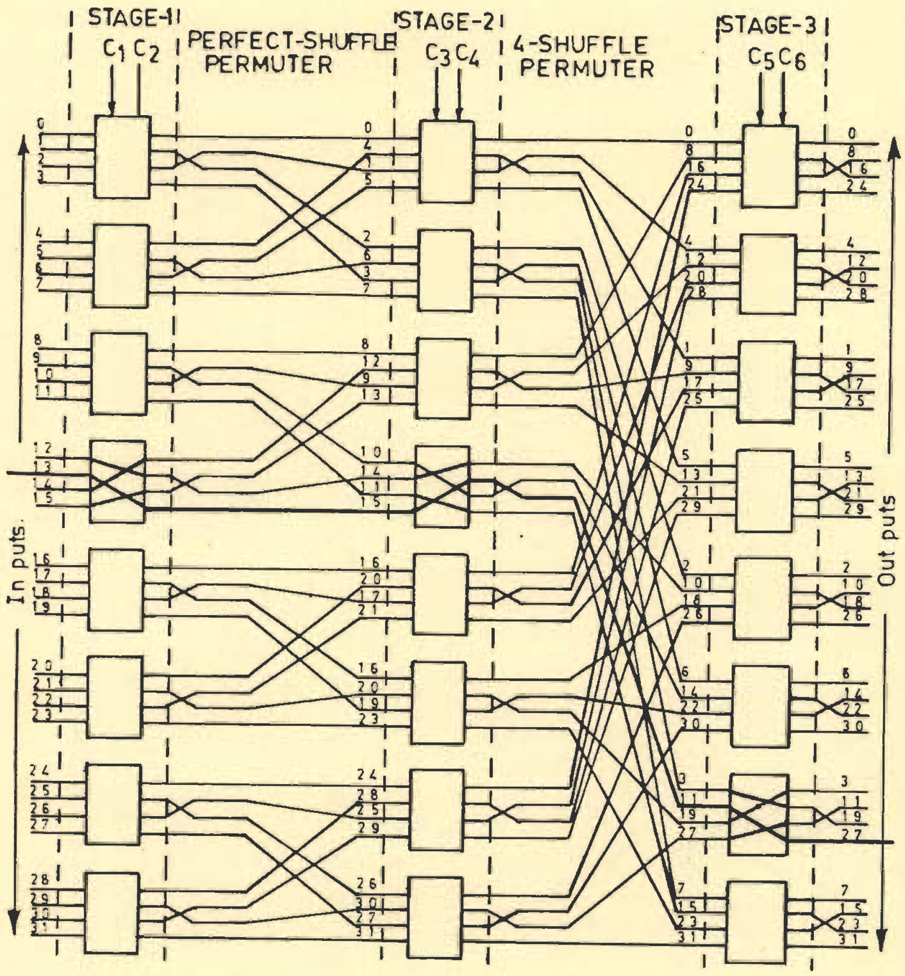


FIGURE 7.3 3-STAGE TDCMIN WITH M=32 DARK LINES INDICATE THE ROUTE FROM SOURCE-13 TO DESTINATION 27.

In a TDCMIN topology a k-stage network T_k, $k \geq 3$, can be constructed from a stack of 4-units of T_{k-1} and a stack of M/4 DIMSEs according to the following equation:

$$T_k = (T_{k-1} + T_{k-1} + T_{k+1} + T_{k-1}).P.S$$

where,

P = 4-shuffle permuter,

S = stack of M/4 DIMSEs,

'+' sign and '.' sign are used for mathematical representation of stack and cascade respectively.

7.2 CONTROL STRATEGY OF TDCMIN

The basic switching element being 4x4 DIMSE, each stage will need two control signals. First and second stages are connected through perfect shuffle permuter and the remaining topology is similar to DCMIN. Hence the control strategy of first and second stages needs special consideration as follows.

In Fig.7.2 for arbitrary input/output connections stage-1 is required to operate in all the four working modes 0,1,2 and 3 of DIMSE as discussed in Section 4.1. This is depicted in Fig.7.4(a). In stage-2 all the four inputs of any DIMSE are not from the four different DIMSEs. Two outputs of any one DIMSE of stage-1 are connected as two inputs to a particular DIMSE of stage-2. This is demonstrated in Fig.7.4(b). In the light of the mapping matrices of DIMSEs, an analysis of Figs.7.4(a) and 7.4(b), reveals that for full-access property, stage-2 in TDCMIN, is required to operate only in two modes, i.e., mode '0' and '2'. Thus the control line C_4 can be permanently deactivated by connecting it to logical '0' value.

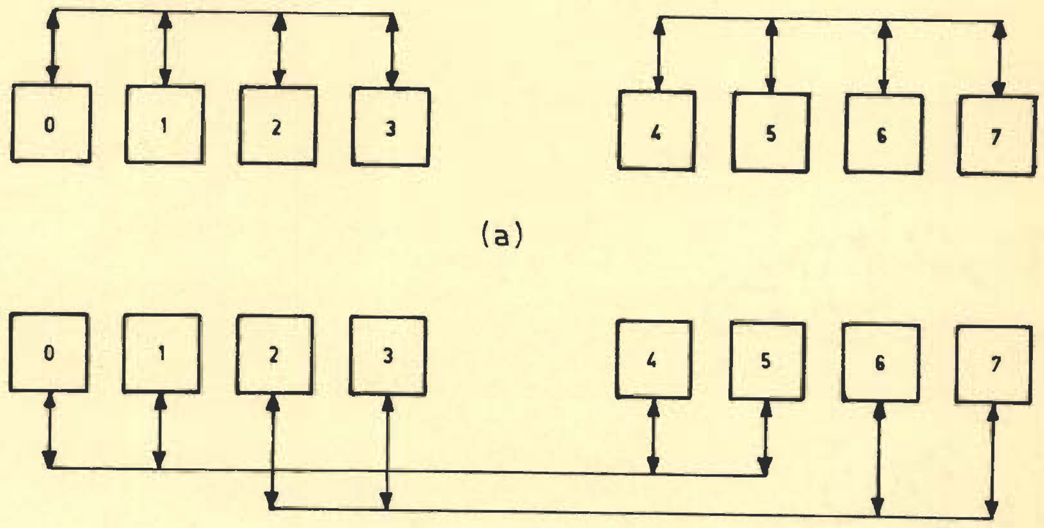


FIGURE 7.4: 8 – PE_s AS CONNECTED TO $DIMSE_s$ INPUTS.
(a) Stage-1 Connections of PEs
(b) Stage-2 Connections of PEs

Hence only 3 control lines will be operative for the first two stages.

Since the remaining stages are connected through the 4-shuffle permuters, both the control signals of all other stages should remain operative.

7.3 ROUTING SCHEME OF TDCMIN

In TDCMIN topology the control strategy remains identical to DCMIN topology, except that the control signal C_4 is deactivated permanently. Accordingly, the routing scheme of DCMIN can also be applied to TDCMIN, after incorporating appropriate modification for C_4. The derivation of Exclusive-OR tag scheme for TDCMIN is explained hereunder.

As the number of PEs in this case is 2^{2n-1} for any n-stage network, the addresses will have odd number of binary bits. However, since each stage requires two control signals, the the routing tag needs even number of binary bits. Accordingly, the procedure to find the routing tag is as follows:

Let $S = s_{2n-1}s_{2n-2}.....s_i.....s_2s_1$ be the source address in binary

and $D = d_{2n-1}d_{2n-2}.....d_i.....d_2d_1$ The destination address in binary

Then bitwise Exclusive-OR will give

$R' = r'_{2n-1}r'_{2n-2}.....r'_i.....r'_2r'_1$

where $r'_i = s_i \oplus d_i$

Thus, the routing tag will be obtained by shifting all the bits from r'_4 to r'_{2n-1}, one bit position left and the bit position of r'_4 is made 'O' giving routing tag as

$$R = r_{2n} r_{2n-1} \cdots\cdots r_i \cdots\cdots r_5 \; 0 \; r_2 \; r_3 \; r_1$$

where

$r_1 = r'_1$

$r_2 = r'_2$

$r_3 = r'_3$

$r_4 = 0$ and

$r_i = r'_{i-1}$ where $5 \leq i \leq 2n$

Then r_{2n-1} r_{2n} specify the values (00, 01, 10 or 11) of the control signals of nth stage.

Example 7.1

Consider finding the routing tag for connecting source 13 (01101) to destination 27 (11011) in Fig.7.3

$$\begin{array}{lccccc} S = & 0 & 1 & 1 & 0 & 1 \\ D = & 1 & 1 & 0 & 1 & 1 \\ \hline R' = & 1 & 0 & 1 & 1 & 0 \\ \hline \end{array}$$

Bitwise Exclusive-OR

Since the network is a three stage network, it requires 6-bit routing tag. Thus R' may be converted to R.

$$\begin{array}{lcccccccc} R' & = & & 1 & 0 & 1 & 1 & 0 \\ & & 6 & 5 & 4 & 3 & 2 & 1 \\ & & \downarrow & \downarrow & & \downarrow & \downarrow & \downarrow \\ R & = & 1 & 0 & 0 & 1 & 1 & 0 \\ & & C_6 & C_5 & C_4 & C_3 & C_2 & C_1 \end{array}$$

Accordingly, the path from Source (13) to Destination (27) is depicted in Fig.7.3 by dark lines.

7.4 PARTITIONING OF TDCMIN

Like DCMIN, the TDCMIN can also be partitioned into smaller subnetworks. However, because of the topology of TDCMIN, the network will be divided into two independent subnetworks when partitioning is affected through stage-1 or stage-2 and into four independent subnetworks through any other stage i, where $3 \leq i \leq n$. Accordingly, there are n ways to partition an n-stage TDCMIN. Partitioning can be affected either by forcing a particular stage in mode 'O' or bypassing it through multiplexer and demultiplexers as was discussed in Section 6.2.

7.4.1 Partitioning Results

Let the input terminals be physically numbered from O to (M-1) in a TDCMIN of size M. In binary representation, the addresses will have (2n-1) binary bits. The networks can be partitioned as follows:

(i) Stage-1 is forced to Mode 'O' condition

Subnetwork A: Containing input terminals with addresses having Ist positional binary bit (2^1-position) as 'O'.

Subnetwork B: Containing input terminals with addresses having Ist positional binary bit (2^1-position) as '1'.

(ii) Stage-2 is forced to Mode 'O' condition

Subnetwork A: Containing input terminals with addresses having 2nd positional binary bit (2^2-position) as 'O'.

Subnetwork B: Containing input terminals with addresses having 2nd positional binary bit (2^2-position) as '1'.

(iii) Stage-i, $3 \leq i \leq n$, is forced to Mode 'O' condition

In this case there will be four subnetworks:

Subnetwork A: Containing input terminals with addresses having $(2i-2)^{th}$ positional binary bit (2^{2i-2}-position)as'O' and also $(2i-3)^{th}$ positional binary bit (2^{2i-3}-position)as'O'

Subnetwork B: Containing input terminals with addresses having $(2i-2)^{th}$ positional binary bit as 'O' and $(2i-3)^{th}$ positional binary bit as '1'

Subnetwork C: Containing input terminals with addresses having $(2i-2)^{th}$ positional binary bit as '1' and $(2i-3)^{th}$ positional binary bit as 'O'

Subnetwork D: Containing input terminals with addresses having $(2i-2)^{th}$ positional binary bit as '1' and also $(2i-3)^{th}$ positional binary bit as '1'

Example 7.2

Figs.7.5-7.7 show three stage TDCMIN partitioned through first second and third stage respectively. Partitioning through first and second stages divides the networks into two subnetworks in each case. Partitioning through stage-3 divides it into four subnetworks.

7.5 MATRIX REPRESENTATION OF TDCMIN

The matrix representation of DCMIN was discussed in Section 6.1. Since the topology of TDCMIN is similar to DCMIN, it can also be represented mathematically by Kronecker product of mapping matrices of DIMSEs. The results derived in Section 6.1.2 for DCMIN can be applied in this case also after carrying out the modifications according to the changes in the topology of TDCMIN. For this purpose Equation (6.11) is reproduced below, as Equation (7.1).

$$[M]_{C_1C_2\ldots C_{2n-1}C_{2n}} = [M]_{C_{2n-1}C_{2n}} \otimes [M]_{C_{2n-3}C_{2n-2}} \otimes \ldots\ldots \otimes [M]_{C_3C_4} \otimes [M]_{C_1C_2} \qquad ..(7.1)$$

It has been demonstrated in Section 7.3 that in TDCMIN topology, control line C_4 is always connected to logical value 'O'. Accordingly, the stage-2 has only two operative modes-mode 'O' (C_3=C_4=O) and mode '2' (C_3=1, C_4=O). Thus, the control signals C_3 and C_4, with C_4 remaining constant at $\bar{C}_4$, will have only two possible combinations as $(\bar{C}_3\bar{C}_4)$ and $(C_3\bar{C}_4)$. Hence,

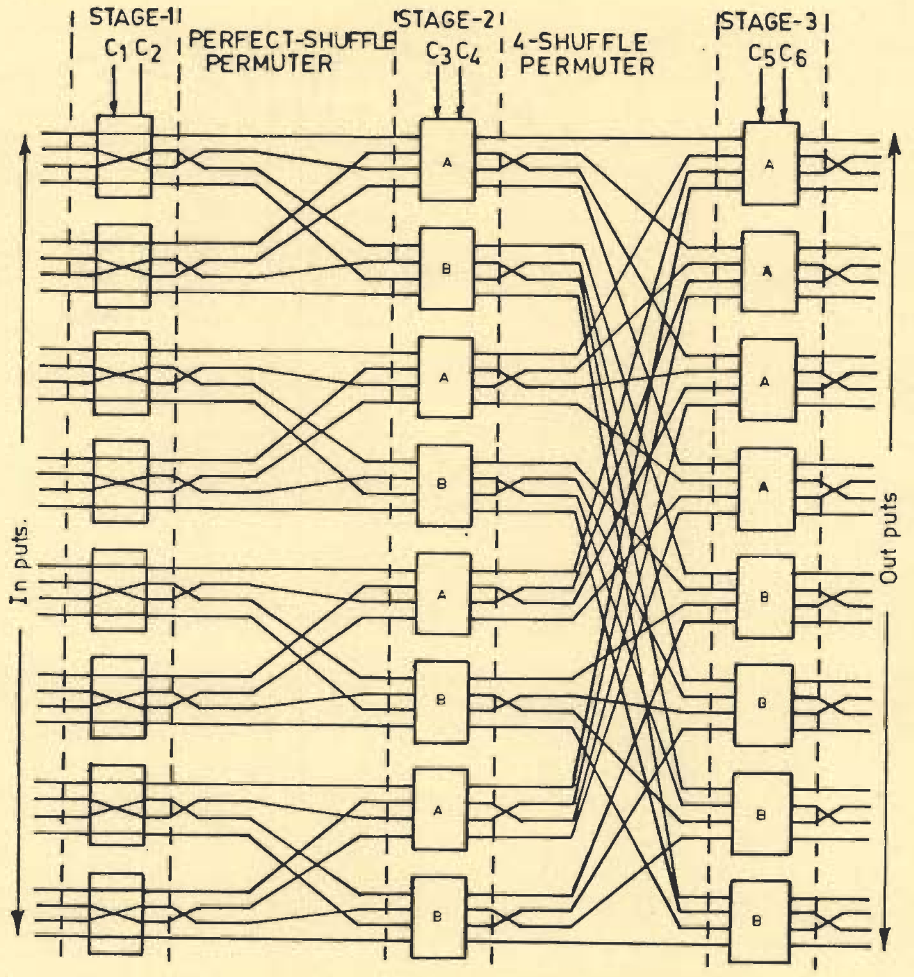


FIGURE 7.5 TDCMIN OF SIZE M=32, PARTITIONED INTO TWO SUBNETWORKS BY FORCING STAGE-1 TO MODE 'O' CONDITION. A AND B INDICATE THE DIMSEs USED IN TWO SUBNETWORKS RESPECTIVELY.

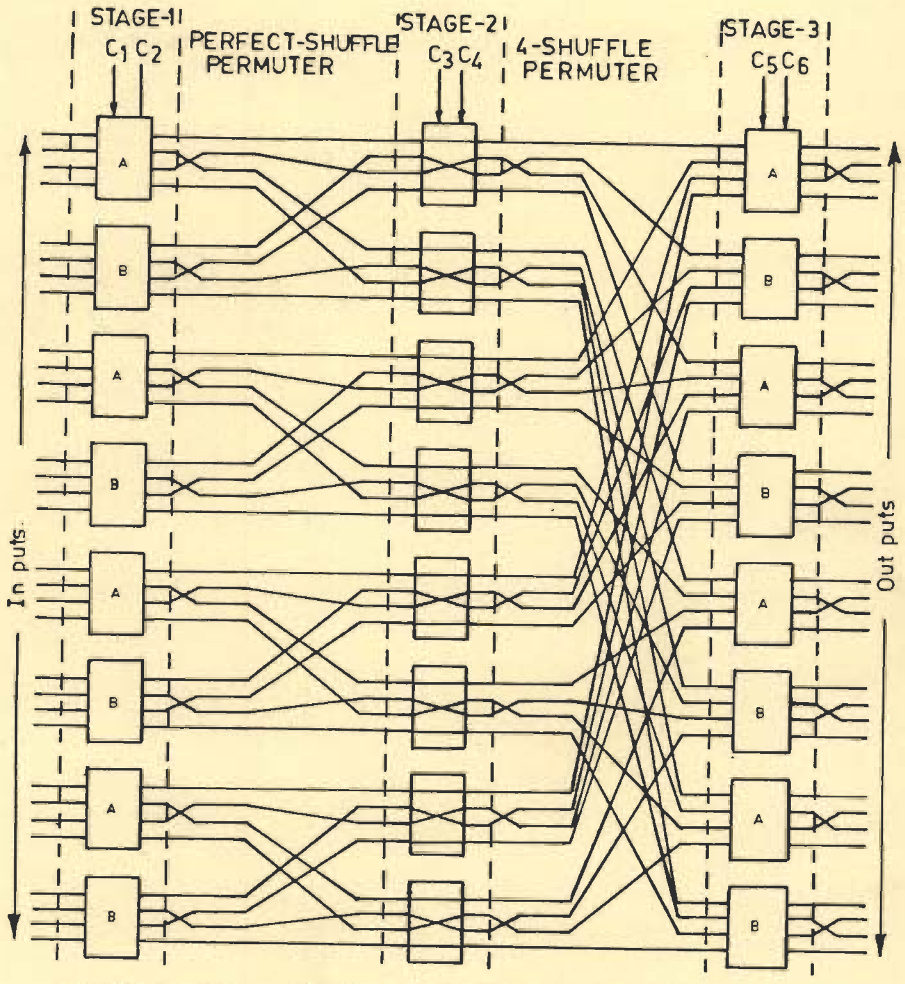


FIGURE 7.6 TDCMIN OF SIZE M=32, PARTITIONED INTO TWO SUBNETWORKS BY FORCING STAGE-2 TO MODE '0' CONDITION. A AND B INDICATE THE DIMSEs USED IN TWO SUBNETWORKS RESPECTIVELY.

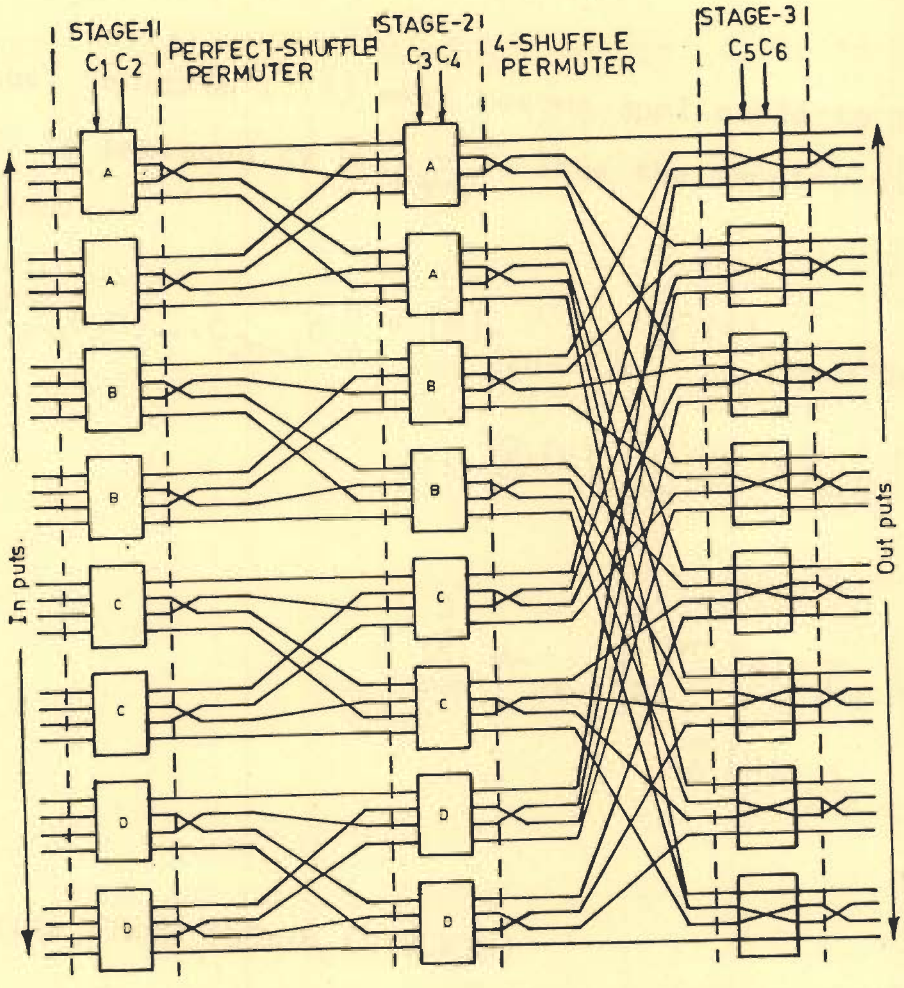


FIGURE 7.7 TDCMIN OF SIZE M = 32, PARTITIONED INTO FOUR SUBNETWORKS BY FORCING STAGE-3 TO MODE 'O' CONDITION. A, B, C AND D INDICATE DIMSEs USED IN FOUR SUBNETWORKS RESPECTIVELY.

$[M]_{C_3C_4}$ will be modified as

$$[M]_{C_3\bar{C}_4} = \begin{bmatrix} \bar{C}_3\bar{C}_4 & C_3\bar{C}_4 \\ & \\ C_3\bar{C}_4 & \bar{C}_3\bar{C}_4 \end{bmatrix} \qquad ..(7.2)$$

Thus, Equation (7.1) will become applicable to TDCMIN. If $[M]_{C_3C_4}$ is replaced by $[M]_{C_3\bar{C}_4}$. Thus the mapping matrix of TDCMIN becomes

$$[M]_{C_1C_2C_3\bar{C}_4C_5C_6\cdots C_{2n-1}C_{2n}} = [M]_{C_{2n-1}C_{2n}} \otimes [M]_{C_{2n-3}C_{2n-2}} \otimes \cdots$$
$$\cdots \otimes [M]_{C_3\bar{C}_4} \otimes [M]_{C_1C_2} \qquad ..(7.3)$$

$$= \bigotimes_{k=n}^{3} M_{C_{2k-1}C_{2k}} \otimes [M]_{C_3\bar{C}_4} \otimes [M]_{C_1C_2} \qquad ..(7.4)$$

7.6 THE EXTRA STAGE TDCMIN (ETDCMIN)

As in the topology of DCMIN, the TDCMIN topology can also be made fault-tolerant by adding an additional stage as (n+1)th stage. Under fault-free condition, any n consecutive stages are activated for unique path network. In case of occurrence of any fault all the (n+1) stages are activated. This generates four alternate paths for any source/destination pair - one primary and three auxiliary. These four paths will have no link in common. However, as discussed in Section 7.3, while passing through stage-2, each DIMSE will enroute two paths. Thus, if a DIMSE of stage-2 becomes faulty then there will be only two alternate paths for the corresponding input/output pair.

Example 7.3

In Fig.7.8, a (3+1) stage ETDCMIN is depicted. The four alternate paths from source (O) to destination (4) have been shown by dark lines. It can be seen that the four paths are independent, as long as the DIMSEs of stage-2 are fault free. However, if the DIMSEs of stage-2 becomes faulty, the choice of alternate paths will be limited to two.

7.6.1 Routing Scheme

As discussed in Section 7.4 the Exclusive-OR routing scheme can be followed for ETDCMIN conveniently under fault-free condition. The network is used as unique path network and the data follows the primary path. Hence, the routing tag can be derived as discussed in Section 7.4. However, if one of the secondary path is selected for routing the message, the mode of operation of (n+1)th stage is to be found. The routing tag from stage-2 to stage-n remains same as for the primary route. The routing tag for (n+1)th stage can be derived as follows:

Let s'_1 and s'_0 be the two lower binary bits of the link address selected to feed the stage-2 and d'_1, d'_0 be the lower binary bits of the destination address, then the routing tag bits for (n+1)th stage will be given as

$$r_{2n+2} = S'_1 \oplus d'_1$$
$$r_{2n+1} = S'_0 \oplus d'_0$$

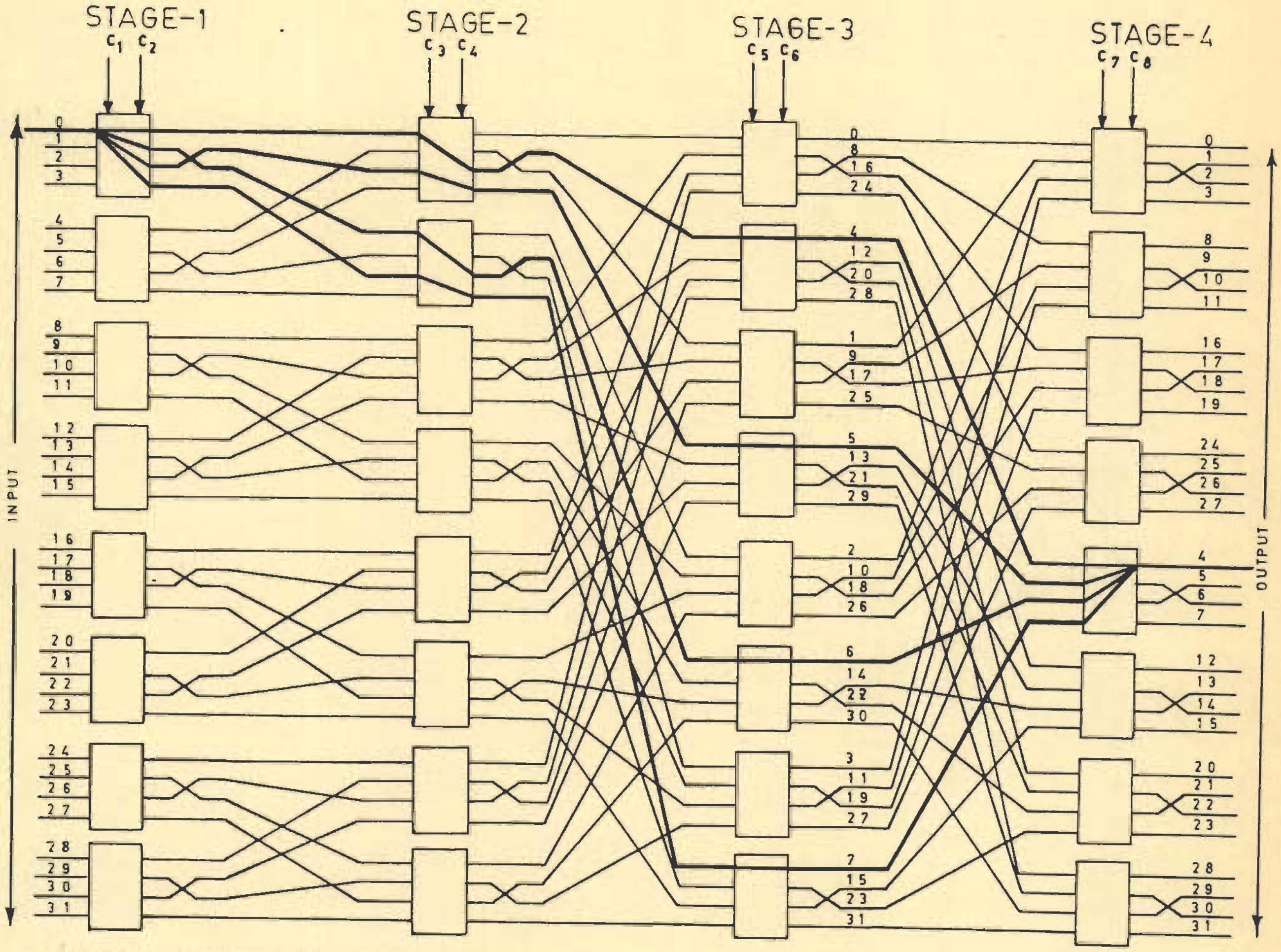


FIGURE 7.8 EXTRA-STAGE TDCMIN OF SIZE N/2 = M. DARK LINES INDICATE FOUR POSSIBLE ROUTES FROM SOURCE 0 TO DESTINATION 4.

7.7 CONCLUSION

In this chapter a derivative of DCMIN, designated as TDCMIN has been proposed. The TDCMIN topology is useful when the PEs need to shuffle the data through a network of size $M = 2^{2n-1}$ for any positive integer $n \geq 2$. Various aspects of TDCMIN such as mathematical modelling, routing scheme, partitionability and fault-tolerance have been discussed.

CHAPTER VIII

CONCLUSIONS

This chapter finally sums up the conclusions that have been arrived at during the design and analysis of dual cube multistage interconnection network. Original contributions made in the thesis have been identified and suggestions for future work have been incorporated.

With the decreasing cost of hardware, parallel processing has assumed greater significance. The parallel processing techniques use interconnection networks to facilitate concurrent operations. Actually, the interconnection networks are assuming greater role in the overall parallel processing systems. Consequently, the problem of designing efficient interconnection networks has become critical. In general, for systems having large number of processing units, MINs are indispensable.

Conventional MINs consist of stages of 2x2 SEs interconnected through perfect shuffle pattern. These MINs have the network and computational complexities which are proportional to ($\lceil \log_2 N \rceil$). In general, the complexity of the network is directly proportional to the number of SEs per stage and the total number of stages in a network. The larger sized SEs with appropriate interconnection scheme between the stages can reduce the complexity of the conventional MINs. With the advances in IC technology, it has become realistic to have a 4x4 switch on a chip having about 80 pins.

The objective of this thesis has been to explore the possibility of alternative MIN topologies which can replace/ augment the currently existing interconnection schemes. Towards this end, the thesis has culminated in the development of a Dual Cube Multistage Interconnection Network (DCMIN). The proposed topology consists of stages of 4x4 switches interconnected through 4-shuffle pattern. The network and computational complexities of DCMIN are proportional to ($\lceil \log_4 N \rceil$). Thus, in the proposed topology the number of stages and external links are reduced to half of the values for conventional MINs. This makes the proposed topology inherently more reliable, computationally faster and cost-effective.

The size of DCMIN is always NxN where $N = 4^n$ for any positive integer n. For practical situations where $N = 2^m \neq 4^n$, a derivative of DCMIN called TDCMIN has been proposed, thus, generalizing the topology for networks of any square size $N = 2^n$.

DCMIN has been evaluated for all attributes fundamental to the design of MINs such as performance, complexity, mathematical modelling, fault-diagnosis, fault-tolerance and partitionability. These are important attributes for any MIN to be feasible for use in parallel processing environment. An analytical model of 4-shuffle based upon cube configuration, has been developed which places the proposed DCMIN topology in the cube family of MINs.

The contributions identified in this thesis are listed below:

(1) A 4x4 Dual Interconnection Modular Switching Element (DIMSE) has been evolved as a building block to replace the conventional 2x2 SEs.

(2) The principle of 4-shuffle permutation has been considered and its applicability for the problems requiring concurrent processing such as fast Fourier transform, polynomial evaluation and matrix transposition, for faster computation, has also been demonstrated.

(3) (a) A new cube type MIN named DCMIN has been proposed as a better and efficient alternative to conventional MINs for use in parallel architectures.

(b) It has been shown that the proposed topology is recursive in nature, i.e., larger-size networks can be derived by adding smaller-size networks.

(c) Established that it uses lesser number of stages and external links compared to the conventional MIN topologies. This makes DCMIN inherently more reliable and reduces the network communication delay significantly.

(d) An analytical model of DCMIN has been formulated using Kronecker product of mapping matrices of DIMSE.

(4) It has been proved that the DCMIN can easily be partitioned into smaller size subnetworks.

(5) Procedures for fault-detection and location in control lines or external links have been proposed. Also, it has been established that DCMIN can be easily made fault-tolerant either by following multiple passes or by providing multiple paths. The capabilities of DCMIN for DFA have been demonstrated. A new fault-tolerant topology of DCMIN called EDCMIN has been proposed. EDCMIN generates multiple paths between any source destination pair if the fault occurs.

(6) TDCMIN, a derivative of DCMIN, has been proposed for the networks of size $M = N/2 = 2^m \neq 4^n$.

8.1 SUGGESTIONS FOR FUTURE INVESTIGATIONS

The following are the suggestions for extending the work carried out in this thesis.

(1) It is envisaged that 4x4 modular switching element would be a strong candidate to replace 2x2 SEs from the existing interconnection network topologies with improved reliability and reduced complexity. In this thesis, only one topology, i.e., DCMIN using DIMSEs has been taken up. Topologies parallel to other networks such as Delta, Benes, Baselines etc. can also be investigated.

(2) It has been shown that the use of 4-shuffle interconnection scheme, instead of perfect shuffle, increases the computational speed. The applicability of 4-shuffle in other problems requiring concrurent

processing can also be investigated.

(3) It is hoped that the network reliability can be further improved by employing the multiple copies of DCMIN in parallel. Using this concept a more reliable DCMIN topology can be investigated. Such an attempt for reliability improvement of conventional MINs has been made in the study of INDRA network [109].

(4) A network topology using DIMSEs as 2x(2x2) switching elements for improved fault-tolerance of ICNs can be investigated for masking the failure effects in the switches.

(5) In the design of DCMIN non broadcast routing have been assumed. DCMIN topology can be evaluated for broadcast routings also.

(6) The technique to locate the fault-source without back tracking, proposed in this thesis, can be extended for fault-diagnosis of existing ICNs. Thus, it may be worthwhile to develop more general algorithms for fault-diagnosis of ICNs which do not involve backtracking.

(7) In the present work only flip control has been used. Network permutational-capabilities using distributed control can be explored. Also, the conditions for the possibility of any arbitrary permutation on DCMIN can be characterized.

APPENDIX

$$A_1 = \begin{bmatrix}
1010 & & & \\
0101 & & & \\
1010 & & & \\
0101 & & & \\
 & 0011 & & \\
 & 0011 & & \\
 & 1100 & & \\
 & 1100 & & \\
 & & 0101 & \\
 & & 1010 & \\
 & & 0101 & \\
 & & 1010 & \\
 & & & 1100 \\
 & & & 1100 \\
 & & & 0011 \\
 & & & 0011
\end{bmatrix}$$

FIGURE F1

$$A_2 = \begin{bmatrix}
1111 & & & \\
 & 1111 & & \\
 & & 1111 & \\
 & & & 1111 \\
1111 & & & \\
 & 1111 & & \\
 & & 1111 & \\
 & & & 1111 \\
1111 & & & \\
 & 1111 & & \\
 & & 1111 & \\
 & & & 1111 \\
1111 & & & \\
 & 1111 & & \\
 & & 1111 & \\
 & & & 1111
\end{bmatrix}$$

FIGURE F2

$$R = \begin{bmatrix} 1111000011110000 \\ 0000111100001111 \\ 1111000011110000 \\ 0000111100001111 \\ 0000000011111111 \\ 0000000011111111 \\ 1111111100000000 \\ 1111111100000000 \\ 0000111100001111 \\ 1111000011110000 \\ 0000111100001111 \\ 1111000011110000 \\ 1111111100000000 \\ 1111111100000000 \\ 0000000011111111 \\ 0000000011111111 \end{bmatrix}$$

FIGURE F3

$$R_2 = R^2 = \begin{bmatrix} 1111111111111111 \\ 1111111111111111 \\ 1111111111111111 \\ 1111111111111111 \\ 1111111111111111 \\ 1111111111111111 \\ 1111111111111111 \\ 1111111111111111 \\ 1111111111111111 \\ 1111111111111111 \\ 1111111111111111 \\ 1111111111111111 \\ 1111111111111111 \\ 1111111111111111 \\ 1111111111111111 \\ 1111111111111111 \end{bmatrix}$$

FIGURE F4

$$A_1 = \left[\begin{array}{cccccccccccccccc}
1010 \\
0101 \\
1010 \\
0101 \\
 & 1100 \\
 & 1100 \\
 & 0011 \\
 & 0011 \\
 & & 0101 \\
 & & 1010 \\
 & & 0101 \\
 & & 1010 \\
 & & & 0011 \\
 & & & 0011 \\
 & & & 1100 \\
 & & & 1100 \\
 & & & & 0011 \\
 & & & & 0011 \\
 & & & & 1100 \\
 & & & & 1100 \\
 & & & & & 0101 \\
 & & & & & 1010 \\
 & & & & & 0101 \\
 & & & & & 1010 \\
 & & & & & & 1010 \\
 & & & & & & 0101 \\
 & & & & & & 1010 \\
 & & & & & & 0101 \\
 & & & & & & & 1100 \\
 & & & & & & & 1100 \\
 & & & & & & & 0011 \\
 & & & & & & & 0011 \\
 & & & & & & & & 1100 \\
 & & & & & & & & 1100 \\
 & & & & & & & & 0011 \\
 & & & & & & & & 0011 \\
 & & & & & & & & & 0011 \\
 & & & & & & & & & 0011 \\
 & & & & & & & & & 1100 \\
 & & & & & & & & & 1100 \\
 & & & & & & & & & & 1010 \\
 & & & & & & & & & & 0101 \\
 & & & & & & & & & & 1010 \\
 & & & & & & & & & & 0101 \\
 & & & & & & & & & & & 0101 \\
 & & & & & & & & & & & 1010 \\
 & & & & & & & & & & & 0101 \\
 & & & & & & & & & & & 1010 \\
 & & & & & & & & & & & & 0011 \\
 & & & & & & & & & & & & 0011 \\
 & & & & & & & & & & & & 1100 \\
 & & & & & & & & & & & & 1100 \\
 & & & & & & & & & & & & & 1100 \\
 & & & & & & & & & & & & & 1100 \\
 & & & & & & & & & & & & & 0011 \\
 & & & & & & & & & & & & & 0011 \\
 & & & & & & & & & & & & & & 1010 \\
 & & & & & & & & & & & & & & 0101 \\
 & & & & & & & & & & & & & & 1010 \\
 & & & & & & & & & & & & & & 0101 \\
 & & & & & & & & & & & & & & & 0101 \\
 & & & & & & & & & & & & & & & 1010 \\
 & & & & & & & & & & & & & & & 0101 \\
 & & & & & & & & & & & & & & & 1010
\end{array}\right]$$

FIGURE F5

$$A_2 = \left[\begin{array}{cccccccccccccccc}
0101 \\
& 1010 \\
& & 1100 \\
& & & 0011 \\
1010 \\
& 0101 \\
& & 1100 \\
& & & 0011 \\
0101 \\
& 1010 \\
& & 0011 \\
& & & 1100 \\
1010 \\
& 0101 \\
& & 0011 \\
& & & 1100 \\
& & & & 0101 \\
& & & & & 1010 \\
& & & & & & 0011 \\
& & & & & & & 1100 \\
& & & & 1010 \\
& & & & & 0101 \\
& & & & & & 0011 \\
& & & & & & & 1100 \\
& & & & 0101 \\
& & & & & 1010 \\
& & & & & & 1100 \\
& & & & & & & 0011 \\
& & & & 1010 \\
& & & & & 0101 \\
& & & & & & 1100 \\
& & & & & & & 0011 \\
& & & & & & & & 1100 \\
& & & & & & & & & 1010 \\
& & & & & & & & & & 0101 \\
& & & & & & & & & & & 0011 \\
& & & & & & & & 1100 \\
& & & & & & & & & 0101 \\
& & & & & & & & & & 1010 \\
& & & & & & & & & & & 0011 \\
& & & & & & & & 0011 \\
& & & & & & & & & 1010 \\
& & & & & & & & & & 0101 \\
& & & & & & & & & & & 1100 \\
& & & & & & & & 0011 \\
& & & & & & & & & 0101 \\
& & & & & & & & & & 1010 \\
& & & & & & & & & & & 1100 \\
& & & & & & & & & & & & 0011 \\
& & & & & & & & & & & & & 0101 \\
& & & & & & & & & & & & & & 1100 \\
& & & & & & & & & & & & & & & 1010 \\
& & & & & & & & & & & & 0011 \\
& & & & & & & & & & & & & 1010 \\
& & & & & & & & & & & & & & 1100 \\
& & & & & & & & & & & & & & & 0101 \\
& & & & & & & & & & & & 1100 \\
& & & & & & & & & & & & & 0101 \\
& & & & & & & & & & & & & & 0011 \\
& & & & & & & & & & & & & & & 1010 \\
& & & & & & & & & & & & 1100 \\
& & & & & & & & & & & & & 1010 \\
& & & & & & & & & & & & & & 0011 \\
& & & & & & & & & & & & & & & 0101
\end{array}\right]$$

FIGURE F6

$$A_3 = \begin{bmatrix}
1111 \\
& 1111 \\
& & 1111 \\
& & & 1111 \\
& & & & 1111 \\
& & & & & 1111 \\
& & & & & & 1111 \\
& & & & & & & 1111 \\
& & & & & & & & 1111 \\
& & & & & & & & & 1111 \\
& & & & & & & & & & 1111 \\
& & & & & & & & & & & 1111 \\
& & & & & & & & & & & & 1111 \\
& & & & & & & & & & & & & 1111 \\
& & & & & & & & & & & & & & 1111 \\
& & & & & & & & & & & & & & & 1111 \\
1111 \\
& 1111 \\
& & 1111 \\
& & & 1111 \\
& & & & 1111 \\
& & & & & 1111 \\
& & & & & & 1111 \\
& & & & & & & 1111 \\
& & & & & & & & 1111 \\
& & & & & & & & & 1111 \\
& & & & & & & & & & 1111 \\
& & & & & & & & & & & 1111 \\
& & & & & & & & & & & & 1111 \\
& & & & & & & & & & & & & 1111 \\
& & & & & & & & & & & & & & 1111 \\
& & & & & & & & & & & & & & & 1111 \\
1111 \\
& 1111 \\
& & 1111 \\
& & & 1111 \\
& & & & 1111 \\
& & & & & 1111 \\
& & & & & & 1111 \\
& & & & & & & 1111 \\
& & & & & & & & 1111 \\
& & & & & & & & & 1111 \\
& & & & & & & & & & 1111 \\
& & & & & & & & & & & 1111 \\
& & & & & & & & & & & & 1111 \\
& & & & & & & & & & & & & 1111 \\
& & & & & & & & & & & & & & 1111 \\
& & & & & & & & & & & & & & & 1111 \\
1111 \\
& 1111 \\
& & 1111 \\
& & & 1111 \\
& & & & 1111 \\
& & & & & 1111 \\
& & & & & & 1111 \\
& & & & & & & 1111 \\
& & & & & & & & 1111 \\
& & & & & & & & & 1111 \\
& & & & & & & & & & 1111 \\
& & & & & & & & & & & 1111 \\
& & & & & & & & & & & & 1111 \\
& & & & & & & & & & & & & 1111 \\
& & & & & & & & & & & & & & 1111 \\
& & & & & & & & & & & & & & & 1111
\end{bmatrix}$$

FIGURE F7

R =

```
0000111100001111000000000000000011111111000000000000000000000000
0000000000000000111100001111000000000000000000000000000011111111
0000111100001111000000000000000011111111000000000000000000000000
0000000000000000111100001111000000000000000000000000000011111111
1111000011110000000011110000111100000000000000000000000000000000
1111000011110000000011110000111100000000000000000000000000000000
0000000000000000000000000000000011111111000000000000000011111111
0000000000000000000000000000000011111111000000000000000011111111
0000000000000000111100001111000000000000000000001111111100000000
0000111100001111000000000000000000000000111111110000000000000000
0000000000000000111100001111000000000000000000001111111100000000
0000111100001111000000000000000000000000111111110000000000000000
0000000000000000000000000000000000000000111111111111111100000000
0000000000000000000000000000000000000000111111111111111100000000
1111000011110000000011110000111100000000000000000000000000000000
1111000011110000000011110000111100000000000000000000000000000000
0000000000000000000000000000000000000000111111111111111100000000
0000000000000000000000000000000000000000111111111111111100000000
0000111100001111111100001111000000000000000000000000000000000000
0000111100001111111100001111000000000000000000000000000000000000
0000000000000000000011110000111100000000000000001111111100000000
1111000011110000000000000000000000000000111111110000000000000000
0000000000000000000011110000111100000000000000001111111100000000
1111000011110000000000000000000000000000111111110000000000000000
0000111100001111000000000000000011111111000000000000000000000000
0000000000000000111100001111000000000000000000000000000011111111
0000111100001111000000000000000011111111000000000000000000000000
0000000000000000111100001111000000000000000000000000000011111111
1111000011110000000011110000111100000000000000000000000000000000
1111000011110000000011110000111100000000000000000000000000000000
0000000000000000000000000000000011111111000000000000000011111111
0000000000000000000000000000000011111111000000000000000011111111
1111111100000000111100001111000000000000000000000000000000000000
1111111100000000111100001111000000000000000000000000000000000000
0000000000000000000000000000000000001111000011110000000011111111
0000000000000000000000000000000000001111000011110000000011111111
0000000000000000000000000000000000001111000011110000000011111111
0000000000000000000000000000000000001111000011110000000011111111
1111111100000000000011110000111100000000000000000000000000000000
1111111100000000000011110000111100000000000000000000000000000000
0000000011111111000000000000000000001111000011110000000000000000
0000000000000000111100001111000000000000000000001111111100000000
0000000011111111000000000000000000001111000011110000000000000000
0000000000000000111100001111000000000000000000001111111100000000
0000000000000000000011110000111100000000000000001111111100000000
0000000011111111000000000000000011110000111100000000000000000000
0000000000000000000011110000111100000000000000001111111100000000
0000000011111111000000000000000011110000111100000000000000000000
0000000000000000000000000000000011111111000000001111000011110000
0000000000000000000000000000000011111111000000001111000011110000
0000000011111111000011110000111100000000000000000000000000000000
0000000011111111000011110000111100000000000000000000000000000000
0000000011111111111100001111000000000000000000000000000000000000
0000000011111111111100001111000000000000000000000000000000000000
0000000000000000000000000000000011111111000000000000111100001111
0000000000000000000000000000000011111111000000000000111100001111
1111111100000000000000000000000000000000111111110000000000000000
0000000000000000000011110000111100000000000000001111000011110000
1111111100000000000000000000000000000000111111110000000000000000
0000000000000000000011110000111100000000000000001111000011110000
0000000000000000111100001111000000000000000000000000111100001111
1111111100000000000000000000000000000000111111110000000000000000
0000000000000000111100001111000000000000000000000000111100001111
1111111100000000000000000000000000000000111111110000000000000000
```

FIGURE F8

$R_2 = R^2 =$

```
1111111111110000111111111111111111111111111111111111111111111111
1111111100001111111111111111111111111111111111111111111111111111
1111111111110000111111111111111111111111111111111111111111111111
1111111100001111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111
1111111100000000111111111111111111111111111111111111111111111111
1111111100000000111111111111111111111111111111111111111111111111
0000111111111111111111111111111111111111111111111111111111111111
1111000011111111111111111111111111111111111111111111111111111111
0000111111111111111111111111111111111111111111111111111111111111
1111000011111111111111111111111111111111111111111111111111111111
0000000011111111111111111111111111111111111111111111111111111111
0000000011111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111
0000000011111111111111111111111111111111111111111111111111111111
0000000011111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111
1111000011111111111111111111111111111111111111111111111111111111
0000111111111111111111111111111111111111111111111111111111111111
1111000011111111111111111111111111111111111111111111111111111111
0000111111111111111111111111111111111111111111111111111111111111
1111111111110000111111111111111111111111111111111111111111111111
1111111100001111111111111111111111111111111111111111111111111111
1111111111110000111111111111111111111111111111111111111111111111
1111111100001111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111
1111111100000000111111111111111111111111111111111111111111111111
1111111100000000111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111110000111111111111111111111111
1111111111111111111111111111111111110000111111111111111111111111
1111111111111111111111111111111100001111111111111111111111111111
1111111111111111111111111111111100001111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111110000111111111111111111111111
0000111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111110000111111111111111111111111
0000111111111111111111111111111111111111111111111111111111111111
1111000011111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111100001111111111111111111111111111
1111000011111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111100001111111111111111111111111111
1111111111111111111111111111111111111111111111111111000011111111
1111111111111111111111111111111111111111111111111111000011111111
1111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111110000111111111111
1111111111111111111111111111111111111111111111110000111111111111
1111111111111111111111111111111111111111111111111111111111111111
1111111111111111000011110000111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111
1111111111111111000011110000111111111111111111111111111111111111
1111111111111111111100001111000011111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111
1111111111111111111100001111000011111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111
```

FIGURE F9

$R_3 = R^3 =$

```
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
```

FIGURE F 10

$$A_1 = \left[\begin{array}{cccccccccccccccc}
0111 \\
0111 \\
0111 \\
0111 \\
& 1111 \\
& 1111 \\
& 1111 \\
& 1111 \\
& & 1101 \\
& & 1101 \\
& & 1101 \\
& & 1101 \\
& & & 1011 \\
& & & 1011 \\
& & & 1011 \\
& & & 1011 \\
& & & & 1011 \\
& & & & 1011 \\
& & & & 1011 \\
& & & & 1011 \\
& & & & & 1111 \\
& & & & & 1111 \\
& & & & & 1111 \\
& & & & & 1111 \\
& & & & & & 1101 \\
& & & & & & 1101 \\
& & & & & & 1101 \\
& & & & & & 1101 \\
& & & & & & & 1110 \\
& & & & & & & 1110 \\
& & & & & & & 1110 \\
& & & & & & & 1110 \\
& & & & & & & & 1101 \\
& & & & & & & & 1101 \\
& & & & & & & & 1101 \\
& & & & & & & & 1101 \\
& & & & & & & & & 1111 \\
& & & & & & & & & 1111 \\
& & & & & & & & & 1111 \\
& & & & & & & & & 1111 \\
& & & & & & & & & & 1011 \\
& & & & & & & & & & 1011 \\
& & & & & & & & & & 1011 \\
& & & & & & & & & & 1011 \\
& & & & & & & & & & & 1111 \\
& & & & & & & & & & & 1111 \\
& & & & & & & & & & & 1111 \\
& & & & & & & & & & & 1111 \\
& & & & & & & & & & & & 1101 \\
& & & & & & & & & & & & 1101 \\
& & & & & & & & & & & & 1101 \\
& & & & & & & & & & & & 1101 \\
& & & & & & & & & & & & & 1111 \\
& & & & & & & & & & & & & 1111 \\
& & & & & & & & & & & & & 1111 \\
& & & & & & & & & & & & & 1111 \\
& & & & & & & & & & & & & & 1111 \\
& & & & & & & & & & & & & & 1111 \\
& & & & & & & & & & & & & & 1111 \\
& & & & & & & & & & & & & & 1111 \\
& & & & & & & & & & & & & & & 1111 \\
& & & & & & & & & & & & & & & 1111 \\
& & & & & & & & & & & & & & & 1111 \\
& & & & & & & & & & & & & & & 1111
\end{array}\right]$$

FIGURE F11

$$A_2=\left[\begin{array}{cccccccccccccccc}
0000 \\
 & 1111 \\
 & & 1111 \\
 & & & 1110 \\
1111 \\
 & 1111 \\
 & & 1111 \\
 & & & 1110 \\
1111 \\
 & 1111 \\
 & & 0000 \\
 & & & 1110 \\
1111 \\
 & 0000 \\
 & & 1111 \\
 & & & 1110 \\
 & & & & 0111 \\
 & & & & & 0000 \\
 & & & & & & 1111 \\
 & & & & & & & 1111 \\
 & & & & 0111 \\
 & & & & & 1111 \\
 & & & & & & 1111 \\
 & & & & & & & 1111 \\
 & & & & 0111 \\
 & & & & & 1111 \\
 & & & & & & 0000 \\
 & & & & & & & 1111 \\
 & & & & 0111 \\
 & & & & & 1111 \\
 & & & & & & 1111 \\
 & & & & & & & 0000 \\
 & & & & & & & & 1011 \\
 & & & & & & & & & 1111 \\
 & & & & & & & & & & 0000 \\
 & & & & & & & & & & & 0111 \\
 & & & & & & & & 1011 \\
 & & & & & & & & & 1111 \\
 & & & & & & & & & & 1111 \\
 & & & & & & & & & & & 0111 \\
 & & & & & & & & 1011 \\
 & & & & & & & & & 0000 \\
 & & & & & & & & & & 1111 \\
 & & & & & & & & & & & 0111 \\
 & & & & & & & & 1011 \\
 & & & & & & & & & 1111 \\
 & & & & & & & & & & 1111 \\
 & & & & & & & & & & & 0111 \\
 & & & & & & & & & & & & 1110 \\
 & & & & & & & & & & & & & 1110 \\
 & & & & & & & & & & & & & & 0000 \\
 & & & & & & & & & & & & & & & 1111 \\
 & & & & & & & & & & & & 1110 \\
 & & & & & & & & & & & & & 1110 \\
 & & & & & & & & & & & & & & 1111 \\
 & & & & & & & & & & & & & & & 1111 \\
 & & & & & & & & & & & & 1110 \\
 & & & & & & & & & & & & & 1110 \\
 & & & & & & & & & & & & & & 1111 \\
 & & & & & & & & & & & & & & & 1111 \\
 & & & & & & & & & & & & 1110 \\
 & & & & & & & & & & & & & 1110 \\
 & & & & & & & & & & & & & & 1111 \\
 & & & & & & & & & & & & & & & 1111
\end{array}\right]$$

FIGURE F12

$$A_3 = \left[\begin{array}{cccccccccccccccc}
1111 \\
&1111 \\
&&1111 \\
&&&1111 \\
&&&&1111 \\
&&&&&1111 \\
&&&&&&1111 \\
&&&&&&&1111 \\
&&&&&&&&1111 \\
&&&&&&&&&1111 \\
&&&&&&&&&&1111 \\
&&&&&&&&&&&1111 \\
&&&&&&&&&&&&1111 \\
&&&&&&&&&&&&&1111 \\
&&&&&&&&&&&&&&1111 \\
&&&&&&&&&&&&&&&0000 \\
0000 \\
&1111 \\
&&1111 \\
&&&1111 \\
&&&&1111 \\
&&&&&1111 \\
&&&&&&1111 \\
&&&&&&&1111 \\
&&&&&&&&1111 \\
&&&&&&&&&1111 \\
&&&&&&&&&&1111 \\
&&&&&&&&&&&1111 \\
&&&&&&&&&&&&1111 \\
&&&&&&&&&&&&&1111 \\
&&&&&&&&&&&&&&1111 \\
&&&&&&&&&&&&&&&1111 \\
1111 \\
&0000 \\
&&1111 \\
&&&1111 \\
&&&&1111 \\
&&&&&1111 \\
&&&&&&1111 \\
&&&&&&&1111 \\
&&&&&&&&1111 \\
&&&&&&&&&1111 \\
&&&&&&&&&&1111 \\
&&&&&&&&&&&1111 \\
&&&&&&&&&&&&0000 \\
&&&&&&&&&&&&&1111 \\
&&&&&&&&&&&&&&1111 \\
&&&&&&&&&&&&&&&1111 \\
1111 \\
&1111 \\
&&1111 \\
&&&0000 \\
&&&&1111 \\
&&&&&1111 \\
&&&&&&1111 \\
&&&&&&&0000 \\
&&&&&&&&1111 \\
&&&&&&&&&1111 \\
&&&&&&&&&&1111 \\
&&&&&&&&&&&1111 \\
&&&&&&&&&&&&1111 \\
&&&&&&&&&&&&&1111 \\
&&&&&&&&&&&&&&0000 \\
&&&&&&&&&&&&&&&1111
\end{array}\right]$$

FIGURE F13

$$R = \left[\begin{matrix}
00000000000000001111111111111111111111111111111111111111111100 \\
00000000000000001111111111111111111111111111111111111111111100 \\
00000000000000001111111111111111111111111111111111111111111100 \\
00000000000000001111111111111111111111111111111111111111111100 \\
11111111111111111111111111111111111111111111111111111111111100 \\
11111111111111111111111111111111111111111111111111111111111100 \\
11111111111111111111111111111111111111111111111111111111111100 \\
11111111111111111111111111111111111111111111111111111111111100 \\
11111111111111111111111111111111000000000000000011111111111100 \\
11111111111111111111111111111111000000000000000011111111111100 \\
11111111111111111111111111111111000000000000000011111111111100 \\
11111111111111111111111111111111000000000000000011111111111100 \\
11111111111111110000000000000000111111111111111111111111111100 \\
11111111111111110000000000000000111111111111111111111111111100 \\
11111111111111110000000000000000111111111111111111111111111100 \\
11111111111111110000000000000000111111111111111111111111111100 \\
00001111111111110000000000000000111111111111111111111111111100 \\
00001111111111110000000000000000111111111111111111111111111111 \\
00001111111111110000000000000000111111111111111111111111111111 \\
00001111111111110000000000000000111111111111111111111111111111 \\
00001111111111111111111111111111111111111111111111111111111111 \\
00001111111111111111111111111111111111111111111111111111111111 \\
00001111111111111111111111111111111111111111111111111111111111 \\
00001111111111111111111111111111111111111111111111111111111111 \\
00001111111111111111111111111111000000000000000011111111111111 \\
00001111111111111111111111111111000000000000000011111111111111 \\
00001111111111111111111111111111000000000000000011111111111111 \\
00001111111111111111111111111111000000000000000011111111111111 \\
00001111111111111111111111111111111111111111111100000000000000 \\
00001111111111111111111111111111111111111111111100000000000000 \\
00001111111111111111111111111111111111111111111100000000000000 \\
00001111111111111111111111111111111111111111111100000000000000 \\
11110000111111111111111111111111000000000000000000001111111111 \\
11110000111111111111111111111111000000000000000000001111111111 \\
11110000111111111111111111111111000000000000000000001111111111 \\
11110000111111111111111111111111000000000000000000001111111111 \\
11110000111111111111111111111111111111111111111100001111111111 \\
11110000111111111111111111111111111111111111111100001111111111 \\
11110000111111111111111111111111111111111111111100001111111111 \\
11110000111111111111111111111111111111111111111100001111111111 \\
11110000111111110000000000000000111111111111111100001111111111 \\
11110000111111110000000000000000111111111111111100001111111111 \\
11110000111111110000000000000000111111111111111100001111111111 \\
11110000111111110000000000000000111111111111111100001111111111 \\
11110000111111111111111111111111111111111111111100001111111111 \\
11110000111111111111111111111111111111111111111100001111111111 \\
11110000111111111111111111111111111111111111111100001111111111 \\
11110000111111111111111111111111111111111111111100001111111111 \\
11111111111100001111111111110000000000000000000111111110000111 \\
11111111111100001111111111110000000000000000000111111110000111 \\
11111111111100001111111111110000000000000000000111111110000111 \\
11111111111100001111111111110000000000000000000111111110000111 \\
11111111111100001111111111110000111111111111111111111110000111 \\
11111111111100001111111111110000111111111111111111111110000111 \\
11111111111100001111111111110000111111111111111111111110000111 \\
11111111111100001111111111110000111111111111111111111110000111 \\
11111111111100001111111111110000111111111111111111111110000111 \\
11111111111100001111111111110000111111111111111111111110000111 \\
11111111111100001111111111110000111111111111111111111110000111 \\
11111111111100001111111111110000111111111111111111111110000111 \\
11111111111100001111111111110000111111111111111111111110000111 \\
11111111111100001111111111110000111111111111111111111110000111 \\
11111111111100001111111111110000111111111111111111111110000111 \\
11111111111100001111111111110000111111111111111111111110000111
\end{matrix}\right]$$

FIGURE F14

$$R_2 = R^2 = \left[\begin{array}{c}
1111111111111111111111111111111111111111111111111111111111111111 \\
1111111111111111111111111111111111111111111111111111111111111111 \\
1111111111111111111111111111111111111111111111111111111111111111 \\
1111111111111111111111111111111111111111111111111111111111111111 \\
1111111111111111111111111111111111111111111111111111111111111111 \\
1111111111111111111111111111111111111111111111111111111111111111 \\
1111111111111111111111111111111111111111111111111111111111111111 \\
1111111111111111111111111111111111111111111111111111111111111111 \\
1111111111111111111111111111111111111111111111111111111111111111 \\
1111111111111111111111111111111111111111111111111111111111111111 \\
1111111111111111111111111111111111111111111111111111111111111111 \\
1111111111111111111111111111111111111111111111111111111111111111 \\
1111111111111111111111111111111111111111111111111111111111111111 \\
1111111111111111111111111111111111111111111111111111111111111111 \\
1111111111111111111111111111111111111111111111111111111111111111 \\
1111111111111111111111111111111111111111111111111111111111111111 \\
1111111111111111111111111111111111111111111111111111111111111111 \\
1111111111111111111111111111111111111111111111111111111111111111 \\
1111111111111111111111111111111111111111111111111111111111111111 \\
1111111111111111111111111111111111111111111111111111111111111111 \\
1111111111111111111111111111111111111111111111111111111111111111 \\
1111111111111111111111111111111111111111111111111111111111111111 \\
1111111111111111111111111111111111111111111111111111111111111111 \\
1111111111111111111111111111111111111111111111111111111111111111 \\
1111111111111111111111111111111111111111111111111111111111111111 \\
1111111111111111111111111111111111111111111111111111111111111111 \\
1111111111111111111111111111111111111111111111111111111111111111 \\
1111111111111111111111111111111111111111111111111111111111111111 \\
1111111111111111111111111111111111111111111111111111111111111111 \\
1111111111111111111111111111111111111111111111111111111111111111 \\
1111111111111111111111111111111111111111111111111111111111111111 \\
1111111111111111111111111111111111111111111111111111111111111111 \\
1111111111111111111111111111111111111111111111111111111111111111 \\
1111111111111111111111111111111111111111111111111111111111111111 \\
1111111111111111111111111111111111111111111111111111111111111111 \\
1111111111111111111111111111111111111111111111111111111111111111 \\
1111111111111111111111111111111111111111111111111111111111111111 \\
1111111111111111111111111111111111111111111111111111111111111111 \\
1111111111111111111111111111111111111111111111111111111111111111 \\
1111111111111111111111111111111111111111111111111111111111111111 \\
1111111111111111111111111111111111111111111111111111111111111111 \\
1111111111111111111111111111111111111111111111111111111111111111 \\
1111111111111111111111111111111111111111111111111111111111111111 \\
1111111111111111111111111111111111111111111111111111111111111111 \\
1111111111111111111111111111111111111111111111111111111111111111 \\
1111111111111111111111111111111111111111111111111111111111111111 \\
1111111111111111111111111111111111111111111111111111111111111111 \\
1111111111111111111111111111111111111111111111111111111111111111 \\
1111111111111111111111111111111111111111111111111111111111111111 \\
1111111111111111111111111111111111111111111111111111111111111111 \\
1111111111111111111111111111111111111111111111111111111111111111 \\
1111111111111111111111111111111111111111111111111111111111111111 \\
1111111111111111111111111111111111111111111111111111111111111111 \\
1111111111111111111111111111111111111111111111111111111111111111 \\
1111111111111111111111111111111111111111111111111111111111111111 \\
1111111111111111111111111111111111111111111111111111111111111111 \\
1111111111111111111111111111111111111111111111111111111111111111 \\
1111111111111111111111111111111111111111111111111111111111111111 \\
1111111111111111111111111111111111111111111111111111111111111111 \\
1111111111111111111111111111111111111111111111111111111111111111 \\
1111111111111111111111111111111111111111111111111111111111111111 \\
1111111111111111111111111111111111111111111111111111111111111111 \\
1111111111111111111111111111111111111111111111111111111111111111 \\
1111111111111111111111111111111111111111111111111111111111111111
\end{array}\right]$$

FIGURE F 15

$$A_1 = \begin{bmatrix}
0111 \\
0111 \\
0111 \\
0111 \\
& 1111 \\
& 1111 \\
& 1111 \\
& 1111 \\
& & 1101 \\
& & 1101 \\
& & 1101 \\
& & 1101 \\
& & & 1111 \\
& & & 1111 \\
& & & 1111 \\
& & & 1111 \\
& & & & 1011 \\
& & & & 1011 \\
& & & & 1011 \\
& & & & 1011 \\
& & & & & 1111 \\
& & & & & 1111 \\
& & & & & 1111 \\
& & & & & 1111 \\
& & & & & & 1110 \\
& & & & & & 1110 \\
& & & & & & 1110 \\
& & & & & & 1110 \\
& & & & & & & 1111 \\
& & & & & & & 1111 \\
& & & & & & & 1111 \\
& & & & & & & 1111 \\
& & & & & & & & 1011 \\
& & & & & & & & 1011 \\
& & & & & & & & 1011 \\
& & & & & & & & 1011 \\
& & & & & & & & & 1101 \\
& & & & & & & & & 1101 \\
& & & & & & & & & 1101 \\
& & & & & & & & & 1101 \\
& & & & & & & & & & 1111 \\
& & & & & & & & & & 1111 \\
& & & & & & & & & & 1111 \\
& & & & & & & & & & 1111 \\
& & & & & & & & & & & 1111 \\
& & & & & & & & & & & 1111 \\
& & & & & & & & & & & 1111 \\
& & & & & & & & & & & 1111 \\
& & & & & & & & & & & & 0111 \\
& & & & & & & & & & & & 0111 \\
& & & & & & & & & & & & 0111 \\
& & & & & & & & & & & & 0111 \\
& & & & & & & & & & & & & 1011 \\
& & & & & & & & & & & & & 1011 \\
& & & & & & & & & & & & & 1011 \\
& & & & & & & & & & & & & 1011 \\
& & & & & & & & & & & & & & 1111 \\
& & & & & & & & & & & & & & 1111 \\
& & & & & & & & & & & & & & 1111 \\
& & & & & & & & & & & & & & 1111 \\
& & & & & & & & & & & & & & & 1110 \\
& & & & & & & & & & & & & & & 1110 \\
& & & & & & & & & & & & & & & 1110 \\
& & & & & & & & & & & & & & & 1110
\end{bmatrix}$$

FIGURE F 16

$$A_2 = \begin{bmatrix}
0000 &&&&&&&&&&&&&&& \\
& 1011 &&&&&&&&&&&&&& \\
&& 1111 &&&&&&&&&&&&& \\
&&& 0111 &&&&&&&&&&&& \\
1111 &&&&&&&&&&&&&&& \\
& 1011 &&&&&&&&&&&&&& \\
&& 1111 &&&&&&&&&&&&& \\
&&& 0111 &&&&&&&&&&&& \\
1111 &&&&&&&&&&&&&&& \\
& 1011 &&&&&&&&&&&&&& \\
&& 0000 &&&&&&&&&&&&& \\
&&& 0111 &&&&&&&&&&&& \\
1111 &&&&&&&&&&&&&&& \\
& 1011 &&&&&&&&&&&&&& \\
&& 1111 &&&&&&&&&&&&& \\
&&& 0111 &&&&&&&&&&&& \\
&&&& 1100 &&&&&&&&&&& \\
&&&&& 0000 &&&&&&&&&& \\
&&&&&& 0111 &&&&&&&&& \\
&&&&&&& 1111 &&&&&&&& \\
&&&& 1100 &&&&&&&&&&& \\
&&&&& 1111 &&&&&&&&&& \\
&&&&&& 0111 &&&&&&&&& \\
&&&&&&& 1111 &&&&&&&& \\
&&&& 0011 &&&&&&&&&&& \\
&&&&& 1111 &&&&&&&&&& \\
&&&&&& 0111 &&&&&&&&& \\
&&&&&&& 0000 &&&&&&&& \\
&&&& 0011 &&&&&&&&&&& \\
&&&&& 1111 &&&&&&&&&& \\
&&&&&& 1111 &&&&&&&&& \\
&&&&&&& 1111 &&&&&&&& \\
&&&&&&&& 0111 &&&&&&& \\
&&&&&&&&& 0000 &&&&&& \\
&&&&&&&&&& 1111 &&&&& \\
&&&&&&&&&&& 1011 &&&& \\
&&&&&&&& 0111 &&&&&&& \\
&&&&&&&&& 1111 &&&&&& \\
&&&&&&&&&& 0000 &&&&& \\
&&&&&&&&&&& 1011 &&&& \\
&&&&&&&& 0111 &&&&&&& \\
&&&&&&&&& 1111 &&&&&& \\
&&&&&&&&&& 1111 &&&&& \\
&&&&&&&&&&& 1011 &&&& \\
&&&&&&&& 0111 &&&&&&& \\
&&&&&&&&& 1111 &&&&&& \\
&&&&&&&&&& 1111 &&&&& \\
&&&&&&&&&&& 1011 &&&& \\
&&&&&&&&&&&& 0000 &&& \\
&&&&&&&&&&&&& 1111 && \\
&&&&&&&&&&&&&& 1110 & \\
&&&&&&&&&&&&&&& 1111 \\
&&&&&&&&&&&& 1111 &&& \\
&&&&&&&&&&&&& 0000 && \\
&&&&&&&&&&&&&& 1110 & \\
&&&&&&&&&&&&&&& 1111 \\
&&&&&&&&&&&& 1111 &&& \\
&&&&&&&&&&&&& 1111 && \\
&&&&&&&&&&&&&& 1110 & \\
&&&&&&&&&&&&&&& 1111 \\
&&&&&&&&&&&& 1111 &&& \\
&&&&&&&&&&&&& 1111 && \\
&&&&&&&&&&&&&& 1110 & \\
&&&&&&&&&&&&&&& 0000
\end{bmatrix}$$

FIGURE F 17

$$A_3 = \left[\begin{array}{cccccccccccccccc}
1111 \\
& 0011 \\
& & 1100 \\
& & & 0101 \\
& & & & 1010 \\
& & & & & 0000 \\
& & & & & & 1111 \\
& & & & & & & 0011 \\
& & & & & & & & 1111 \\
& & & & & & & & & 1100 \\
& & & & & & & & & & 0101 \\
& & & & & & & & & & & 1111 \\
& & & & & & & & & & & & 0000 \\
& & & & & & & & & & & & & 1111 \\
& & & & & & & & & & & & & & 1010 \\
& & & & & & & & & & & & & & & 0101 \\
1111 \\
& 0011 \\
& & 1100 \\
& & & 1010 \\
& & & & 0101 \\
& & & & & 1111 \\
& & & & & & 1111 \\
& & & & & & & 0011 \\
& & & & & & & & 0000 \\
& & & & & & & & & 1100 \\
& & & & & & & & & & 1010 \\
& & & & & & & & & & & 1111 \\
& & & & & & & & & & & & 1111 \\
& & & & & & & & & & & & & 1111 \\
& & & & & & & & & & & & & & 0101 \\
& & & & & & & & & & & & & & & 1010 \\
0000 \\
& 1100 \\
& & 0011 \\
& & & 0101 \\
& & & & 1010 \\
& & & & & 1111 \\
& & & & & & 1111 \\
& & & & & & & 1100 \\
& & & & & & & & 1111 \\
& & & & & & & & & 0011 \\
& & & & & & & & & & 0101 \\
& & & & & & & & & & & 1111 \\
& & & & & & & & & & & & 1111 \\
& & & & & & & & & & & & & - \; 0000 \\
& & & & & & & & & & & & & & 1010 \\
& & & & & & & & & & & & & & & 0101 \\
1111 \\
& 1100 \\
& & 0011 \\
& & & 1010 \\
& & & & 0101 \\
& & & & & 1111 \\
& & & & & & 1111 \\
& & & & & & & 1100 \\
& & & & & & & & 1111 \\
& & & & & & & & & 0011 \\
& & & & & & & & & & 1010 \\
& & & & & & & & & & & 0000 \\
& & & & & & & & & & & & 1111 \\
& & & & & & & & & & & & & 1111 \\
& & & & & & & & & & & & & & 0101 \\
& & & & & & & & & & & & & & & 1010
\end{array}\right]$$

FIGURE F18

R =

```
0000000000000000101000001111001111111100010111110000111110100101
0000000000000000101000001111001111111100010111110000111110100101
0000000000000000101000001111001111111100010111110000111110100101
0000000000000000101000001111001111111100010111110000111110100101
1111001111000101101000001111001111111100010111110000111110100101
1111001111000101101000001111001111111100010111110000111110100101
1111001111000101101000001111001111111100010111110000111110100101
1111001111000101101000001111001111111100010111110000111110100101
1111001111000101101000001111001100000000000000000000111110100101
1111001111000101101000001111001100000000000000000000111110100101
1111001111000101101000001111001100000000000000000000111110100101
1111001111000101101000001111001100000000000000000000111110100101
1111001111000101101000001111001111111100010111110000111110100101
1111001111000101101000001111001111111100010111110000111110100101
1111001111000101101000001111001111111100010111110000111110100101
1111001111000101101000001111001111111100010111110000111110100101
1111001100000000000000000000000000001100101011111111111101011010
1111001100000000000000000000000000001100101011111111111101011010
1111001100000000000000000000000000001100101011111111111101011010
1111001100000000000000000000000000001100101011111111111101011010
1111001100000000010111111111001100001100101011111111111101011010
1111001100000000010111111111001100001100101011111111111101011010
1111001100000000010111111111001100001100101011111111111101011010
1111001100000000010111111111001100001100101011111111111101011010
0000000011001010010111111111001100001100101011110000000000000000
0000000011001010010111111111001100001100101011110000000000000000
0000000011001010010111111111001100001100101011110000000000000000
0000000011001010010111111111001100001100101011110000000000000000
0000000011001010010111111111001100001100101011111111111101011010
0000000011001010010111111111001100001100101011111111111101011010
0000000011001010010111111111001100001100101011111111111101011010
0000000011001010010111111111001100001100101011111111111101011010
0000110000110101000000000000000011110011010111111111000010100101
0000110000110101000000000000000011110011010111111111000010100101
0000110000110101000000000000000011110011010111111111000010100101
0000110000110101000000000000000011110011010111111111000010100101
0000110000110101101011111111110000000000000000001111000010100101
0000110000110101101011111111110000000000000000001111000010100101
0000110000110101101011111111110000000000000000001111000010100101
0000110000110101101011111111110000000000000000001111000010100101
0000110000110101101011111111110011110011010111111111000010100101
0000110000110101101011111111110011110011010111111111000010100101
0000110000110101101011111111110011110011010111111111000010100101
0000110000110101101011111111110011110011010111111111000010100101
0000110000110101101011111111110011110011010111111111000010100101
0000110000110101101011111111110011110011010111111111000010100101
0000110000110101101011111111110011110011010111111111000010100101
0000110000110101101011111111110011110011010111111111000010100101
0000000000000000010111111111110011110011101000001111111101011010
0000000000000000010111111111110011110011101000001111111101011010
0000000000000000010111111111110011110011101000001111111101011010
0000000000000000010111111111110011110011101000001111111101011010
1111110000111010000000000000000011110011101000001111111101011010
1111110000111010000000000000000011110011101000001111111101011010
1111110000111010000000000000000011110011101000001111111101011010
1111110000111010000000000000000011110011101000001111111101011010
1111110000111010010111111111110011110011101000001111111101011010
1111110000111010010111111111110011110011101000001111111101011010
1111110000111010010111111111110011110011101000001111111101011010
1111110000111010010111111111110011110011101000001111111101011010
1111110000111010010111111111110011110011101000000000000000000000
1111110000111010010111111111110011110011101000000000000000000000
1111110000111010010111111111110011110011101000000000000000000000
1111110000111010010111111111110011110011101000000000000000000000
```

FIGURE F19

$R_2 = R^2 =$

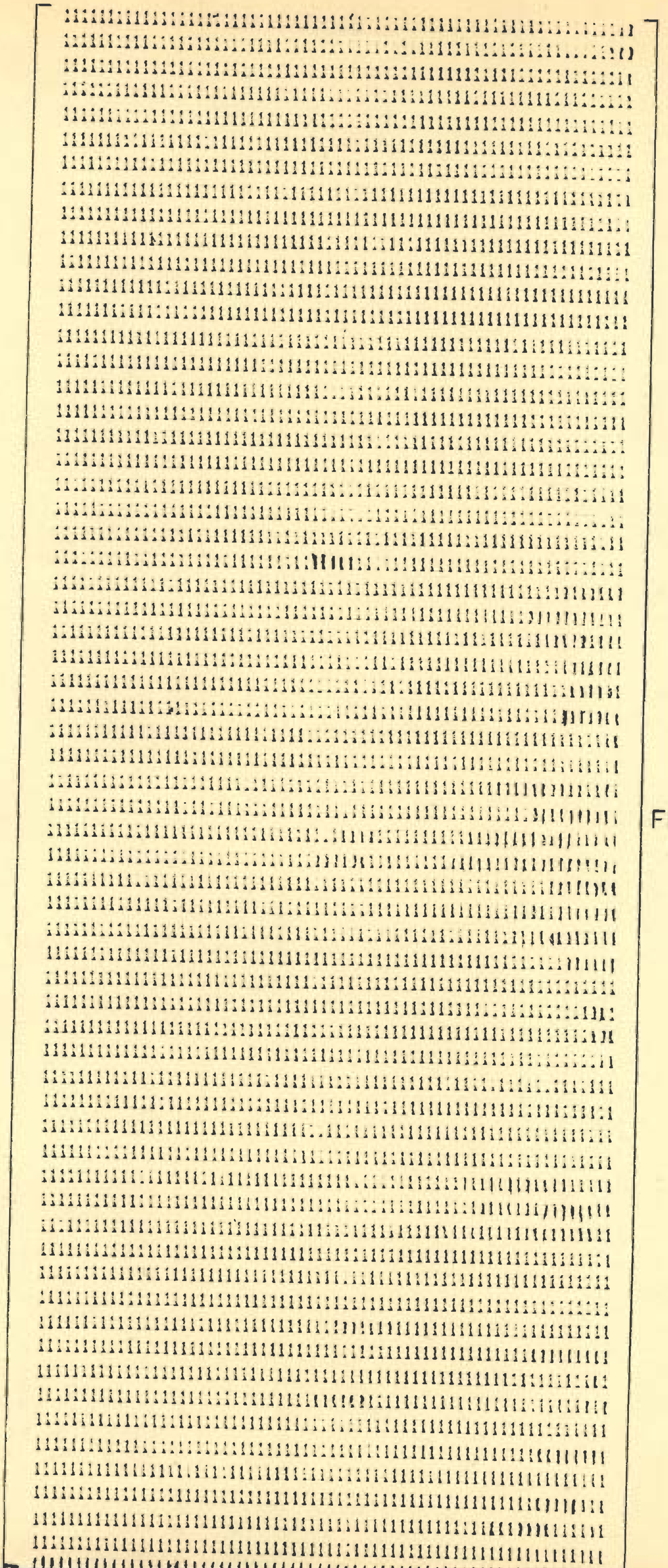


FIGURE F 20

$$A_1 = \begin{bmatrix}
01 & & & & & & & \\
01 & & & & & & & \\
 & 10 & & & & & & \\
 & 10 & & & & & & \\
 & & 10 & & & & & \\
 & & 10 & & & & & \\
 & & & 11 & & & & \\
 & & & 11 & & & & \\
 & & & & 10 & & & \\
 & & & & 10 & & & \\
 & & & & & 01 & & \\
 & & & & & 01 & & \\
 & & & & & & 11 & \\
 & & & & & & 11 & \\
 & & & & & & & 10 \\
 & & & & & & & 10
\end{bmatrix}$$

FIGURE F 21

$$A_2 = \begin{bmatrix}
00 & & & & & & & \\
 & 11 & & & & & & \\
11 & & & & & & & \\
 & 00 & & & & & & \\
 & & 01 & & & & & \\
 & & & 00 & & & & \\
 & & 01 & & & & & \\
 & & & 11 & & & & \\
 & & & & 11 & & & \\
 & & & & & 00 & & \\
 & & & & 00 & & & \\
 & & & & & 11 & & \\
 & & & & & & 10 & \\
 & & & & & & & 11 \\
 & & & & & & 10 & \\
 & & & & & & & 00
\end{bmatrix}$$

FIGURE F 22

$$A_3 = \begin{bmatrix}
11 & & & & & & & \\
 & & 01 & & & & & \\
 & 01 & & & & & & \\
 & & & 10 & & & & \\
00 & & & & & & & \\
 & & 01 & & & & & \\
 & 01 & & & & & & \\
 & & & 10 & & & & \\
 & & & & 10 & & & \\
 & & & & & & 11 & \\
 & & & & & 10 & & \\
 & & & & & & & 01 \\
 & & & & 10 & & & \\
 & & & & & & 00 & \\
 & & & & & 10 & & \\
 & & & & & & & 01
\end{bmatrix}$$

FIGURE F 23

$$A_4 = \begin{bmatrix}
1&1& & & & & & & & & & & & & & \\
 & & & & & & & &1&1& & & & & & \\
 & &0&0& & & & & & & & & & & & \\
 & & & & & & & & & &1&1& & & & \\
 & & & &0&0& & & & & & & & & & \\
 & & & & & & & & & & & &1&1& & \\
 & & & & & &1&1& & & & & & & & \\
 & & & & & & & & & & & & & &0&0\\
1&1& & & & & & & & & & & & & & \\
 & & & & & & & &0&0& & & & & & \\
 & &1&1& & & & & & & & & & & & \\
 & & & & & & & & & &0&0& & & & \\
 & & & &1&1& & & & & & & & & & \\
 & & & & & & & & & & & &1&1& & \\
 & & & & & &0&0& & & & & & & & \\
 & & & & & & & & & & & & & &1&1
\end{bmatrix}$$

FIGURE F 24

$$R = \begin{bmatrix}
0&0&0&0&0&0&1&1&0&0&1&1&0&0&0&0\\
0&0&0&0&0&0&1&1&0&0&1&1&0&0&0&0\\
1&1&0&0&0&0&0&0&1&1&0&0&1&1&0&0\\
1&1&0&0&0&0&0&0&1&1&0&0&1&1&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&1&1&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&1&1&0&0\\
0&0&0&0&0&0&1&1&0&0&1&1&1&1&0&0\\
0&0&0&0&0&0&1&1&0&0&1&1&1&1&0&0\\
1&1&0&0&1&1&0&0&0&0&0&0&1&1&0&0\\
1&1&0&0&1&1&0&0&0&0&0&0&1&1&0&0\\
0&0&1&1&0&0&0&0&0&0&0&0&0&0&1&1\\
0&0&1&1&0&0&0&0&0&0&0&0&0&0&1&1\\
1&1&1&1&0&0&0&0&0&0&0&0&0&0&1&1\\
1&1&1&1&0&0&0&0&0&0&0&0&0&0&1&1\\
1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0
\end{bmatrix}$$

FIGURE F25

$$R_2 = R^2 = \begin{bmatrix}
0&0&1&1&0&0&1&1&0&0&1&1&1&1&1&1\\
0&0&1&1&0&0&1&1&0&0&1&1&1&1&1&1\\
1&1&1&1&1&1&1&1&0&0&1&1&1&1&1&1\\
1&1&1&1&1&1&1&1&0&0&1&1&1&1&1&1\\
1&1&1&1&0&0&0&0&0&0&0&0&0&0&1&1\\
1&1&1&1&0&0&0&0&0&0&0&0&0&0&1&1\\
1&1&1&1&0&0&1&1&0&0&1&1&1&1&1&1\\
1&1&1&1&0&0&1&1&0&0&1&1&1&1&1&1\\
1&1&1&1&0&0&1&1&0&0&1&1&1&1&1&1\\
1&1&1&1&0&0&1&1&0&0&1&1&1&1&1&1\\
1&1&0&0&0&0&0&0&1&1&0&0&1&1&0&0\\
1&1&0&0&0&0&0&0&1&1&0&0&1&1&0&0\\
1&1&0&0&0&0&1&1&1&1&1&1&1&1&0&0\\
1&1&0&0&0&0&1&1&1&1&1&1&1&1&0&0\\
0&0&0&0&0&0&1&1&0&0&1&1&0&0&0&0\\
0&0&0&0&0&0&1&1&0&0&1&1&0&0&0&0
\end{bmatrix}$$

FIGURE F 26

$$R_3 = R^3 = \begin{bmatrix}
1&1&1&1&0&0&1&1&1&1&1&1&1&1&1&1\\
1&1&1&1&0&0&1&1&1&1&1&1&1&1&1&1\\
1&1&1&1&0&0&1&1&1&1&1&1&1&1&1&1\\
1&1&1&1&0&0&1&1&1&1&1&1&1&1&1&1\\
1&1&0&0&0&0&1&1&1&1&1&1&1&1&0&0\\
1&1&0&0&0&0&1&1&1&1&1&1&1&1&0&0\\
1&1&1&1&0&0&1&1&1&1&1&1&1&1&1&1\\
1&1&1&1&0&0&1&1&1&1&1&1&1&1&1&1\\
1&1&1&1&0&0&1&1&1&1&1&1&1&1&1&1\\
1&1&1&1&0&0&1&1&1&1&1&1&1&1&1&1\\
1&1&1&1&1&1&1&1&0&0&1&1&1&1&1&1\\
1&1&1&1&1&1&1&1&0&0&1&1&1&1&1&1\\
1&1&1&1&1&1&1&1&0&0&1&1&1&1&1&1\\
1&1&1&1&1&1&1&1&0&0&1&1&1&1&1&1\\
0&0&1&1&0&0&1&1&0&0&1&1&1&1&1&1\\
0&0&1&1&0&0&1&1&0&0&1&1&1&1&1&1
\end{bmatrix}$$

FIGURE F 27

$$R_4 = R^4 = \begin{bmatrix}
1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1\\
1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1\\
1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1\\
1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1\\
1&1&1&1&1&1&1&1&0&0&1&1&1&1&1&1\\
1&1&1&1&1&1&1&1&0&0&1&1&1&1&1&1\\
1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1\\
1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1\\
1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1\\
1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1\\
1&1&1&1&0&0&1&1&1&1&1&1&1&1&1&1\\
1&1&1&1&0&0&1&1&1&1&1&1&1&1&1&1\\
1&1&1&1&0&0&1&1&1&1&1&1&1&1&1&1\\
1&1&1&1&0&0&1&1&1&1&1&1&1&1&1&1\\
1&1&1&1&0&0&1&1&1&1&1&1&1&1&1&1\\
1&1&1&1&0&0&1&1&1&1&1&1&1&1&1&1
\end{bmatrix}$$

FIGURE F28

$$R_5 = R^5 = \begin{bmatrix}
1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1\\
1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1\\
1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1\\
1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1\\
1&1&1&1&0&0&1&1&1&1&1&1&1&1&1&1\\
1&1&1&1&0&0&1&1&1&1&1&1&1&1&1&1\\
1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1\\
1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1\\
1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1\\
1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1\\
1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1\\
1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1\\
1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1\\
1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1\\
1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1\\
1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1
\end{bmatrix}$$

FIGURE F29

$$R_6 = R^6 = \begin{bmatrix}
1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1\\
1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1\\
1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1\\
1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1\\
1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1\\
1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1\\
1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1\\
1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1\\
1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1\\
1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1\\
1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1\\
1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1\\
1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1\\
1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1\\
1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1\\
1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1
\end{bmatrix}$$

FIGURE F30

$$A_1 = \begin{bmatrix}
1&1 \\
1&1 \\
&&1&1 \\
&&1&1 \\
&&&&1&1 \\
&&&&1&1 \\
&&&&&&1&1 \\
&&&&&&1&1 \\
&&&&&&&&1&1 \\
&&&&&&&&1&1 \\
&&&&&&&&&&1&1 \\
&&&&&&&&&&1&1 \\
&&&&&&&&&&&&1&1 \\
&&&&&&&&&&&&1&1 \\
&&&&&&&&&&&&&&1&1 \\
&&&&&&&&&&&&&&1&1
\end{bmatrix}$$

FIGURE F31

$$A_2 = \begin{bmatrix}
1&0&0&0 \\
0&0&0&1 \\
0&1&0&0 \\
0&0&1&0 \\
&&&&0&1&0&0 \\
&&&&0&0&1&0 \\
&&&&1&0&0&0 \\
&&&&0&0&0&1 \\
&&&&&&&&0&1&0&0 \\
&&&&&&&&0&0&1&0 \\
&&&&&&&&1&0&0&0 \\
&&&&&&&&0&0&0&1 \\
&&&&&&&&&&&&1&0&0&0 \\
&&&&&&&&&&&&0&0&0&1 \\
&&&&&&&&&&&&0&1&0&0 \\
&&&&&&&&&&&&0&0&1&0
\end{bmatrix}$$

FIGURE F32

$$A_3 = \begin{bmatrix}
1&1 \\
&&&&1&1 \\
&&1&1 \\
&&&&&&1&1 \\
1&1 \\
&&&&1&1 \\
&&1&1 \\
&&&&&&1&1 \\
&&&&&&&&1&1 \\
&&&&&&&&&&&&1&1 \\
&&&&&&&&&&1&1 \\
&&&&&&&&&&&&&&1&1 \\
&&&&&&&&1&1 \\
&&&&&&&&&&&&1&1 \\
&&&&&&&&&&1&1 \\
&&&&&&&&&&&&&&1&1
\end{bmatrix}$$

FIGURE F33

$$A_4 = \left[\begin{array}{cccccccccccccccc}
1 & 1 &&&&&&&&&&&&&& \\
&&&&&&&& 1 & 1 &&&&&& \\
&& 1 & 1 &&&&&&&&&&&& \\
&&&&&&&&&& 1 & 1 &&&& \\
&&&& 1 & 1 &&&&&&&&&& \\
&&&&&&&&&&&& 1 & 1 && \\
&&&&&& 1 & 1 &&&&&&&& \\
&&&&&&&&&&&&&& 1 & 1 \\
1 & 1 &&&&&&&&&&&&&& \\
&&&&&&&& 1 & 1 &&&&&& \\
&& 1 & 1 &&&&&&&&&&&& \\
&&&&&&&&&& 1 & 1 &&&& \\
&&&& 1 & 1 &&&&&&&&&& \\
&&&&&&&&&&&& 1 & 1 && \\
&&&&&& 1 & 1 &&&&&&&& \\
&&&&&&&&&&&&&& 1 & 1
\end{array}\right]$$

FIGURE F34

$$R = \left[\begin{array}{c}
1100001111000011 \\
1100001111000011 \\
0011110000111100 \\
0011110000111100 \\
0011110000111100 \\
0011110000111100 \\
1100001111000011 \\
1100001111000011 \\
0011110000111100 \\
0011110000111100 \\
1100001111000011 \\
1100001111000011 \\
1100001111000011 \\
1100001111000011 \\
0011110000111100 \\
0011110000111100
\end{array}\right]$$

FIGURE F35

$$R_2 = R^2 = \left[\begin{array}{c}
1111111111111111 \\
1111111111111111 \\
1111111111111111 \\
1111111111111111 \\
1111111111111111 \\
1111111111111111 \\
1111111111111111 \\
1111111111111111 \\
1111111111111111 \\
1111111111111111 \\
1111111111111111 \\
1111111111111111 \\
1111111111111111 \\
1111111111111111 \\
1111111111111111 \\
1111111111111111
\end{array}\right]$$

FIGURE F36

B I B L I O G R A P H Y

1. J.A.Abraham, 'An Improved Algorithm for Network Reliability', IEEE Trans. on Reliability, Vol. R-23, April 1979, pp.58-61.

2. G.B.Adams III, H.J.Siegel, 'On the Number of Permutations Performable By the Augmented Data Manipulator Network', IEEE Trans. on Computers, Vol.C-31, April 1982, pp.270-277.

✓ 3. G.B.Adams III, H.J.Siegel, 'The Extra-Stage Cube: A Fault Tolerant Interconnection Network for Supersystems', IEEE Trans. on Computers, Vol. C-31, May 1982, pp.443-454.

4. D.P.Agrawal, 'Testing and Fault-Tolerance of Multistage Interconnection Networks', Computer, Vol.15, April 1982, pp.41-53.

5. D.P.Agrawal, 'Graph Theoretical Analysis and Design of Multistage Interconnection Networks', IEEE Trans. on Computers, Vol.C-32, July 1983, pp.637-648.

✓ 6. D.P.Agrawal, D.Kaur, 'Fault Tolerant Capabilities of Redundant Multistage Interconnection Networks', Proc. Real Time Systems Symp., December 1983, pp.119-127.

7. D.P.Agrawal, S.C.Kim, 'On Non-Equivalent Multistage Interconnection Networks', Proc. Int. Conf. Parallel Processing, August 1981, pp.234-237.

✓ 8. D.P.Agrawal, J.S.Leu, 'Dynamic Accessibility Testing and Path Length Optimization of Multistage Interconnection Networks', Proc. 4th Int. Conf. on Distributed Computing Systems, May 1984, pp.266-277.

9. G.A.Anderson, E.D.Jensen, 'Computer Interconnection Networks: Texonomy, Characteristics and Examples', ACM Computing Surveys, Vol.7, December 1975, pp.197-213.

✓10. J.L.Baer, 'Multiprocessing Systems', IEEE Trans. on Computers, Vol.C-25, December 1976, pp.1271-1277.

✓11. J.L.Baer, 'Computer System Architecture', (Computer Science Press), 1980.

12. G.H.Barnes et al., 'The ILLIAC IV Computer', IEEE Trans. on Computers, Vol.C-17, August 1968, pp.746-757.

13. G.H.Barnes, S.F.Lundstrom, 'Design and Validation of a Connection Network for Many-Processor Multiprocessor Systems', Computer, Vol.14, December 1981, pp.31-41.

14. K.E.Batcher, 'Sorting Networks and their Applications', Proc. of SJCC, AFIPS, Vol.32, 1968, pp.307-314.

✓15. K.E.Batcher, 'The Flip Network in STARAN', Proc. Int. Conf. on Parallel Processing, August 1976, pp.65-71.

✓16. K.E.Batcher, 'Design of a Massively Parallel Processor', IEEE Trans. on Computers, Vol.C-29, September 1980,pp.836-840.

17. K.E.Batcher, 'Bit Serial Parallel Processing Systems', IEEE Trans. on Computers, Vol.C-31, May 1982, pp.377-384.

18. L.H.Bauer, 'Implementation of Data Manipulating Functions on the STARAN Associative Processor', Proc. Sagamore Conf. on Parallel Processing, August 1974, pp.209-227.

19. J.S.Bedi, K.Singh, 'Hierarchical Distributed Control for a Robotic Network', Proc. Intelligent Systems and Machines Conf., April 1986, pp.182-186.

20. V.E.Benes,'On Rearrangeable Three Stage Connecting Networks', BSTJ, Vol.41, September 1962, pp.1481-1492.

✓ 21. V.E.Benes, 'Mathematical Theory of Connecting Networks and Telephone Traffic, (Academic Press, New York), 1965.

22. C.Berge, 'Graphs and Hypergraphs', (North-Holland Publishing Company, Amsterdam), 1973.

23. D.P.Bhandarkar, 'Analysis of Memory Interference in Multiprocessors', IEEE Trans. on Computers, Vol.C-24, September 1983, pp.897-908.

✓ 24. L.N.Bhuyan, D.P.Agrawal, 'Design and Performance of Generalized Interconnection Networks', IEEE Trans. on Computers, Vol.C-32, December 1983, pp.1081-1090.

✓ 25. W.J.Bouknight et al., 'The Illiac IV System, 'Proc. IEEE, Vol.60, April 1972, pp.369-388.

26. F.A.Briggs et al., 'PM 4 - a Reconfigurable Multimicro - processor System for Pattern Recognition and Image Process - ing', AFIPS Conf. Proc., June 1979, pp.255-265.

27. F.A.Briggs et al., 'PUMPS Architecture for Pattern Analysis and Image Database Management', IEEE Trans. on Computers, Vol.C-31, October 1982, pp.969-982.

28. G.Broomell, J.R.Heath, 'Classification, Catagories, and Historical Development of Circuit Switching Topologies', Computing Surveys, Vol.15, June 1983, pp.95-133.

29. P.Y.Chen, P.C.Yew, D.A.Padua, 'Interconnection Networks Using Shuffles', Computer, Vol.14, December 1981, pp.55-64.

30. P.Y.Chen, P.C.Yew, D.H.Lawrie, 'Performance of Packet Switching in Buffered Single Stage Shuffle-Exchange Networks', Proc. of the 3rd Int. Conf. on Distributed Computing Systems, October 1982, pp.622-629.

31. V.Cherkassky, E. Opper, M.Malek, 'Reliability and Fault Diagnosis Analysis of Fault-Tolerant Multistage Interconnection Networks', Proc. Int. Symp. on Fault-Tolerant Computing, June 1984, pp.246-253.

32. C.Y.Chin, K.Hwang, 'Connection Principles for Multipath Packet Switching Networks', Proc. Int. Symp. on Computer Architecture, June 1984, pp.99-108.

33. Y.Chow et al., 'Routing Techniques for Rearrangeable Interconnection Networks', Proc. Workshop on Interconnection Networks, April 1980, pp.64-69.

34. L.Ciminiera, 'Quick Fault Location in Multistage Interconnection Networks', Electronics Letters, Vol.20, Ist March 1984, pp.193.

✓ 35. L.Ciminiera, A.Serra, 'A Fault-Tolerant Connecting Networks for Multiprocessor Systems', Proc. Int. Conf. on Parallel Processing, August 1982, pp.113-122.

✓ 36. C.Clos, 'A Study of Nonblocking Switching Networks', BSTJ, Vol.32, March 1953, pp.406-424.

37. J.W.Cooley, J. W.Tukey, 'An Algorithm for the Machine Calculation of Complex Fourier Series', Maths. Computing, Vol.19, April 1965, pp.297-301.

✓ 38. Special Issue on Multiprocessing Technology, Computer, Vol. 18, June 1985.

39. C.R.Das, L.N.Bhuyan, 'Reliability Simulation of Multiprocessor Systems', Proc. Int. Conf. on Parallel Processing, August 1985, pp.764-771.

40. N.J.Davis et al., 'Fault Location Techniques for Distributed Control Interconnection Networks', IEEE Trans. on Computers, Vol.C-34, October 1985, pp.902-910.

41. M.Davio, 'Kronecker Product and Shuffle Algebra', IEEE Trans. on Computers, Vol.C-30, February 1981, pp.116-125.

42. J.B.Dennis et al., 'Building Blocks for Data Flow Prototypes', Seventh Annual Symp. on Computer Architecture, May 1980, pp.1-8.

43. N.Deo, 'Graph Theory: With Applications to Engineering and Computer Science', (Prentice-Hall, New Delhi), 1980.

✓44. D.M.Dias, J.R.Jump, 'Analysis and Simulation of Buffered Delta Networks', IEEE Trans. on Computers, Vol.C-30, April 1981, pp.273-282.

✓45. P.H.Enslow, 'Multiprocessor and Parallel Processing', (John Wiley, New York), 1974.

✓46. P.H.Enslow, 'Multiprocessor Organization - A Survey', ACM Computing Surveys, Vol.9, March 1977, pp.103-129.

47. K.M.Falavarajani, D.K.Pradhan, 'Fault Diagnosis of Parallel Processor Interconnection Networks', Proc. Symp. on Fault Tolerant Computing, June 1981.

48. T.Feng, 'Data Manipulation Functions in Parallel Processors and their Implementations', IEEE Trans. on Computers, Vol. C-23, March 1974, pp.309-318.

49. T.Feng, 'A Survey of Interconnection Networks', Computer, Vol.14, December 1981, pp.309-318.

50. T.Feng, I.P.Kao, 'On Fault-Diagnosis of Some Multistage Networks', Proc. Int. Conf. on Parallel Processing, 1982, pp.99-101.

51. T.Feng, C.Wu, 'Fault-Diagnosis for a Class of Multistage Interconnection Networks', IEEE Trans. on Computers, Vol. C-30, October 1981, pp.743-758.

52. R.A.Finkel, M.H.Solomon, 'Processor Interconnection Strategies', IEEE Trans. on Computers, Vol.C-29, May 1980, pp. 360-371.

53. J.P.Fishburn, R.A.Finkel, 'Quotient Networks', IEEE Trans. on Computers, Vol.C-31, April 1982, pp.288-295.

54. M.J.Flynn, 'Very High-Speed Computing Systems', Proc. IEEE, Vol.54, December 1966, pp.1901-1909.

✓55. M.J.Flynn, 'Some Computer Organizations and Their Effectiveness', IEEE Trans. on Computers, Vol.C-21, September 1972, pp.948-960.

56. T.J.Fountain, 'CLIP 4: Progress Report', In Languages and Architectures for Image Processing, Ed. M.J.B. Duff and S.Levialdi (Academic Press, London), 1981.

57. M.A.Franklin, D.F.Wann, W.J.Thomas, 'Pin Limitations and Partitioning of VLSI Interconnection Networks', IEEE Trans. on Computers, Vol.C-31, November 1982, pp.1109-1116.

58. W.K.Fuchs et al., 'Concurrent Error Detection in VLSI Interconnection Networks', Proc. tenth Int. Symp. on Computer Architecture, June 1983, pp.309-315.

59. W.M.Gentleman, 'Some Complexity Results for Matrix Computations on Parallel Processors', J. ACM, Vol. 25, January 1978, pp.112-115.

60. L.R.Goke, G.J.Lipovski, 'Banayan Networks for Partitioning Multiprocessor Systems', Proc. of the First Annual Symposium on Computer Architecture, 1973, pp.21-28.

61. A.R.Gottlieb et al., 'The NYU Ultracomputer - Designing an MIMD Shared-Memory Parallel Computer', IEEE Trans. on Computers, Vol.C-22, February 1983, pp.175-189.

62. F.Harary, 'Graph Theory', (Addison Wesley, Mass.), 1972.

63. J.A.Harris, D.R.Smith, 'Hierarchical Multiprocessor Organisations', Proc. Fourth Symp. on Computer Architecture, March 1977, pp.41-48.

64. R.W.Hockney, C.R.Jeshope, 'Parallel Computers',(Adam Hilger, Bristol, England), 1981.

65. D.J.Hunt, 'The ICL DAP and Its Application to Image Processing', Languages and Architecture for Image Processing, Ed. M.J.B. Duff and S.Levialdi, (Academic Press, London), 1981.

✓66. K.Hwang, 'Supercomputers: Design and Applications', Tutorial IEEE Computer Society, 1984.

✓67. K.Hwang, F.Briggs, 'Computer Architecture and Parallel Processing', (Mc Graw-Hill, New York), 1984.

68. A.Joel, 'On Permutation Switching Networks', BSTJ, Vol.47, May-June 1968, pp.813-822.

69. A.K.Jones et al., 'Software Management of C_m^* - a Distributed Multiprocessor', AFIPS Conf.Proc., June 1977, pp.657-663.

70. S.L.Khanduja, R.C.Joshi, K.Singh, 'Fault Analysis for F.F.T. Processors', Proc. Symp. on Communications, August 1984, pp.57-60.

71. C.K.Kothari, S.Lakshmivarahan, H.Peyravi, 'A Note on Re - arrangeable Networks', Technical Report, School of Engineering and Computer Science, University of Oklahoma, November 1983.

72. V.P.Krothapalli et al., 'Fault Diagnosis for a Multistage Banyan Interconnection Network', Proc. IEE, Vol.132, Pt.E, May 1985, pp.146-154.

73. C.P.Kruskal, M.Snir, 'The Performance of Multistage Interconnection Networks for Multiprocessors', IEEE Trans. on Computers, Vol.C-32, December 1983, pp.1091-1098.

✓ 74. D.J.Kuck, 'A Survey of Parallel Machine Organization and Programming', ACM Computing Surveys, Vol.9, March 1977, pp.29-59.

75. D.J.Kuck, 'The Structure of Computers and Computations', Vol., (John Wiley, New York), 1978.

76. M.Kumar, J.R.Jump, 'Generalized Delta Networks', Proc. Int. Conf. on Parallel Processing, August 1983, pp.10-18.

77. Prem Kumar et al., 'Design and Implementation of the Banyan Interconnection Network in TRAC', AFIPS Conf. Proc. June 1980, pp.643-653.

78. H.T.Kung, 'Why Systolic Architectures?', Computer, Vol.15, January 1982, pp.37-46.

9 79. D.H.Lawrie, 'Access and Alignment of Data in an Array Processor', IEEE Trans. on Computers, Vol.C-24, December 1975, pp.1145-1155.

80. D.H.Lawrie, D.A. Padua, 'Analysis of Message Switching with Shuffle-Exchanges in Multiprocessors', Proc. Workshop on Interconnection Networks for Parallel and Distributed Processing, April 1980, pp.116-123.

81. K.Y.Lee, 'On the Rearrangeability of ($2\log_2 N-1$) - Stage Permutation Networks', IEEE Trans. on Computers, Vol.C-34, May 1985, pp.412-425.

82. W.E.Leland, 'Density and Reliability of Interconnection Topologies for Multicomputers', Ph.D. Thesis, University of Wilsconsin, Madison, July 1982.

83. J.Lenfant, 'Parallel Permutations of Data: A Benes Network Control Algorithm for Frequently Used Permutations', IEEE Trans. on Computers, Vol.C-27, July 1978, pp.637-647.

✓ 84. W.Lin, C.Wu, 'Design of 2x2 Fault-Tolerant Switching Element', Proc. Int. Symp. on Computer Architecture, April 1982, pp.181-189.

85. M.T.Liu, 'Distributed Loop Computer Networks', Advances in Computers, Vol.17, (Academic Press, London), 1978.

86. M.Malek, E.Opper, 'Multiple Fault Diagnosis of SW Banyan Networks', Proc. Fault-Tolerant Computing Symp. June 1983, pp.446-449.

87. G.M.Masson et al., 'A Sampler of Circuit Switching Networks', Computer, Vol.12, June 1979, pp.32-48.

88. R.J.McMillen, G.B.Adams III, H.J.Siegel, 'Performance and Implementation of 4x4 Switching Nodes in an Interconnection Network for PASM', Proc. Int. Conf. on Parallel Processing, August 1981, pp.229-233.

89. R.J.McMillen, H.J.Siegel, 'MIMD Machine Communication Using the Augmented Data Manipulator Network', Proc. Seventh Annual Symp. on Computer Architecture, May 1980, pp.51-58.

90. R.J.McMillen, H.J.Siegel, 'Routing Schemes for the Augmented Data Manipulator Network in an MIMD System', IEEE Trans. on Computers, Vol.C-31, December 1982, pp.1202-1214.

91. C.Mead, L.Conway, 'Introduction to VLSI Systems', (Addison-Wesley, Mass.), 1980.

92. D.P.Mital et al., 'Matrix Representation of Staran Interconnecting Network', Computers and Elect. Engg. Vol.11, January 1984, pp.59-66.

93. D.Nassimi, S.Sahni, 'An Optimal Routing Algorithm for Mesh-Connected Parallel Computers', J. ACM, Vol.27, January 1980, pp.6-29.

94. D.Nassimi, S.Sahni, 'Data Broadcasting in SIMD Computers', IEEE Trans. on Computers, Vol.C-30, February 1981, pp.101-107.

95. D.Nassimi, S.Sahni, 'Optimal BPC Permutations on a Cube Connected Computer', IEEE Trans. on Computers, Vol. C-31, April 1982, pp.338-341.

96. D.C.Opferman, N.T.Tsao-Wu, 'On a Class of Rearrangeable Switching Networks', BSTJ, Vol.50, May - June 1971, pp.1579-1600.

97. S.E.Orcutt, 'Implementation of Permutation Functions in Illiac IV Type Computers', IEEE Trans. on Computers, Vol. C-25, September 1976, pp.929-936.

98. K.Padmanabhan, D.H.Lawrie, 'Fault-Tolerance Schemes in Shuffle/Exchange Type Interconnection Networks', Proc. Int. Conf. on Parallel Processing, August 1983, pp.71-75.

99. K.Padmanabhan, D.H.Lawrie, 'A Class of Redundant Path Multi-stage Interconnection Networks', IEEE Trans. on Computers, Vol.C-32, December 1983, pp.1099-1108.

100. S.K.Paranjpe et al., 'A New Concept for Supermodular Alignment Network', Int. J. of Electronics, Vol.56, June 1984, pp.815-822.

101. S.K.Paranjpe, R.Mitra, 'Modular Implementation of Inter - connecting Networks in VLSI', J. of Inst. of Electronic and Radio Engineers (UK), Vol.55, January 1985, pp.11-14.

102. D.S.Parker, 'Notes on Shuffle/Exchange Type Switching Networks', IEEE Trans. on Computers, Vol.C-29, March 1980, pp.213-222.

✓103. D.S.Parker, C.S.Raghavendra, 'The Gamma Network', IEEE Trans. on Computers, Vol.C-33, April 1984, pp.367-373.

104. J.H.Patel, 'Performance of Processor-Memory Interconnections for Multiprocessors', IEEE Trans. on Computers, Vol.C-30, October 1981, pp.771-780.

105. M.C.Pease, 'An Adaptation of the Fast Fourier Transform for Parallel Processing', J. ACM, Vol.15, April 1968, pp.252-264.

✓ 106. M.C.Pease, 'The Indirect Binary n-Cube Microprocessor Array', IEEE Trans. on Computers, Vol.C-26, May 1977, pp.458-473.

✓ 107. G.F.Pfister et al., 'The IBM Research Parallel Processor Prototype (RP3): Introduction and Architecture', Proc. Int. Conf. on Parallel Processing, August 1985, pp.764-771.

108. D.K.Pradhan, K.L.Kodandapani, 'A Uniform Representation of Single and Multistage Interconnection Networks Used in SIMD Machine', IEEE Trans. on Computers, Vol.C-29, September 1980, pp.777-790.

✓ 109. C.S.Raghavendra, A.Varma, 'INDRA: A Class of Interconnection Networks with Redundant Paths', Proc. Real-Time Systems Symp. December 1984, pp.153-164.

110. C.S.Raghavendra, A.Varma, 'Fault-Tolerant Multiprocessors with Redundant Path Interconnection Networks', IEEE Trans. on Computers, Vol.C-35, April 1986, pp.307-316.

✓ 111. S.M.Reddy, V.P.Kumar, 'On Fault-Tolerant Multistage Interconnection Networks', Proc. Int. Conf. on Parallel Processing, August 1984, pp.155-164.

112. A.P.Reeves, R.Rindfuss, 'The BASE 8 Binary Array Processor', Proc. IEEE Computer Society Conf. on Pattern Recognition and Image Processing, August 1979, pp.250-255.

113. A.Satyanarayana, J.N.Hagstrom, 'A New Algorithm for Reliability Analysis of Multiterminal Networks', IEEE Trans. on Reliability, Vol.R-30, October 1981, pp.325-334.

114. C.L.Sietz, 'The Cosmic Cube', Communications of the ACM, Vol.28, January 1985, pp.22-33.

115. M.C.Sejnowski et al., 'An Overview of the Texas Reconfigurable Array Computer', AFIPS Conf. Proc., June 1980, pp.631-641.

✓ 116. D.K.Sharma, K.Singh, 'Matrix Representation and Fault Tolerance of Delta Network', Proc. All India Seminar on Computers, August 1986, pp.A.01-A.13.

✓ 117. J.P.Shen, 'Fault-Tolerance of β-Networks in Interconnected Multicomputer Systems', Ph.D. Dissertation, USCEE Technical Report No.510, August 1981.

✓ 118. J.P.Shen, J.P.Hayes, 'Fault-Tolerance of Dynamic-Full-Access Interconnection Networks', IEEE Trans. on Computers, Vol.C-33, March 1984, pp.241-248.

119. J.P.Shen et al., 'Fault-Tolerance and Performance Analysis of Beta-Networks', Parallel Computing, Vol.3, July 1986, pp.231-249.

120. H.J.Siegel, 'The Universality of Various Types of SIMD Machine Interconnection Networks', Proc. 4th Annual Symp. on Computer Architecture, March 1977, pp.70-79.

✓ 121. H.J.Siegel, 'A Model of SIMD Machines and a Comparison of Various Interconnection Networks', IEEE Trans. on Computers, Vol.C-28, December 1979, pp.907-917.

122. H.J.Siegel, 'The Theory Underlying the Partitioning of Permutation Networks', IEEE Trans. on Computers, Vol.C-29, September 1980, pp.791-801.

123. H.S.Siegel, 'Introduction to Computer Architecture', 2nd ed., (Science Research Associates, Chicago), 1980,pp.363-425.

124. H.J.Siegel, R.J.McMillen, 'Using the Augmented Data Manipulator in PASM', Computer, Vol.14, February 1981, pp.25-33.

125. H.J.Siegel, R.J.McMillen, 'The Multistage Cube: A Versatile Interconnection Network', Computer, Vol.14, December 1981, pp.65-76.

✓126. K.Singh, N.K.Nanda, R.C.Joshi, 'Design of MIN - A New Approach', J. AMSE, A, Vol.12, 1987, pp.53-64 (To Appear).

127. K.Singh, N.K.Nanda, R.C.Joshi, S.K.Paranjpe, 'Multistage Interconnection Network Using VLSI - 4x4 Modular Switching Element',(Communicated).

128. K.Singh, N.K.Nanda, R.C.Joshi, 'Testing and Fault - Tolerance of 4-Shuffle Network', (Communicated).

129. K.Singh, N.K.Nanda, R.C.Joshi, '4-Shuffle Interconnection Pattern In Parallel Processing', (Communicated).

130. S.D.Smith et al., 'Use of Augmented Data Manipulator Multistage Network for SIMD Machines', Proc. Int. Conf. on Parallel Processing, August 1980, pp.75-78.

131. K.M.So, J.J.Narraway, 'On Line Fault Diagnosis of Switching Networks', IEEE Trans. On Circuits and Systems, CAS-26, July 1979, pp.575-583.

132. J.Sovis, 'Uniform Theory of the Shuffle-Exchange Type Permutation Networks', Proc. 10th Annual Int. Symp. on Computer Architecture, June 1983,pp.185-191.

133. A.K.Sood, 'Design of Multistage Interconnection Networks', Proc. IEE, Vol.130, Pt.E, July 1983, pp.109-115.

134. D.Steinberg, 'Invariant Properties of the Shuffle-Exchange and A Simplified Cost-Effective Version of the Omega Network', IEEE Trans. on Computers, Vol.C-32, May 1983, pp. 444-450.

135. H.S.Stone, 'Parallel Processing with the Perfect Shuffle', IEEE Trans. on Computers, Vol.C-20, . February 1971, pp.153-161.

✓136. H.S.Stone, 'Parallel Computers', Introduction to Computer Architecture, 2nd Ed., (SRA Inc., Chicago, USA), 1980.

137. R.Sugarman, 'Superpower Computers', IEEE Spectrum, Vol.17, April 1980, pp.28-34.

138. R.J.Swan et al., 'The Implementation of the Cm* Multi - Microprocessor', AFIPS Conf. Proc., June 1977, pp.645-655.

139. E.Swartzlander, B.Gilbert, 'Supersystems: Technology and Architecture', IEEE Trans. on Computers, Vol.C-31, May 1982, pp.399-409.

140. S.L.Tanimoto, 'A Pyramidal Approach to Parallel Processing', Proc. 10th Annual Int. Symp. on Computer Architecture, June 1983, pp. 372-378.

141. S.Thanawastien, V.P.Nelson, 'Interference Analysis of Shuffle/Exchange Networks', Proc. Thirteenth Annual Int. Symp. on Fault-Tolerant Computing, June 1983, pp.442-445.

142. C.D.Thompson, 'Generalized Connection Networks for Parallel Processor Intercommunication', IEEE Trans. on Computers, C-27, December 1978, pp.1119-1125.

143. K.J.Thurber, 'Circuit Switching Technology: A State-of-The-Art Survey', IEEE Computer Society Compcon, September 1978, pp.116-124.

144. K.J.Thurber, 'Parallel Processor Architectures - Part 1: General Purpose System', Computer Design, Vol.18, January 1979, pp.89-97.

145. K.J.Thurber, G.M.Masscn, 'Distributed - Processor Communication Architecture, (Lexington Books, Lexington, Mass.), 1979.

146. A.Varma, C.S.Raghavendra, 'Performance Analysis of a Redundant Path Interconnection Network', Proc. Int. Conf. on Parallel Processing, August 1985, pp.474-479.

147. A.Varma, C.S.Raghavendra, 'Realization of Permutations on Generalized Indra Networks', Proc. Int. Conf. on Parallel Processing, August 1985, pp.328-333.

148. B.W.Wah, A.Hicks, 'Distributed Scheduling of Resources on Interconnection Networks', AFIPS Conf. Proc. June 1982, pp.692-709.

149. A.Waksman, 'A Permutation Network', J. ACM, Vol.15, January 1968, pp.159-163.

150. C.W.Weiss, 'Bounds on the Length of Terminal Stuck Fault Tests', IEEE Trans. on Computers, Vol.C-21, March 1972, pp.305-309.

151. L.D.Wittie, 'Communication Structures for Large Networks of Microcomputers', IEEE Trans. on Computers, Vol.C-30, April 1981, pp.264-273.

152. C.L.Wu, T.Feng, 'On a Class of Multistage Interconnection Networks', IEEE Trans. on Computers, Vol.C-29, August 1980, pp.694-702.

153. C.L.Wu, T.Y.Feng, 'The Universality of the Shuffle-Exchange Network', IEEE Trans. on Computers, Vol.C-30, May 1981, pp.324-332.

154. C.L.Wu, T.Feng, 'Fault Diagnosis for a Class of Multistage Shuffle/Exchange Networks', IEEE Trans. on Computers, Vol. C-30, October 1981, pp.743-758.

155. C.L.Wu, T.Feng, 'Introduction: Interconnection Networks for Parallel and Distributed Processing', IEEE Computer Society Press, 1984, pp.1-3.

156. J.Wu, T.Lin, 'A New Cell-Based Interconnection Network', Int. J. Electronics, Vol.59, September 1985, pp.375-382.

157. W.A.Wulf, C.G.Bell, 'C.mmp- A Multi-miniprocessor', AFIPS Conf. Proc. December 1972, pp.765-777.

158. P.C.Yew, D.H.Lawrie, 'An Easily Controlled Network for Frequently Used Permutations', IEEE Trans. on Computers Vol. C-30, April 1981, pp.296-298.