USCEE TECHNICAL REPORT NO. 510

FAULT TOLERANCE OF β-NETWORKS IN INTERCONNECTED MULTICOMPUTER SYSTEMS

by
John P. Shen

August, 1981

This research was supported by the Air Force Office of Scientific Research under Grant No. AFOSR-77-3352.

ABSTRACT

Several proposals have been made recently for using a class of connecting networks called β-networks in multicomputer systems, such as systems containing large numbers of microprocessors. A β-network is a connecting network composed of 2×2 crossbar switches called β-elements. This thesis presents a study of the fault tolerance of β-networks intended for multicomputer applications.

A fault model is specified which allows β-elements to be stuck in either of their two normal states. A new connectivity property called dynamic full access (DFA) is introduced which serves as the criterion for fault tolerance in β-networks. A fault is called critical if it destroys the DFA property. A minimal critical fault (MCF) is a critical fault none of whose proper subsets constitutes a critical fault. Two graph-theoretical characterizations of the minimal critical and noncritical faults of a β-network are presented. A β-network is defined to be k-fault tolerant or k-FT if the failure of any k or fewer β-elements does not destroy DFA. The largest k for which a β-network is k-FT is called the fault-tolerance (FT) parameter of the β-network.

In the synthesis of practical fault-tolerant β-networks, network performance must also be considered. A performance criterion called the communication-delay (CD) parameter is introduced, which is defined as the worst possible transmission delay through the β-network, measured in terms of the number of intervening β-elements between any pair of communicating devices. It is proven that the FT parameter k and CD parameter d of any β-network with n β-elements must satisfy the following bounds:

$$0 \leqq k \leqq n-1$$

$$\lfloor \log_2 n \rfloor + 1 \leqq d \leqq n.$$

These bounds are shown to be tight. The design of fault-tolerant β-networks for multicomputer systems typically involves striking a balance between fault tolerance and communication delay. Two β-network designs are obtained which possess extreme values for k and d. The modified inverse shuffle-exchange (MISE) β-network is shown to have FT parameter $k = 1$ and CD parameter $d = \lfloor \log_2 n \rfloor + 1$. Another β-network called the double parallel ring (DPR) network is shown to have FT parameter $k = n-1$ and CD parameter $d = n$. It is further demonstrated that the DPR-network is unique in achieving the maximum value n-1 of the FT parameter.

The preceding theoretical results are applied to the analysis of various β-network structures. Some general

properties of cascaded β-networks are derived. FT and CD parameters are obtained for each of the following well-known β-networks: the double tree (DOT) network, the indirect binary m-cube (m-IBC) network, and the Benes rearrangeable (BRS) network. Several promising topics for further research are discussed.

TABLE OF CONTENTS

CHAPTER page

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Computer design frequently involves the interconnection of many functional units to form a complex system. In standard von Neumann computers these functional units include data processors, controllers, memory units, and input/output devices. The functional units and resulting systems vary greatly in complexity. For example, registers and arithmetic logic units (ALUs) can be interconnected to form a central processing unit (CPU). A multiprocessor is formed by interconnecting a number of CPUs, memory units, and controllers. With the advent of low-cost microprocessors in the 1970s complex computer systems, such as relatively powerful multi-microprocessors, are becoming economically feasible [Thurber and Masson 1977,Rattner 1977]. Consequently, in recent years the study of distributed systems of interconnected computers, or multicomputer systems, has been an active area of computer architecture research [Anderson and Jensen 1975, Lipovski and Doty 1978]. In this chapter we present the rationale for using connecting networks to realize the interconnection structure of these systems. Motivations for studying the fault tolerance of these connecting networks are also discussed.

1.1 CONNECTING NETWORKS

An *interconnected multicomputer system* is informally defined here as a system of complex functional units or computing units supported by an *interconnection and communication network (ICN)*. Figure 1.1 illustrates this model of an interconnected multicomputer system. Each functional computing unit, or simply unit, is capable of data processing and storage, and has input and output interfaces to the ICN. The INC provides the necessary communication paths for information transfer among the units. Interconnected multicomputer systems include multiple computer systems [Russo 1977], MIMD (Multiple Instruction, Multiple Data stream) systems [Flynn 1972] and distributed computer systems [Jensen et al. 1976, Thurber and Masson 1977]. Such systems are well suited to executing computing tasks that can be divided into interacting and concurrently-executable subtasks.

1.1.1 Interconnection and Communication Networks (ICNs)

It is envisioned that as the component density of integrated circuits (ICs) continues to increase, computer architecture will be more concerned with the interconnection of processors and less concerned with processor design [Thurber and Masson 1977]. Furthermore, when the number

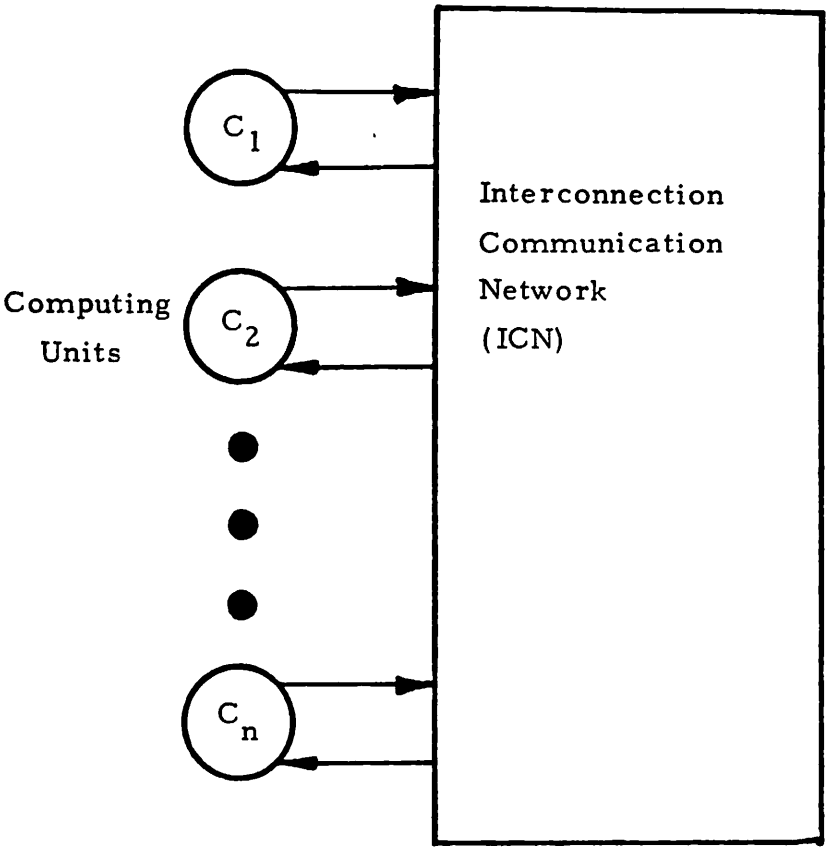


Figure 1.1. Logical model of an interconnected multicomputer system.

of units in an interconnected multicomputer system becomes large, the ICN becomes a major component of the system [Lipovski and Malek 1981]. The performance and fault tolerance of such systems can become more dependent upon the ICN than on the individual units. Hence the design of ICNs is usually a key part of interconnected multicomputer system design.

Various designs for ICNs have been proposed [Pease 1977, Sullivan and Bashkow 1977, Despain and Patterson 1978], and a few have been constructed [Swan et al. 1977, Sejnowski et al. 1980, Lundstrom and Barnes 1980]. Recently, Lipovski and Malek have proposed a classification scheme for ICNs based on elementary graph theory [Lipovski and Malek 1981]. In this study we loosely classify ICNs into three broad categories: (1) shared-bus, (2) dedicated-link, and (3) connecting networks; see Figure 1.2. The shared-bus design makes use of one or more common communication links called buses to support inter-unit communication. Each bus in the system is controlled by a bus controller which arbitrates the use of the bus by ensuring only two units are communicating via the bus at any given time. A well-known multicomputer system utilizing the shared-bus approach is the Cm* system developed at Carnegie-Mellon University [Swan et al. 1977]. Dedicated-link ICNs use dedicated or unshared links to interconnect the units. Each link allows communication between a fixed

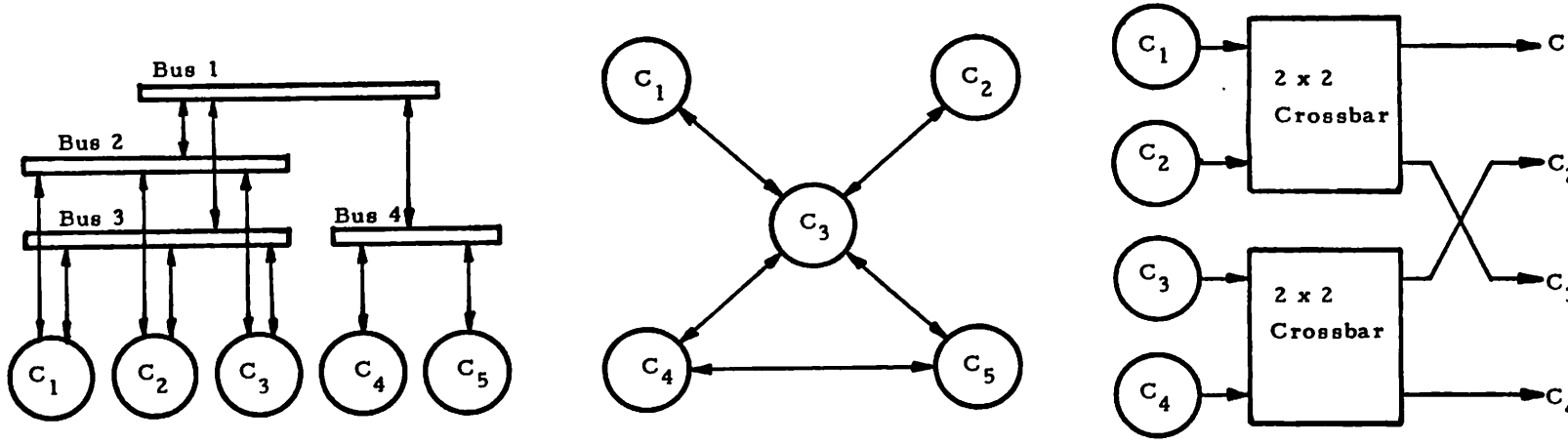


Figure 1.2. Possible ICN designs: (a) shared-bus; (b) dedicated-link; (c) connecting network.

pair of units. The most common interconnection topologies used for dedicated-link designs are star, ring, cube, tree and array structures [Lipovski and Malek 1981]. A *connecting network* is a system of switches that can be opened or closed to provide interconnecting paths from a set of input terminals to a set of output terminals [Masson et al. 1979]; see Figure 1.3a. In an interconnected multicomputer system the computing units represent both the set of input terminals and the set of output terminals, as illustrated in Figure 1.3b. A β-*network* is a connecting network of 2×2 crossbar switches called β-elements. β-networks constitute a very common class of connecting networks. It has been shown that the most efficient connecting networks can be realized using β-networks [Waksman 1968].

There are distinct advantages to using connecting networks as ICNs for large-scale interconnected systems. Because of their interconnection structure, connecting networks are capable of supporting large numbers of computing units and large numbers of simultaneous connections using a reasonable number of switches. With proper design a connecting network can provide connections with reasonably short delays, and have a relatively simple control structure. Connecting networks have built-in flexibility and redundancy which can be utilized for fault tolerance. Lastly, connecting networks have been extensively studied

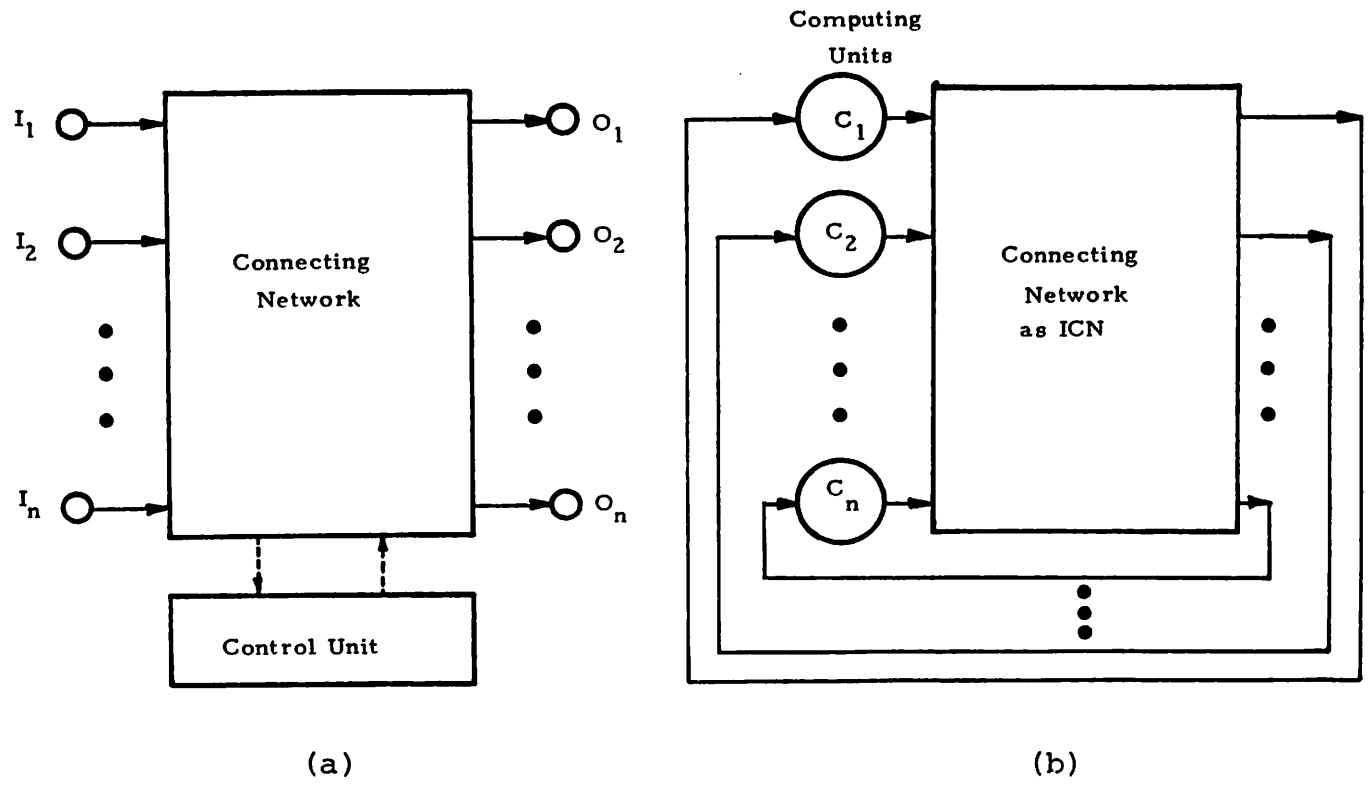


Figure 1.3. Applications of a connecting network: (a) in a general connecting system; (b) as an ICN in a multicomputer system.

for telephone switching systems, and have a sound theoretical basis [Benes 1965].

1.1.2 ICNs as Connecting Networks

Several proposals have been made for using connecting networks as ICNs in multicomputer systems. A connecting network called the indirect binary m-cube has been proposed for use in systems composed of a large array of microprocessors [Pease 1977]. The indirect binary m-cube provides logical interconnections resembling an m-dimensional cube. This same network was chosen by Sullivan and Bashkow for their proposed large-scale multicomputer system [Sullivan and Bashkow 1977]. A connecting network called the omega network, which is essentially similar to the indirect binary m-cube, has been proposed as a data alignment network for connecting processors and memory modules in an array processor [Lawrie 1975]. Researchers at Burroughs Corporation have selected the "baseline" network, suggested by Wu and Feng [Wu and Feng 1980], in their design of a MIMD Flow Model Processor for numerical aerodynamic simulation [Lundstrom and Barnes 1980]. A class of connecting networks called banyan networks has been extensively studied at the University of Texas at Austin [Tripathi and Lipovski 1979]. These networks serve the dual purpose of partitioning the functional units in an interconnected system and

providing communication links between them. A specific banyan network is used in the implementation of the Texas Reconfigurable Array Computer (TRAC) [Kapur et al. 1980, Premkumar et al. 1980]. Detailed descriptions of some of these connecting networks are provided in subsequent chapters.

1.2 FAULT TOLERANCE

In the previous section, we outlined the motivations for interconnected multicomputer systems and demonstrated the relevance of using connecting networks as their ICNs. In this section we discuss the rationale for studying the fault tolerance of connecting networks, summarize previous work in this area, and present a list of proposed tasks for this study.

Fault tolerance is the ability of a system to continue operating in the presence of faults. It is an important issue in the design of the ICNs for several reasons. In some applications, a fault can have serious consequences, and cannot be allowed to cause the system to fail. These applications include those systems in which the failure of a computing unit can result in the loss of human lives or valuable nonhuman resources, or cause severe inconvenience. In many real-time applications the ability to tolerate faults is a necessity in order to ensure uninterrupted system operation. The natural redundancy of having a

multiplicity of computing units in an interconnected multicomputer system can be utilized to tolerate faulty units. Fault tolerance is possible because the tasks performed by a faulty computing unit can be assumed by other units. With the maturation of very large-scale integration (VLSI), it is likely each unit will consist of only a few highly integrated circuit chips. If a fault occurs in some unit, the entire unit can easily be switched off and replaced. Hence the fault tolerance of a large-scale interconnected multicomputer system is likely to be a function of the fault tolerance of its ICN.

1.2.1 Previous Work

Most previous research has focused on the performance aspects of connecting networks. Various specific network structures and their associated combinatorial properties have been investigated. A primary goal of that work is to achieve the greatest connecting capability using the fewest switches. Permutation or symmetric group theory has been frequently used in the analysis of connecting networks. Another major area of research is the design of control algorithms to set the switches in proper states according to connection requests. Because of their combinatorial complexity, connecting networks often are difficult to control. The design of an efficient control algorithm

is crucial in providing a large number of fast and simultaneous connections in a connecting network. The performance and traffic patterns of connecting networks have also been studied using stochastic models. Numerous new applications of connecting networks have been proposed in recent papers [Siegel 1979, Hopper and Wheeler 1979, Leung and Dennis 1980].

The fault-tolerance aspects of connecting networks, despite their growing importance, have received little attention. There is therefore no general theory for the fault tolerance of connecting networks. This study is an attempt to remedy this situation for β-networks. We now summarize previous work dealing with the fault tolerance of β-networks.

Levitt, Green, and Goldberg of Stanford Research Institute (now SRI International) investigated the fault tolerance of a class of β-networks called CPCU (complete permutation-complete utilization) networks as part of a study for a self-repairable multiprocessor [Levitt et al. 1968]. Connecting networks, or "commutation networks," as they are called by Levitt et al., are used to connect memory modules, processors, control units, and I/O device controllers. CPCU networks are similar to Benes rearrangeable connecting networks [Benes 1965]. Levitt et al. use a fault model which allows β-elements to be stuck in either

of their two normal states. The two inputs of a β-element can be connected to the two outputs in two different ways. They devised a fault-detecting and fault-correcting β-network which can be appended to a CPCU network to make the composite network tolerate single β-element faults; see Figure 1.4.

An $n \times n$ β-network provides one-to-one connection paths from the set of n input terminals to the set of n output terminals. Each connection of the n inputs to the n outputs can be represented by a permutation of the inputs. The set of all possible connections in a network is then equivalent to the set of all permutations of inputs realizable by the network. Opferman and Tsao-Wu of Bell Laboratories have enumerated the permutations achievable in certain rearrangeable connecting networks [Opferman and Tsao-Wu 1971]. They also use the β-element stuck-at fault model of Levitt et al. Theorems for constructing a set of connecting patterns for diagnosing faults are given in [Opferman and Tsao-Wu 1971]. They describe a procedure for detecting and locating faulty β-elements, and show that by using only a pair of test permutations, the failure of any number of faulty β-elements can be easily detected. Furthermore, if only one or two β-elements fail, they can be located by employing a simple procedure.

Sowrirajan and Reddy have recently pursued the work of Opferman et al., using the same class of rearrangeable

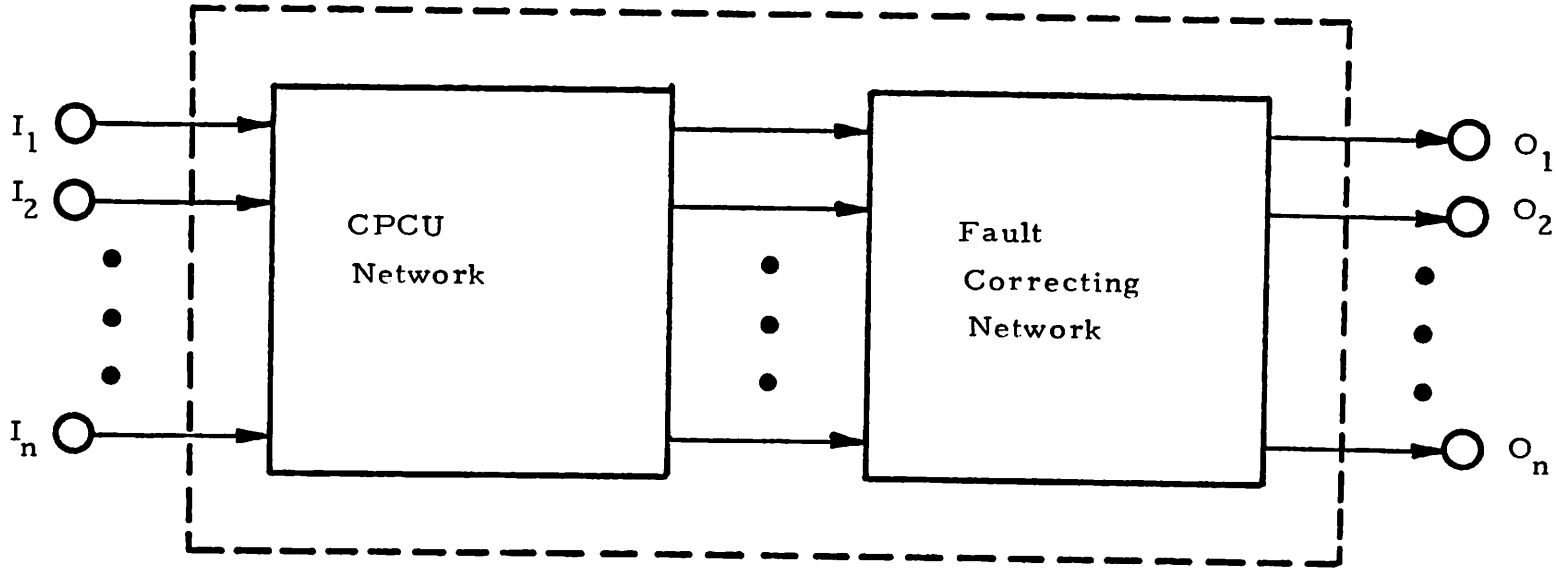


Figure 1.4. A fault-tolerant CPCU (complete permutation-complete utilization) network.

connecting networks and the same fault model [Sowrirajan and Reddy 1980]. They showed how one such network can be made to tolerate single faults by adding a redundant β-element. Their fault tolerance criterion is the maintenance of rearrangeability.

Several other researchers have touched upon the fault tolerance of β-networks in other contexts [Leung and Dennis 1980, Lundstrom and Barnes 1980, Thanawastien and Nelson 1981]. Most of these contributions are of a general and philosophical nature. As can be observed from the foregoing summary, previous research has focused on particular classes of β-networks. Little progress has been made in establishing a general theory with which the fault tolerance of all β-networks can be analyzed. Our goal is to establish such a theory.

1.2.2 Proposed Tasks

We now present a list of the major objectives for our study of the fault tolerance of β-networks used as ICNs in interconnected multicomputer systems.

1. To devise a structural model for β-networks on which a general theory of fault tolerance can be based.
2. To formulate useful fault models and fault-tolerance criteria for β-networks which are

appropriate for interconnected multicomputer system applications.

3. To develop measures of fault tolerance which can be used as figures of merit for different β-network types.

4. To apply the foregoing models and fault-tolerance measures to the analysis of well-known β-networks.

5. To develop methods for synthesizing β-networks with specified levels of fault tolerance.

1.3 THESIS OUTLINE

Chapter 2 presents a general introduction to β-networks. Connecting networks, β-networks, and measures of their connecting capability are formally defined. A formal technique for describing the structure of complex β-networks is derived. A graph model of β-networks is described along with a classification scheme for β-networks based on this model.

In Chapter 3 a fault model and a new fault-tolerance criterion called dynamic full access (DFA) are introduced. A network fault is defined as a collection of β-element stuck-at faults. A fault is said to be critical if it destroys the DFA property of the network. Minimal critical

faults of a β-network. We present a characterization of the set of all MCFs of a β-network, and necessary and sufficient conditions for a fault to be non-critical.

Chapter 4 discusses the tradeoffs between fault tolerance and communication delay in β-networks. A β-network called the double-parallel ring (DPR) network is shown to possess the maximum fault tolerance and to be unique. A modification method for β-networks with minimal delay is described which introduces fault tolerance without increasing delay.

In Chapter 5 we apply our theoretical results to the analysis of several well-known families of β-networks. These networks include the fault-correcting network of Levitt et al., the indirect binary m-cube network proposed by Pease, and the Benes rearrangeable network.

A thesis summary is provided in Chapter 6, and some further research topics are discussed.

CHAPTER 2

β-NETWORKS

In this chapter previous work on β-networks is summarized and the possible applications of these networks are reviewed. A formal technique for describing the structure of complex β-networks is discussed, and a new graph model for analyzing β-networks is introduced.

2.1 INTRODUCTION

According to Benes' definition [Benes 1965], a *connecting system* is a physical communication system consisting of three major components:

1. a set of input and output terminals,
2. a set of control units which process requests for connection, usually between pairs of terminals,
3. a connecting network, which provides the requested connections.

Figure 2.1 illustrates the structure of a general connecting system. A well-known example is a telephone switching system. In such a system, the input terminals may corres-

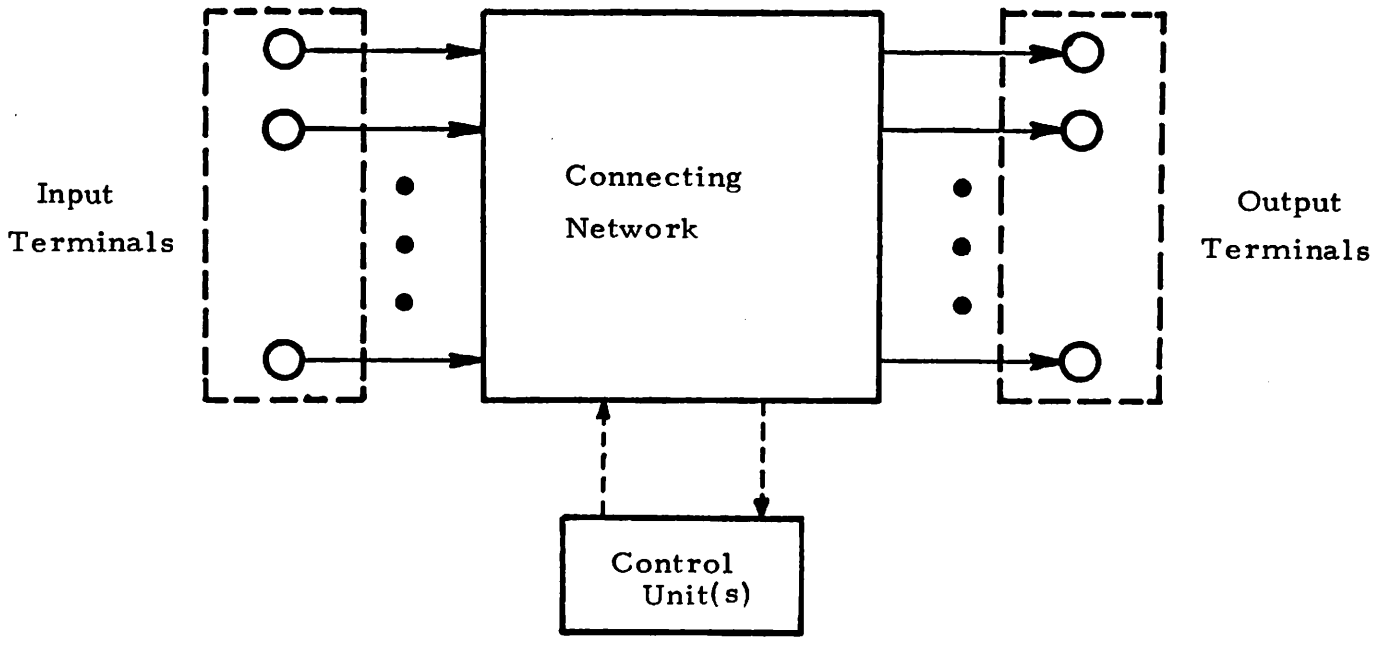


Figure 2.1. A general connecting system.

pond to telephone lines and the output terminals correspond to trunk lines. The control unit, which in recent telephone systems is a stored-program digital computer, is responsible for examining connection requests, determining the appropriate connection routes, and setting switches in the connecting network to provide these connections. The connecting network thus provides the physical communication paths between the input and output terminals.

Typically, a connecting network operates in the following way:

1. Requests for connection between pairs of idle input and output terminals arise.
2. The requests are processed by the control unit and, if possible, the desired connections are established.
3. The established connections are maintained in the network until communication ends.
4. Terminals are returned to the idle state when the associated communication ends.

2.1.1 Connecting Networks

The connecting system component of interest to us here is the connecting network. In general, a *connecting network* is a set of simple switches or *crosspoints*, and a

set of circuits or *links* that join the crosspoints to one another and to the input and output terminals. The links are the physical transmission media through which the terminals communicate. The crosspoints can be either open or closed to provide the requested communication paths from the input terminals to the output terminals. It is assumed throughout this work that no feedback paths exist inside a connecting network. Typically the connections are one-to-one, i.e., each input terminal can be connected to only one output terminal at a time. A network with n input terminals and m output terminals is called an $n \times m$ *connecting network*.

In many connecting networks, the number of input terminals n is equal to the number of output terminals m, and all the connections are one-to-one. We are primarily interested in $n \times n$ connecting networks of this kind. Each connection of the n inputs to the n outputs can be represented by a permutation of the inputs, since the data received at the n outputs can be viewed as a permutation of the data at the n inputs. Hence, the name *permutation network* is sometimes used for this type of connecting network [Joel 1968]. The $n \times n$ crossbar switch in Figure 2.2 is an example of an $n \times n$ permutation network. This network consists of n^2 switches or crosspoints, with one crosspoint connecting every pair of input and output terminals. Due to the large number of crosspoints it contains, an $n \times n$

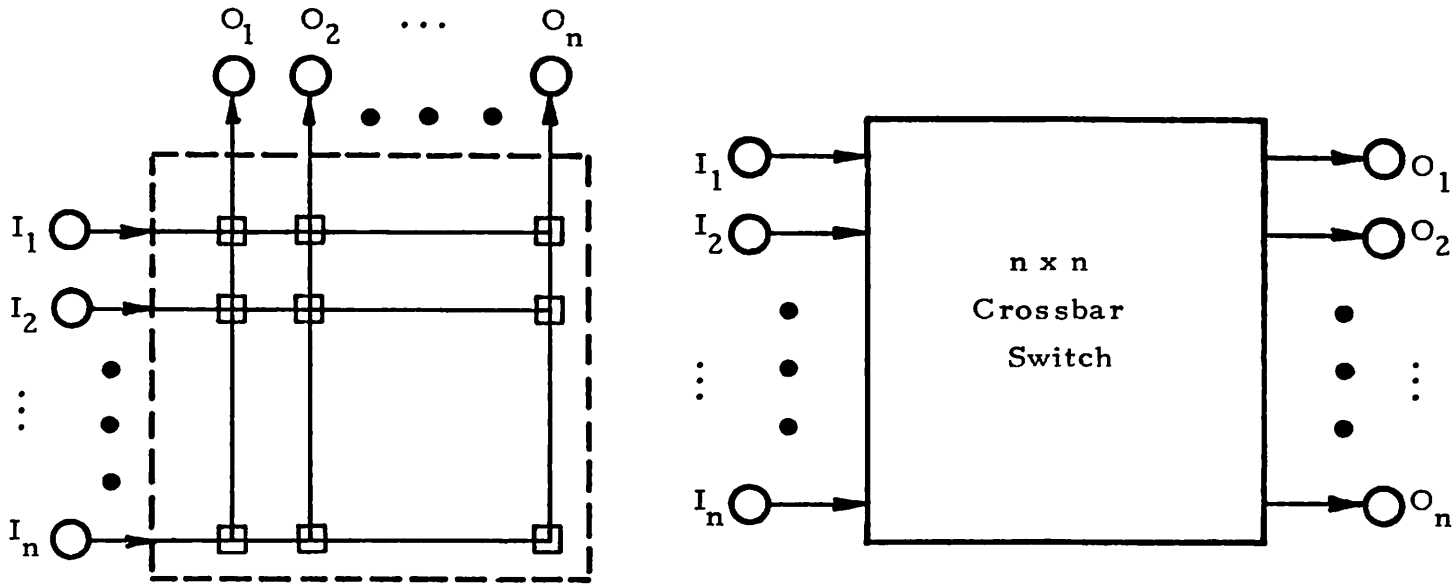


Figure 2.2. An n × n crossbar switch.

crossbar switch is very costly, especially when n is large. The connecting capability of an $n \times n$ crossbar switch can be achieved with fewer than n^2 crosspoints by interconnecting several smaller crossbar switches. Consequently, a permutation network generally is implemented as a network of many small permutation networks. The smallest permutation network has 2 inputs and 2 outputs and is commonly called a β-*element*. A β-element has two states corresponding to the two possible permutations of its input terminals. Diagrams depicting a β-element and its states appear in Figure 2.3. The two states of a β-element will be called here the through-state or *T-state*, and cross-state or *X-state*.

Definition 2.1: An $n \times n$ β-*network* is a (feedforward) connecting network with n input and n output terminals formed by interconnecting a set of β-elements.

Any $n \times n$ crossbar switch can always be implemented by a network of β-elements. Hence every $n \times n$ connecting network consisting of a network of crossbar switches can be decomposed into a β-network possessing the same connecting capability. For example, Figure 2.4a shows a connecting network called the Clos network [Benes 1965], and Figure 2.4b shows an equivalent β-network. An important advantage of a β-network is its ease of control; each β-element re-

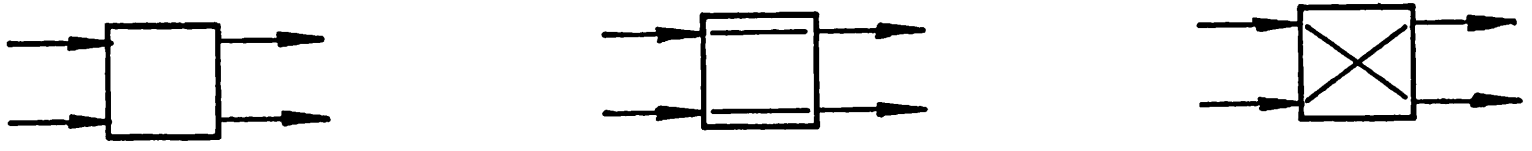
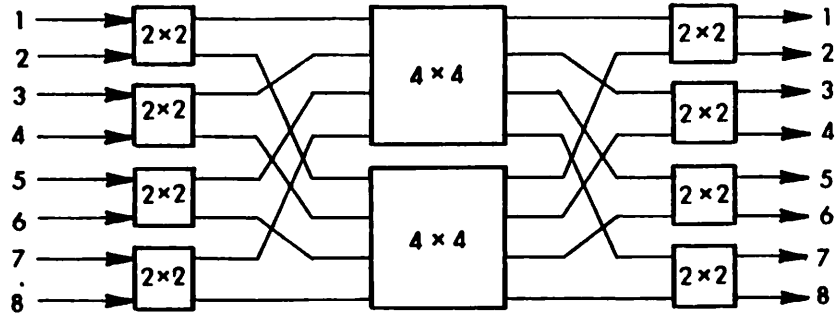
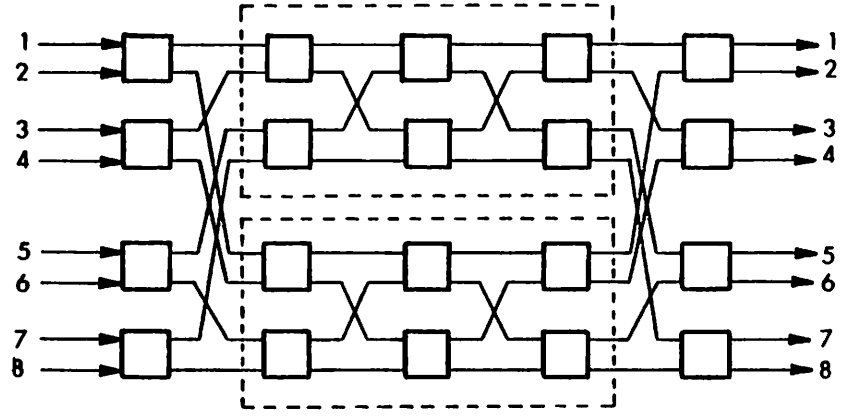


Figure 2.3. A β-element and its two possible states: (a) β-element in unspecified state; (b) β-element in through or T-state; (c) β-element in cross or X-state.



(a)



(b)

Figure 2.4. An example of a connecting network and an equivalent β-network: (a) the three-stage 8 × 8 Clos network; (b) an equivalent 8 × 8 β-network (Benes network).

quires only one binary control signal to change its state. Furthermore, connecting networks that are efficient in the sense of achieving a certain connecting capability with the fewest crosspoints, can be realized with β-networks [Waksman 1968]. Many well-studied connecting networks, such as the Benes rearrangeable switching network [Benes 1965], the Flip network used in Staran [Batcher 1976], Pease's indirect binary m-cube [Pease 1977], Stone's shuffle exchange network [Stone 1971], and Patel's base-2 delta network [Patel 1979], are β-networks. Since β-networks appear to constitute the most important class of connecting networks, they were selected as the subject of this study.

An $n \times n$ β-network N can be viewed as a set of z β-elements $B = \{b_1, b_2, \ldots, b_z\}$ interconnected by a set of 2z links $L = \{\ell_1, \ell_2, \ldots, \ell_{2z}\}$. In an interconnected computer system, the n input terminals and the n output terminals of the β-network are identical. The n links originating from the (input) terminals and the n links terminating at the (output) terminals can be considered to be identical, and are called the *terminal links* $T = \{t_1, t_2, \ldots t_n\}$ of the β-network. A *connection* of N is a one-to-one mapping from the input to the output terminals, and is represented by a permutation p(T) of T. Following the usual mathematical notation, p(T) can be defined by the expression

$$p(T) = \begin{pmatrix} t_1 & t_2 & \cdots & t_n \\ p(t_1) & p(t_2) & \cdots & p(t_n) \end{pmatrix}$$

where t_i and $p(t_i)$ are in T, and $p(t_i)$ is the output terminal mapped onto the input terminal t_i for $i = 1,2,\ldots,n$. The *state* of N is determined by the states of its z β-elements. If $s_i = s(b_i) \in \{T,X\}$ denotes the state of the β-element b_i, then a state of N is represented by a z-tuple $s(B) = s(b_1,b_2,\ldots,b_z) = (s_1,s_2,\ldots,s_z)$. If some of the β-elements are not specified, or their states are not of interest, then we can characterize s(B) by the *partial state*, $s(B) = (s_1,s_2,\ldots,s_z)$, where $s_i \in \{T,X,d\}$, and d represents an unspecified or don't care state. A connection p of N is *realizable* if there exists a state s of N such that by setting N to state s, the one-to-one mapping specified by P is established. It is possible that a connection of N can be realized by more than one state of N. For example, to realize the connection

$$p = \begin{pmatrix} t_1 & t_2 & t_3 & t_4 \\ t_2 & t_3 & t_4 & t_1 \end{pmatrix}$$

in the β-network of Figure 2.5, N must be set to the state $s(b_1,b_2,b_3,b_4) = (T,X,X,T)$.

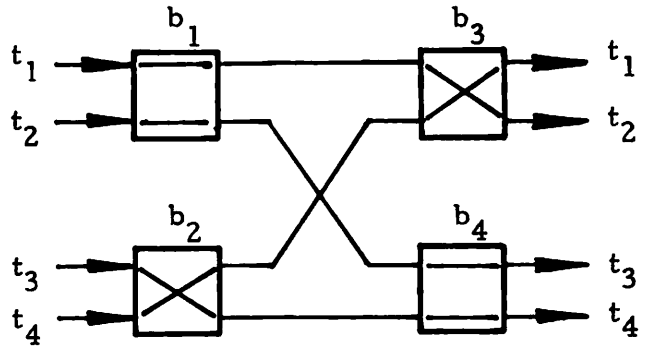


Figure 2.5. A 4×4 β-network realizing the connection $\begin{pmatrix} t_1 & t_2 & t_3 & t_4 \\ t_2 & t_3 & t_4 & t_1 \end{pmatrix}$.

2.1.2 Connecting Capability

The performance of a connecting network can be measured by its ability to provide connections between the input and output terminals. There are several well-known measures of connecting capability which we now summarize in order of decreasing capability [Benes 1965]. A connecting network is said to be in a *blocking state* when some pair of idle terminals cannot be connected. A connecting network is *nonblocking in the strict sense* if it has no blocking states. A connecting network is *nonblocking in the wide sense* if by suitably choosing routes for new requests, all blocking states can be avoided and all new requests can be completed. An $n \times n$ crossbar switch is an example of a strictly nonblocking network. There exists a unique crosspoint connecting every pair of input and output terminals. A request to connect any pair of idle input and output terminals can be completed by simply closing the corresponding crosspoint.

An $n \times n$ crossbar switch requires n^2 crosspoints. However, a strictly nonblocking connecting network can be realized using fewer than n^2 crosspoints. Clos in his classic paper [Clos 1953] presented one such nonblocking connecting network consisting of three stages of small crossbar switches. The three-stage Clos network is illustrated in Figure 2.6. The first stage contains r $n \times m$

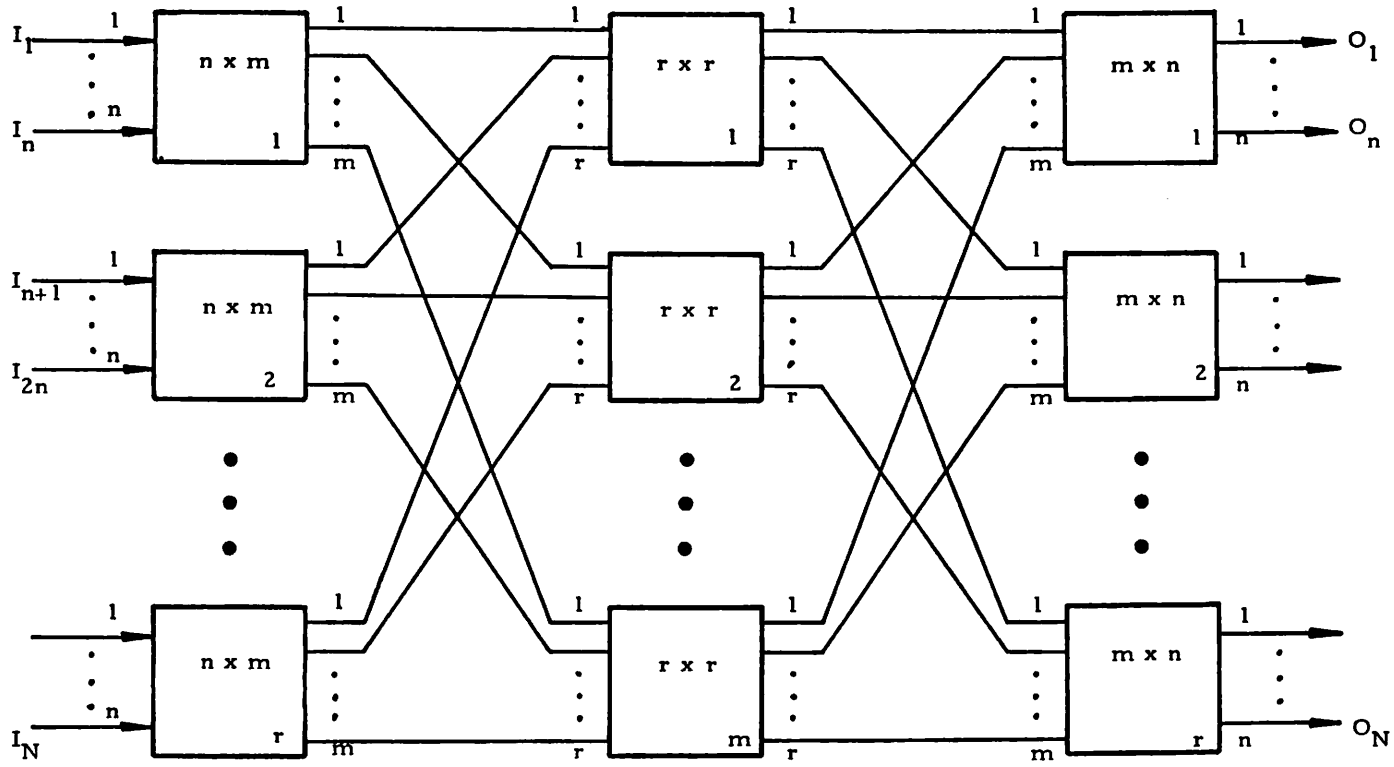


Figure 2.6. The general three-stage Clos network.

crossbar switches, and the third stage contains r $m \times n$ crossbar switches. The network has N input and N output terminals where $N = n \times r$. The middle stage contains m $r \times r$ crossbar switches. For every switch in the middle stage, there is an input link connecting it with each switch in the first stage and an output link connecting it with each switch in the third stage. Clos has shown that for $m \geq 2n-1$, the three-stage network of Figure 2.6 is nonblocking in the strict sense. For reasonably large N, the three-stage Clos networks require on the order of $N^{3/2}$ crosspoints. Although this represents an improvement over a simple crossbar switch, nonblocking connecting networks generally require many crosspoints and thus tend to be very expensive. Furthermore, the property of being nonblocking is not necessary in many applications. Hence, large nonblocking connecting networks are rarely implemented.

A type of connecting network which is less powerful but more useful than a nonblocking network is a rearrangeable network. A connecting network is *rearrangeable* if any set of active connections can be assigned new routes to permit connecting any idle pair of terminals. Blocking can occur in a rearrangeable network, but any request for connection can be completed after rearranging ongoing connections. A state exists in a rearrangeable network which realizes each possible connection, i.e., all N! connections

are realizable. The well-known Slepian-Duguid theorem states that the three-stage Clos network is rearrangeable if and only if $m \geq n$ [Benes 1965]. Figure 2.4a depicts the rearrangeable three-stage Clos network where $N = 8$ and $m = n = 2$. Benes has designed a class of rearrangeable connecting networks composed of multiple stages of identical square crossbar switches. An 8×8 Benes rearrangeable network made up of five stages of 2×2 crossbar switches is shown in Figure 2.4b. This network is equivalent to that of Figure 2.4a in terms of connecting capability.

A class of connecting networks having less connecting capability than the rearrangeable networks are the full-access networks. A network has the *full-access* property if each of the input terminals can be connected to any one of the output terminals. Pease's indirect binary m-cube is an example of a full-access network which is neither rearrangeable nor nonblocking [Pease 1977]. The 8×8 indirect binary 3-cube network is illustrated in Figure 2.7. As can be observed, there exists a unique route from each input terminal to each output terminal. An example of a network without the full-access property can be obtained by deleting the third stage of β-elements from Figure 2.7.

Clearly, 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,

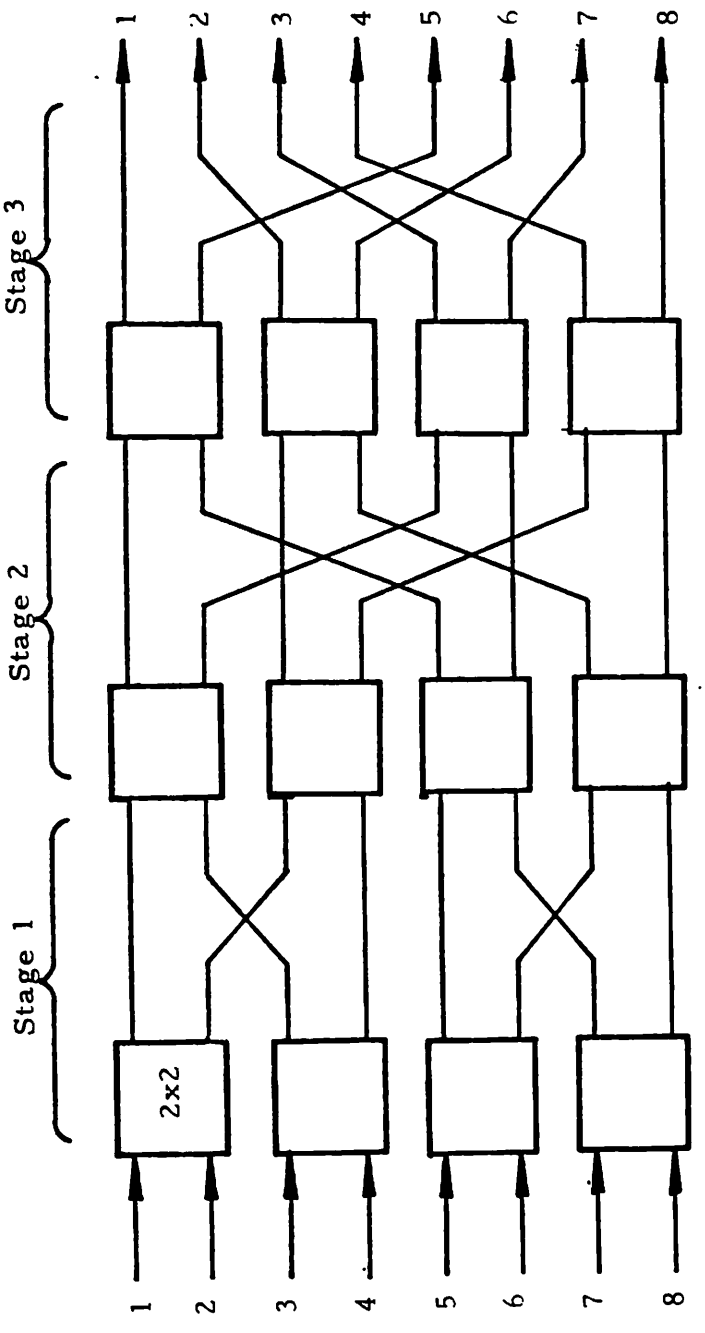


Figure 2.7. The indirect binary 3-cube network.

which may be measured by the number of crosspoints and links used, and the complexity of the control algorithm software. Greater connecting capability normally implies higher cost. As stated earlier, large nonblocking networks are rarely implemented because of their high cost. Rearrangeability is a very desirable property for telephone switching networks. Hence rearrangeable networks have been extensively studied in that context, and many rearrangeable networks have been implemented. The most efficient $n \times n$ rearrangeable networks can be implemented using on the order of $n \times (\log_2 n)$ crosspoints. Unlike telephone switching systems which are primarily communication systems, interconnected computers utilize interunit communication only to support their computation. The demands for connecting capability in these two types of systems are therefore quite different. It appears that rearrangeability is not essential for computer systems. In fact rearrangeable networks may be undesirable due to the complexity of their control algorithms. The full-access property guarantees connectivity between all the input and output terminals, and is achievable with low hardware and control software cost. It therefore appears to be the key connecting capability of interest in the study of connecting networks for interconnected multicomputer systems.

2.1.3 Applications

Telecommunication systems constitute the earliest and most important application of connecting networks. Examples of such systems include telephone central offices, telegraph networks, and military communication systems. In these systems, the connecting networks perform circuit switching. Upon request a dedicated communication path or circuit is created between two terminals. Typically, analogue signals, e.g., voice signals, are transmitted over these circuits, but digital signal transmission is increasingly being used. These networks may be controlled by special-purpose analogue control units, or, more recently, by general-purpose digital computers. All these applications share three common properties [Benes 1965]:

1. great combinatorial complexity in terms of the number of circuit possible,
2. definite geometrical structure,
3. the occurrence of random requests in the system.

Connecting networks have other applications besides the design of circuit switching communication systems. In recent years, it has become technologically feasible to build computing systems containing large numbers of powerful components, such as microprocessors, microcomputers,

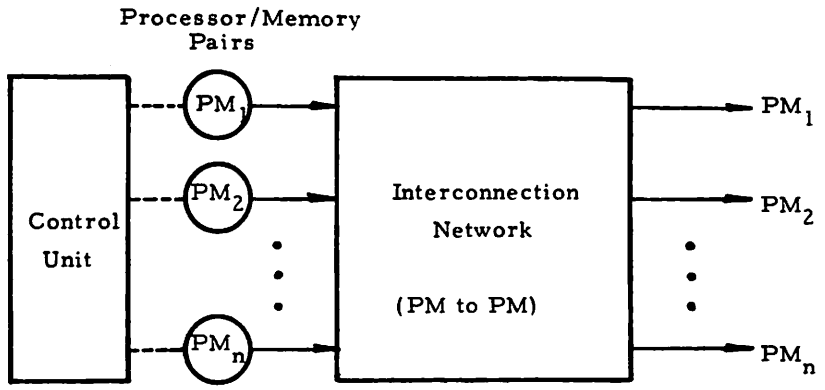
intelligent memories, and peripheral controllers. These systems, which are referred to as interconnected multicomputer systems in Chapter 1, can make effective use of an interconnection and communication network (ICN) to support communication among the various system components. The various properties of connecting networks described in Chapter 1 make them suitable as ICNs in interconnected multicomputer systems.

Local computer networks have undergone rapid development in the last few years [Clark 1968]. The primary aims of these networks are to allow efficient resource sharing, and to enhance overall system performance within a computing facility. A local computer network consists of a set of computers located close to one another, e.g., in the same building, and linked by a communication network. Each computer typically has its own operating system, and communicates with other computers through its I/O ports via the communication network. Because of their flexible structure and simple control requirements, β-networks have been proposed to provide communication in local computer networks. Two specific β-networks, implemented as packet switching networks, have been studied at Cambridge University [Hopper 1979] and MIT [Dennis 1980].

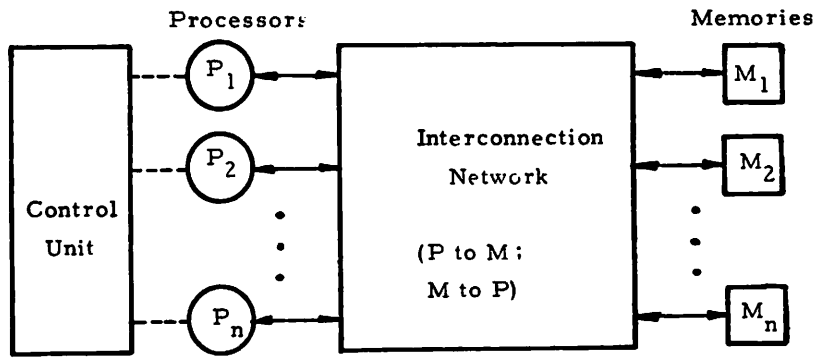
Presently, the principal application of connecting networks in computing-oriented systems is as interconnection networks in SIMD (single instruction stream-multiple

data stream) computers. SIMD systems are used to perform massive parallel computations such as those found in signal and image processing [Siegel 1979]. Typically, an SIMD system consists of a control unit, n processors, n memory units, and an interconnection network. Each processor is paired with a memory unit. All n processors execute the same instruction, broadcast to them from the control unit. Each processor executes the instruction on an operand located in its own memory unit. Several proposed and constructed interconnection networks for SIMD systems are β-networks [Pease 1977, Siegel 1978]. There are two possible uses for β-networks in SIMD systems. Each processor P and memory M pair can form a single physical module PM as shown in Figure 2.8a. In this case the set of input terminals and the set of output terminals of the β-network coincide. An alternative SIMD organization uses the β-network to provide connections between processors and memory units as shown in Figure 2.8b. These β-networks are sometimes referred to as data alignment networks [Lawrie 1975].

Several proposals have been made for using the structure of Figure 2.8a to implement MIMD (multiple instruction stream-multiple data stream) systems. In this case, each processor-memory unit contains its own controller and operating system, and is capable of functioning as an inde-



(a)



(b)

Figure 2.8. Two computer organizations employing interconnection networks: (a) interconnection of processor-memory pairs; (b) interconnection of processors and memories.

pendent computer. This type of organization is sometimes called a *multicomputer*. In such a system, there is usually no single central operating system. Instead the operating system functions are distributed throughout the computer; the control for the interconnection network can also be distributed. A multicomputer system differs from a local computer network in having a tighter physical and functional coupling among the computers. Figure 2.9 depicts the structure of a multicomputer system. β-networks appear to be very promising interconnection networks for multicomputers. Several such systems have been proposed [Sullivan and Bashkow 1977,Pease 1977,Lundstrom 1980].

Currently, a type of computer architecture called data flow architecture is being studied by several researchers [Dennis 1979,Davis 1979,Watson 1979]. The computational activities in a data flow machine are driven by the availability of the operand data rather than the availability of instructions as in a conventional computer. Consequently, a large number of computational activities can be in progress concurrently as long as all the needed operands are available. At MIT a data flow processor with the architecture illustrated in Figure 2.10 is being studied [Leung and Dennis 1980]. Operand packets are transmitted via an ICN called the distribution network to instruction cells. Each instruction cell sends an oper-

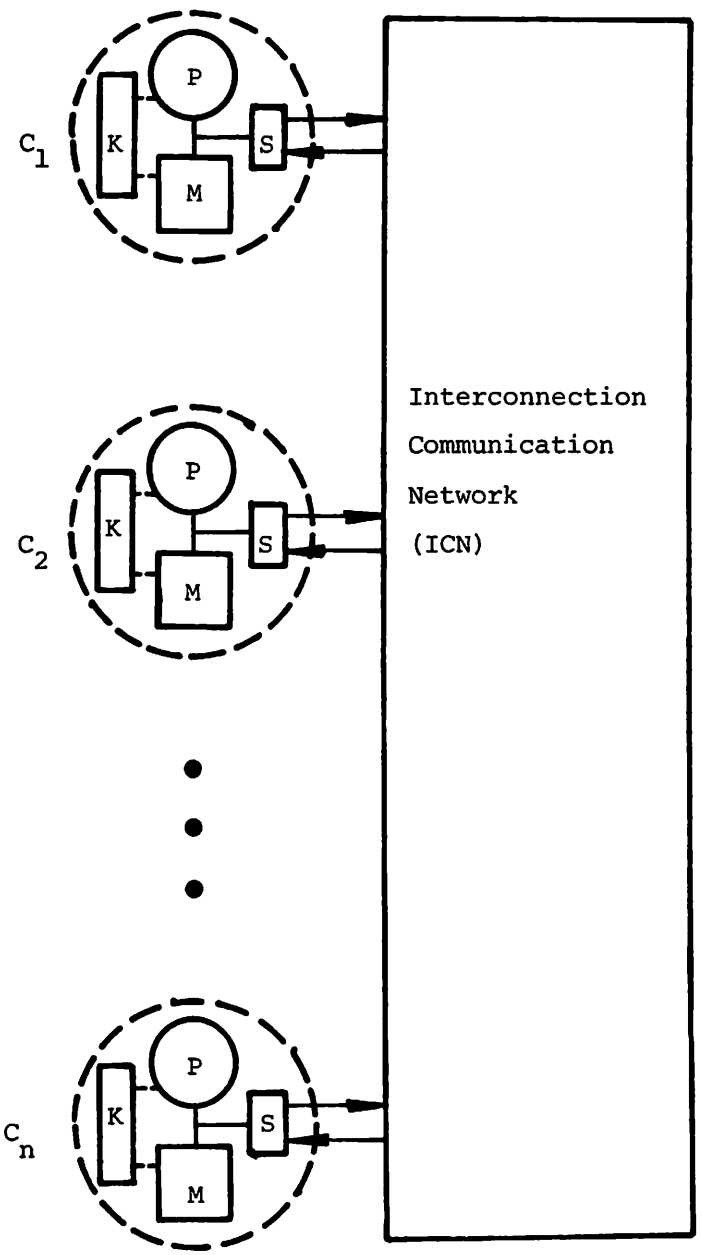


Figure 2.9. Detailed structure of a multicomputer system: P = processor; M = memory unit; K = control unit; S = interface switch; C = computer.

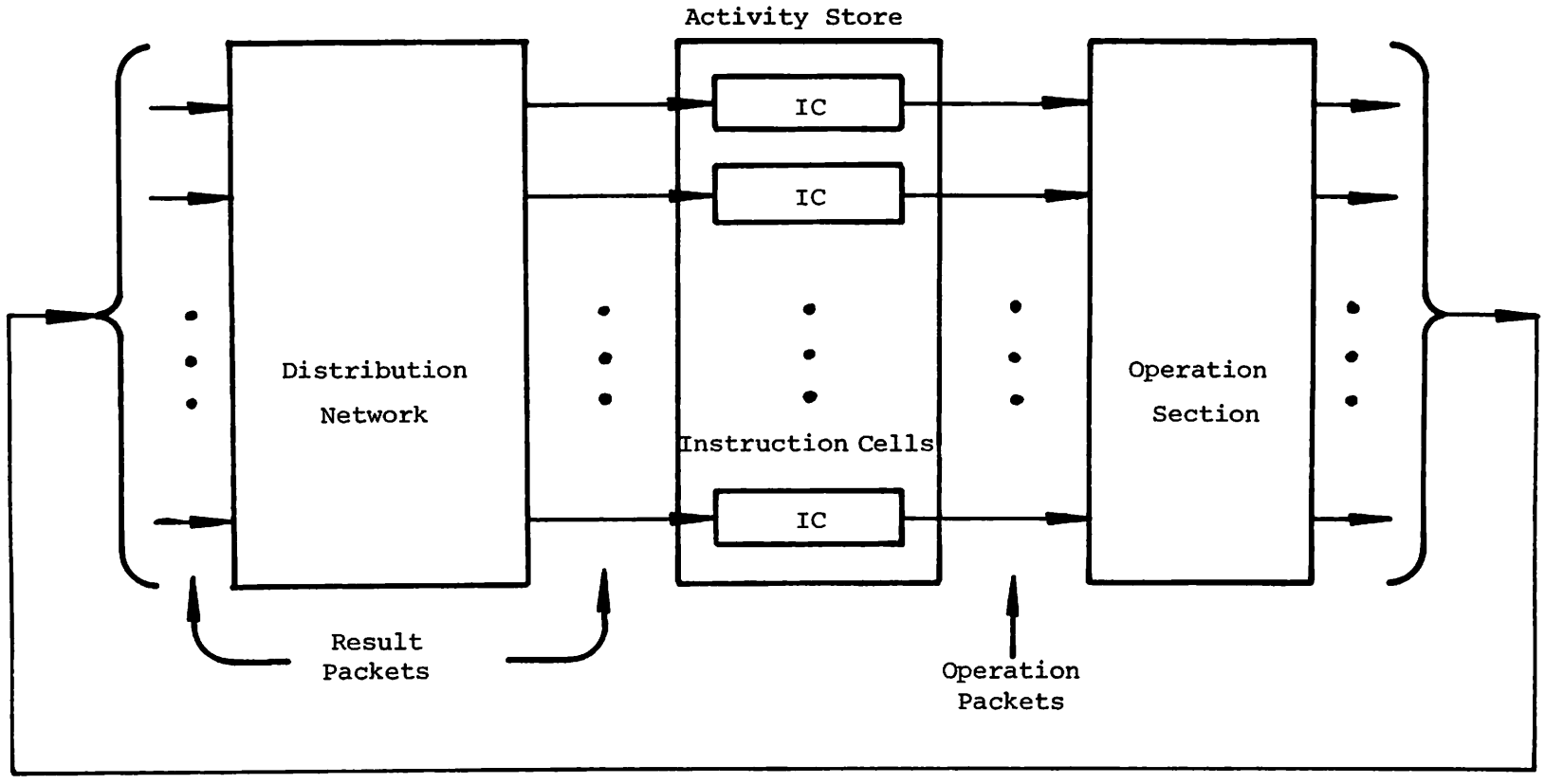


Figure 2.10. The MIT data flow processor architecture.

ation packet to the operation section when all the operands have been received. The operation section executes the instruction and forwards the result packet to another target instruction via the distribution network. A key component of the system is the distribution network which should be flexible and able to support high throughput. Because of these requirements, β-networks are being considered by Dennis et al. as candidate distribution networks in their data flow processor prototypes.

Besides providing connecting paths in an interconnected system, a β-network can also function as a partitioning network, where it is used to partition the resource units of an MIMD system into disjoint subsets. Each subset can then function independently by executing its own program(s) [Goke and Lipovski 1973]. β-networks can also be used to sort data items [Batcher 1968,Tarjan 1972]. By entering unordered data items at the inputs of the network, the items can be obtained in various ordered or sorted forms at the outputs of the network.

2.2 β-NETWORK STRUCTURES

A β-network is a collection of β-elements interconnected by fixed links. A β-element can be viewed as containing flexible links which can be programmed, i.e., set to certain states, to provide desired communication paths

from the inputs to the outputs of the β-element. This section develops a formal technique for concisely describing the topological structure of β-networks.

Large and complex β-networks are typically constructed from smaller β-networks; the smallest being the 2×2 β-element. Frequently, many identical subnetworks are connected to form a large network with a regular interconnection struction. Two very general interconnection methods for β-networks are now discussed.

2.2.1 Cascade and Stack Connections

The *dimension* of an $n \times n$ β-network N is $|N| = n$. By numbering the inputs and outputs from top to bottom, the set of inputs and the set of outputs of N can be denoted by two ordered sets $I(N) = (I_1, I_2, \ldots, I_n)$ and $O(N) = (O_1, O_2, \ldots, O_n)$, respectively. In an interconnected multicomputer system the inputs and outputs of N coincide, hence we can say that $I(N) = O(N)$, which means that I_i is connected to O_i for $i = 1,2,\ldots,n$. β-networks with the same dimension can be connected to form a cascade or series network; we now define this concept formally.

Definition 2.2: A β-network N is a *cascade* of β-networks $N_1, N_2, \ldots, N_y$, denoted $N = N_1 \circ N_2 \circ \cdots \circ N_y$, if $|N| = |N_1| = |N_2| = \cdots = |N_y|$, and $I(N) = I(N_1)$, $O(N_1) = I(N_2), \ldots, O(N_{y-1}) = I(N_y)$, $O(N_y) = O(N)$. If all the subnetworks are identical,

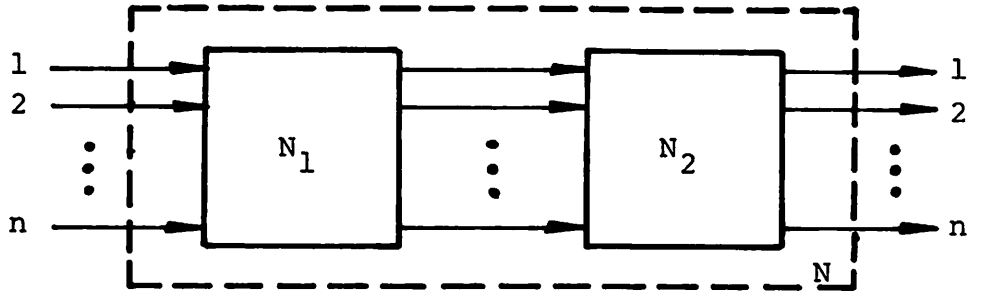
i.e., if $N_1 = N_2 = \cdots = N_y$, then we will write $N = N_1^y$.

Figure 2.11a shows a cascade of two networks N_1 and N_2. A cascaded β-network and all its subnetworks must have the same dimension. Many well-known β-networks are cascades of other networks. For example, the indirect binary 2-cube β-network [Pease 1977] of Figure 2.12b is a cascade of two copies of the 4 × 4 shuffle-exchange β-network [Stone 1971] in Figure 2.12a.

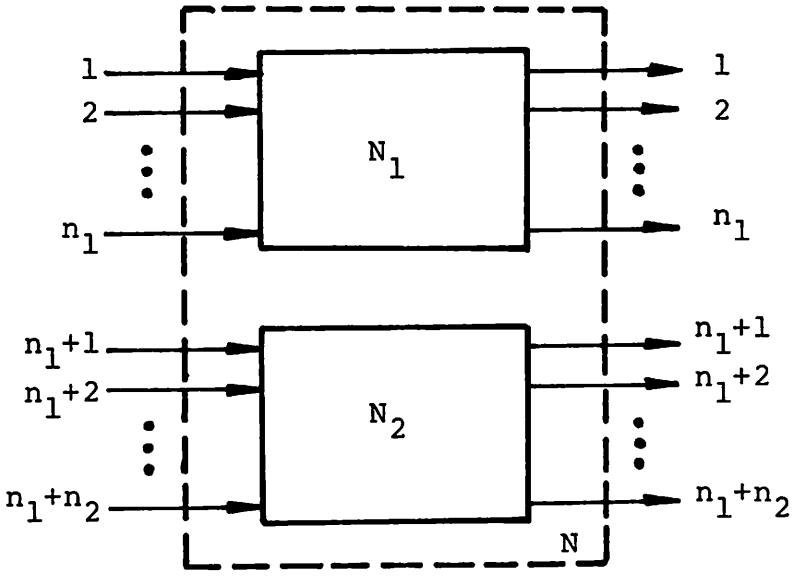
Given two ordered sets of terminals, $X = (X_1, X_2, \ldots, X_a)$ and $Y = (Y_1, Y_2, \ldots, Y_b)$, the *union* of X and Y, denoted $X \cup Y$, is another ordered set of terminals $Z = \{Z_1, Z_2, \ldots, Z_c\}$, such that $c = a+b$, and $Z_i = X_i$ for $i = 1,2,\ldots,a$, and $Z_i = Y_{i-a}$ for $i = a+1, a+2, \ldots, a+b$. We can now define another interconnection method involving the vertical composition or juxtaposition of networks.

Definition 2.3: A β-network N is a *stack* of β-networks $N_1, N_2, \ldots, N_w$, denoted $N = N_1 + N_2 + \cdots + N_w$, if $|N| = |N_1| + |N_2| + \cdots + |N_w|$ and $I(N) = I(N_1) \cup I(N_2) \cup \cdots \cup I(N_w)$ and $O(N) = O(N_1) \cup O(N_2) \cup \cdots \cup O(N_w)$. If all the subnetworks are identical, i.e., if $N_1 = N_2 = \cdots = N_w$, then we will write $N = wN_1$.

Figure 2.11b depicts a stack β-network formed from two subnetworks. The above two interconnection methods, cascade and stack, can be combined in the construction of complex β-networks. For example, the indirect binary



(a)



(b)

Figure 2.11. (a) Cascade $N = N_1 \circ N_2$ of two networks N_1 and N_2; (b) Stack $N = N_1 + N_2$ of N_1 and N_2.

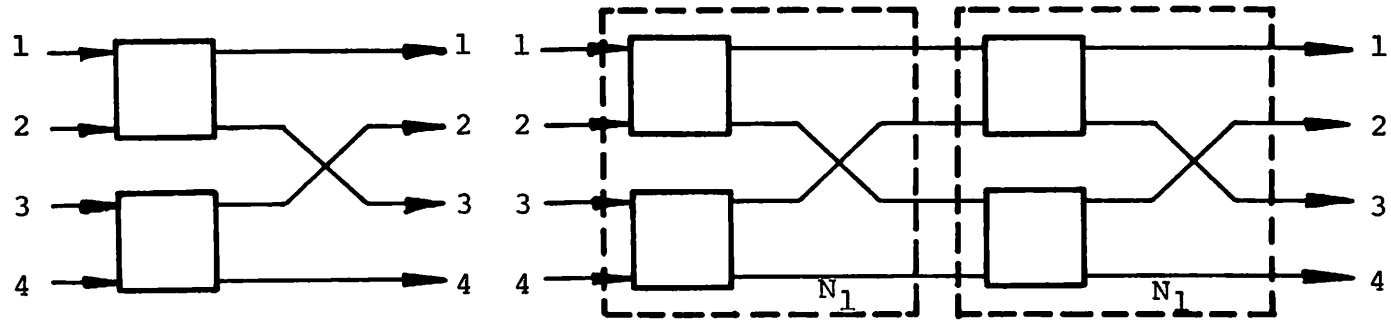


Figure 2.12. An example of a cascaded β-network: (a) the 4×4 shuffle-exchange network N_1; (b) the indirect binary 2-cube network $N_2 = N_1 \circ N_1$.

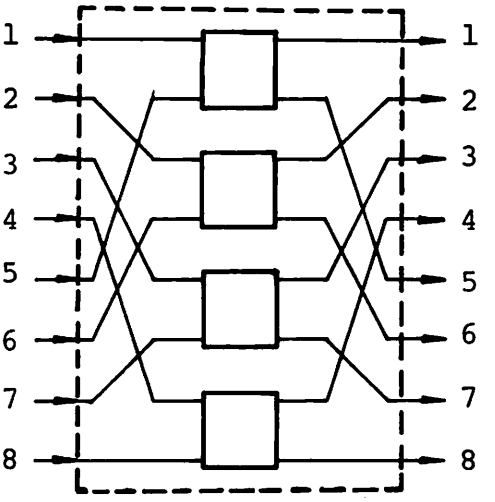
3-cube β-network of Figure 2.13b is a stack of two indirect binary 2-cube β-networks from Figure 2.12b cascaded with the β-network in Figure 2.13a.

The interconnection topology of the n links in a β-network can be conveniently described by a permutation of its terminals. An $n \times n$ *permuter* ρ is defined here as a network consisting of n fixed links connecting two sets of n terminals. The connections realized by the permuter ρ can be represented by a permutation of n elements thus

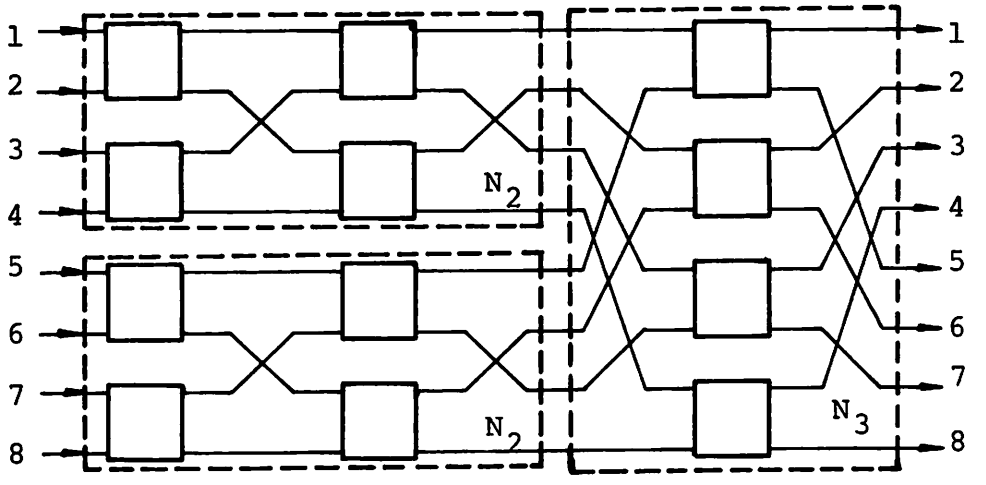
$$\rho = \begin{pmatrix} t_1 & t_2 & & t_n \\ \rho(t_1) & \rho(t_2) & \cdots & \rho(t_n) \end{pmatrix}$$

where t_i is connected to $\rho(t_i)$ for $i = 2,\ldots,n$. Figure 2.14a depicts a general $n \times n$ permuter, and Figure 2.14b shows an example of an 8×8 permuter. An $n \times n$ permuter can be considered to be a degenerate or empty $n \times n$ β-network, and hence can be used as a subnetwork in the construction of large networks.

Since a β-network is composed of β-elements and fixed links, β-elements and permuters can be considered as the most primitive elements used in the construction of β-networks. The two interconnection methods, cascade and stack, can be defined as operators on the primitive elements. All β-networks of interest can be formed by applying the cascade (•) and stack (+) interconnections to a



(a)



(b)

Figure 2.13. Combined use of cascade and stack constructions: (a) an 8×8 single stage β-network N_3; (b) the indirect binary 3-cube $N_4 = (N_2+N_2) \cdot N_3$.

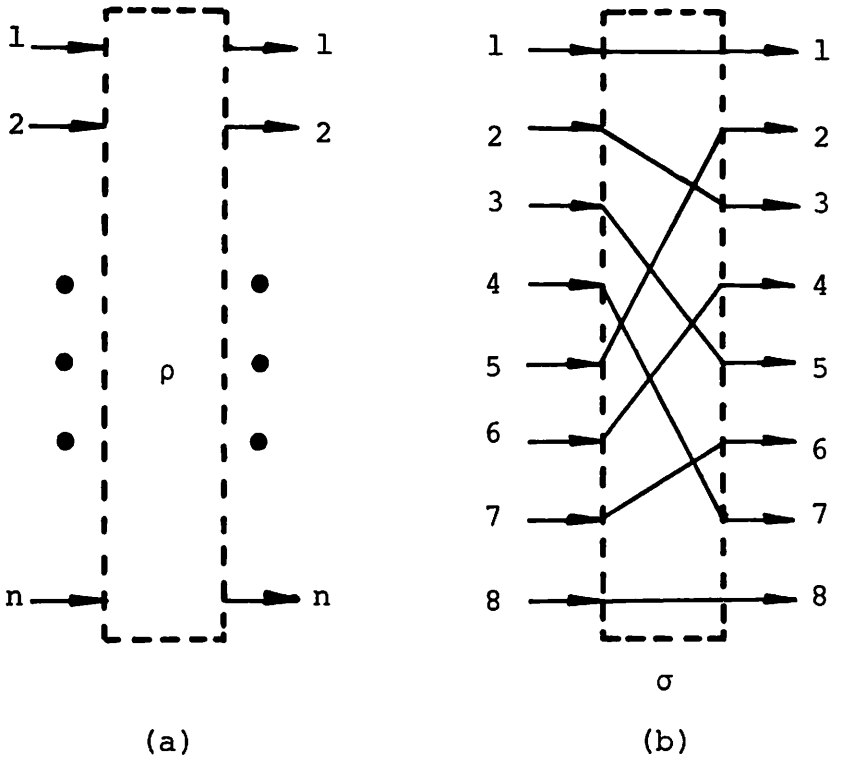


Figure 2.14. (a) A general $n \times n$ permuter ρ;
(b) the 8×8 perfect-shuffle permuter σ.

set of β-elements and permuters. Consequently the structure of a β-network can be represented as a cascade of stacks of β-elements and permuters.

2.2.2 Single and Multiple Stage β-networks

An $n \times n$ β-*stack* is an $n \times n$ β-network consisting of a stack of $n/2$ β-elements. Many $n \times n$ β-networks contain multiple stages of β-stacks. The $n \times n$ β-stack is denoted $\mathcal{S} = B_1 + B_2 + \cdots + B_{n/2} = (n/2)B$, where B and B_i for $i = 1,2,\ldots,$ $n/2$ are β-elements. An $n \times n$ *t-stage* β-*network* is a β-network containing t stages of β-elements where each stage constitutes an $n \times n$ β-stack. A 1-stage β-network is commonly called a single-stage β-network. Two typical implementations of single-stage β-networks, consisting of an $n \times n$ β-stack $\mathcal{S}$ cascaded with an $n \times n$ permuter ρ, are shown in Figure 2.15. A well-known single-stage β-network, the $n \times n$ *shuffle-exchange network* or *SE-network*, $\mathcal{P}$ [Stone 1971], is illustrated in Figure 2.16a. The $n \times n$ permuter σ used here, which resembles the perfect shuffling of a deck of cards, is called the *perfect shuffle*. If the terminals are numbered from 0 to n-1, then the perfect shuffle permutation $\sigma = \begin{pmatrix} 0 & 1 & & n-1 \\ \sigma(0) & \sigma(1) & \cdots & \sigma(n-1) \end{pmatrix}$ can be defined as follows

$$\sigma(i) = \left(2i + \left\lfloor 2i/n \right\rfloor\right) \bmod n, \quad \text{for } i = 0,1,\ldots,n-1 .$$

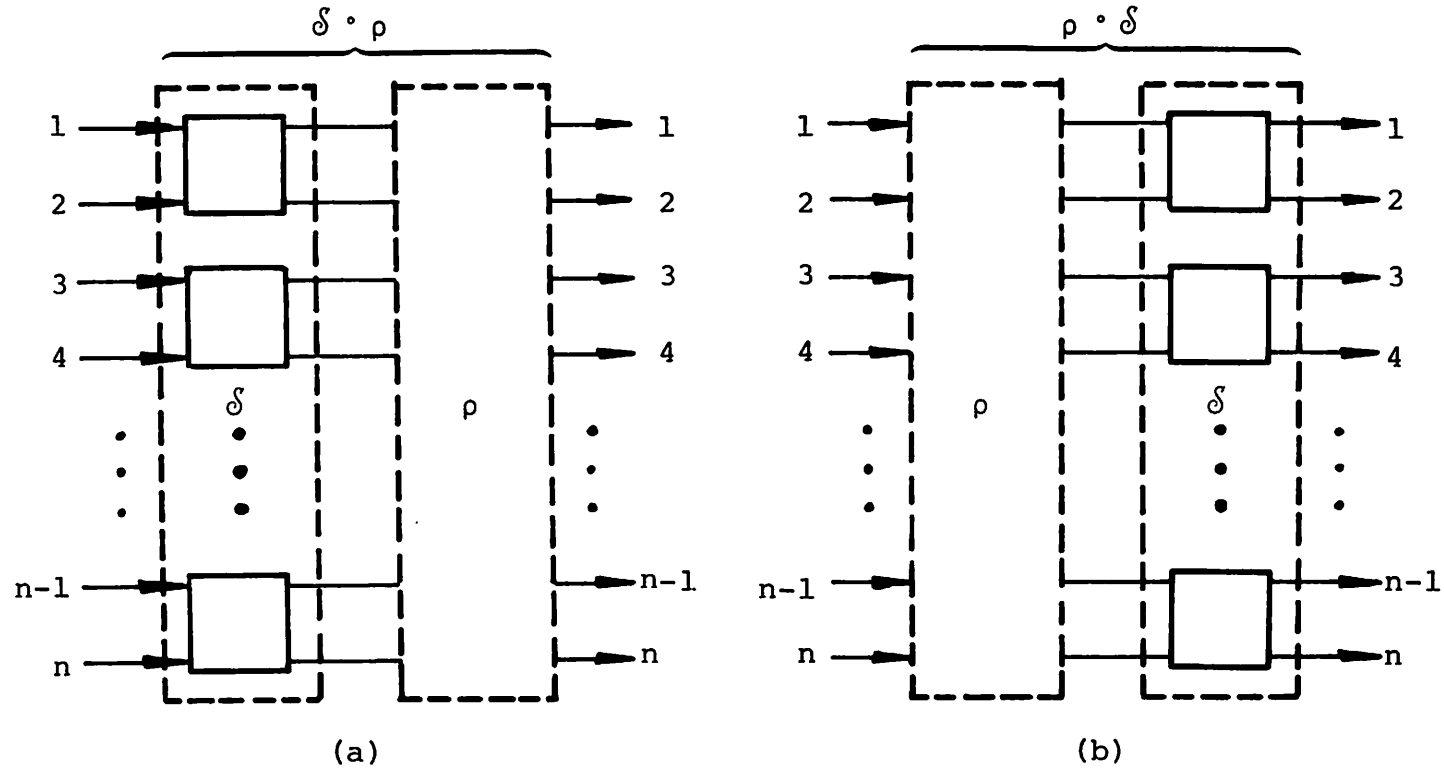


Figure 2.15. Two possible representations of a single-stage β-network.

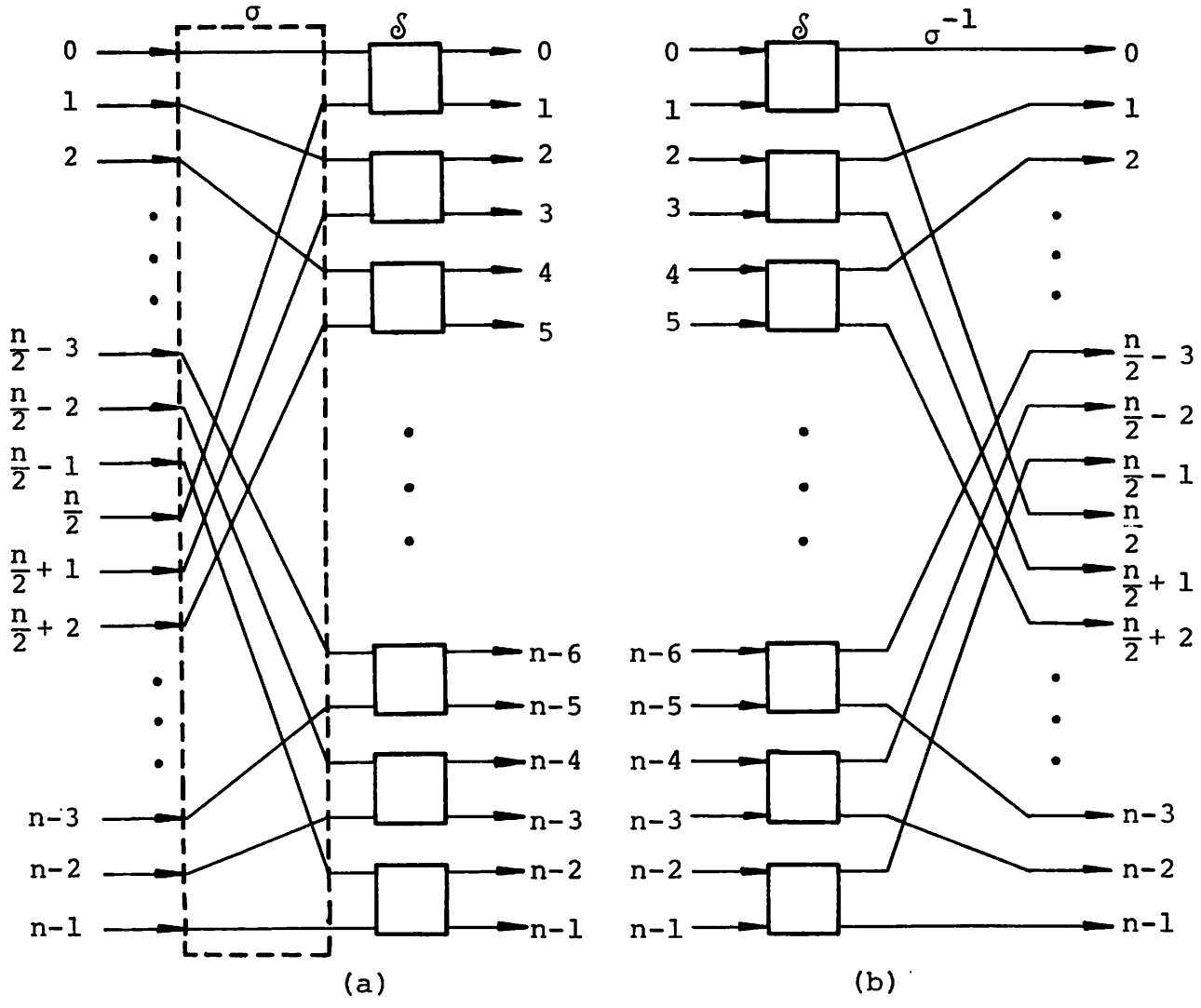


Figure 2.16. (a) The shuffle-exchange network, $\mathcal{P} = \sigma \circ \mathcal{S}$; (b) The inverse shuffle-exchange network, $\mathcal{P}^{-1} = \mathcal{S} \circ \sigma^{-1}$.

A t-stage β-network can be viewed as a cascade of t single-stage β-networks. For example, the indirect binary 3-cube of Figure 2.7 is a 3-stage β-network consisting of a cascade of three single-stage β-networks.

The *inverse* of an $n \times n$ β-network N, denoted N^{-1}, is another $n \times n$ β-network that is the same as N except that the direction of all the links is reversed. The input terminals become the output terminals, and vice versa. Clearly the inverse of an $n \times n$ permuter is represented by the corresponding inverse permutation. The $n \times n$ *inverse shuffle exchange network*, or *ISE-network*, is the inverse of the $n \times n$ SE-network. Figure 2.16b depicts the $n \times n$ ISE-network $\mathcal{P}^{-1}$. Although $N \circ N = N^2$, $N \circ N^{-1}$ is not always defined, unless N is a permuter. In fact if both N_1 and N_2 are permuters, then the cascade operator $\circ$ used in $N_1 \circ N_2$ becomes identical to the usual composition operator in permutation groups. It can be shown that $(N^t)^{-1} = (N^{-1})^t = N^{-t}$.

2.3 GRAPH MODEL FOR β-NETWORKS

The principal aim of this thesis is to study the fault-tolerance properties of β-networks used as ICNs in interconnected multicomputer systems. As illustrated in Figure 1.3b, the set of n computing units in an interconnected multicomputer system can be equated with both the set of input terminals and the set of output terminals

of the β-network. Because of the existence of the n feedback paths through the n computing units, the n input links and the n output links of the β-network are considered to be identical and have been defined as the n terminal links of the network, as shown in Figure 2.17. In our analysis of the nature of faults and fault tolerance in β-networks, we make extensive use of the following graph model of a β-network.

Definition 2.4: The *labeled β-graph* of a β-network is a labeled directed graph with vertices representing the β-elements, and edges representing the links of the β-network. There is an edge denoted (i,j) from vertex i to vertex j of the β-graph if an output link of β-element i is also an input link of β-element j. An edge in the β-graph is called a *terminal edge* if it corresponds to a terminal link of the β-network, otherwise it is called an *intermediate edge*. The terminal edges are labeled, e.g., with the indices of the corresponding terminal links. Intermediate edges are not labeled. An *unlabeled β-graph*, or simply a β-graph, is a labeled β-graph with all its labels deleted.

Figure 2.18 shows the labeled β-graph of the indirect binary 2-cube network which connects four computing units {1,2,3,4}. Each computing unit is implicitly represented by a terminal edge in the β-graph. The terminal edges are

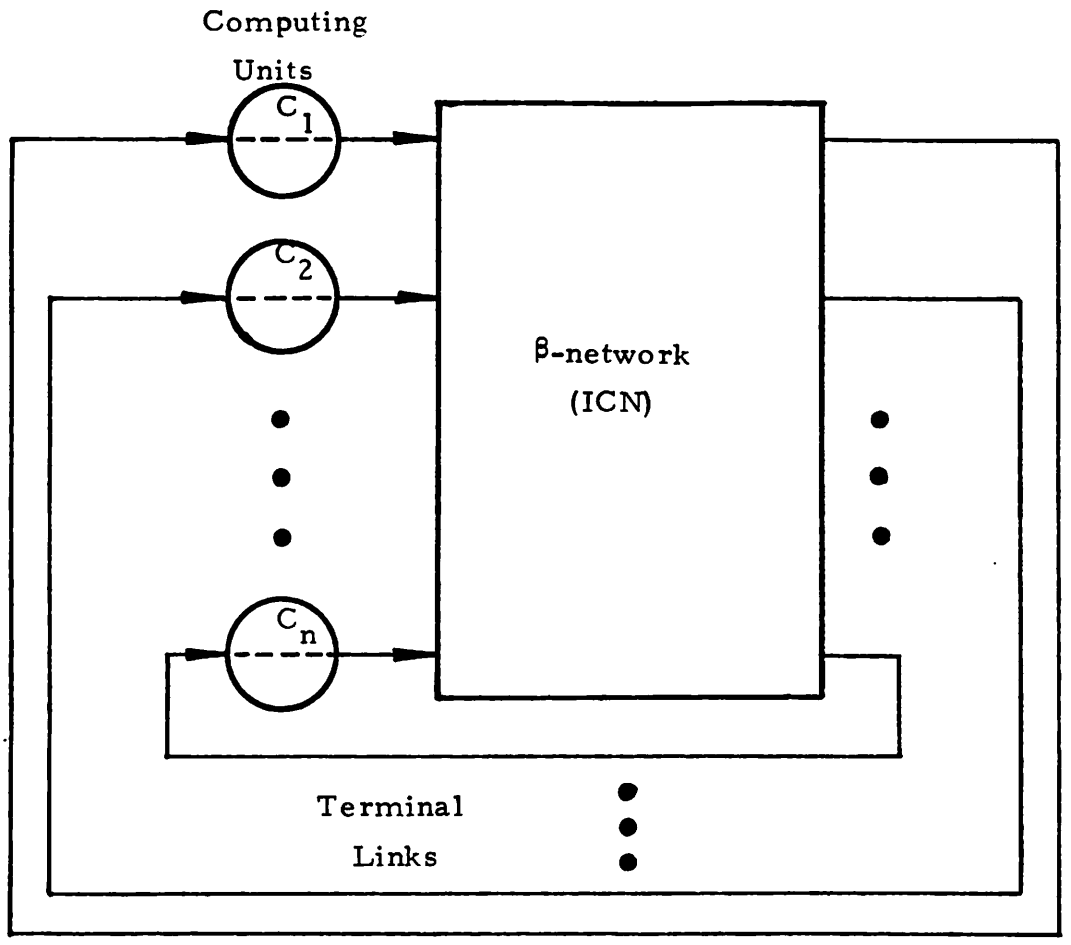
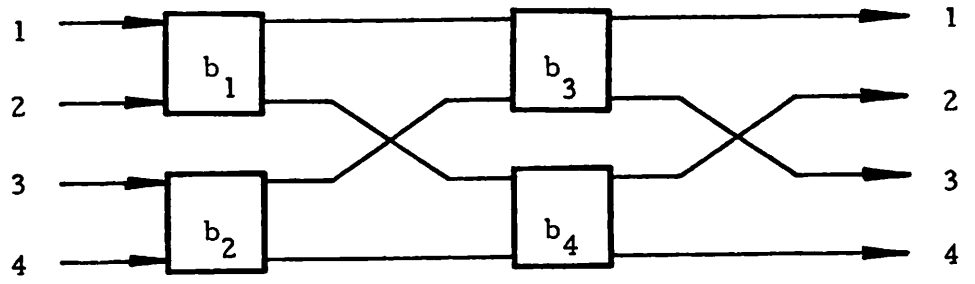
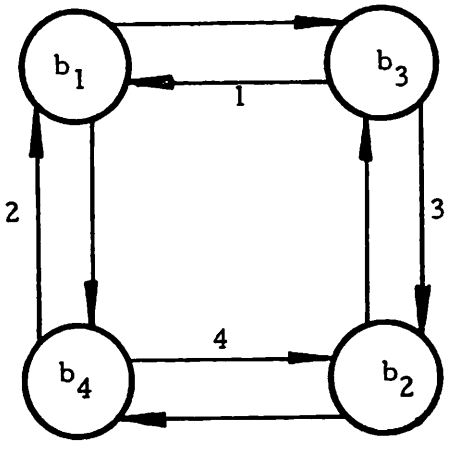


Figure 2.17. Computing units as feedback paths for the β-network of a multicomputer system.



(a)



(b)

Figure 2.18. (a) The indirect binary 2-cube network; (b) Its labeled β-graph.

labeled with the indices of the associated computing units or terminal links as shown in Figure 2.18b. The β-graph of an n × n β-network with z β-elements contains z vertices, 2z edges, and n terminal edges. Every vertex has two incoming and two outgoing edges. In the case of a single-stage β-network, all the edges are terminal edges. The β-graph of an n × n single-stage β-network contains n/2 vertices and n edges, all of which are terminal edges. The 8 × 8 ISE-network in Figure 2.19 is an example. A directed graph is an *Eulerian graph* [Harary 1969] if every vertex has the same number of outgoing edges as incoming edges. Clearly every β-graph is Eulerian. In fact, every β-graph is a regular Eulerian graph of degree four.

2.3.1 β-graph Equivalence

Two β-networks are *isomorphic* if they have the same labeled β-graphs. The actual symbols used for the labelling are insignificant. There exist one-to-one correspondences between the β-elements, links and terminal links of two isomorphic β-networks. Isomorphic β-networks also have the same connecting capability and network structure. However, the diagrams representing two isomorphic β-networks may not look identical. They can be made to look identical by rearranging the positions of the β-elements without breaking and reconnecting any link. For example,

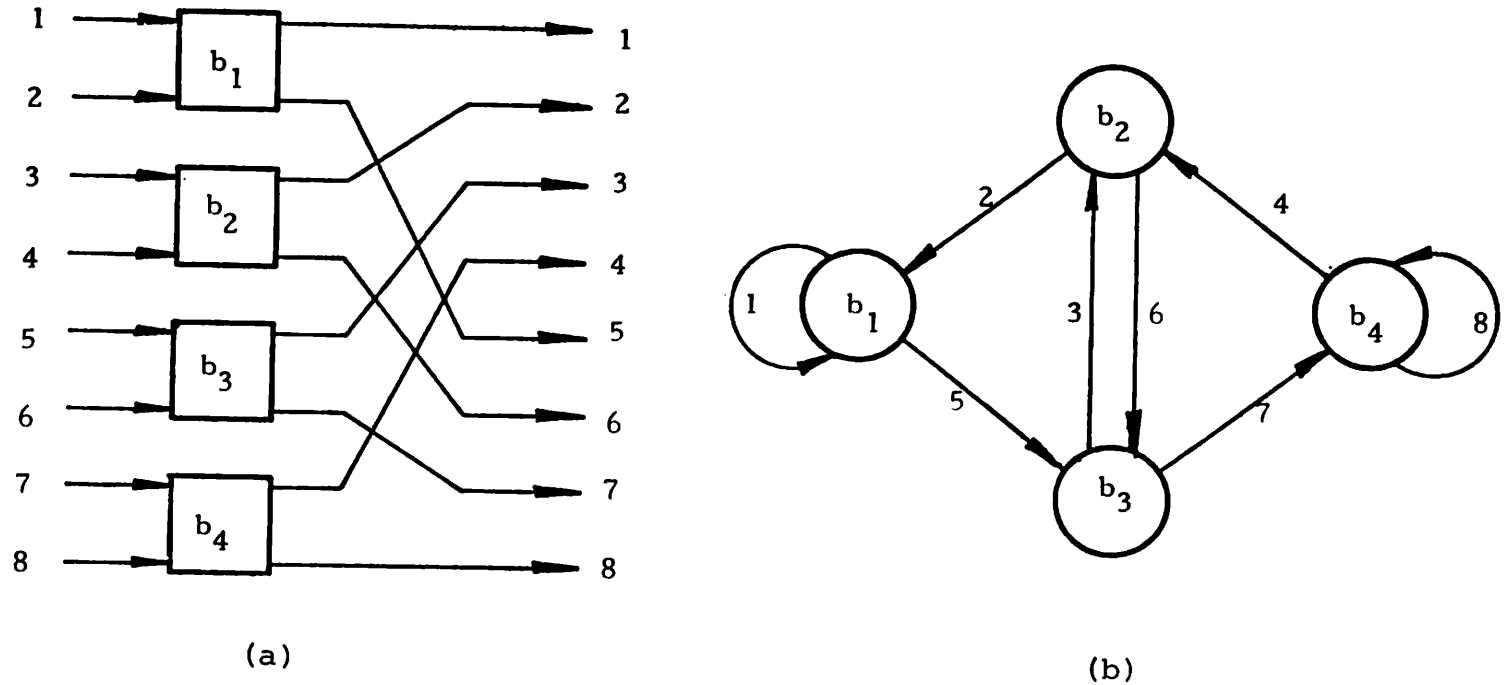


Figure 2.19. (a) The 8 × 8 inverse shuffle-exchange (ISE) network; (b) Its β-graph.

the 8×8 omega network [Lawrie 1975] in Figure 2.20 is isomorphic to the indirect binary 3-cube network of Figure 2.7. By alternating the positions of the second and third β-elements in the middle stage of Figure 2.7, the layout of Figure 2.20 is obtained.

Every β-network has a unique (unlabeled) β-graph, but a β-graph can represent more than one β-network. For example, the single-stage β-network of Figure 2.21a has the same unlabeled β-graph as the indirect binary 2-cube in Figure 2.18. The difference between the two β-graphs lies in the labeling of their terminal edges. All eight edges in Figure 2.21b are terminal edges, whereas only four of the edges are terminal edges in Figure 2.18b. Hence an unlabeled β-graph represents a class of β-networks all having the same unlabeled β-graph. We define two β-networks to be *BG-equivalent* (β-graph equivalent) if they have the same unlabeled β-graph.

All the β-networks belonging to the same BG-equivalence class can be viewed as possible realizations of the same unlabeled β-graph. Each β-network corresponds to a specific labeling of the edges of the β-graph. In a β-graph of z vertices and 2z edges, there are 2^{2z} distinct ways of labeling its edges. Hence the β-graph represents a BG-equivalence class of 2^{2z} distinct β-networks, not all of which may have practical significance. All the edges in the β-graph of a single-stage β-network are terminal

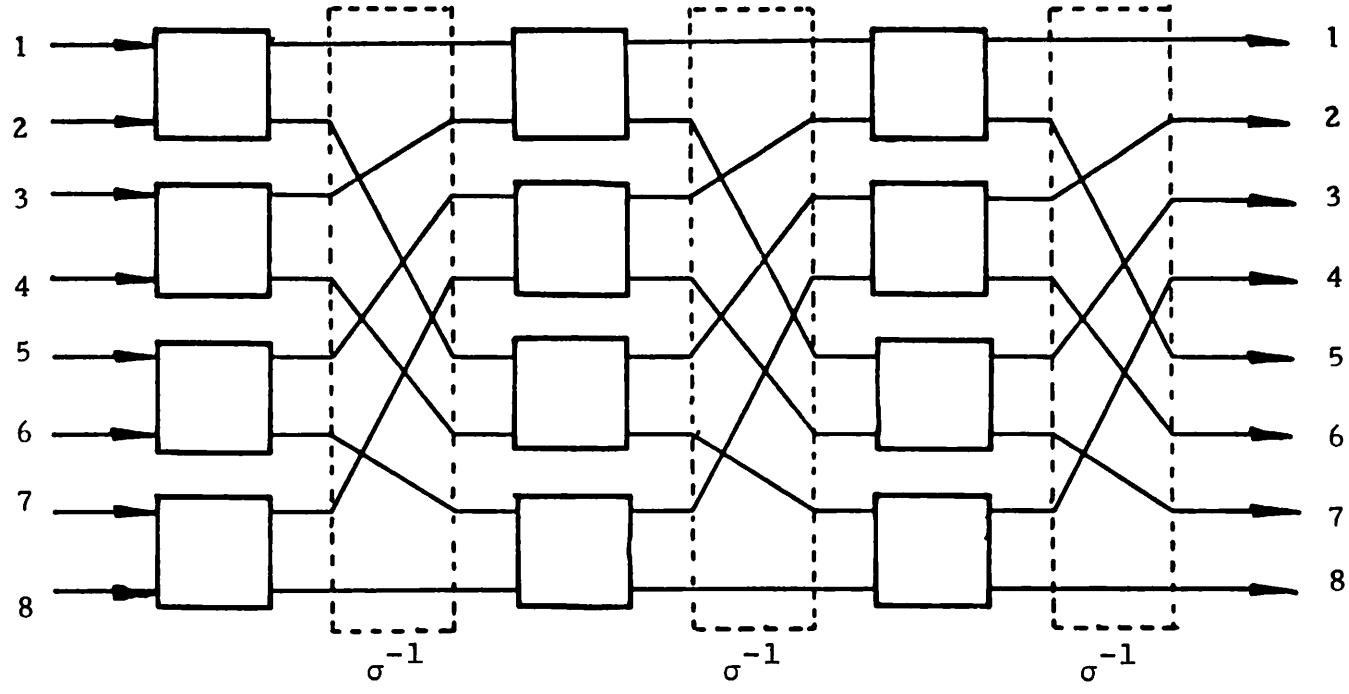


Figure 2.20. The 8 × 8 omega network.

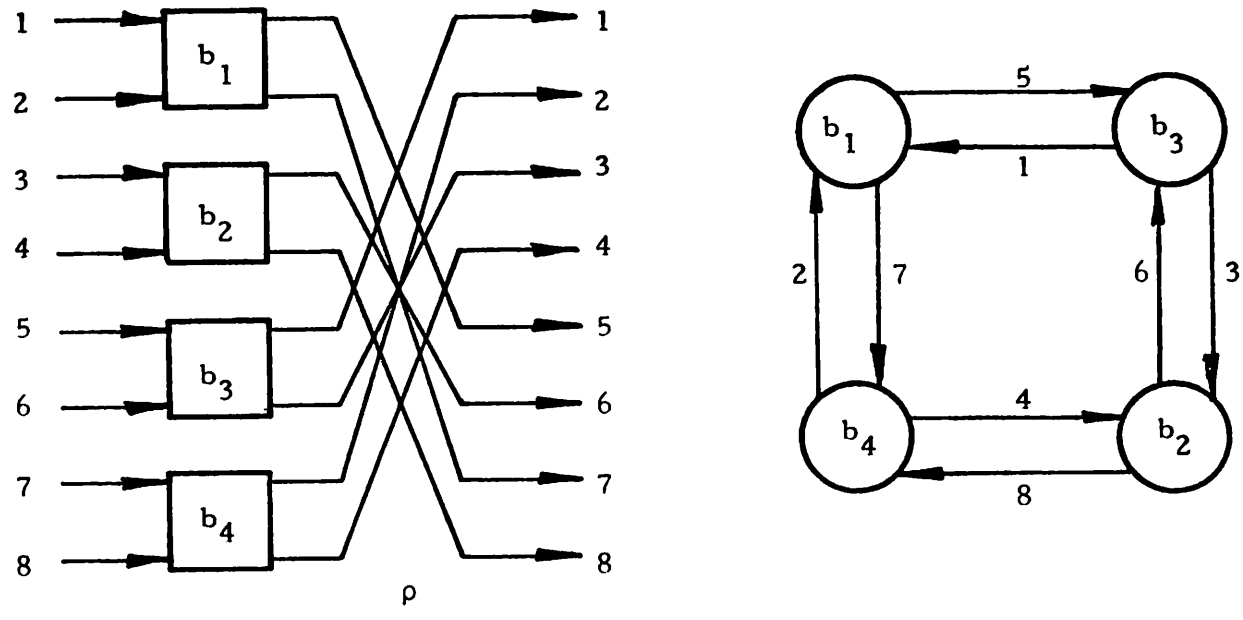


Figure 2.21. (a) A single-stage β-network with permuter $\rho = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 5 & 7 & 6 & 8 & 1 & 3 & 2 & 4 \end{pmatrix}$;
(b) Its β-graph.

edges. Hence each β-graph represents a unique single-stage β-network.

We are interested in the fault tolerance characteristics of β-networks. These characteristics depend strictly on the structure of the β-networks, and not on the computing units. It appears that β-graphs capture all the useful structural properties of a β-network, including connecting and switching properties needed for fault-tolerance analysis. Thus β-networks in the same BG-equivalence class have the same fault-tolerance properties.

2.3.2 Residual Networks

Let N be a β-network with z β-elements $B = \{b_1, b_2, \ldots, b_z\}$. A (partial) state of N, denoted $s(B) = (s_1, s_2, \ldots, s_z)$, is an assignment of each of the z β-elements to the T, X, or d state, where $s_i \in \{T, X, d\}$ denotes the state of the β-element b_i for $i = 1, 2, \ldots, z$. The β-elements which have been assigned the T or X states are called the *specified* β-elements. The *residual network* of a β-network N with respect to a (partial) state s, denoted N/s, is the β-network obtained from N by replacing all the specified β-elements of s by fixed links according to the specified states. The number of β-elements in N/s is equal to the number of unspecified β-elements in s. The residual network of N with respect to a completely specified state is

simply a collection of links and is not of much interest. Figure 2.22a depicts a β-network N with seven β-elements $B = \{b_1, b_2, \ldots, b_7\}$ connecting eight computing units. The residual β-network of N with respect to the partial state $s(B) = (d,d,d,d,X,X,T)$, denoted $M = N/s$, is illustrated in Figure 2.22b. We can extend this concept to β-graphs. If G is the β-graph of a β-network N, then the *residual β-graph* of G with respect to a state s, denoted G/s, is the β-graph of the residual β-network N/s. Figure 2.22c and Figure 2.22d are the β-graphs of the β-network of Figure 2.22a and its residual network $M = N/s$ of Figure 2.22b, respectively. It can easily be seen that residual networks of a β-network are also legitimate β-networks.

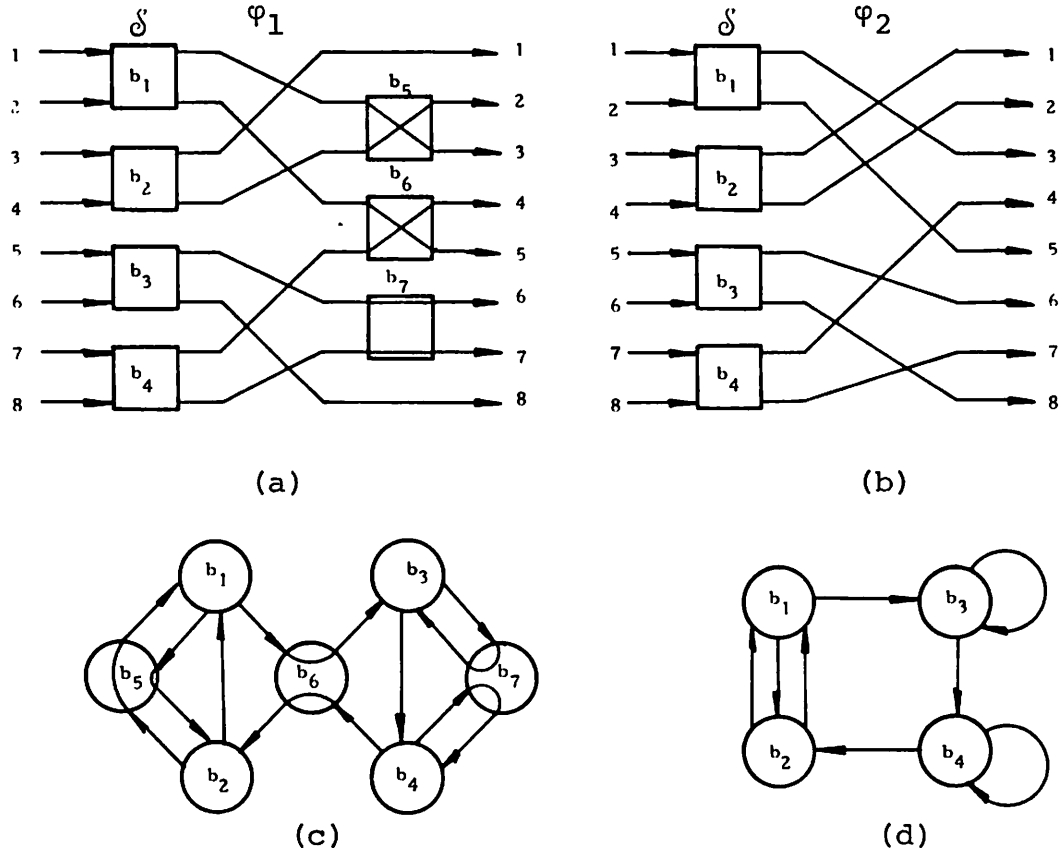


Figure 2.22. Illustration of a residual network: (a) A β-network N; (b) A single-stage β-network, M = N/s, s = (d,d,d,d,X,X,T); (c) β-graph of Figure 2.22a; (d) β-graph of Figure 2.22b.

CHAPTER 3

CRITICAL FAULTS IN β-NETWORKS

Fault tolerance has been defined as "the ability of a system to execute specified algorithms correctly regardless of hardware failures" [Avizienis 1967]. In a β-network, the algorithms in question perform the task of establishing connecting paths among the computing units. The hardware failures of concern here are β-element failures. This chapter begins the analysis of the fault tolerance of β-networks. A fault model for β-element failures is presented, and a new connectivity property called dynamic full access (DFA) is introduced which serves as the criterion for fault tolerance. A fault is called critical if it destroys the DFA property of a β-network. Graph-theoretical characterizations of the minimal critical faults and noncritical faults of a β-network are presented.

3.1 FAULT TOLERANCE OF β-NETWORKS

The primary function of a β-network is to provide connections. The connecting capability of a β-network depends on the interconnection structure of the links and β-elements. Our interest here lies in the effects permanent

hardware failures have on this interconnection structure and, in turn, on the connecting capability of the β-network.

3.1.1 Fault Model

We assume that β-network faults are mainly caused by β-element failures, since the β-elements are composed of active switching devices like transistors. The passive interconnecting links of a β-network are less likely to fail. Many link failures manifest themselves as line stuck-at 0/1 faults. These faults are equivalent to output failures in β-elements, and can therefore be modeled by β-element faults. Some link failures, such as open-circuited or short-circuited links, may not correspond to β-element faults, and are not treated in this study.

As illustrated in Figure 2.3, each β-element can be in one of two states: the "through" (T) state or the "cross" (X) state. A β-element is considered faulty when it is permanently stuck in one of its states; thus it can fail in two ways: stuck at through (s-a-T) or stuck at cross (s-a-X). Levitt et al. have shown that if a logic design of the type shown in Figure 3.1 is used for β-elements, the stuck-at-T/X fault model is quite valid [Levitt et al. 1968]. Any single component failure results in either the β-element being stuck in a state, or else causes a faulty signal to be produced at one of the β-element's output

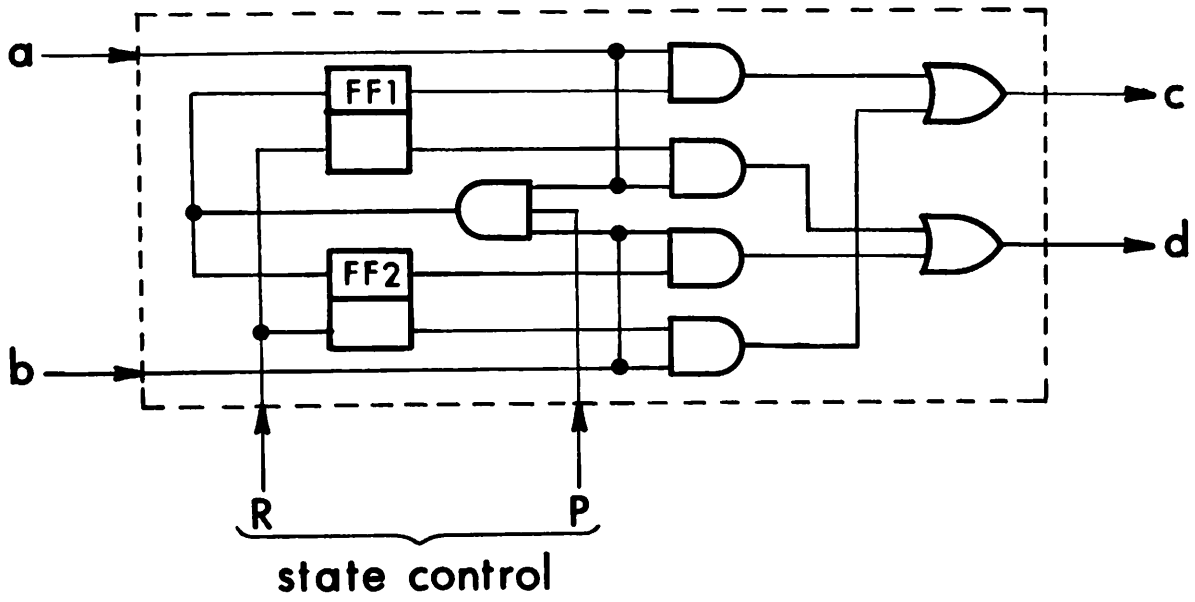


Figure 3.1. Logical design of a β-element [Levitt et al. 1968].

lines. Faulty output failures do not affect the interconnection structure of the β-network and can be readily detected. Furthermore, methods for correcting faulty output failures have been developed [Levitt et al. 1968]. Opferman and Tsao-Wu use the same β-element fault model in their study of the fault diagnosis of rearrangeable networks [Opferman and Tsao-Wu 1971].

Each β-element in a β-network is modeled by a vertex with two incoming and two outgoing edges in the corresponding β-graph. Under fault-free conditions each of the two incoming edges can be connected to each of the two outgoing edges. When the β-element is stuck at one of its two states, each incoming edge can only be connected to one of the outgoing edges. A β-element stuck-at fault can be modeled by the splitting of a vertex in the β-graph, as illustrated in Figure 3.2. A *vertex split* is therefore defined as the decomposition of a vertex in a β-graph into two subvertices, each with one incoming and one outgoing edge.

3.1.2 Fault-Tolerance Criterion

In an interconnected system the set of computing units represents both the set of input terminals and the set of output terminals of the β-network, as shown in Figure 2.17. The computing units provide a set of feedback paths from the outputs back to the inputs of the β-network.

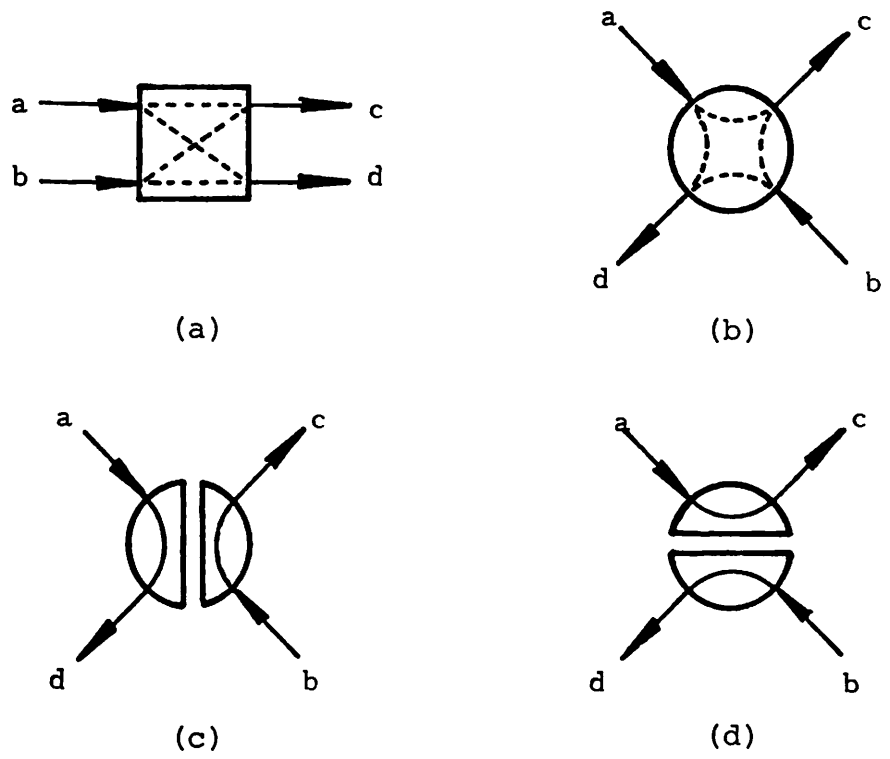


Figure 3.2. β-element stuck-at faults and vertex splits: (a) a fault-free β-element; (b) the corresponding β-graph vertex; (c) the s-a-X split; (d) the s-a-T split.

Hence data can be transmitted from computing unit to computing unit via one or more passes through the β-network. For example data transmitted from unit i to unit j can be retransmitted by unit j to another unit k in a second pass through the β-network. In essence, unit i can communicate with unit k via two passes through the network. With multiple passes, the connectivity among the set of computing units is enhanced, but the total communication delay is increased. If additional delay can be tolerated, a simpler and thus cheaper β-network can be used to achieve the same connecting capability. Furthermore, multiple passes can be used to improve fault tolerance. Data transmitted from unit i to unit k can sometimes be rerouted via additional passes to circumvent a faulty β-element in the original route. Hence a β-element failure may only degrade the communication speed instead of incapacitating an ICN.

In light of multiple-pass data transmission, a new definition of full-access can be constructed.

Definition 3.1: A β-network of an interconnected system has the *dynamic full-access* (DFA) property if each of its inputs can be connected to any one of its outputs by a finite number of passes through the β-network.

For example, a β-network consisting of stages 1 and 2 of the network of Figure 2.6 has the DFA property but not the

full-access property; whereas a β-network consisting of just stage 1 does not have the DFA property.

The purpose of the ICN in an interconnected system is to provide interconnections among the computing units. Thus the most basic requirement for any ICN is the ability to connect any pair of computing units. The fulfillment of this basic requirement is equivalent to the maintenance of the dynamic full-access property. Hence, DFA can be regarded as a fundamental fault-tolerance criterion for ICNs of interconnected systems.

In this study, the fault tolerance of a β-network is measured by its ability to maintain DFA in spite of the presence of stuck-at-T/X faults in its β-elements. A β-network is more fault tolerant if it is able to tolerate a greater number of β-element stuck-at faults. We define a β-network with DFA as *k-fault tolerant* or *k-FT* if the failure, either s-a-T or s-a-X, of any k or fewer β-elements does not destroy DFA. The largest k for which a β-network is k-FT is called the *fault tolerance (FT) parameter* of the β-network.

It is easily seen that a β-network has the DFA property if and only if the corresponding β-graph is strongly connected. Every useful β-network must have the DFA property and thus its β-graph must be strongly connected. The occurrence of β-element stuck-at faults is equivalent to the splitting of the corresponding vertices in the β-graph.

When a set of vertex splits disconnects a β-graph, DFA is lost in that faulty β-network. Fault tolerance in terms of the β-graph, can thus be defined as the ability of the β-graph to stay strongly connected in spite of the splitting of its vertices. Intuitively speaking, fault tolerance is a measure of how tightly the edges are tied together by the vertices of the β-graph. Table 3.1 summarizes some important properties of a β-network in terms of its β-graph.

3.2 CRITICAL-FAULT CHARACTERIZATION

The fault-tolerance parameter k is a basic measure of the fault tolerance of a β-network. However, simple characterizations of k in terms of the structural properties of the β-graph do not appear to exist. In order to determine k we need to know which combinations of stuck-at faults destroy the DFA property of the β-network, and which do not. In this section we characterize those combinations of β-element stuck-at faults which destroy the DFA property of a β-network.

3.2.1 Critical Faults

We first need to formalize the concept of a fault of a β-network.

Definition 3.2: A *fault* in a β-network is a collection of β-elements that are stuck at X or T. A fault is *critical*

Table 3.1. Correspondence between β-networks and β-graphs.

	β-Network	β-Graph
FT Criterion:	DFA	Strongly connected
Fault Model:	s-a-T, s-a-X	Vertex splits
Fault Tolerance:	Maintenance of DFA in the presence of s-a-T/X faults	β-graph remains strongly connected with vertex splits
Critical Fault:	Set of β-element stuck-at faults which destroys DFA	Set of vertex splits which disconnects the β-graph

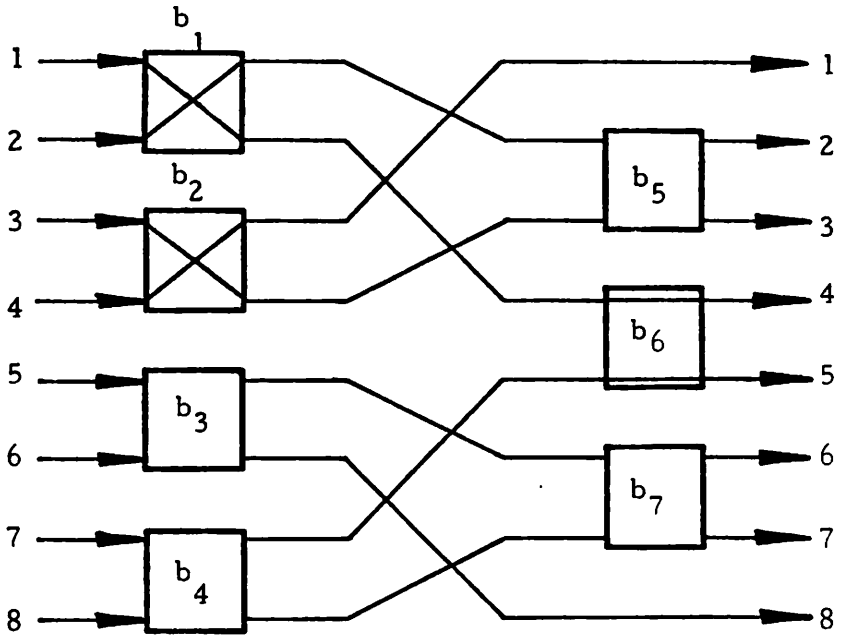
if it destroys the DFA property of the β-network. A *minimal critical fault* (MCF) is a critical fault none of whose proper subsets constitutes a critical fault. The *critical fault set* of a β-network is the set of all its MCFs.

A fault in a β-network can be represented by a (partial) state of the β-network. The specified entries represent faulty β-elements. The unspecified entries represent fault-free β-elements. A β-element b_i is assigned the T or X state if it is stuck at T or X, respectively. We define a *fault state* of a β-network as a state which represents a fault of the β-network. For example, the fault state $f(b_1,b_2,b_3,b_4) = (X,d,d,T)$ represents a fault consisting of β-elements b_1 s-a-X and b_4 s-a-T. A fault state differs from a normal state of a β-network in that the states assigned to the specified β-elements are permanent and represent faulty conditions. The fault state of a critical fault is called a *critical fault state*. A minimal critical fault state or *MCF state* is similarly defined.

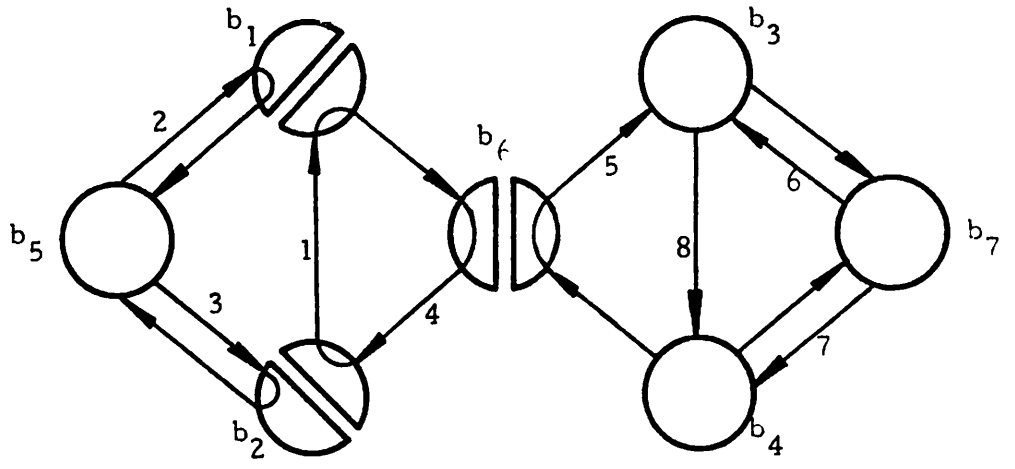
Every fault in a β-network is equivalent to a collection of vertex splits in the corresponding β-graph. We call the β-graph resulting from the vertex splits the *faulty β-graph*. A fault is critical if and only if the corresponding set of vertex splits disconnects the β-graph, i.e., the faulty β-graph is disconnected. Every set of vertex splits corresponding to a fault in the β-network

produces a faulty β-graph which is Eulerian like the original β-graph. Hence a minimal critical fault is a set of vertex splits which disconnects the fault-free β-graph to form a faulty β-graph consisting of two disconnected strongly-connected components. The set of computing units represented by the terminal edges in each of the two components cannot communicate with the set of computing units in the other component.

Figure 3.3 illustrates the effect of a fault f in the β-network of Figure 2.22a. Figure 3.3a is the faulty β-network, while Figure 3.3b depicts the corresponding faulty β-graph. The fault f consists of β-elements b_1 and b_2 s-a-X and b_6 s-a-T, and is denoted by $f(b_1,b_2,b_3,b_4,b_5,b_6,b_7) = (X,X,d,d,d,T,d)$. f is critical because the β-graph in Figure 3.3b consists of three disconnected components. Computing units 2 and 3 are isolated by f, as are computing units 1 and 4. However f is not an MCF. In fact f is the union of two MCFs f_1 and f_2, where $f_1 = (d,d,d,d,d,T,d)$ and $f_2 = (X,X,d,d,d,d,d)$. Clearly each of the faults f_1 and f_2 disconnects the β-graph. By exhaustively searching its β-graph we can obtain all the MCFs of a β-network. There are nine MCFs in the critical fault set of the β-network of Figure 3.3a. They are represented by the following nine MCF states:



(a)



(b)

Figure 3.3. Example of the effect of a fault in a β-network: (a) the β-network of Figure 2.22a with b_1 and b_2 s-a-X, and b_6 s-a-T; (b) faulty β-graph of this β-network.

f_1 = (d,d,d,d,d,T,d)

f_2 = (X,X,d,d,d,d,d)

f_3 = (d,d,X,X,d,d,d)

f_4 = (X,d,d,d,T,d,d)

f_5 = (d,X,d,d,T,d,d)

f_6 = (d,d,X,d,d,d,T)

f_7 = (d,d,d,X,d,d,T)

f_8 = (T,T,d,d,X,d,d)

f_9 = (d,d,T,T,d,d,X)

The critical fault set can be used to characterize the vulnerability of a β-network to β-element stuck-at faults. An MCF represents a smallest set of β-element stuck-at faults which, if present, destroys DFA regardless of the states of other β-elements. Clearly any fault that contains an MCF must be a critical fault. A more fault-tolerant β-network tends to have fewer and larger MCFs in its critical fault set. The smaller MCFs represent the more vulnerable areas of a β-network. For example, the critical fault set of the β-network in Figure 3.3a consists of one single fault, six double faults, and two triple faults as shown in the list above. There is only one single fault, namely f_1, involving β-element b_6 s-a-T. Hence we can conclude that b_6 is the most vulnerable β-element in the β-network. In the ensuing subsections we

characterize the critical fault set of a β-network using the properties of its β-graph.

3.2.2 Circuit Partitions

To characterize the MCFs of a β-network we introduce a graph called the circuit adjacency (CA) graph which is obtained by considering the elementary circuits of the β-graph. An *elementary circuit* of a directed graph is a directed circuit that has no repeated vertex [Harary 1969].

Definition 3.3: A *circuit partition* π of a β-graph is a collection of edge-disjoint elementary circuits of the graph such that each vertex belongs to exactly two elementary circuits, and each edge belongs to exactly one elementary circuit.

Each elementary circuit can be denoted by the set of vertices belonging to the circuit. For example, if an elementary circuit contains vertices b_1, b_2 and b_3, then the elementary circuit is written as $b_1b_2b_3$. A circuit partition can be represented by the set of edge-disjoint elementary circuits that define it. Figure 3.4 illustrates the two circuit partitions of the β-graph of Figure 2.22a. These two circuit partitions are denoted by $\pi_1 = \{b_1b_5, b_5b_2, b_1b_6b_2, b_3b_4b_6, b_3b_7, b_7, b_4\}$, and $\pi_2 = \{b_1b_6b_2b_5, b_1b_5b_2, b_3b_7b_4b_6, b_3b_4b_7\}$.

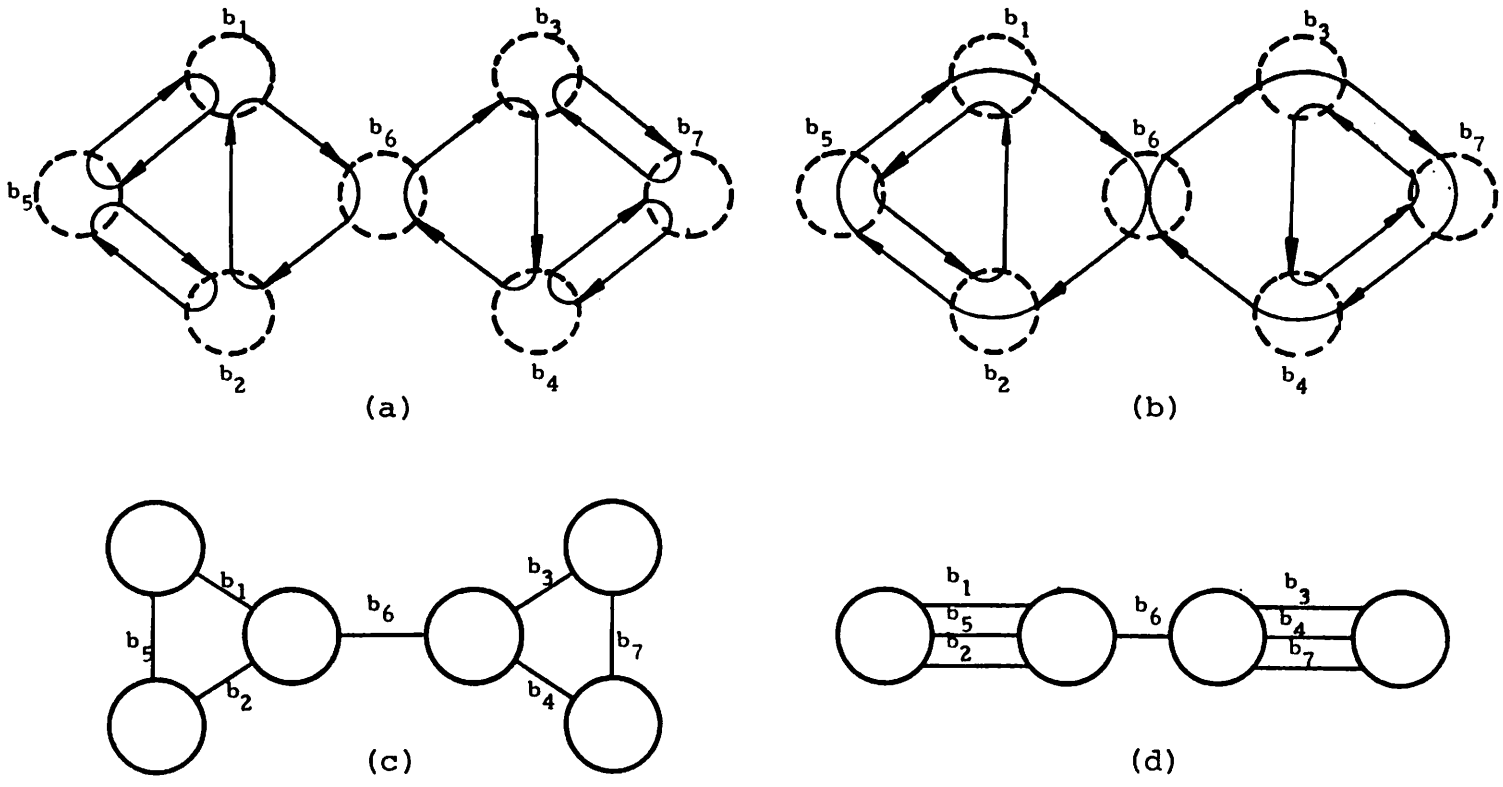


Figure 3.4. Examples of circuit partitions and CA-graphs: (a) circuit partition π_1 of the β-graph of Figure 2.22c; (b) circuit partition π_2 of the β-graph of Figure 2.22c; (c) CA-graph of π_1; (d) CA-graph of π_2.

Using Berge's concept of a factor [Berge 1973], it can be shown that every β-graph possesses at least one circuit partition. A *factor* of a directed graph is a collection of edge-disjoint elementary circuits such that every vertex belongs to exactly one of the elementary circuits. Berge has shown that in every regular Eulerian graph of degree 2n, the edges can be decomposed into n mutually exclusive factors. Since every β-graph is a regular Eulerian graph of degree four, the edges of the β-graph must be able to be partitioned into two factors. The edge-disjoint elementary circuits of the two factors constitute a circuit partition. For example, in Figure 3.4b the set of elementary circuits $F_1 = \{b_1b_6b_2b_5, b_3b_4b_7\}$ constitutes a factor, and the set $F_2 = \{b_1b_5b_2, b_3b_7b_4b_6\}$ constitutes the second factor.

Definition 3.4: For every circuit partition π of a β-graph G, a *circuit adjacency graph* or *CA-graph* of π is an undirected graph whose vertices represent the elementary circuits of π and whose edges represent the vertices of G. An edge exists between vertices a and b of a CA-graph if the corresponding vertex in the β-graph belongs to both elementary circuits a and b of the β-graph.

The CA-graphs of π_1 and π_2 of Figures 3.4a and 3.4b are illustrated in Figures 3.4c and 3.4d, respectively. The

number of edges in a CA-graph is the number of β-elements in the corresponding β-network. The number of vertices in a CA-graph is the number of elementary circuits in the corresponding circuit partition.

From Figures 3.4a and 3.4b we see that a circuit partition can be obtained by appropriately setting each of the β-elements to either the T or the X state. A circuit partition of a β-graph corresponds to a unique state of the β-network, and can therefore be represented by the notation used for a β-network state. We define a *circuit partition state* or *CP-state* of a β-network as a state of the β-network which represents a circuit partition of its β-graph. For example, the circuit partitions π_1 and π_2 in Figures 3.4a and 3.4b can be represented by the CP-states s_1 = (X,X,X,X,T,T,T) and s_2 = (T,T,T,T,X,T,X), respectively. Henceforth, we will use CP-states to denote circuit partitions instead of the original more cumbersome notation using sets of elementary circuits.

Every edge in a CA-graph represents a β-element. In a circuit partition every β-element is set to the T or X state. Hence every edge in a CA-graph can be labeled with T or X to denote the corresponding β-element state. We define a *labeled CA-graph* as a CA-graph with its edges labeled according to the entries of the corresponding CP-state. An edge is labeled T (X) if the corresponding β-

element is assigned the T (X) state in the circuit partition. Hence we can identify the state of a CA-graph with the corresponding CP-state.

Every CA-graph has as many edges as the corresponding β-graph has vertices. The two vertices connected by an edge in a CA-graph represent two elementary circuits of the β-graph which share, or are connected by, a particular β-element. If the β-element is set to the T (X) state in a circuit partition, then the removal of the corresponding edge in the CA-graph is analogous to the β-element being stuck at the T (X) state. This concept leads to the following characterization of MCFs in β-networks.

3.2.3 Minimal Critical Faults

A *cutset* of a graph is a minimum set of edges whose removal disconnects the graph. The *system of cutsets* of a graph is the set of all its cutsets. In this section we characterize MCFs of a β-network using the cutsets of its CA-graphs.

Lemma 3.1: Let G be the β-graph of a β-network. Every cutset of a CA-graph H of G represents an MCF of the β-network. The edges of a cutset correspond to the faulty β-elements of the MCF. The edge labels of the cutset correspond to the stuck-at states of the faulty β-elements.

Proof: Let the ordered set of edges $E = (e_1, e_2, \ldots, e_y)$ denote a cutset of H corresponding to the set of β-elements $B = (b_1, b_2, \ldots, b_y)$ of G. Let the labels of E be $s = (s_1, s_2, \ldots, s_y)$ where s_i is the label of e_i or , equivalently, the state of the β-element b_i. The removal of E disconnects H into two subgraphs H_1 and H_2 as shown in Figure 3.5. Let $V(H_1)$ and $V(H_2)$ be the vertex sets of H_1 and H_2 respectively. Since every vertex in H represents an elementary circuit in G, $V(H_1)$ and $V(H_2)$ must represent two disconnected sets of elementary circuits C_1 and C_2 of G.

The removal of an edge e_i in H corresponds to the β-element b_i being stuck at the state s_i, or to the appropriate splitting of the vertex b_i in the β-graph G. Let f be a fault of the β-network obtained when β-elements are stuck at the states specified by s, i.e., b_i is stuck at state s_i for $i = 1,2,\ldots,y$. The fault f must disconnect the two sets of elementary circuits C_1 and C_2 of G. Hence f must be a critical fault. Furthermore, f must be a minimal critical fault, otherwise there exists a subset of f that constitutes a critical fault. If that is the case, then there must exist a cutset of H which is a proper subset of E. This contradicts the assumption that E is a cutset of H. Therefore the fault f represented by the cutset E and its associated labels must be an MCF of the β-network. □

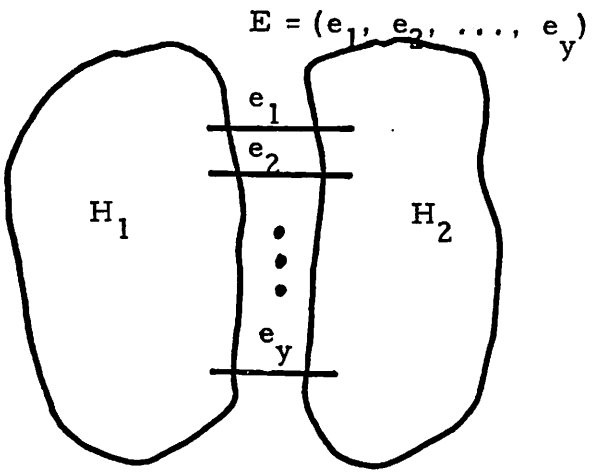


Figure 3.5. The cutset E of a CA-graph H.

Lemma 3.2: Let G be the β-graph of a β-network N. Every MCF of N can be represented by a cutset of a CA-graph H of G. The faulty β-elements and their stuck-at states correspond to the edges and the labels of the cutset, respectively.

Proof: Let $B = (b_1, b_2, \ldots, b_y)$ be a set of faulty β-elements constituting an MCF. Let the stuck-at states of B be denoted by $F = (f_1, f_2, \ldots, f_y)$ where f_i is the stuck-at state of β-element b_i for $i = 1, 2, \ldots, y$. The set of vertex splits corresponding to B with f present must disconnect G into two subgraphs G_1 and G_2 as depicted in Figure 3.6. Since the splitting of a vertex in G produces two subvertices each having one incoming and one outgoing edge, both G_1 and G_2 are Eulerian graphs like G. We know that every Eulerian graph possesses at least one circuit partition. Hence there must exist two circuit partitions π_1 and π_2 for the graphs G_1 and G_2 respectively. Since π_1 and π_2 are disjoint sets of elementary circuits of G, the union of π_1 and π_2 must be a circuit partition π of G. Let the CA graph induced by π be denoted by H. The removal of a set of edges E of H corresponding to the set of β-elements B must disconnect H into two subgraphs H_1 and H_2. The vertices of H_1 and H_2 represent the elementary circuits of π_1 and π_2, respectively. Hence E is a cutset of H which represents the MCF f of B. □

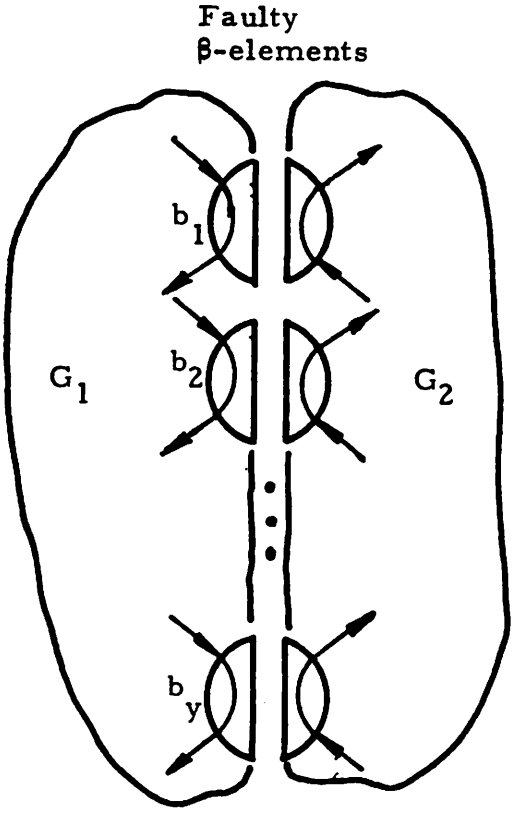


Figure 3.6. An MCF of a β-graph G.

Lemmas 3.1 and 3.2 imply the following result.

Theorem 3.1: A one-to-one correspondence exists between the MCFs of a β-network and the labeled cutsets of its CA-graphs. The faulty β-elements of an MCF correspond to the edges of a cutset. The stuck-at states of the faulty β-elements correspond to the edge labels of the cutset.

Two distinct CA-graphs of the same β-network have the same set of edges, but different edge labels. Hence two cutsets of two different CA-graphs may also contain the same set of edges but different edge labels. Consequently the two cutsets can represent two different MCFs. Hence every labeled cutset must be identified by the edges involved and their labels. Similarly every MCF is defined by the faulty β-elements involved and their stuck-at states. By Theorem 3.1, the set of all distinct labeled cutsets of the CA-graphs characterize the critical fault set of the β-network.

Example 3.1: The β-network of Figure 2.22a has two circuit partitions π_1 and π_2 as illustrated in Figures 3.4a and 3.4b. The CA-graphs H_1 and H_2 of these two circuit partitions are shown in Figure 3.4c and 3.4d, respectively. π_1 and π_2, and hence the labels of the edges of H_1 and H_2, can be represented by the two CP-states $s_1(b_1,b_2,b_3,b_4,b_5,$

$b_6,b_7) = (X,X,X,X,T,T,T)$ and $s_2(b_1,b_2,b_3,b_4,b_5,b_6,b_7) = (T,T,T,T,X,T,X)$. The two CA-graphs H_1 and H_2 possess two systems of cutsets E_1 and E_2 respectively, where $E_1 = \{b_1b_5, b_5b_2, b_1b_2, b_6, b_3b_7, b_7b_4, b_3b_4\}$ and $E_2 = \{b_1b_5b_2, b_6, b_3b_4b_7\}$. We see that H_1 has seven cutsets, while H_2 has three. Taking into account the labels of the edges of these cutsets, the MCFs corresponding to the cutsets of E_1 and E_2 can be represented by the following two sets of MCF states, F_1 and F_2, respectively.

$$F_1 = \{(X,d,d,d,T,d,d),$$
$$(d,X,d,d,T,d,d),$$
$$(X,X,d,d,d,d,d),$$
$$(d,d,d,d,d,T,d),$$
$$(d,d,X,d,d,d,T),$$
$$(d,d,d,X,d,d,T),$$
$$(d,d,X,X,d,d,d)\}$$

$$F_2 = \{(T,T,d,d,X,d,d),$$
$$(d,d,d,d,d,T,d),$$
$$(d,d,T,T,d,d,X)\}$$

F_1 contains seven MCF states and F_2 contains three MCF states. The fault state (d,d,d,d,d,T,d) is in both F_1 and F_2, hence their union contains nine distinct MCF states. These MCF states define the critical fault set of the β-network of Figure 2.22a, and are identical to the nine MCF

states obtained earlier by exhaustive search. □

Example 3.2: The β-graph of the indirect binary 2-cube of Figure 2.18 has two circuit partitions, and hence two CA-graphs H_1 and H_2, as shown in Figure 3.7. The circuit partitions can be represented by the CP-states $s_1 = (T,T,T,T)$ and $s_2 = (X,X,X,X)$. As can be seen from Figure 3.7, the cutsets of H_1 are $\{b_1b_2, b_1b_3, b_1b_4, b_2b_3, b_2b_4, b_3b_4\}$, while the sole cutset of H_2 is $\{b_1b_2b_3b_4\}$. These seven cutsets along with their edge labels characterize the seven MCFs of the critical fault set of the indirect binary 2-cube. The MCFs are listed below.

$$f_1 = (T,T,d,d)$$
$$f_2 = (T,d,T,d)$$
$$f_3 = (T,d,d,T)$$
$$f_4 = (d,T,T,d)$$
$$f_5 = (d,T,d,T)$$
$$f_6 = (d,d,T,T)$$
$$f_7 = (X,X,X,X)$$

As can be seen from this list, the critical fault set of the indirect binary 2-cube contains six double faults and one fault involving all four of its β-elements. Clearly any single β-element failure is not critical because no MCF is a single fault. Hence we can conclude that the in-

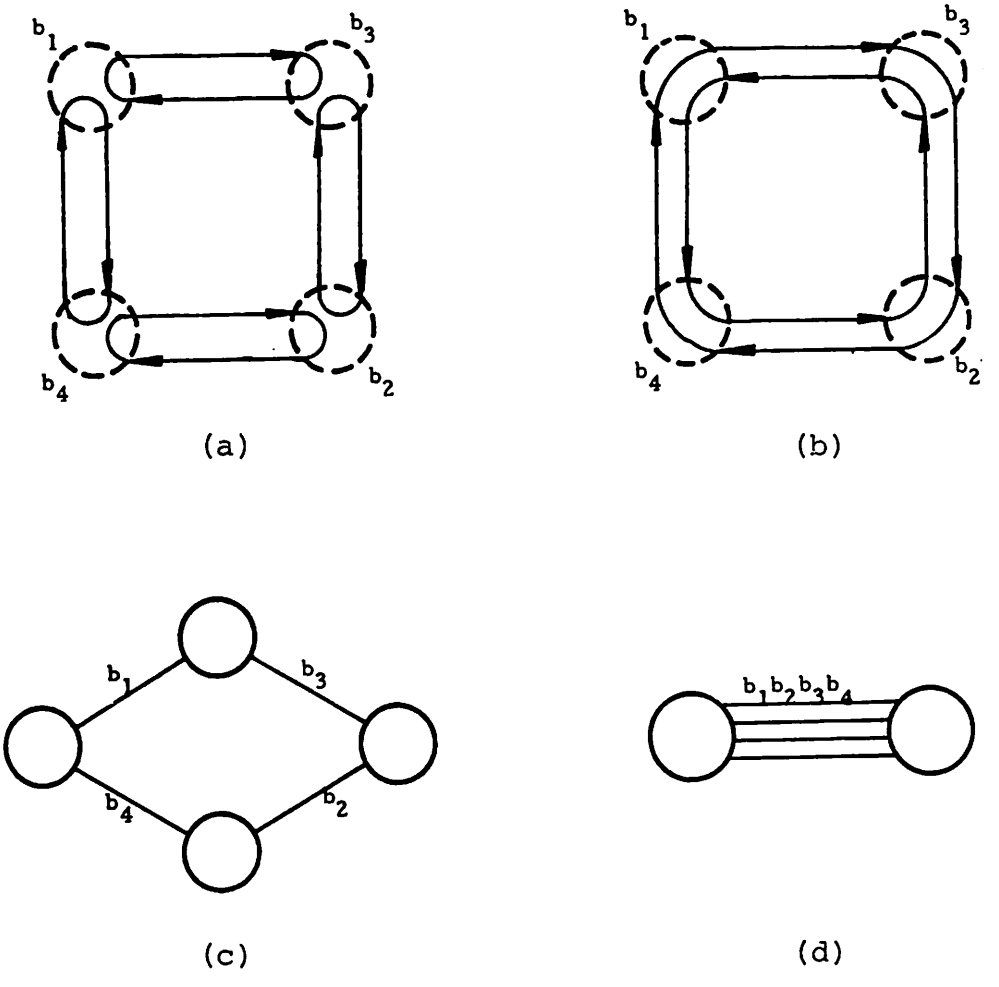


Figure 3.7. Circuit partitions and CA-graphs of the indirect binary 2-cube: (a) circuit partition $\pi_1 = (T,T,T,T)$; (b) circuit partition $\pi_2 = (X,X,X,X)$; (c) CA-graph H_1 of π_1; (d) CA-graph H_2 of π_2.

direct binary 2-cube is single-fault tolerant with respect to DFA, i.e., it is able to tolerate the failure of any single β-element. □

3.3 NONCRITICAL-FAULT CHARACTERIZATION

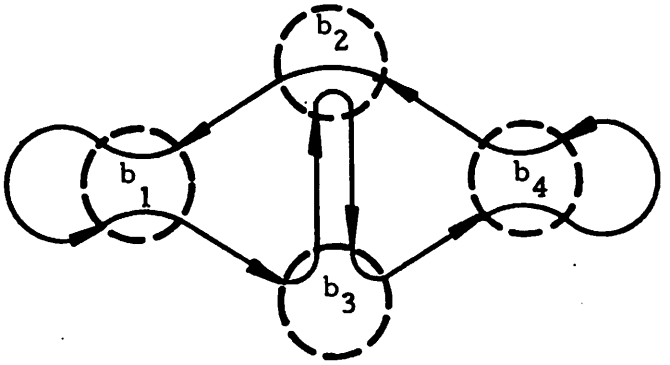
Theorem 3.1 implies that all MCFs of a β-network can be obtained by constructing all its CA-graphs and identifying their cutsets. Once the critical fault set of a β-network is determined, the fault tolerance parameter k can be easily obtained. If the cardinality of the smallest MCF is k+1, then the β-network is k-fault tolerant or k-FT. Theorem 3.1 characterizes the smallest faults of a β-network which destroy the DFA property. In this section we characterize the noncritical faults of a β-network. Using this characterization, it can be determined whether a given fault is critical or not without having to construct the entire set of MCFs. This approach makes use of the Eulerian circuits of the β-graph, and does not require the construction of the circuit partitions or CA-graphs of the β-graph. The theory of Eulerian circuits is well developed [Mayeda 1972], and some of its results can be applied here. For example, unlike the number of circuit partition, the exact number of Eulerian circuits in a β-graph can be easily computed.

An *Eulerian circuit* of a directed graph is a circuit that includes every edge in the graph exactly once. A directed graph possesses an Eulerian circuit if and only

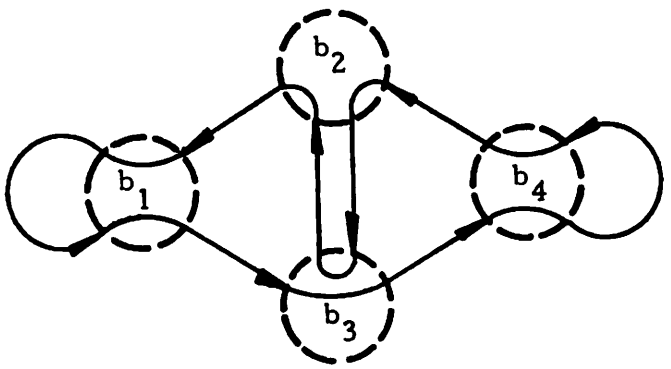
if it is a connected Eulerian graph. Every β-graph having the DFA property is a connected Eulerian graph, and must possess an Eulerian circuit. The β-graph of the 8 × 8 ISE-network, or inverse shuffle-exchange network, of Figure 2.19 has two Eulerian circuits which are illustrated in Figure 3.8. As can be seen from Figure 3.8, an Eulerian circuit, like a circuit partition, can be obtained by setting each of the β-graph vertices to an appropriate state, either X or T. Hence an Eulerian circuit of a β-graph corresponds directly to a particular state of the β-network. We define an *Eulerian circuit state*, or *EC-state*, of a β-network as the state of the β-network that corresponds to an Eulerian circuit of its β-graph. The two Eulerian circuits in Figures 3.8a and 3.8b correspond, respectively, to the EC-states $s_1(b_1,b_2,b_3,b_4) = (X,X,T,X)$ and $s_2(b_1,b_2,b_3,b_4) = (X,T,X,X)$.

3.3.1 Noncritical Faults

Noncritical faults are those faults that do not destroy DFA, and so can be tolerated by a β-network. The number and size of the noncritical faults of a β-network are closely related to the degree of fault tolerance of the network. Given all the EC-states, the noncritical faults of a β-network can be easily characterized, as we now show.



(a)



(b)

Figure 3.8. The two Eulerian circuits of the 8×8 inverse shuffle-exchange network of Figure 2.19: (a) $s_1 = (X,X,T,X)$; (b) $s_2 = (X,T,X,X)$.

Definition 3.5: Two complete or partial states $s_1 = (u_1, u_2, \ldots, u_z)$ and $s_2 = (v_1, v_2, \ldots, v_z)$ of a β-network are *compatible* if $u_i = d$, $v_i = d$, or $u_i = v_i$, for $i = 1,2,\ldots,z$. In other words, if both s_1 and s_2 have specified i^{th} entries, then u_i and v_i must have the same value.

This definition allows us to establish compatibility between two states of any types. We can speak of compatibility between two fault states or between a fault state and an Eulerian circuit state. For example, suppose that two faults f_1 and f_2 are represented by the two fault states s_1^f and s_2^f, respectively where

$$s_1^f = (d,X,T,d,d,d)$$
$$s_2^f = (d,d,T,X,X,d).$$

Let an Eulerian circuit be represented by the EC-state

$$s^e = (X,X,T,T,X,X).$$

We see that the two fault states s_1^f and s_2^f are compatible. s_1^f and s^e are also compatible, while s_2^f and s^e are not compatible.

Theorem 3.2: Let G be the β-graph of a β-network N. A fault f of N corresponding to the fault state s^f is non-critical if and only if s^f is compatible with an EC-state

of N.

Proof: Assume that the fault f is noncritical. We want to show that s^f must be compatible with an EC-state. If f is noncritical then the faulty β-graph induced by f, denoted G^f, must be connected. We know that any faulty β-graph is Eulerian, hence G^f must possess an Eulerian circuit e. Clearly e is also an Eulerian circuit of G, and therefore corresponds to an EC-state of N, say s^e. Since s^e contains all the specified entries of s^f, s^e and s^f must be compatible.

Suppose that the fault state s^f is compatible with an EC-state of N, say s_1^e. The specified entries of s^f constitute a subset of the specified entries of s_1^e. This means that an Eulerian circuit of G can still be constructed from the faulty β-graph G^f by specifying the unspecified entries of s^f according to the entries of s_1^e. We know that a graph possesses an Eulerian circuit if and only if it is a connected Eulerian graph. Hence the faulty β-graph G^f must be connected and f must be concritical. □

Theorem 3.2 provides a method for determining whether a given fault state s^f is critical by comparing s^f to the EC-states and checking for compatibility. The number of Eulerian circuits, and hence EC-states, in a β-graph may be large. However, algorithms for finding an Eulerian

circuit in an Eulerian graph are known [Edmonds and Johnson 1973]. The Eulerian circuits need only be computed once during the design of the β-network. They can subsequently be stored in dictionary form for use in identifying critical faults during system operation.

Example 3.3: The two EC-states of the 8 × 8 inverse shuffle-exchange network shown in Figure 3.8 are $s_1^e = (X,X,T,X)$ and $s_2^e = (X,T,X,X)$. The four MCF states of this β-network are

$$s_1^{mcf} = (T,d,d,d)$$

$$s_2^{mcf} = (d,d,d,T)$$

$$s_3^{mcf} = (d,X,X,d)$$

$$s_4^{mcf} = (d,T,T,d).$$

It can be easily verified that these MCF states are incompatible with both s_1^e and s_2^e. Three examples of noncritical faults in this network are

$$s_1^f = (X,d,T,X)$$

$$s_2^f = (X,X,d,X)$$

$$s_3^f = (d,T,X,d).$$

We see that s_1^f and s_2^f are compatible with s_1^e, and s_3^f is compatible with s_2^e. □

Example 3.4: The indirect binary 2-cube of Example 3.2 has four EC-states

$$s_1^e = (T,X,X,X)$$

$$s_2^e = (X,T,X,X)$$

$$s_3^e = (X,X,T,X)$$

$$s_4^e = (X,X,X,T)$$

The corresponding four Eulerian circuits are illustrated in Figure 3.9. It can be seen that each of the seven MCF states is indeed incompatible with all four EC-states. Every EC-state has exactly one β-element in the T state and three in the X state. Six of the MCF states have exactly two s-a-T β-elements, and the seventh MCF state has all four β-elements s-a-X. □

3.3.2 Computation of Eulerian Circuits

Theorem 3.2 requires the calculation of all EC-states of a β-network to determine if a particular fault is non-critical. In general, the number of Eulerian circuits in an Eulerian graph can be very large, and the procedures for obtaining them can be very time-consuming. β-graphs constitute a special class of Eulerian graphs, in which every vertex has exactly two incoming and two outgoing

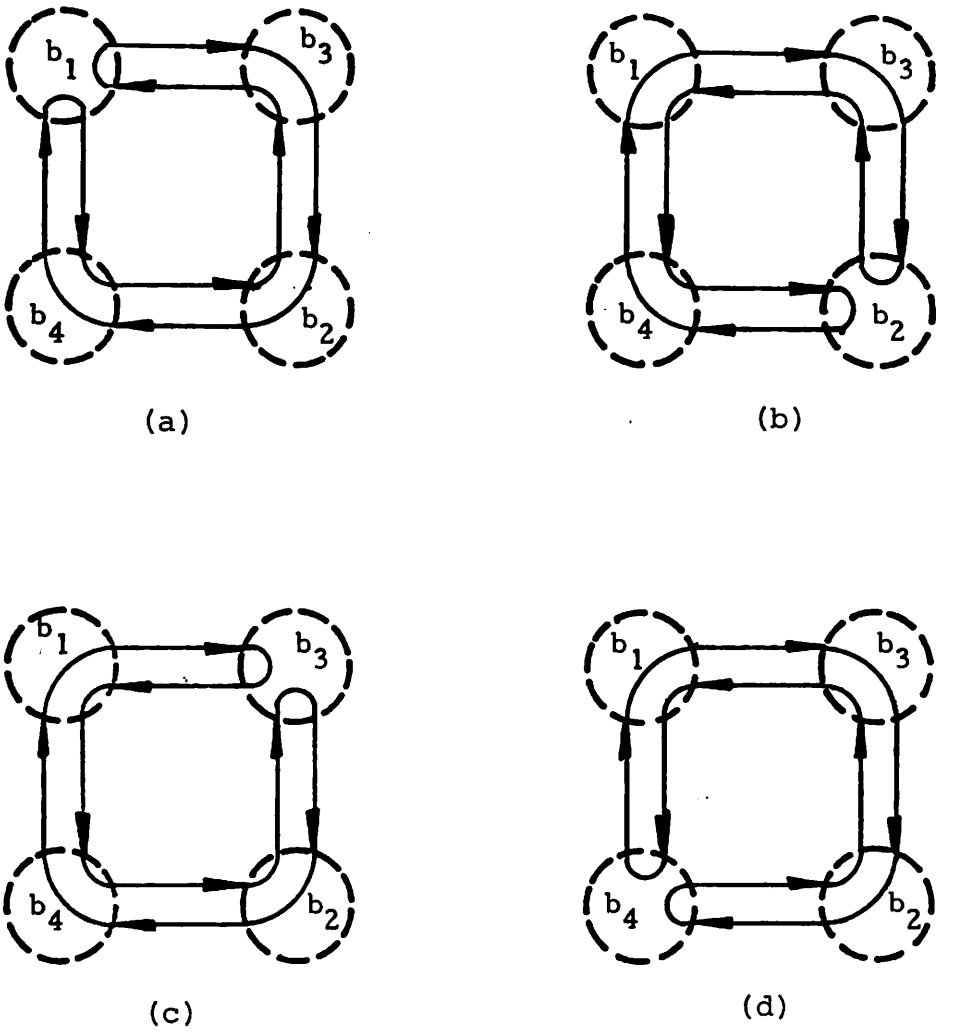


Figure 3.9. The four Eulerian circuits of the indirect binary 2-cube of Figure 2.18: (a) $s_1^e = (T,X,X,X)$; (b) $s_2^e = (X,T,X,X)$; (c) $s_3^e = (X,X,T,X)$; (d) $s_4^e = (X,X,X,T)$.

edges. We now show that the number of Eulerian circuits in a β-graph is typically much smaller than the number in an arbitrary Eulerian graph. Moreover, the number of Eulerian circuits in a β-graph is very easy to compute.

The (*vertex*) *adjacency matrix* of a directed graph having n vertices is an $n \times n$ matrix A such that the $(i,j)^{th}$ entry $a_{i,j}$ of A is the number of edges in the graph from vertex v_i to vertex v_j. The *indegree* (*outdegree*) of a vertex v_i, denoted id_i (od_i) of an Eulerian graph G is the number of incoming (outgoing) edges at v_i. Every vertex in an Eulerian graph has the same number of outgoing as incoming edges. Hence $id_i = od_i = d_i$ for all i. Let the $n \times n$ *diagonal matrix T* of an Eulerian graph be defined as follows: $t_{i,j} = 0$ if $i \neq j$ and $t_{i,j} = d_i$ if $i = j$. The *Kirchhoff matrix K* [Deo 1974] of an Eulerian graph G is defined as the $n \times n$ matrix T-A, where A is the adjacency matrix and T is the diagonal matrix of G. The i^{th} *minor* of K denoted Δ_i is the determinant of the submatrix K_i of K obtained by removing the i^{th} column and the i^{th} row of K. It has been shown [Bott and Mayberry 1954] that for an Eulerian graph with n vertices $\Delta_1 = \Delta_2 = \cdots = \Delta_n = \Delta$. Thus we may refer to Δ unambiguously as the *minor* of K.

Lemma 3.3: [Aardenne-Ehrenfest and DeBruijn 1951]. A connected Eulerian graph G of n vertices has δ distinct Eulerian circuits given by

$$\mathcal{E} = \Delta \prod_{i=1}^{n} (d_i - 1)!$$

where Δ is the minor of the Kirchhoff matrix of G, and d_i is the indegree (outdegree) of the i^{th} vertex of G. In the case of β-graphs $d_i = 2$ for all i, hence by the above theorem the number of Eulerian circuits $\mathcal{E}$ in a β-graph is Δ. It was also shown by Bott and Mayberry that the number of spanning trees rooted at a vertex v_i of an Eulerian graph is equal to Δ_i. Combining these two facts we obtain the following result.

Theorem 3.3: Let G be the β-graph of a β-network containing n β-elements. The number $\mathcal{E}$ of Eulerian circuits in G is equal to any minor of the $n \times n$ (Kirchhoff) matrix $K = 2I - A$, where I is the $n \times n$ identity matrix and A is the vertex adjacency matrix of G. $\mathcal{E}$ is also equal to the number of spanning trees rooted at any vertex of G.

For example, the vertex adjacency matrix of the β-graph of the inverse shuffle-exchange network of Figure 2.19 is

$$A = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}.$$

Hence the Kirchhoff matrix $K = 2I - A$ is

$$K = \begin{bmatrix} 1 & -1 & 0 & 0 \\ 0 & 2 & -1 & -1 \\ -1 & -1 & 2 & 0 \\ 0 & 0 & -1 & 1 \end{bmatrix}.$$

The first minor of K is then

$$\Delta = \det \begin{bmatrix} 2 & -1 & -1 \\ -1 & 2 & 0 \\ 0 & -1 & 1 \end{bmatrix} = 2 .$$

Therefore the β-graph of Figure 2.19b has exactly two distinct Eulerian circuits. These Eulerian circuits are illustrated in Figure 3.8.

In general, the determination of the Eulerian circuits of an arbitrary β-network is a computationally complex problem. For obtaining just one Eulerian circuit a simple algorithm exists [Edmonds and Johnson 1973], whose complexity is proportional to the number of β-elements in the β-network. This algorithm finds a spanning tree rooted at a particular vertex, and makes use of the fact that there exists one Eulerian circuit corresponding to each spanning tree rooted at a particular vertex. To determine all the Eulerian circuits of the β-graph, all the spanning trees rooted at the vertex must be identified. Most useful β-networks and their β-graphs have considerable structural symmetry. By taking advantage of this symmetry, the

computational burden of determining all Eulerian circuits can be greatly eased.

3.4 MCF-STATES AND EC-STATES

The MCF characterization of Section 3.2 allows us to compute all the MCF-states of a β-network by constructing all the circuit partitions and then obtaining the cutsets of the corresponding CA-graphs. The noncritical fault characterization of Section 3.3 enables us to judge whether a given fault is critical or not by checking it for compatibility with the EC-states of the β-network. Although we dealt with MCF-states and EC-states from two different viewpoints, these two sets of states of a β-network are intimately related. Both are indicators of the fault tolerance of a β-network. The MCF-states represent the simplest fault states which must be avoided to maintain DFA; whereas the EC-states represent the ultimate failsafe states of a β-network, i.e., states in which all the specified β-elements are faulty but DFA is still maintained. In this section we make use of Boolean algebra to characterize the relationship between EC-states and MCF-states.

3.4.1 Partial State Expansion

A partial state of a β-network N is a state some of whose entries are unspecified. A (complete) state is a state

all of whose entries are specified. We define the *expansion of a (partial) state* s, denoted $\langle s \rangle$, as the set of all complete states of N compatible with s. For example, the expansion of the partial state s = (d,d,X,T) is the following set of complete states:

$$\langle s \rangle = \{(T,T,X,T),(T,X,X,T),(X,T,X,T),(X,X,X,T)\} .$$

A partial state s_1 is compatible with a complete state s_2, or vice versa, if and only if s_2 is an element of $\langle s_1 \rangle$. The expansion of a partial state consists of all possible ways of fully specifying the entries of the partial state. Clearly the expansion of a complete state is the complete state itself.

By analogy with the notion of an MCF-state we define a *maximal noncritical fault (MNF) state* as a fault state s^f representing a noncritical fault such that no other noncritical fault state $s \neq s^f$ contains all the specified entries of s^f. It can easily be shown that a fault state s^f is an MNF-state if and only if it is an EC-state. We can restate Theorem 3.2 in the following form.

Corollary 3.1: A fault state s^f is a noncritical fault state if and only if its expansion $\langle s^f \rangle$ contains an EC-state.

We can also characterize MCFs using the same approach.

Corollary 3.2: A fault state s^f is an MCF-state if and only if $\langle s^f \rangle$ does not contain any EC-state, and $\langle s^f \rangle$ is not a proper subset of $\langle s_1^f \rangle$ for any other critical fault state s_1^f.

To illustrate the above results, consider the β-network of Figure 2.19 again. This β-network has the two EC-states $s_1^e = (X,X,T,X)$ and $s_2^e = (X,T,X,X)$ illustrated in Figure 3.8. Consider the two faults $s_1^f = (T,T,T,d)$ and $s_2^f = (d,X,X,d)$. The expansions of s_1^f and s_2^f are

$$\langle s_1^f \rangle = \{(T,T,T,T),(T,T,T,X)\}$$

$$\langle s_2^f \rangle = \{(T,X,X,T),(T,X,X,X),(X,X,X,T),(X,X,X,X)\}.$$

Neither $\langle s_1^f \rangle$ nor $\langle s_2^f \rangle$ contains s_1^e or s_2^e, hence both s_1^f and s_2^f are critical. However, s_1^f is not an MCF-state because $\langle s_1^f \rangle$ is a proper subset of $\langle s_3^f \rangle$, where s_3^f is the MCF-state (T,d,d,d).

3.4.2 Switching Theoretic Interpretation

The state $(s_1,s_2,\ldots,s_z)$ of a β-network N with z β-elements can be viewed as an element in a 2^z-element "vector" Boolean algebra B_{2^z} [Lee 1976]. Here we reexamine the foregoing concepts in terms of B_{2^z} using some standard concepts and notation from switching theory [Friedman and

Menon 1975]. Every complete state of N corresponds to a minterm of B_{2z}. Similarly every partial state can be represented by a (incomplete) product term defined on B_{2z}. For example, the complete state $s_1 = (X,T,X,T)$ can be represented by the minterm $m_1 = b_1\bar{b}_2b_3\bar{b}_4$, where b_i ($\bar{b}_i$) denotes the i^{th} β-element in the X (T) state. The partial state $s_2 = (X,T,d,d)$ corresponds to the product term $m_2 = b_1\bar{b}_2$.

Since every EC-state is a complete state, it can be represented by a minterm. We define an *EC-term* as a minterm that corresponds to an EC-state. Similarly, an *MCF-term* is a product term that represents an MCF-state. If a partial state s is represented by the product term p, then the expansion $\langle s \rangle$ is equivalent to the set of all minterms covered by p. We can then restate Corollary 3.1 in the follwoing form. A product term p represents a critical fault state if and only if p does not cover any EC-term. For example, let a β-network containing three β-elements have two EC-states $s_1^e = (X,T,X)$ and $s_2^e = (T,T,T)$. The two corresponding EC-terms are $m_1 = b_1\bar{b}_2b_3$ and $m_2 = \bar{b}_1\bar{b}_2\bar{b}_3$. The fault state $s_1^f = (T,X,d)$ is obviously critical because the product term $m_3 = \bar{b}_1b_2$ does not cover either m_1 or m_2. The Karnaugh map [Friedman and Menon 1975] of Figure 3.10 shows the two EC-terms m_1 and m_2 which are marked "e," and the product term m_3 representing the critical fault s_1^f. From Figure 3.10 we can also conclude that the fault state represented by the product term $m_4 = \bar{b}_2\bar{b}_3$ is not critical

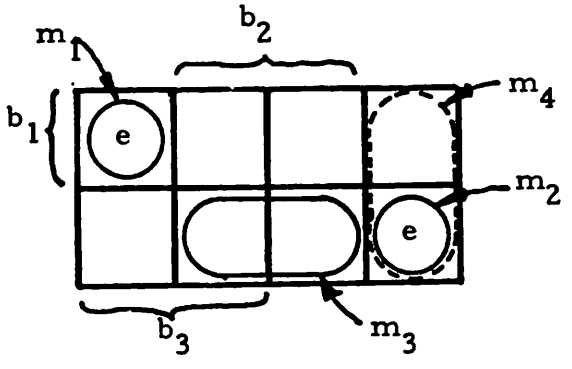


Figure 3.10. A three-variable Karnaugh map with two EC-terms m_1 and m_2, and a critical fault term m_3.

because m_4 covers m_2.

We define the *EC-expression* f^{ec} of a β-network as the Boolean expression consisting of the Boolean sum of all the EC-terms. For example, the indirect binary 2-cube of Example 3.4 has four EC-states, viz.

$$s_1^e = (T,X,X,X)$$

$$s_2^e = (X,T,X,X)$$

$$s_3^e = (X,X,T,X)$$

$$s_4^e = (X,X,X,T) .$$

These four EC-states correspond to the following four EC-terms:

$$m_1 = \bar{b}_1 b_2 b_3 b_4$$

$$m_2 = b_1 \bar{b}_2 b_3 b_4$$

$$m_3 = b_1 b_2 \bar{b}_3 b_4$$

$$m_4 = b_1 b_2 b_3 \bar{b}_4$$

The EC-expression f^{ec} of the indirect binary 2-cube is then:

$$f^{ec} = \bar{b}_1 b_2 b_3 b_4 + b_1 \bar{b}_2 b_3 b_4 + b_1 b_2 \bar{b}_3 b_4 + b_1 b_2 b_3 \bar{b}_4 .$$

The Karnaugh map of f^{ec} appears in Figure 3.11.

We now present a result relating EC-terms and MCF terms.

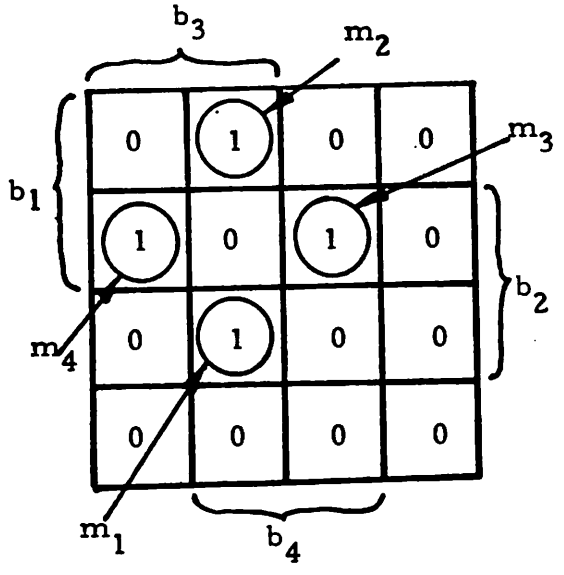


Figure 3.11. The four EC-terms and the EC-expression of the indirect binary 2-cube.

Theorem 3.4: Let N be a β-network with the EC-terms $m_1, m_2, \ldots, m_\delta$. The prime implicates of the function $f^{ec} = \prod_{i=1}^{\delta} m_i$ are the MCF-terms of the β-network.

Proof: If m is a prime implicate of f^{ec}, then m does not cover any EC-terms. Hence m must represent a critical fault state. Since m is prime, there must not exist another product term that covers m. Therefore m must be an MCF-term.

Assume that m_1 is an MCF-term but is not a prime implicate of f^{ec}. There must exist a prime implicate m_2 of f^{ec} that covers m_1. Since m_2 is a prime implicate of f^{ec} it must not cover any EC-term and hence represents a critical fault state. This contradicts the assumption that m_1 is an MCF-term, because MCF-terms cannot be covered by another critical fault term. Therefore m_1 must be a prime implicate of f^{ec}. □

Example 3.5: Figure 3.12 shows the Karnaugh map of the function defined by the EC-expression of the indirect binary 2-cube. Examining Figure 3.12 we see there are seven prime implicates of f^{ec}, namely.

$$m_1 = \bar{b}_1\bar{b}_2$$

$$m_2 = \bar{b}_1\bar{b}_3$$

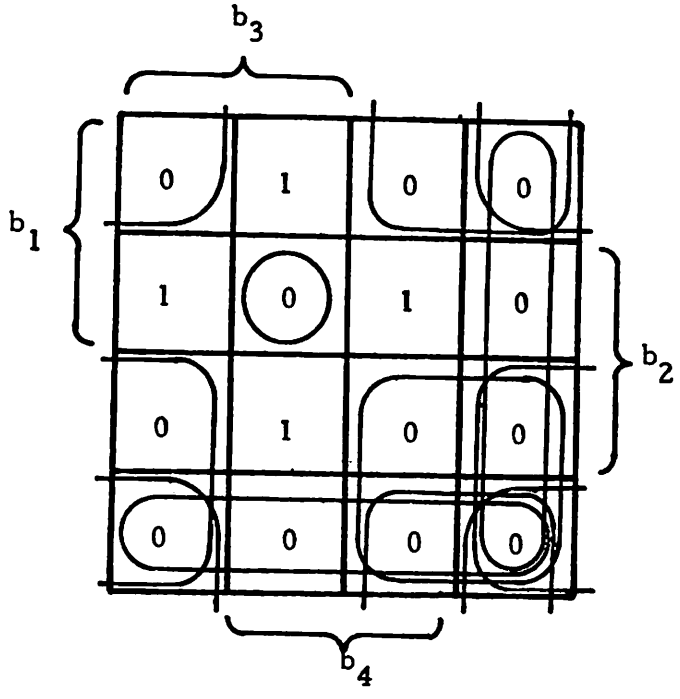


Figure 3.12. The function defined by the EC-expression of the indirect binary 2-cube, and its seven prime implicates.

$$m_3 = \bar{b}_1\bar{b}_4$$

$$m_4 = \bar{b}_2\bar{b}_3$$

$$m_5 = \bar{b}_2\bar{b}_4$$

$$m_6 = \bar{b}_3\bar{b}_4$$

$$m_7 = b_1b_2b_3b_4 .$$

The above list of prime implicates corresponds exactly to the list of MCF-states of the indirect binary 2-cube obtained in Example 3.2. □

Theorem 3.4 provides a link between MCFs and the noncritical faults of a β-network. The Boolean algebraic viewpoint presented here provides a simple and a rigorous characterization of the relationship between the minimal critical fault states and the maximal noncritical fault states.

CHAPTER 4

SYNTHESIS OF FAULT-TOLERANT β-NETWORKS

The critical fault analysis of Chapter 3 is quite general, and does not depend on the specific β-network or β-graph under consideration. It focuses on the detailed structure of individual faults, instead of the overall fault tolerance of the β-network. Most of the characterization results in Chapter 3 involve complex combinatorial properties of β-graphs. Hence the computation required for practical implementation of these results may be prohibitive for large β-networks with arbitrary structures. Fortunately, most practical β-networks have regular structures which can reduce the computation involved in critical fault analysis. By utilizing the regularity in a β-graph, we can often obtain useful overall measures of fault tolerance without performing detailed critical fault analysis. Furthermore, regular β-networks with a specified degree of fault tolerance are relatively easy to synthesize. In the synthesis of fault-tolerant β-networks, network operating speed or performance must also be considered.

In this chapter several classes of β-networks with regular structures are examined, and the analytical re-

sults from previous chapters are applied to the synthesis of fault-tolerant β-networks. The communication delay between computing units is chosen as a measure of the performance of a β-network. A formal definition of communication delay is introduced and compared with the fault tolerance parameter introduced in Section 3.1. Two important specific classes of β-networks are studied, namely DPR-networks and MISE-networks. It is shown that DPR-networks possess the maximal fault tolerance and maximal communication delay; while MISE-networks are minimally fault tolerant, but have minimal communication delay. This chapter concludes with a classification of β-networks based on the fault-tolerance and the communication-delay parameters.

4.1 FAULT TOLERANCE VS. COMMUNICATION DELAY

Traditionally, high speed or high performance has been the primary design objective of connecting networks. Due to the proliferation of fault-critical applications of computers, fault tolerance is becoming an important requirement for interconnected computer systems and their ICNs. It is probable that the design of future β-networks for ICNs will involve striking a balance between performance and fault-tolerance. In Section 3.1 we introduced the fault-tolerance parameter k as a measure of fault tolerance of a β-network. A basic goal in designing a fault-

tolerant β-network is to maximize k. The performance of a β-network can be measured by its connecting capability. Several general connectivity properties were mentioned in earlier chapters, e.g., rearrangeability, full access and dynamic full-access (DFA). All useful β-networks have the DFA property. A finer, and preferably numerical, measure of performance is needed to distinguish β-networks that have DFA. Communication delays between computing units can be used for this purpose.

4.1.1 Communication Delay

The communication delay from computing unit i to computing unit j will be measured here by the minimum number of β-elements that need to be traversed by data being sent from unit i to unit j. A communication delay parameter d for a β-network is obtained by considering the minimum delays between all pairs of computing units and choosing the maximum or worst case value of these delays. To formalize the above definition we again make use of the β-graph model of β-networks introduced in Section 2.3.

Definition 4.1: The *edge-distance*, or simply *distance*, from edge i to edge j in a β-graph is the number of intermediate vertices in the shortest directed path having edges i and j as its first and last edges, respectively. The *edge-diameter*, or simply *diameter*, of a β-graph is the

longest distance between any two edges of the β-graph. The *communication delay (CD) parameter d* of a β-network is the diameter of its β-graph.

The CD parameter d indicates the worst possible delay between any pair of computing units in the interconnected system.

As an illustration, we now consider two single-stage β-networks having very different values of the CD parameter d. The β-network of Figure 4.1a is the 16 × 16 inverse shuffle exchange network, or ISE-network, defined in Section 2.3. The 16 × 16 permuter in the single-stage β-network of Figure 4.1b is $\varphi = \begin{pmatrix} 1 & 2 & 3 & 4 & \cdots & 15 & 16 \\ 2 & 3 & 4 & 5 & \cdots & 16 & 1 \end{pmatrix}$. We call this the 16 × 16 *single cycle shift network*, or SCS-network, which is defined more formally in Section 4.5. Both β-networks are 0-FT. For example, the ISE network cannot tolerate its β-element b_1 s-a-T, and the SCS-network cannot tolerate its β-element b_1 s-a-X. Figures 4.2a and 4.2b depict the β-graphs of Figures 4.1a and 4.1b respectively. The diameters of these two β-graphs, and therefore the CD parameters of the corresponding β-networks, can be easily computed. The CD parameter for the 16 × 16 ISE-network is four, and that of the 16 × 16 SCS-network is eight. ISE-networks and SCS-networks of other sizes can be similarly defined. In general, an n × n ISE-network, where $n = 2^m$ for some integer m, has CD parameter $d_1 = \log_2 n$;

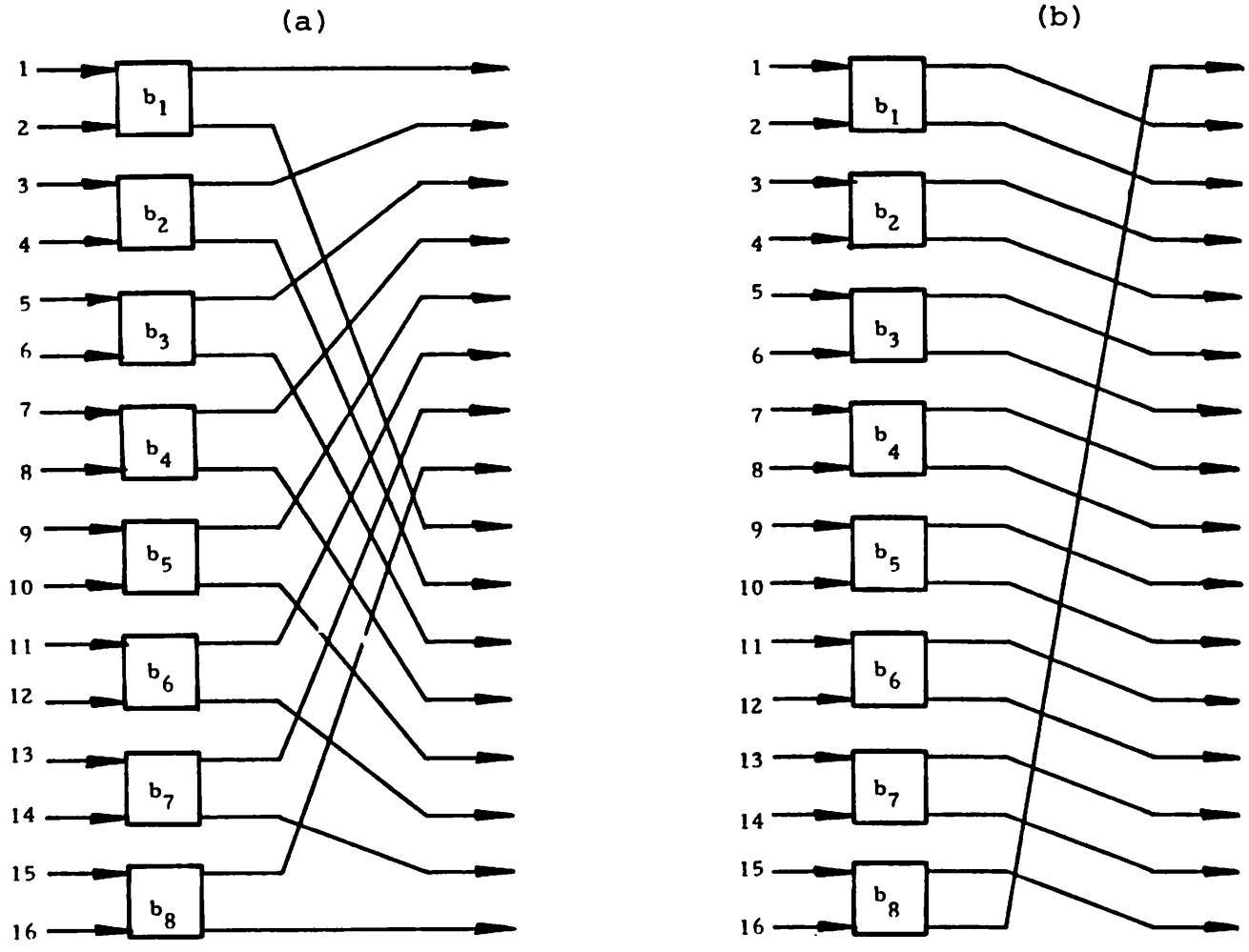


Figure 4.1. (a) The 16 × 16 inverse shuffle exchange (ISE) network; (b) The 16 × 16 single cycle shift (SCS) network.

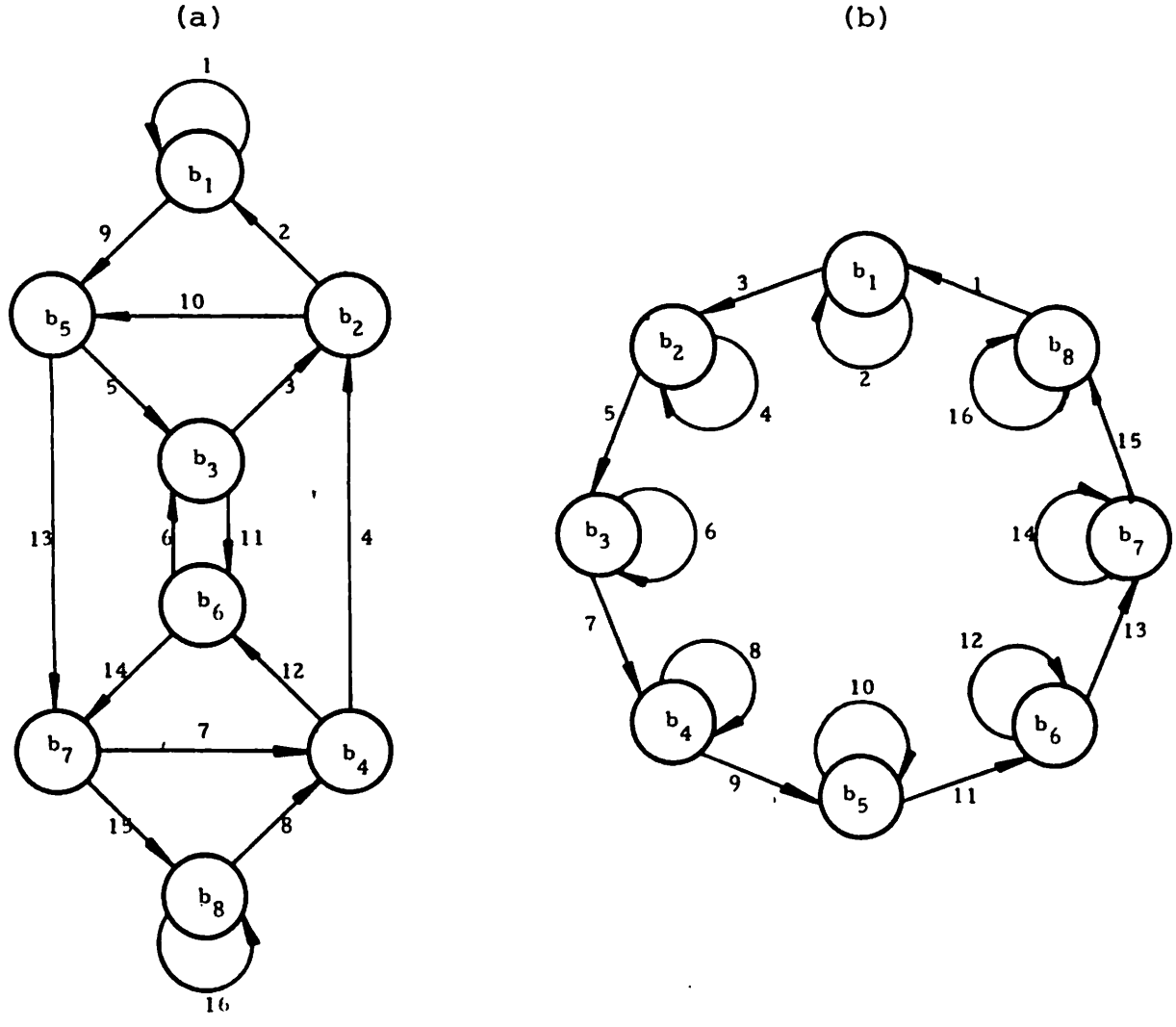


Figure 4.2. (a) β-graph of the 16 × 16 ISE-network; (b) β-graph of the 16 × 16 SCS-network.

and an $n \times n$ SCS-network has CD parameter $d_2 = n/2$.

4.1.2 Bounds on k and d

Both the FT parameter k and the CD parameter d depend on the structure of a β-network, and are therefore properties of the β-graph. In this subsection we derive tight lower and upper bounds for k and d in terms of n, the number of β-elements in a β-network. We also establish a fundamental relationship between k and d.

The smallest possible value for k is clearly zero, while the largest possible value of k is n. An n-FT β-network can tolerate certain faults affecting all its β-elements. This implies that the entire β-network is unnecessary; hence, $k = n$ is impossible. Therefore the largest possible value for k is n-1. We have already shown that both the ISE-network and the SCS-network are 0-FT. In subsequent sections we demonstrate the existence of a class of (n-1)-FT β-networks. Hence 0 and n-1 are tight lower and upper bounds, respectively, for k. The corresponding bounds for d are given in the following lemma.

Lemma 4.1: For any β-network with $n > 2$ β-elements and CD parameter d,

$$\lfloor \log_2 n \rfloor + 1 \leq d \leq n.$$

These bounds are tight.

Proof: The CD parameter d is the diameter of the β-graph. The diameter of a β-graph of n vertices cannot exceed n, the total number of vertices. The value $d = n$ is achievable, for example, in the case of an SCS-network containing n β-elements. Hence n is a tight upper bound on d.

Since each vertex in a β-graph has two incoming and two outgoing edges the outdegree of a vertex is two. The maximum number of edges which are exactly at distance d from any edge i is 2^d. The maximum number of distinct edges reachable from i within distance d is $2^d+2^{d-1}+\ldots+2 = 2^{d+1}-2$; see Figure 4.3. We also know that a β-graph with n vertices has exactly 2n edges, and each edge must be able to reach the other 2n-1 edges. Hence the following inequality must hold:

$$2^{d+1}-2 \geq 2n-1$$

from which it follows that

$$d \geq \lceil \log_2(n+1/2) \rceil .$$

Now $\lceil \log_2 x \rceil = \lfloor \log_2 x \rfloor + 1$ unless x is an integral power of 2, which is impossible when $x = n+1/2$. Hence

$$d \geq \lfloor \log_2 n \rfloor + 1.$$

This lower bound for d is achieved by the ISE-network of n

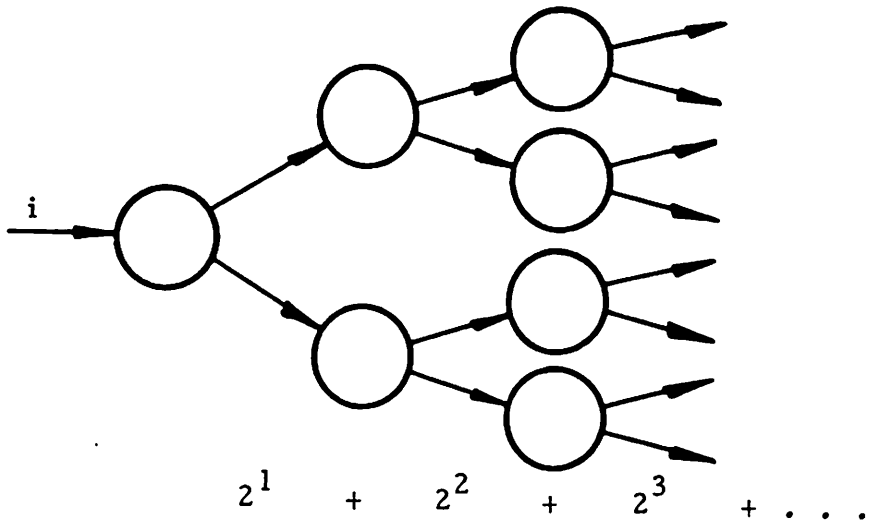


Figure 4.3. Maximum number of edges reachable from an edge i in a β-graph.

β-elements, hence it also is tight. □

The following lemma establishes a relationship between k and d.

Lemma 4.2: For any β-network with FT parameter k and CD parameter d, $k \leq d$.

Proof: Every elementary circuit in a β-graph corresponds to a critical fault of the β-network. The β-elements represented by the vertices of the elementary circuit, can fail in such a way that the edges of the elementary circuit are isolated from the rest of the β-graph as shown in Figure 4.4. Hence the β-graph of a k-FT β-network must not contain any elementary circuit of length less than or equal to k.

Assume that C is an elementary circuit of minimal length $h \geq k+1$ in a k-FT β-graph. Assume also that C contains edges i and j such that the source vertex of i is the destination vertex of j as illustrated in Figure 4.5. Edge i can reach edge j via the edges of C, i.e., there exists a directed path from i to j consisting of the edges of C. Since the length of C is h, the distance from i to j is at most h-1. In fact, it must be exactly h-1, otherwise there must exist a path P from the destination vertex of i to the source vertex of j containing at most h-3 edges. In that case the path P together with edges i and

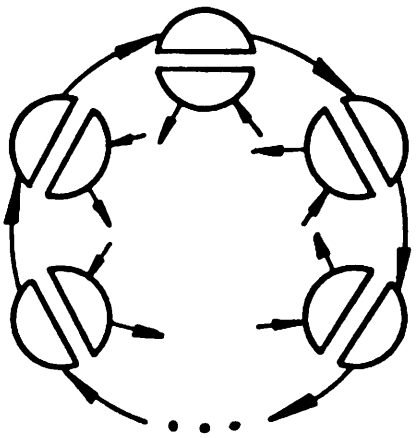


Figure 4.4. A critical fault corresponding to an elementary circuit of a β-graph.

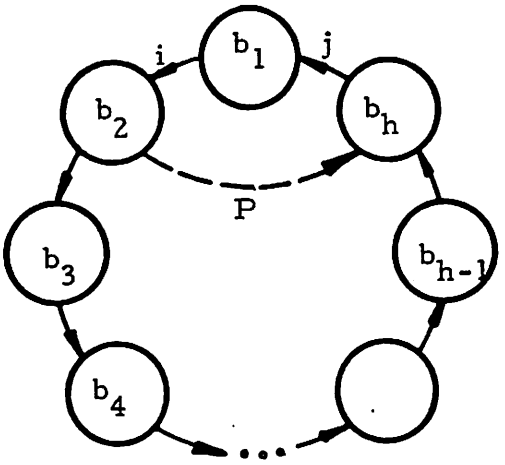


Figure 4.5. An elementary circuit C of length $h \geq k+1$.

j would constitute an elementary circuit of length h-1 or less. This contradicts the minimal-length property of C. Hence the distance from i to j is exactly h-1. Since $h \geqq k+1$, the distance from i to j must be k or more. Hence the diameter of the β-graph cannot possibly be less than k. Therefore $k \leqq d$. □

We can summarize the foregoing results in the following theorem.

Theorem 4.1: Let N be a β-network with n β-elements, where $n > 2$. The FT parameter k and the CD parameter d of N are related as follows:

$$0 \leqq k \leqq n-1 \tag{4.1}$$

$$\lfloor \log_2 n \rfloor + 1 \leqq d \leqq n \tag{4.2}$$

$$k \leqq d \tag{4.3}$$

The bounds in (4.1) and (4.2) are tight.

Every β-network has a unique k and a unique d. As faults occur, k tends to decrease, while d tends to increase. A β-network loses the DFA property when d becomes infinite. The synthesis of practical fault-tolerant β-networks involves finding an appropriate balance between k and d.

4.2 CYCLIC β-NETWORKS AND R-SETS

In recent years we have seen the rapid development of local communication networks for computer systems [Clark 1978]. The primary aims of these networks are to facilitate resource sharing or to enhance overall system performance. Various schemes have been proposed for constructing local communication networks. The most common is the shared bus scheme, which comprises one or more data buses that are shared by a number of computing devices, such as processors, memory subsystems, peripheral controllers, or complete computers. A well-known alternative to the shared bus scheme is the ring system [Farber 1975,Hafner 1974, Wilkes 1975]. There are various designs for ring systems, the most common being the single-loop design [Zafiropulo 1974,Wilkes 1975]. An example of a single-loop network is illustrated in Figure 4.6. It consists of n computing units, each supported by a switching element which acts as a communication interface to other computing units. The n switching elements are connected by a unidirectional communication ring.

The two major problems of single loop systems are long communication delay and low reliability. To circumvent these problems many ring systems employ additional data links, or even additional loops. Figure 4.7 illustrates a ring network with additional data links and

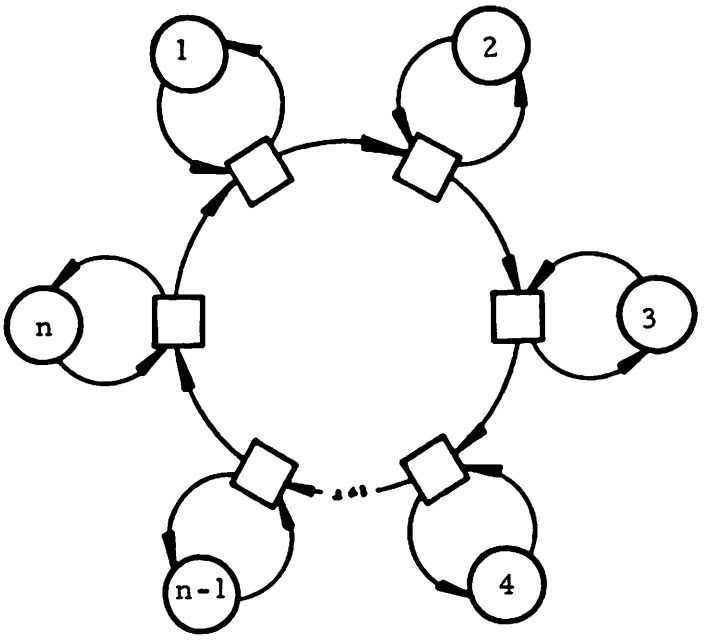


Figure 4.6. The single-loop ring network.

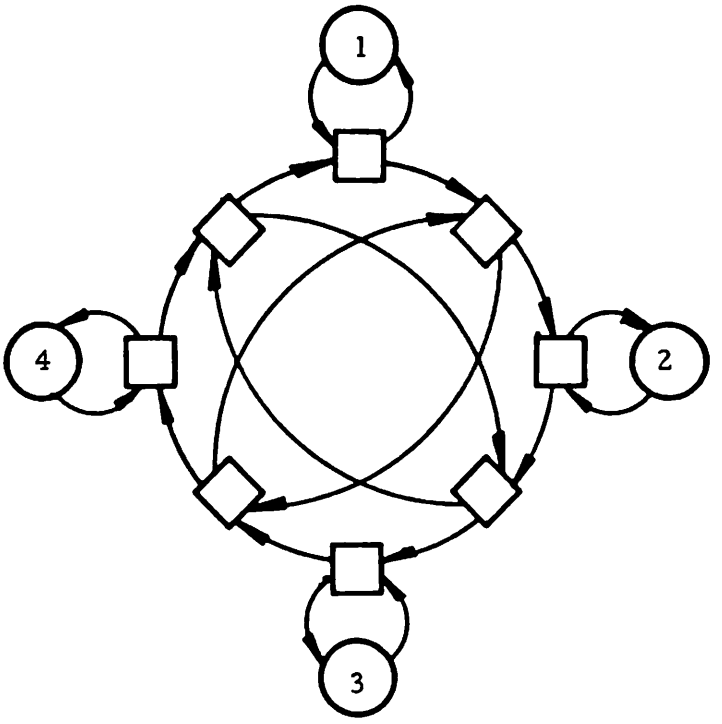


Figure 4.7. A ring network with redundant links.

switching elements. Local computer communication networks are considered here to be interconnected systems in which the ring network constitutes an ICN. Many ring-type communication networks can be realized efficiently by β-networks. For example, the single-loop network of Figure 4.6 can be realized by an SCS-network. A 12×12 SCS-network supporting a ring of six computing units is shown in Figure 4.8. The β-graph in Figure 4.8 is isomorphic with the structure depicted in Figure 4.6. Many of the proposed ring networks can be realized by a class of β-networks called cyclic β-networks which we now introduce.

4.2.1 Cyclic β-networks

A *Hamiltonian circuit* of a directed graph is a circuit that passes through every vertex exactly once [Harary 1969]. The β-graphs of many well-known β-networks contain Hamiltonian circuits. For example, the main ring in Figure 4.8b constitutes a Hamiltonian circuit. Every β-graph of n vertices has 2n edges. A Hamiltonian circuit C of a β-graph consists of n edges. The vertices of C can be labeled $v_0, v_1, \ldots, v_{n-1}$ according to the sequence of the n vertices in C. A typical edge in C connects vertex v_i to vertex $v_{i+1 \pmod n}$ for $i = 0,1,\ldots,n-1$. A cyclic β-network is a β-network that contains a Hamiltonian circuit in its β-graph, with the n edges not in the Hamiltonian circuit satisfying a certain regularity condition. For the present

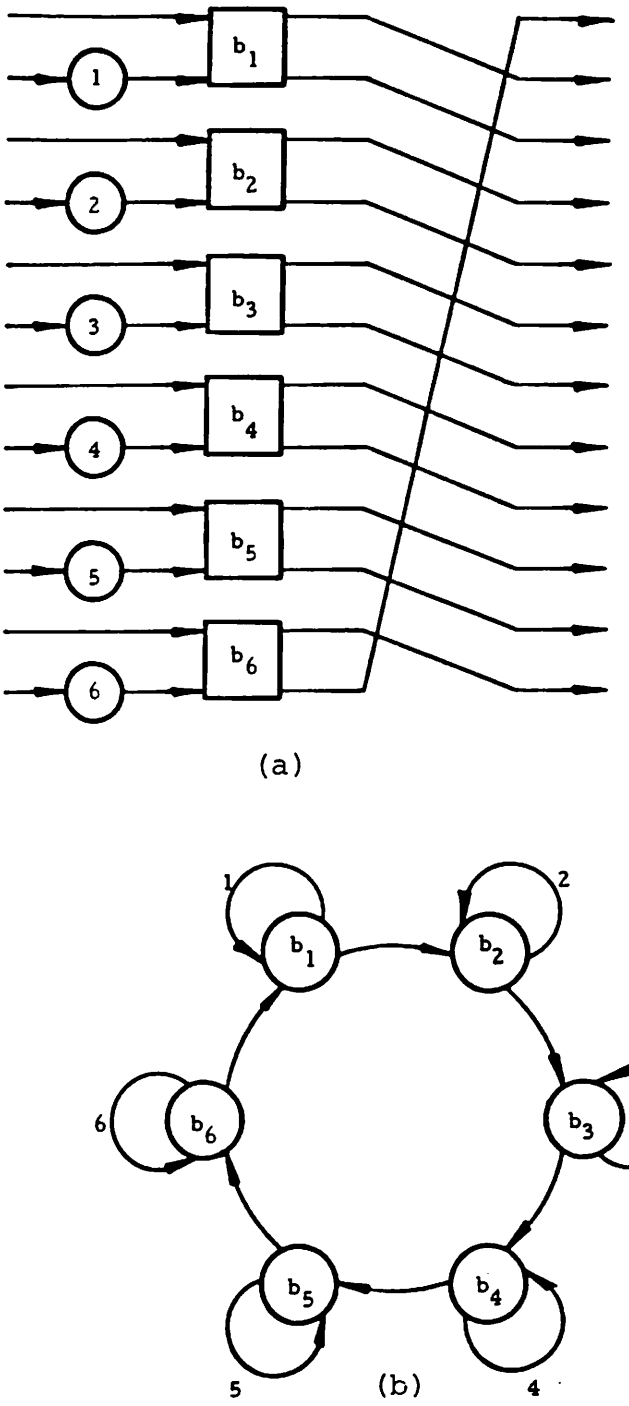


Figure 4.8. Emulation of a ring network by a single-stage β-network: (a) a single-loop β-network; (b) its β-graph.

we do not assign labels to the edges of the β-graph.

Definition 4.2: A β-network of n β-elements is a *cyclic β-network* with generator h if there exists a labeling of the vertices of its β-graph such that for every vertex labeled v_i, the two edges from v_i terminate at vertices $v_{i+1 \pmod n}$ and $v_{i+h \pmod n}$ for some h, $0 \leq h \leq n-1$.

For a given n there are n distinct possible values for h. Hence there are exactly n distinct cyclic β-networks with n β-elements. Figure 4.9 illustrates the β-graphs of the six cyclic β-networks for n = 6. Every cyclic β-network can be characterized by the two parameters n and h.

The β-graphs of cyclic β-networks are isomorphic with Cayley color graphs of cyclic groups [Cayley 1895]. In a two-color Cayley graph there are n vertices and 2n directed edges. The vertices represent elements of a cyclic group. One outgoing edge at each vertex is assigned the color A; the other outgoing edge is assigned the color B. Let vertex v_i represent the element i of the cyclic group. The set of n A-colored (B-colored) edges represents a generator a (b) of the cyclic group in the following sense. If $i+a \pmod n = j$ is in the cyclic group then the A-colored edge from vertex v_i must terminate at vertex v_j. Similarly, if $i+b \pmod n = k$ then the B-colored edge from v_i must terminate at vertex v_k. Hence all the elements of

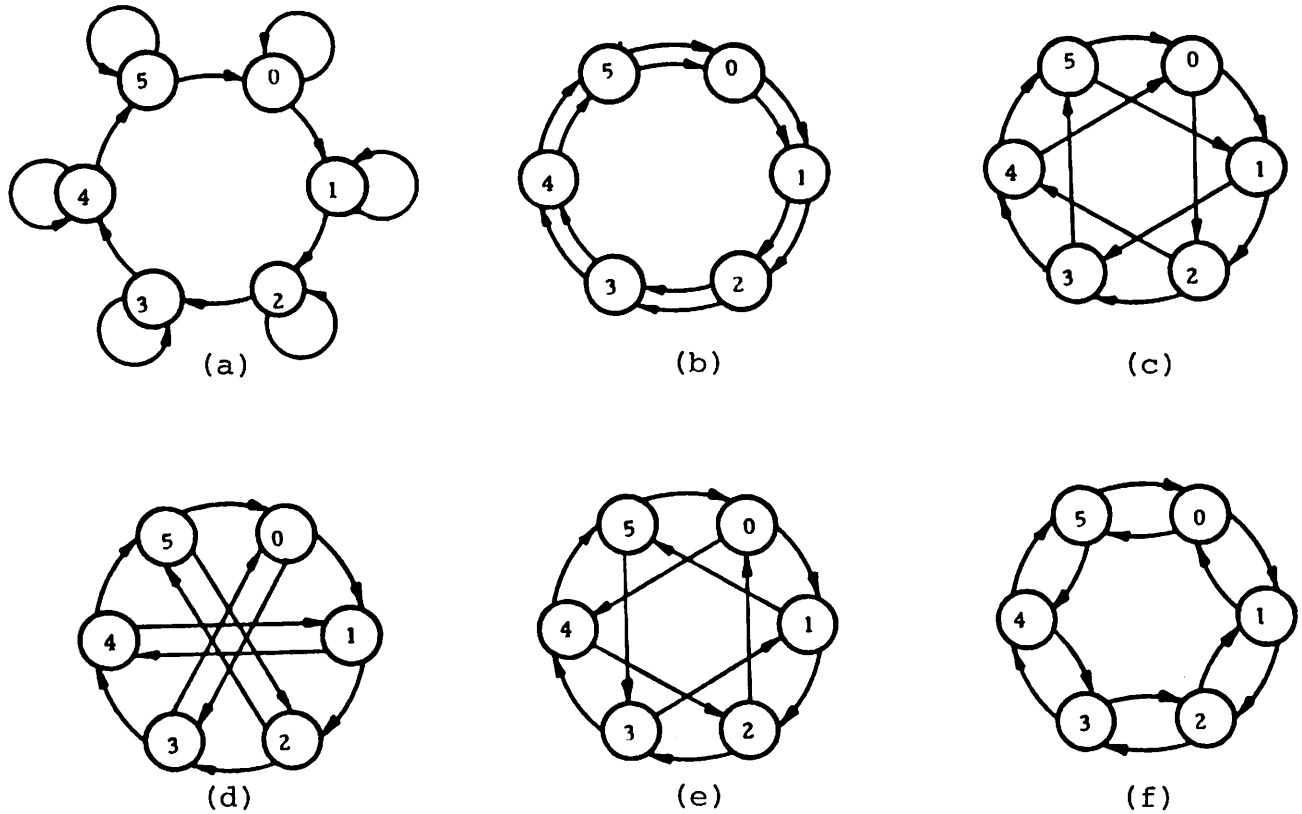


Figure 4.9. The β-graphs of the six cyclic β-networks of order 6: (a) h = 0; (b) h = 1; (c) h = 2; (d) h = 3; (e) h = 4; (f) h = 5.

the cyclic group can be generated from a and b by successively adding a and b to themselves. As can be seen, a cyclic β-network of order n, i.e., one containing n β-elements, with generator h is a Cayley color graph of a cyclic group with generators 1 and h. Because of their regularity, certain fault-tolerance characteristics of cyclic β-networks can be easily analyzed. For example, there are certain easily-identified subsets of β-elements in a β-network which are very tolerant of faults. These subsets are called R-sets.

4.2.2 R-sets

Let S_1 and S_2 be two disjoint subsets of β-elements whose union is the set of all β-elements in a β-network. If any fault involving only the β-elements of S_1 cannot be critical as long as all the β-elements of S_2 are fault-free, then S_1 can be considered to be "redundant" with respect to S_2. Similarly, S_2 can be considered as "hardcore" with respect to S_1. For example, if a β-network is 1-FT then every β-element is redundant with respect to the set of all other β-elements.

Definition 4.3: A subset S of the set of β-elements of a β-network is called a *redundant subset*, or simply an *R-set*, if all faults involving only the β-elements of S are non-

critical. An R-set is *maximal* if it is not a proper subset of another R-set.

Every set of k or fewer β-elements in a k-FT β-network is an R-set. However, a k-FT β-network may have an R-set S containing more than k β-elements. Hence the set of all maximal R-sets of a β-network provides more detailed information about the fault tolerance of a β-network than its fault-tolerance parameter k. Every cyclic β-network possesses a class of maximal R-sets which, as is shown next, can easily be characterized.

Lemma 4.3: Consider a cyclic β-network with generator h containing n β-elements labeled $v_0, v_1, \ldots, v_{n-1}$. Every set $S = (v_{i+1}, v_{i+2}, \ldots, v_{i+n-h})$ of n-h consecutive vertices in the corresponding β-graph is a maximal R-set of the β-network.

Proof: Let G_S be the subgraph of the β-graph G generated by S, i.e., G_S consists of all the vertices of S and all the edges adjacent to S. From the definitions of a cyclic β-network and the set S, we know that G_S must not contain any circuits. Assume that S is not an R-set. There must exist a fault f involving only the β-elements of S which is critical and disconnects G into two component graphs G_1 and G_2. Both G_1 and G_2 are Eulerian graphs, hence each must contain a circuit.

Let T be the set of vertices of the β-graph not in S. All the vertices of T are consecutive and fault-free, hence they must all remain connected in the faulty β-graph induced by f. Therefore the vertices of T must all belong to one of the two component graphs G_1 and G_2. If G_1 contains all the vertices of T, then G_2 must contain all the vertices of S. As stated earlier, G_2 contains a circuit, which implies there exists at least one circuit among the vertices of S. This contradicts the fact that G_S contains no circuits. Consequently S must be an R-set. S is a maximal R-set because S constitutes the largest possible set of consecutive vertices that produces a G_S containing no circuits. □

Lemma 4.3 implies that no fault involving β-elements contained in any set of n-h consecutive β-elements can be critical. This lemma is used later in the study of certain fault-tolerant cyclic β-networks.

In the foregoing discussion of cyclic β-networks, only unlabeled β-graphs are used. Consequently, in this analysis, we can equate the term cyclic β-network with a BG-equivalence class of cyclic β-networks, i.e., all the cyclic β-networks having the same unlabeled β-graph. There are n distinct BG-equivalence classes of cyclic β-networks with n β-elements. Each class consists of 2^{2n} distinct network realizations, i.e., 2^{2n} distinct labelings of

terminal and nonterminal edges. Since most of our analysis of β-networks is based solely on the unlabeled β-graph, all networks in the same BG-equivalence class possess the same fault-tolerance characteristics. Henceforth, we refer to a BG-equivalence class of β-networks as simply a β-network.

4.3 MAXIMALLY FAULT-TOLERANT β-NETWORKS

In Section 4.1 we showed that the largest possible value for the fault-tolerance parameter k is n-1, where n is the number of β-elements. In this section the results of Section 4.2 are used to prove the existence and uniqueness of a class of cyclic β-networks which meets this upper bound on k.

Definition 4.4: The cyclic β-networks with n β-elements and generator h = 1 are called *double parallel ring β-networks*, or *DPR-networks*, of order n.

Figure 4.10 illustrates two DPR-networks of order six. Both networks have six β-elements. All the edges in Figure 4.10c are terminal edges, whereas only six edges in Figure 4.10d are terminal edges. Nevertheless both networks possess the same unlabeled β-graph and belong to the same BG-equivalence class. In this section we refer to the class of BG-equivalent DPR-networks of order n as

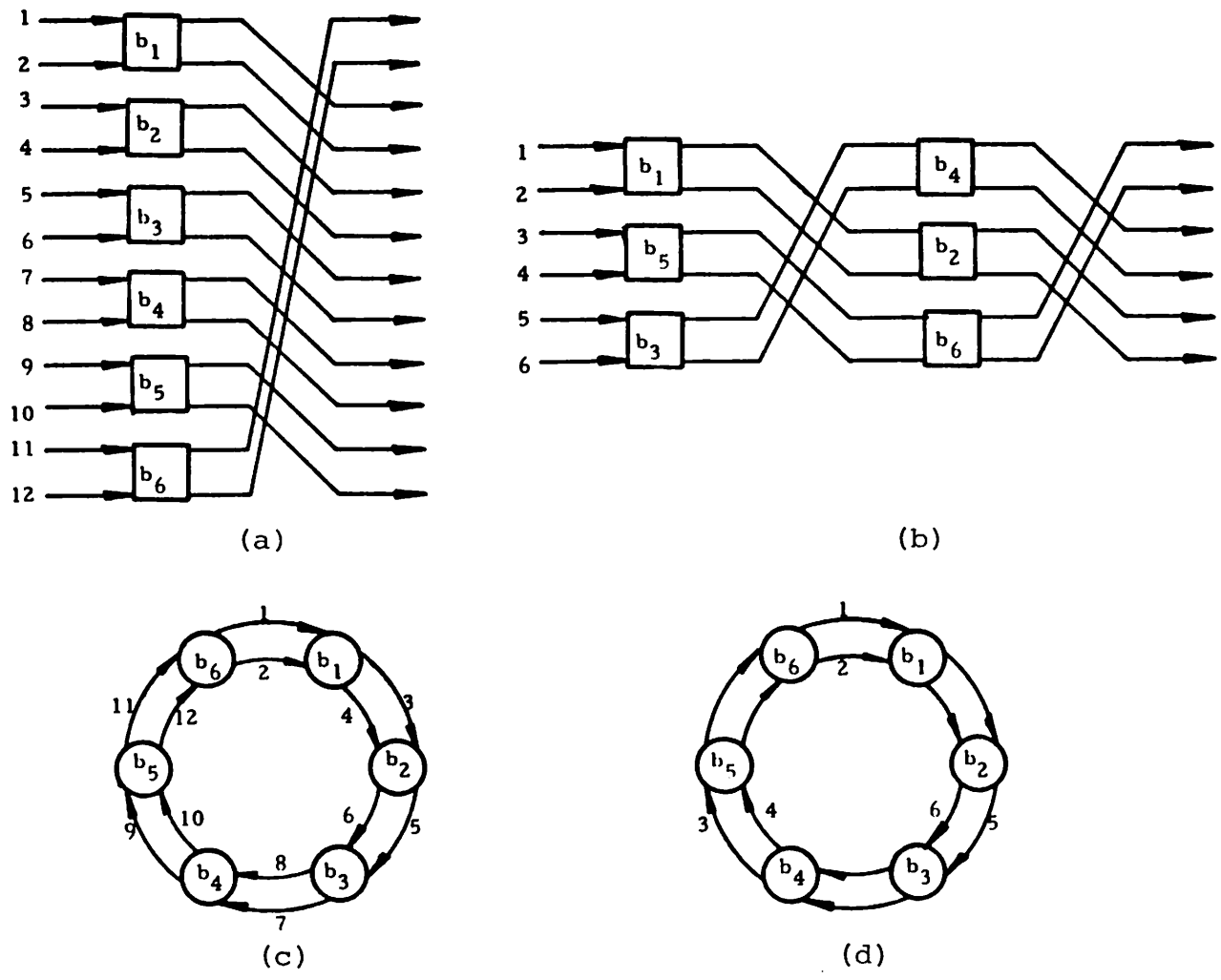


Figure 4.10. Two BG-equivalent DPR-networks: (a) a singel-stage DPR-network of order 6; (b) a two-stage DPR-network of order 6; (c) β-graph of (a); (d) β-graph of (b).

simply the DPR-network of order n, because our analysis is based on the unlabeled β-graph.

The DPR-networks not only possess interesting fault tolerance characteristics, they are also useful in practice. As stated in the previous section, one major shortcoming of the single-loop ring system is its vulnerability to hardware failures. In an effort to combat this problem other ring networks have been proposed which are able to tolerate failures. One such network, called the D-ring or double-ring network [Pierce 1977], makes use of a second or redundant loop to tolerate faults in the network. The DPR-network can be used to implement the D-ring structure.

4.3.1 Fault Tolerance of DPR-networks

Using Lemma 4.3 from the previous section we now determine the fault tolerance of DPR-networks.

Theorem 4.2: The FT parameter k of the DPR-network of order n is n-1.

Proof: Let G be the β-graph of the DPR-network of order n. We know that the DPR-network is a cyclic β-network with generator h = 1. Hence from Lemma 4.3, every set S of n-1 consecutive vertices of G is a maximal R-set. Since G has n vertices, there are exactly n distinct maximal R-sets. There are n possible ways of selecting n-1

vertices from among n vertices. Hence the n maximal R-sets are all the possible subsets of n-1 elements. Consequently any subset of n-1 vertices in G constitutes a maximal R-set, which means that any fault involving n-1 or fewer β-elements cannot possibly be critical. Therefore the DPR-network of order n must be (n-1)-FT. □

Theorem 4.2 can also be proven by examining the MCFs of the DPR-network. Figure 4.11 shows a DPR-network of order n, along with its β-graph and CA-graph. From this we see that there is no elementary circuit of length less than n in the β-graph of the DPR-network of order n. Each circuit partition of the β-graph must consist of exactly two elementary circuits each of length n. Hence every CA-graph of Figure 4.11b must be isomorphic to Figure 4.11c. In fact, there exist exactly 2^{n-1} distinct circuit partitions of Figure 4.11b, each having a CA-graph isomorphic to Figure 4.11c. Clearly each CA-graph has only one cut-set consisting of all its n edges. Hence every MCF consists of n faulty β-elements. In fact, every circuit partition corresponds to an MCF. Therefore all n β-elements must be stuck at T or X to destroy the DFA property of the DPR-network. Consequently, the DPR-network of order n is (n-1)-FT.

From Theorem 4.1 we know that n-1 is a tight upper bound for k in β-networks with n β-elements. The DPR-

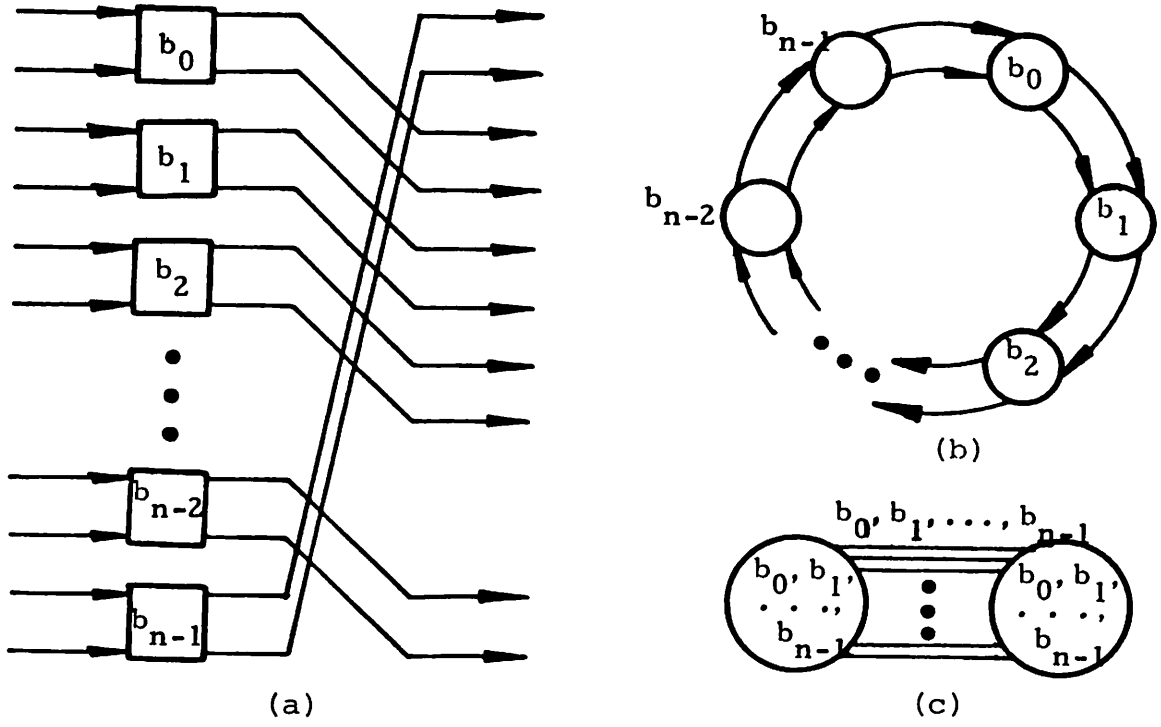


Figure 4.11. The general DPR-network: (a) a DPR-network of order n; (b) its β-graph; (c) its CA-graph.

network clearly meets this upper bound. We can therefore say that the DPR-network of order n is maximally fault tolerant among all β-networks with n β-elements. Thus in a DPR-network only one fault-free β-element is needed to ensure DFA. The maximal fault tolerance of the DPR-network can be viewed from yet another perspective which is developed in the following subsection.

4.3.2 Eulerian-Circuit Interpretation

The fault tolerance of a β-network can be analyzed in terms of the number of Eulerian circuits in its β-graph. We know that a fault is noncritical if and only if it is compatible with an Eulerian circuit (Theorem 3.2). Eulerian circuits represent maximal noncritical states of a β-network. Intuitively, we expect the number of Eulerian circuits to be proportional to the degree of fault tolerance.

A classical result in graph theory [Aardenne-Ehrenfest 1951] states that the number of Eulerian circuits in a β-graph is equal to the number of distinct spanning trees rooted at any one of the vertices. A rooted spanning tree of a β-graph of n vertices consists of n-1 edges all directed toward the root vertex. Since every vertex in the β-graph has two outgoing edges, either one can be chosen for inclusion in a rooted spanning tree. Hence in a β-

graph of n vertices, there are at most 2^{n-1} distinct spanning trees rooted at a particular vertex. As a result, a β-graph of n vertices has at most 2^{n-1} distinct Eulerian circuits. From the structure of the β-graph of the DPR-network, we see that all 2^{n-1} combinations of n-1 edges indeed correspond to distinct rooted spanning trees. Hence we have the following result:

Lemma 4.4: The β-graph of the DPR-network of order n has 2^{n-1} distinct Eulerian circuits, which is the maximal number possible in any β-graph of n β-elements.

We next show that being (n-1)-FT is equivalent to having 2^{n-1} Eulerian circuits. First, we need the following lemma.

Lemma 4.5: If s_1 and s_2 are two distinct Eulerian circuit states of a β-graph, then s_1 and s_2 cannot differ in only one of their entries, i.e., at least two β-elements must be in different states.

Proof: Assume that the states s_1 and s_2 are identical except in the i^{th} entry, i.e., $s_1 = (b_1, b_2, \ldots, b_i, b_{i+1}, \ldots, b_n)$ and $s_2 = (b_1, b_2, \ldots, \bar{b}_i, b_{i+1}, \ldots, b_n)$, where if $b_i = T$ (X) then $\bar{b}_i = X$ (T). We illustrate the Eulerian circuit corresponding to s_1 in Figure 4.12a highlighting the i^{th} β-element. The Eulerian circuit of s_2 is identical to that

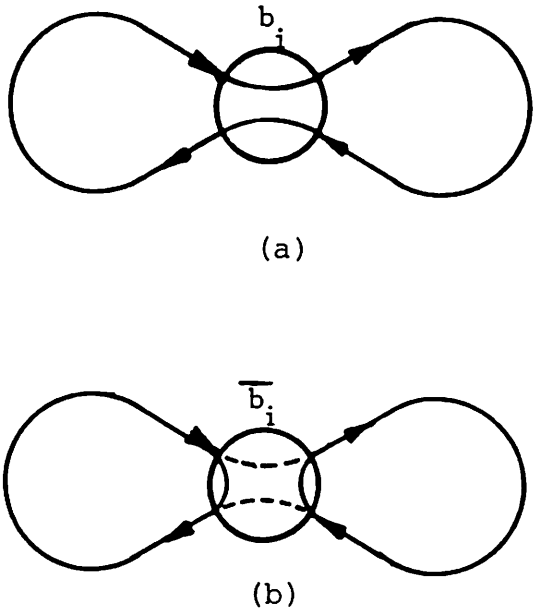


Figure 4.12. Relationship between any two Eulerian circuits: (a) the Eulerian circuit represented by the EC state $s_1 = (b_1, b_2, \ldots, b_i, \ldots, b_n)$; (b) the Eulerian circuit represented by the EC state $s_2 = (b_1, b_2, \ldots, \bar{b}_i, \ldots, b_n)$.

shown in Figure 4.12a except in the i^{th} β-element as shown in Figure 4.12b. Clearly Figure 4.12b consists of two disjoint circuits; it is therefore impossible for s_2 to represent an Eulerian circuit. Hence s_1 and s_2 must differ in more than one entry. □

Theorem 4.3: A β-network of n β-elements is (n-1)-FT if and only if its β-graph has 2^{n-1} Eulerian circuits.

Proof: The necessary condition is straightforward. If a β-network is (n-1)-FT, then any set of n-1 β-elements can be stuck in any of the 2^{n-1} possible faulty states without destroying DFA. Each of these 2^{n-1} states must be compatible with an Eulerian circuit (EC) state. Hence, there must exist 2^{n-1} distinct Eulerian circuits in an (n-1)-FT β-graph.

We now need to show that a β-graph with 2^{n-1} Eulerian circuits must be (n-1)-FT. If not, there exists a critical fault f involving n-1 β-elements. Let β-elements $b_1, b_2, \ldots, b_{n-1}$ be faulty, and let b_n be fault-free. The critical fault state of f must not be compatible with any of the 2^{n-1} EC-states. This implies that none of the 2^{n-1} EC-states has the same values as f in the first n-1 entries. There are 2^{n-1} possible distinct partial states involving only the first n-1 entries. We know that at least one of these 2^{n-1} partial states, namely, the one

corresponding to f, does not appear in the set of 2^{n-1} EC-states. Hence there can be at most $2^{n-1}-1$ distinct partial states involving the first n-1 entries of the 2^{n-1} distinct EC-states. Therefore at least two of the EC-states, s_1 and s_2, must be identical in the first n-1 entries. This means s_1 and s_2 differ in the n^{th} entry, which is impossible by Lemma 4.5. Hence if a β-graph has 2^{n-1} Eulerian circuits, the corresponding β-network must be (n-1)-FT. □

4.3.3 Uniqueness of the DPR-network

So far we have shown that the DPR-network of order n has FT parameter k = n-1 and has 2^{n-1} Eulerian circuits. We show next that these two properties are equivalent, and characterize the maximal fault tolerance achievable in a β-network of n β-elements. An interesting question is whether the DPR-network is the only β-network having this property. The answer is yes, as the following theorem asserts.

Theorem 4.4: The DPR-network of order n is unique among all β-networks of n β-elements in achieving the maximum possible fault tolerance k = n-1.

Proof: Let G be the β-graph of the DPR-network of order n. We need to show that if a β-network is (n-1)-FT

then its β-graph G′ must be isomorphic to G. If G′ is the β-graph of an (n-1)-FT β-network, it cannot contain any elementary circuits of length n-1 or less, because such an elementary circuit would correspond to a critical fault involving n-1 or fewer β-elements. Hence G′ must contain only elementary circuits of length n. Consequently, to prove that G′ and G are isomorphic, all we need to show is that for every vertex in G′, the two outgoing edges terminate at the same vertex.

Assume there exists a vertex v_i in G′ whose two outgoing edges e_j and e_k terminate at vertices v_j and v_k, respectively, and $v_j \neq v_k$, as depicted in Figure 4.13a. Since G′ contains only elementary circuits of length n, the edge e_k must belong to an elementary circuit C_k of length n. G′ has only n vertices and the length of C_k is n, therefore v_j must be a vertex in C_k, as shown in Figure 4.13b. From Figure 4.13b we see that e_j along with some of the edges in C_k constitute another elementary circuit C_j. Since $v_j \neq v_k$, the length of C_j must be less than n. This contradicts the fact that G′ only has elementary circuits of length n. Both outgoing edges of every vertex in G′ terminate at the same vertex. Hence G′ is isomorphic to G, and the DPR-network of order n is unique in achieving the maximal fault tolerance k = n-1. □

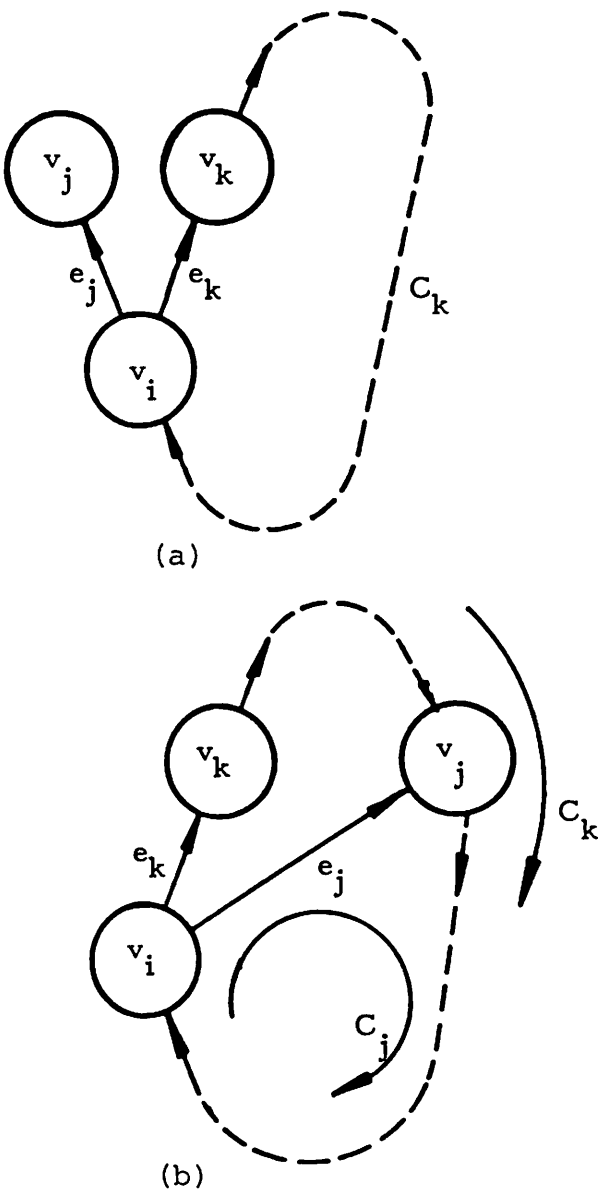


Figure 4.13. (a) Elementary circuit C_k of G';
(b) elementary circuit C_j of G'.

The results in this section can also be presented in the switching theoretic formulation described in Section 3.4. Lemma 4.5 implies that no two EC-terms can be adjacent in the Karnaugh map of the function defined by the EC-expression. Hence such a function can have at most 2^{n-1} EC-terms forming a checkerboard pattern on the Karnaugh map. This statement is equivalent to Lemma 4.4. If the EC-expression f^{ec} of a β-network has 2^{n-1} EC-terms, then every MCF-term must be a minterm of f^{ec}, and thus every MCF-state must be a completely specified state. Consequently every MCF must involve all n faulty β-elements, and thus the β-network must be (n-1)-FT. The converse, which states that being (n-1)-FT and having 2^{n-1} Eulerian circuits are equivalent, can be proven using an argument similar to that used to prove Theorem 4.3.

4.4 FAULT-TOLERANT β-NETWORKS WITH MINIMAL DELAY

In the previous section we introduced DPR-networks, and showed that they have the maximum fault tolerance k = n-1. However, it is easily seen that while a DPR-network is (n-1)-FT, it has the worst possible communication delay d = n. In this section we present a complementary class of β-networks which have the minimum communication delay $d = \lfloor \log_2 n \rfloor + 1$, and the minimum fault tolerance k = 1.

4.4.1 1-FT β-networks

In a typical repairable system diagnostic software is run periodically, and when a faulty β-element is detected it is replaced. If the rate of diagnosis is much higher than the average rate of β-element failures, then it is reasonable to assume that only single β-element failures can occur in β-networks. In light of this single-fault assumption, single-fault tolerance can be treated as a fundamental design goal. As long as single β-element faults can be tolerated and detected before a second fault occurs, DFA can be continuously maintained.

We now present a relatively simple characterization of 1-FT β-networks. For each β-element v_i of a β-network, we denote its two inputs by a_i and b_i, and its two outputs by c_i and d_i as shown in Figure 4.14a. In order for v_i to tolerate the s-a-T fault, there must exist paths in the β-graph from edges c_i and d_i back to edges b_i and a_i, respectively, as shown in Figure 4.14b. Similarly, paths from c_i and d_i to a_i and b_i, respectively, are needed for v_i to tolerate the s-a-X fault, as shown in Figure 4.14c. The foregoing reasoning leads to the following characterization.

Lemma 4.6: Let G be the β-graph of a β-network N. N is 1-FT if and only if for every vertex v_i, there exist four paths in G, one from each of the two outputs of v_i to each

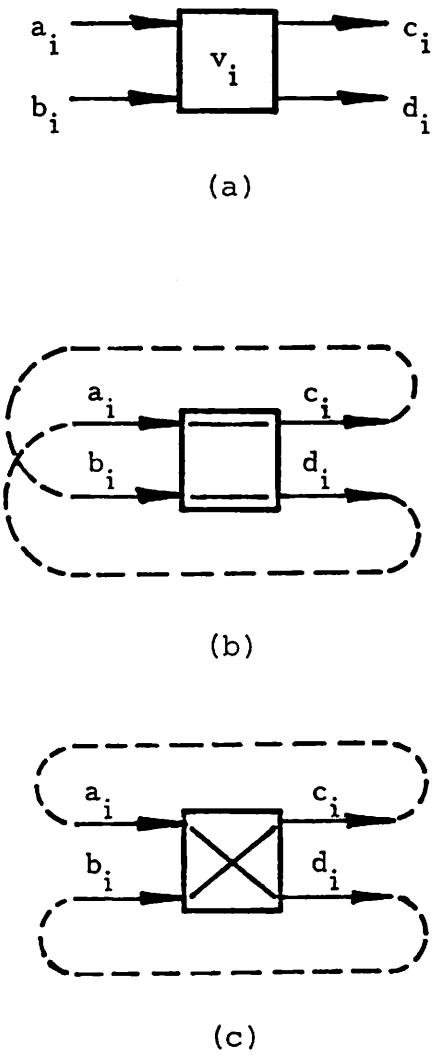


Figure 4.14. 1-FT characterization: (a) A fault-free β-element v_i; (b) tolerance of s-a-T by v_i; (c) tolerance of s-a-X by v_i.

of the two inputs of v_i.

Using this lemma, single-fault tolerance can be verified by checking for feedback paths from the outgoing edges of each vertex back to the incoming edges. If a β-network is not 1-FT, all its single critical faults can be identified in the same checking procedure. The identification and removal of single critical faults in a β-network is a basic step in fault-tolerant design. The characterization of Lemma 4.6 is straightforward, but its usefulness is limited because of the large amount of computation which may be needed to identify the feedback paths. The sufficient condition in the next lemma is more useful.

Lemma 4.7: A β-network is 1-FT if its β-graph contains a Hamiltonian circuit and no self-loops.

This lemma can be restated in the following more useful way. We call a β-element and the corresponding vertex in its β-graph *critical* if one of its stuck-at states constitutes a single critical fault.

Lemma 4.8: If the β-graph G of a β-network N contains a Hamiltonian circuit, then a vertex v_i of G and the corresponding β-element are critical if and only if there exists a self-loop at v_i.

Proof: If there is a self-loop at v_i, then it is obvious that v_i must be critical. Suppose that v_i is critical and has no self-loop. There must then exist a split of v_i which disconnects G into two components G_1 and G_2, as shown in Figure 4.15. Since G contains a Hamiltonian circuit C, one incoming and one outgoing edge of v_i must belong to C. Assume, without loss of generality, that the edges a_i and d_i defined in Figure 4.15 belong to C. There must exist a path from G_2 back to G_1 not containing v_i, in order to complete the Hamiltonian circuit C. This is impossible because it implies that v_i is not critical. If instead of a_i and d_i, a_i and c_i belong to C, then d_i and b_i must constitute a self-loop. Hence if v_i is critical, then v_i must contain a self-loop. □

Lemma 4.8 implies that critical β-elements can be easily identified in β-networks that are known to contain a Hamiltonian circuit.

4.4.2 ISE-networks

As defined in Section 2.2, an inverse shuffle exchange β-network, or ISE-network, of order n is a single stage β-network with n β-elements connected by an inverse perfect shuffle permuter. Figure 4.16 illustrates the ISE-network of order four. For convenience, we restrict our attention to ISE-networks of order $n = 2^m$, where m is an integer.

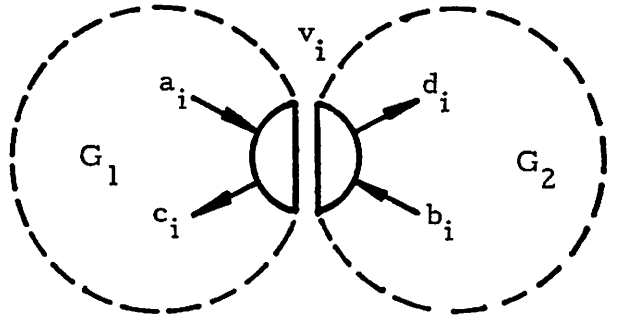


Figure 4.15. A single critical fault v_i in the β-graph G.

Each β-element in an ISE-network can therefore be designated by an m-bit binary number $b_m b_{m-1} \ldots b_1$, where $b_i \in \{0,1\}$. The top β-element is designated 00...0 and the bottom β-element is designated 11...1. Following the same convention, all the 2n links can be labeled from top to bottom by (m+1)-bit binary numbers $b_m b_{m-1} \ldots b_0$, starting from 00...0 and terminating with 11...1, as illustrated in Figure 4.16.

In the above labeling scheme each vertex $b_m b_{m-1} \ldots b_1$ has two incoming links labeled $b_m \ldots b_1 0$ and $b_m \ldots b_1 1$, and two outgoing links labeled $0\, b_m \ldots b_1$ and $1\, b_m \ldots b_1$. When β-element $b_m b_{m-1} \ldots b_1$ is in the T-state, connections are established from $b_m \ldots b_1 0$ to $0\, b_m \ldots b_1$ and from $b_m \ldots b_1 1$ to $1\, b_m \ldots b_1$. If it is in the X-state, these connections are reversed. The labels for β-elements can be translated directly into β-graphs to identify corresponding vertices. The binary (m+1)-tuple labels for β-network links can be used to label edges in the β-graph and, thereby implicitly identifying the computing units. The labeled β-graph of the ISE-network of Figure 4.16 is shown in Figure 4.17.

It has been shown that Pease's indirect binary m-cube is isomorphic to the omega network, which is actually a cascade of m stages of the ISE-network of order 2^{m-1} [Parker 1980]. For example, the $2^3 \times 2^3$ omega network in Figure 2.20 is a cascade of three identical ISE-networks of order $2^{3-1} = 4$. We know from Pease's work [Pease 1977]

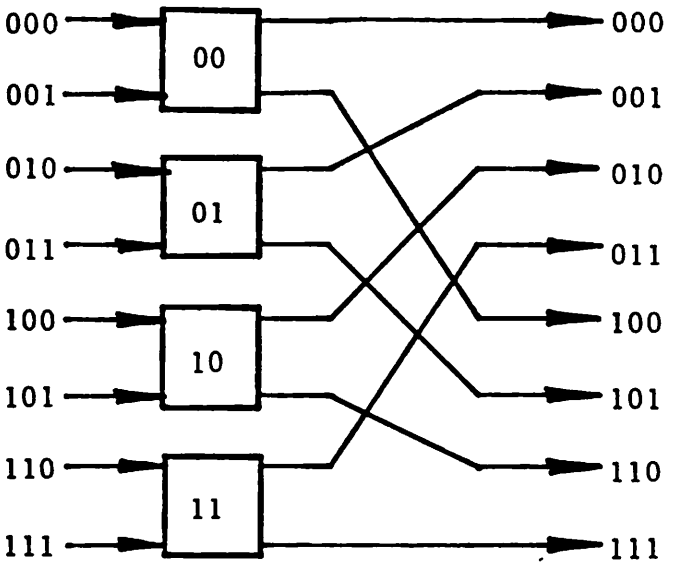


Figure 4.16. The ISE-network of order four.

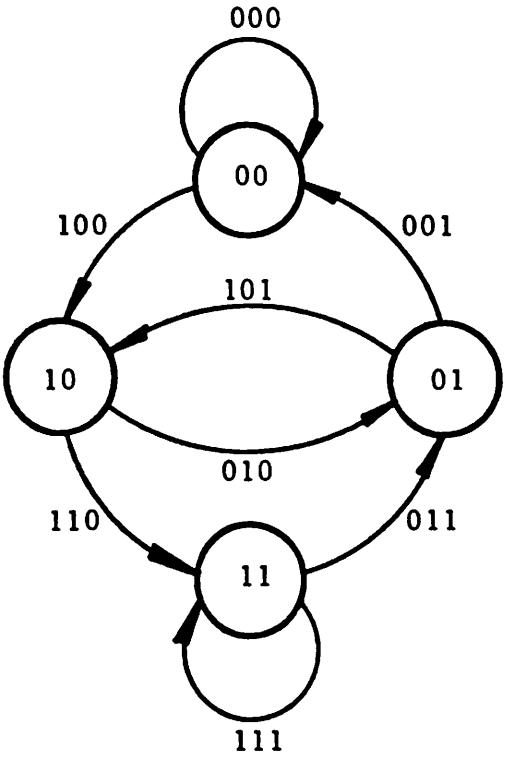


Figure 4.17. Labeled β-graph of the ISE-network of order four.

that the indirect binary m-cube has the full access property, that is, every input terminal of the network can reach any output terminal via one pass through the network. By doing a space-to-time transformation, the i^{th} stage of the indirect binary m-cube can be mapped onto the i^{th} pass through the ISE-network of order 2^{m-1}. Hence if an input terminal of an indirect binary m-cube can reach any one of the output terminals in m stages, then any input terminal of the ISE-network of order 2^{m-1} should be able to reach any other terminal within the distance m. The communication-delay parameter d of the ISE-network of order 2^{m-1} must therefore be m or less. In other words, for the ISE-network of order $n = 2^m$, $d \leq \log_2 n+1$. Since we know that $\log_2 n+1$ is the smallest possible value of d for any β-network (Lemma 4.1), the CD parameter of the ISE-network of order n must be $\log_2 n+1$. Consequently the ISE-network of order n has the minimal communication delay among the β-networks of n β-elements.

It is easy to see that the ISE-network of order n is 0-FT. Both the top and bottom β-elements contain self-loops; by Lemma 4.7 these self-loops constitute single critical faults. The foregoing discussion leads to the following theorem.

Theorem 4.5: The FT and CD parameters of the ISE-network of order $n = 2^m$ for some integer m, are $k = 0$ and $d = \log_2 n+1$, respectively.

The minimal communication delay of ISE-networks makes them very desirable for systems requiring very fast communication. In addition, ISE-networks require very simple control algorithms [Pease 1977]. Clearly, a serious drawback of ISE-networks is their lack of fault tolerance. In the next section we propose a modified ISE-network which is fault tolerant and still possesses the minimum communication delay.

4.4.3 Modified ISE-networks

In Section 4.3 we showed that the DPR-network of order n is maximally fault tolerant and is unique. If the ISE-network of order n were similarly unique in achieving minimal delay, then a fault-tolerant β-network of order n with minimal delay would not exist. Fortunately, ISE-networks are not unique in achieving minimal delay. In this subsection we describe a modified ISE-network which is 1-FT and still has the same minimal communication delay.

In order to make an ISE-network 1-FT, we must first identify all its critical β-elements. We know that the β-elements labeled 00...0 and 11...1 in the β-graph have self-loops, and thus are critical; see Figure 4.17. We first demonstrate that these are the only β-elements that are critical, and therefore need to be modified.

In the context of the mathematical treatment of shift register sequences, Good [Good 1946] and de Bruijn

[de Bruijn 1946] introduced an important graph. A *Good-de Bruijn graph* of order m, G_m, is a directed graph with 2^m vertices representing the 2^m distinct binary m-tuples. A directed edge leads from vertex v_i to v_j in G_m, if the m-tuple v_j is a successor of the m-tuple v_i, i.e., v_j can be obtained from v_i by a single cyclic shift operation. For example, the m-tuple $a_1a_2...a_m$ has two successors, $a_2...a_m0$ and $a_2...a_m1$ and two predecessors, $0\,a_1...a_{m-1}$ and $1\,a_1...a_{m-1}$. The next lemma is a direct consequence of the definitions of the ISE-network of order $n = 2^m$ and the Good-de Bruijn graph of order m.

Lemma 4.9: The β-graph of the ISE-network of order $n = 2^m$ is isomorphic with the Good-de Bruijn graph G_m of order m.

A shift-register sequence of maximum length 2^m corresponds to a Hamiltonian circuit in the Good-de Bruijn graph G_m. Good has proven the existence of such maximum-length sequences [Good 1946]. Combining this fact with Lemma 4.9 we obtain the following result.

Lemma 4.10: The β-graph of the ISE-network of order $n = 2^m$ contains a Hamiltonian circuit.

Lemmas 4.8 and 4.10 together confirm that the top and bottom β-elements are the only critical β-elements in the ISE-network of order n. In order to make the network 1-FT,

the two self-loops associated with these β-elements must be removed.

Sowrirajan and Reddy have recently investigated the design of a fault-tolerant β-network [Sowrirajan and Reddy 1980]. They studied a class of rearrangeable Clos networks called C_2 networks. Their fault-tolerance criterion is the maintenance of rearrangeability. They showed that by adding one redundant β-element, a C_2 network can be made 1-FT with respect to rearrangeability. Employing a similar approach we can easily make the ISE-network 1-FT with respect to the DFA by adding a redundant β-element. Figure 4.18 illustrates a 1-FT version of the ISE-network of order four. During fault-free operation the redundant β-element b_r is set to the T-state which makes the modified network isomorphic to the original network. When either β-element 00 or 11 is stuck-at-T, b_r can be set to the X-state to maintain DFA. With the introduction of a redundant β-element, additional hardware and delay are also introduced. We next describe another modification method for ISE-networks which makes the networks 1-FT but uses no redundant β-element and incurs no additional delay.

Definition 4.5: The *modified ISE-network*, or *MISE-network*, of order n is an ISE-network of order n with two of its links altered as follows. The top output from β-element 00...0 is connected to link 11...1 instead of to link

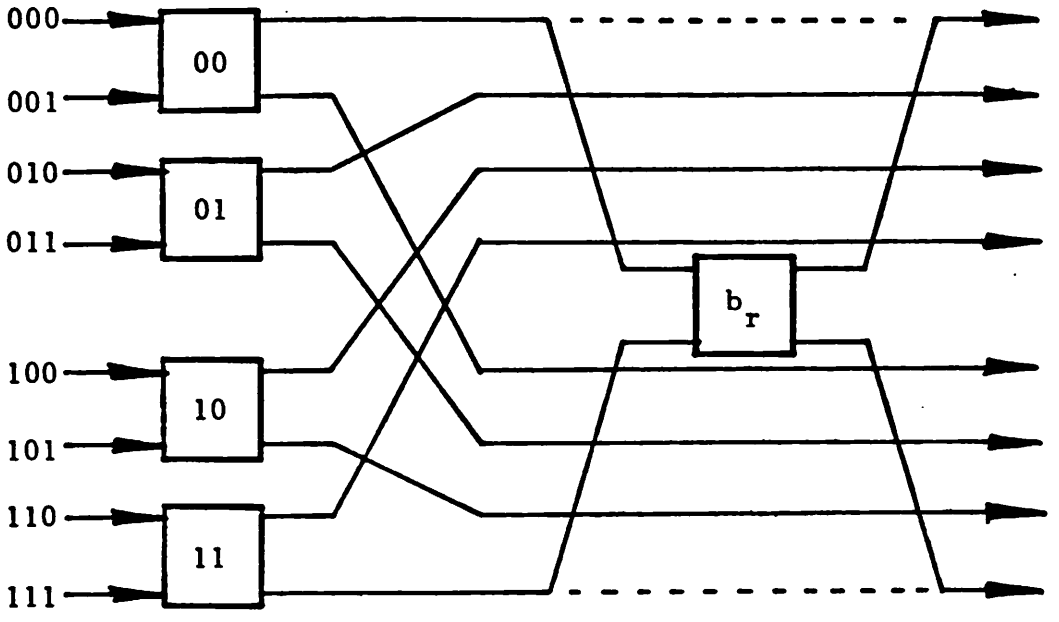


Figure 4.18. The 1-FT redundant ISE-network of order four.

00...0. Similarly, the bottom output of β-element 11...1 is connected to link 00...0 instead of to link 11...1.

Basically in the MISE-network, the destinations of the two original self-loop links are exchanged. Figure 4.19 illustrates the MISE-network of order eight. We now show that the MISE-network of order n is 1-FT and still possesses the communication delay $d = \log_2 n+1$.

Lemma 4.11: The MISE-network has FT parameter $k = 1$.

Proof: The MISE-network is obtained be deleting the two original self-loops from the ISE-network. These self-loops could not have been part of a Hamiltonian circuit in the ISE-network. Hence any such Hamiltonian circuit must still exist in the MISE-network. The MISE-network thus contains a Hamiltonian circuit and has no self-loops, therefore, by Lemma 4.7 it is 1-FT. □

Lemma 4.12: The MISE-network of order $n = 2^m$, for some integer m, has CD parameter $d = \log_2 n+1$.

Proof: If G is the β-graph of an ISE-network and G′ is the β-graph of the corresponding MISE-network, then we must show that the edge diameter of G′ is the same as that of G. Let edges a′ and b′ in G′ be the modified edges of the self-loops a and b in G. Let v_0 and v_{2^m-1} denote the

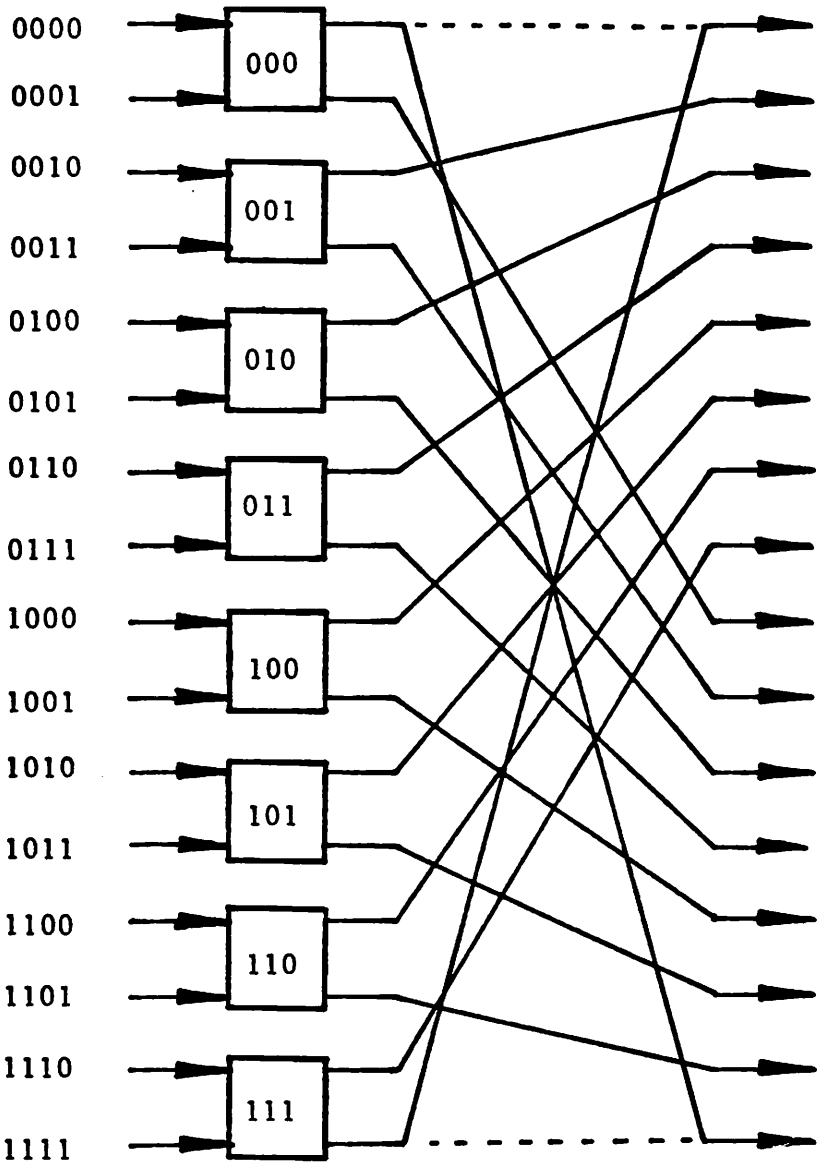


Figure 4.19. The MISE-network of order eight.

source vertices of a′ and b′ respectively. Let the other four edges adjacent to v_0 and v_{2^m-1} be as denoted in Figure 4.20. G′ differs from G only in the edges a′ and b′. We know that the delay in G is $\log_2 n+1$. To show that the delay in G′ is also $\log_2 n+1$, all we need to show is that a′ and b′ can reach and be reached by all other edges in G′ within the distance $\log_2 n+1$.

Any edge in G must reach a and b via d and e, respectively. If a and b are reachable from any other edge within the distance $\log_2 n+1$, then a′ and b′ must also be reachable from any other edge within the distance $\log_2 n+1$. Edges b and a can reach any other edge of G via edges f and c, respectively within the distance $\log_2 n+1$. Hence a′ (b′) in G′ must be able to do the same via edge f (c). Consequently the CD parameter of the MISE-network of order n is the same as that of the ISE-network of order n, namely $d = \log_2 n+1$. □

Combining Lemmas 4.11 and 4.12 we obtain the following result.

Theorem 4.6: The FT and CD parameters of the MISE-network of order $n = 2^m$ for some integer m, are $k = 1$ and $d = \log_2 n+1$, respectively.

The MISE-networks are fault-tolerant β-networks with minimal communication delay. We have thus synthesized a

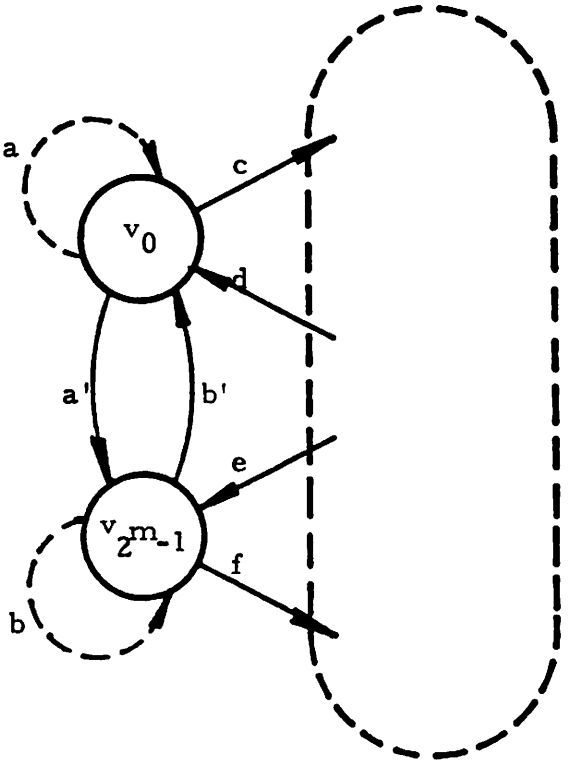


Figure 4.20. A portion of the β-graph of a MISE-network.

fault-tolerant β-network by modifying a non-fault-tolerant β-network. This was accomplished without adding extra β-elements or increasing communication delay. The simple control algorithm used for ISE-networks needs to be modified only very slightly for the MISE-networks, as we now show.

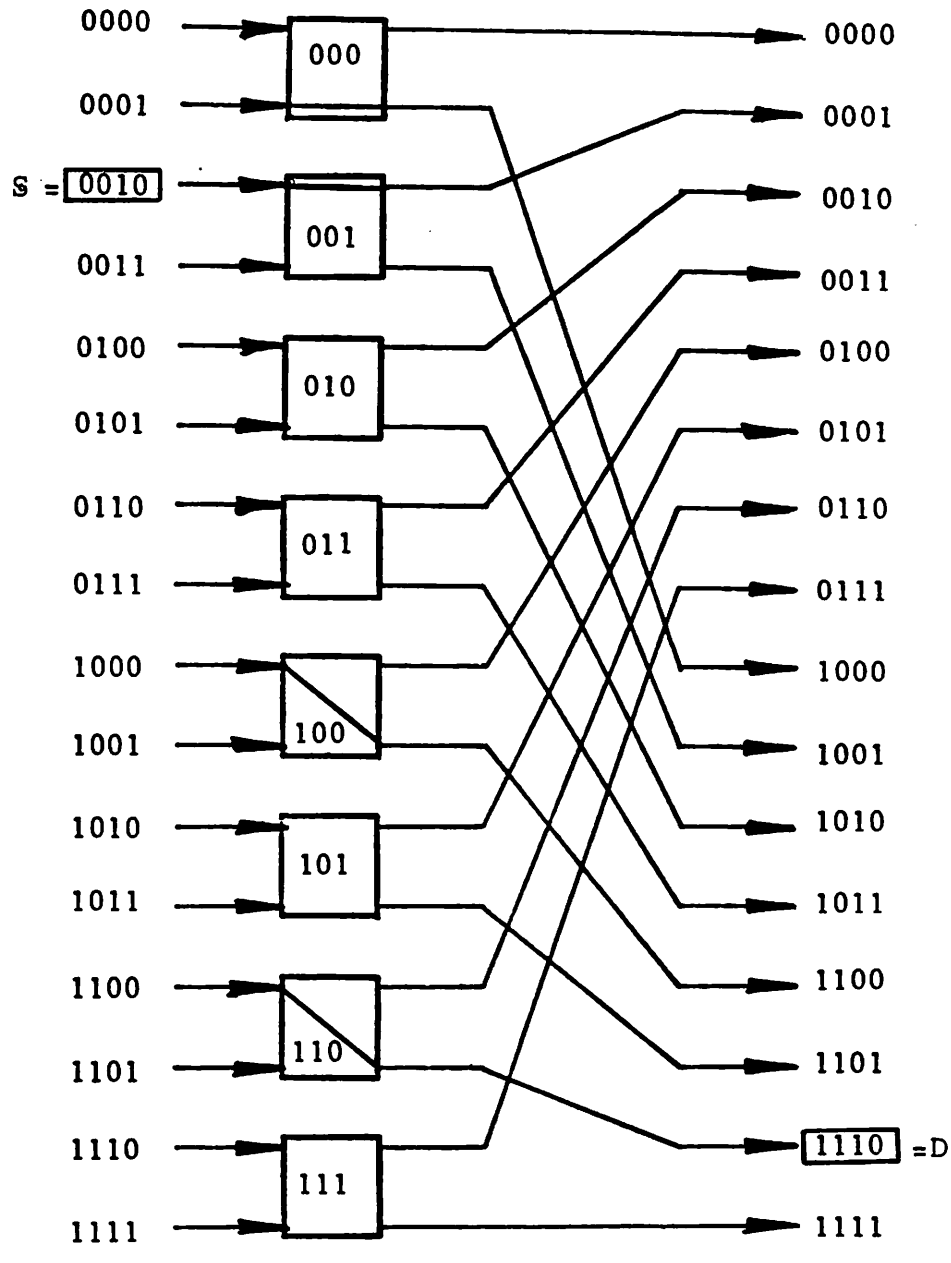
A simple algorithm exists for routing data from a source computing unit to a destination computing unit of an ISE-network. Computing units are implicitly modeled by terminal links. Every link in an ISE-network can reach any other link in exactly $\log_2 n+1$ passes through the network. To establish a route from a source link to a destination link, the states of $\log_2 n+1$ β-elements, one for each pass, must be specified. This specification is made by an (m+1)-bit binary number called a *destination tag*. In the ISE-network of order $n = 2^m$, to provide a route from source link $S = s_m \ldots s_1 s_0$ to destination link $D = d_m \ldots d_1 d_0$, a destination tag $g = g_m \ldots g_1 g_0$ is needed, where $g_i = s_i \oplus d_i$ for $i = 0,1,\ldots,m$. Bit g_i specifies the state to which the β-element being traversed in the $(i+1)^{th}$ pass must be set. If $g_i = 0$ (1) then the β-element is set to the T (X) state. For example, in the ISE-network of order $8 = 2^3$, the destination tag needed for establishing a route from 0010 to 1110 is $0010 \oplus 1110 = 1100$. By tracing through the β-network we see that the four β-elements traversed by data going from 0010 to 1110 are 001, 000, 100, and 110. Since the

destination tag is 1100, these four β-elements are set to the states T,T,X,X, respectively, to establish the required route, as shown in Figure 4.21.

For MISE-networks the same control algorithm is still valid except when the destination computing unit or link is either 00...0 or 11...1. In such cases, the original destination tag must be complemented, and any leading (high-order) zeros must be eliminated. For example, if S = 1101 and D = 0000 in a MISE-network, then $S \oplus D = 1101$. The actual destination tag needed is the complement of 1101 with all leading zeros deleted, namely --10. Hence the data from 1101 can reach 0000 in just two passes through the MISE-network. The β-element traversed in the first pass is set to the T-state while the one traversed in the second pass is set to the X-state. If, for example, S′= 1100 and D′= 1111, then $S' \oplus D' = 0011$. The destination tag needed is the complement of 0011, i.e., 1100. The two connecting routes discussed above are illustrated in Figure 4.22.

4.5 FAMILIES OF β-NETWORKS

We define a *family* of β-networks as a collection of β-networks having similar network structures, but differing in size. For example, the collection of all ISE-networks is a family of β-networks. The order of an ISE-



S = 0010
⊕ D = 1110
1100
T : (001)
T : (000)
X : (100)
X : (110)

Figure 4.21. Establishing a connection from 0010 to 1110 in the ISE-network of order eight.

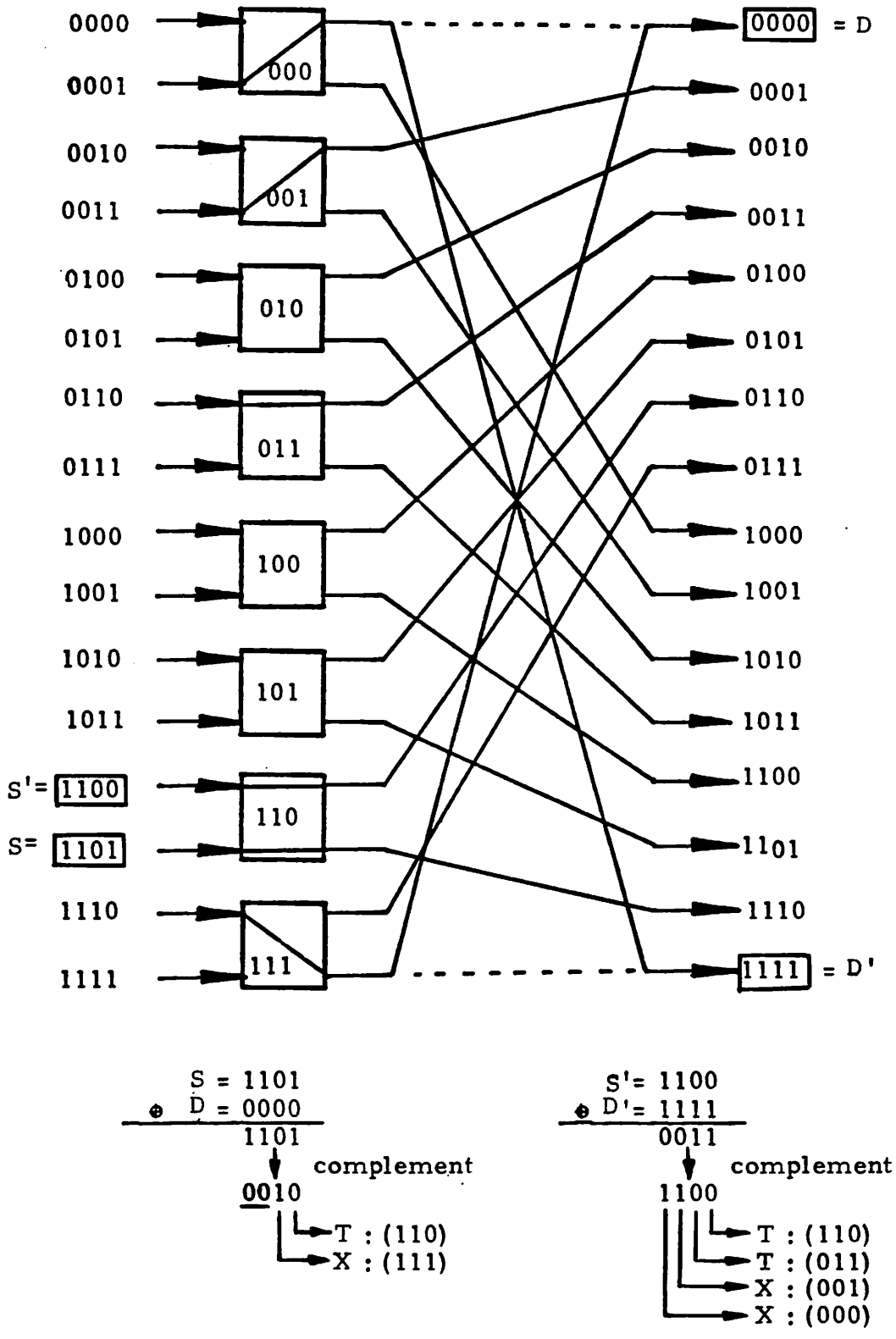


Figure 4.22. Establishing the two connections 1101 to 0000 and 1100 to 1111 in the MISE-network of order eight.

network, which is the number of β-elements it contains, indicates the size of the network and identifies a specific member of the ISE family. Frequently, the FT and CD parameters of a family of β-networks can be expressed as well-defined functions of a suitable size parameter. We will refer to these functions as the *FT function* and the *CD function*, respectively, of the family in question.

For many families, the number n of β-elements used is the most convenient size parameter. The FT function k(n) then denotes the maximum number of β-elements in a network of n β-elements whose failure does not destroy DFA. The CD function d(n) denotes the maximum number of β-elements separating any pair of links in the network. If a family has FT function k(n) and CD function d(n), we call it an *(n,k(n),d(n))-family*. A specific member of the family with FT parameter k and CD parameter d can be called an *(n,k,d)-network*. For example, the family of ISE-networks is an $(n,0,\log_2 n+1)$-family, and the family of MISE-networks is an $(n,1,\log_2 n+1)$-family. The collection of all DPR-networks can be considered to be an (n,n-1,n)-family.

We now derive FT and CD functions for several interesting families of cyclic β-networks, namely the cyclic β-networks with generators 0, 1, n/2, and n-1. The cyclic β-networks with generator h = 0 are called the *single cycle shift β-networks* or *SCS-networks*; see Figure 4.23a. It

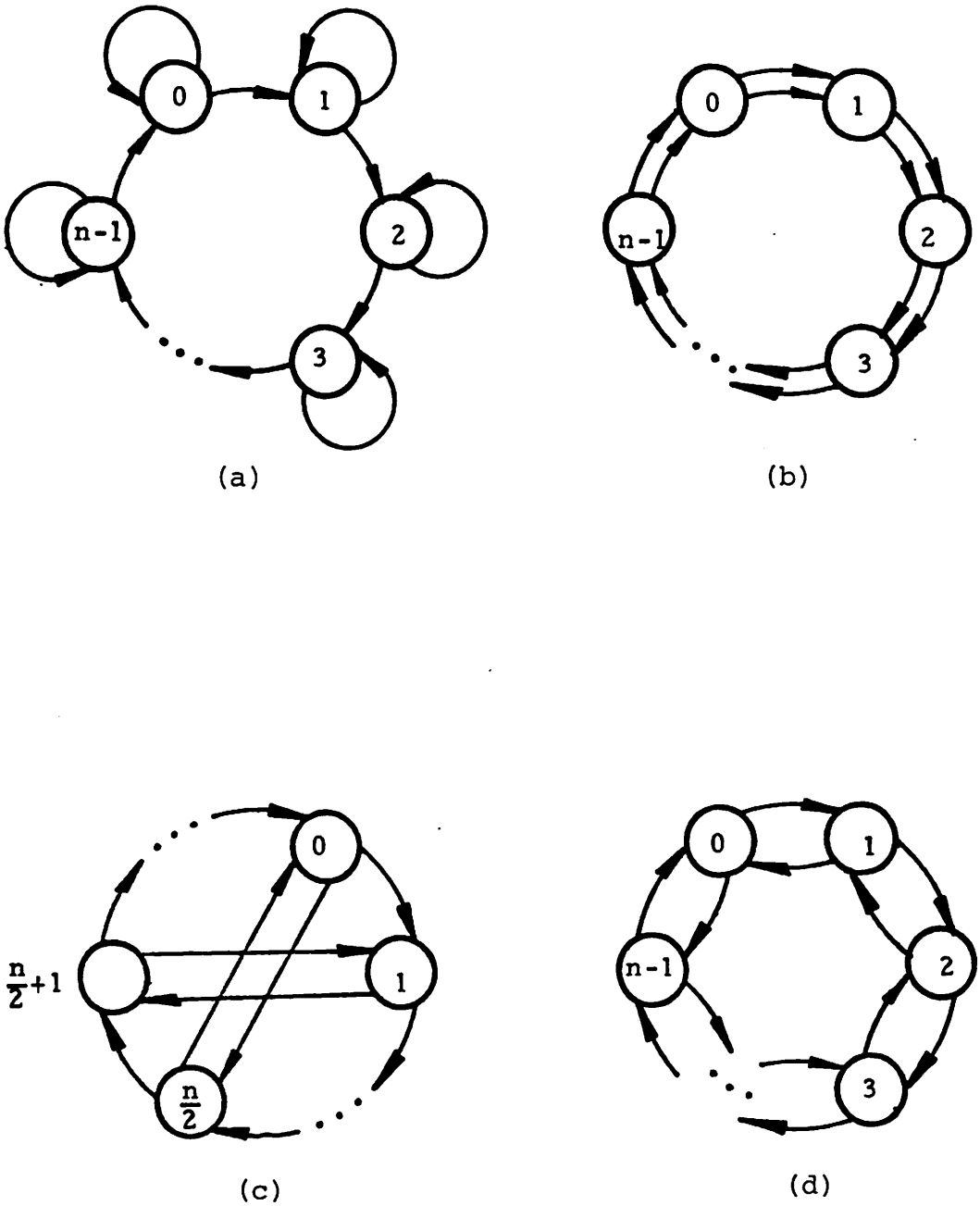


Figure 4.23. β-graphs of four cyclic β-networks: (a) the SCS-network or order n; (b) the DPR-network of order n; (c) the MCC-network of order n; (d) the DRR-network of order n.

has been shown in Section 4.1 that the CD parameter of the SCS-network of order n is n. Because of the presence of self-loops, the FT parameters of the SCS-network of order n is 0. Hence SCS-networks are (n,0,n)-networks.

The cyclic β-networks with generator h = 1 form the family of DPR-networks. As was shown in Section 4.3, DPR-networks are (n,n-1,n)-networks. In the case where the generator h = n/2 and n is even, the cyclic β-network structure of Figure 4.23c is obtained. We call this family the *maximal-chord cyclic β-networks*, or the *MCC-networks*. The edges associated with the generator h can be viewed as chords of the Hamiltonian circuit in a cyclic β-network. When h = n/2, the chords achieve the maximal length and actually become diameters of the Hamiltonian circuit. The FT parameter of the MCC-network of order n is one because it has no self-loops and it contains critical double faults. Because of the availability of the chord edges, the MCC-network of order n has CD parameter d = n/2+1. Hence MCC-networks are (n,1,n/2+1)-networks.

Another interesting family of cyclic β-networks has generator h = n-1. We call these the *double reverse ring β-networks*, or *DRR-networks*; see Figure 4.23d. The DRR-network is identical to the DPR-network, except that the direction of one of the rings is reversed. Consequently, the DRR-networks have rather different FT and CD functions.

Again using Lemma 4.7, we can conclude that the DPR-network of order n has FT parameter $k = 1$. By examination of the network structure we see that the DRR-network of order n has CD parameter $d = \lfloor n/2 \rfloor + 1$. Hence the DRR-networks are $(n, 1, \lfloor n/2 \rfloor + 1)$-networks. While the structure of the DRR-networks resembles that of the DPR-networks, the FT and CD functions of the DRR-networks are almost identical to those of the MCC-networks. They are identical if n is even.

We summarize the foregoing results in Table 4.1. The corresponding parameters of the ISE-networks and the MISE-networks are also included in this table. The families of β-networks we have discussed thus far involve various design tradeoffs which are illustrated in Figure 4.24. The shaded area of the figure represents the feasible design space, and is obtained from the bounds stated in Theorem 4.1. In Chapter 5, we investigate several well-known families of β-networks and obtain their FT and CD functions. These results will contribute several more data points to Figure 4.24. We take as our goal in the design of β-networks to maximize k while keeping d relatively small. Hence we can loosely say that good designs of β-networks will tend to fall into the upper left corner of the shaded area in Figure 4.24.

Table 4.1. Summary of parametric functions of six families of β-networks.

β-Network Family	Fault Tolerance k(n)	Communi-cation Delay d(n)	Generator h (Cyclic Cases Only)
SCS (Single Cycle Shift)	0	n	0
DPR (Double Parallel Ring)	n-1	n	1
MCC (Maximal Chord Cyclic)	1	$n/2 + 1$	$n/2$
DRR (Double Reverse Ring)	1	$\lfloor n/2 \rfloor + 1$	n-1
ISE (Inverse Shuffle Exchange)	0	$\lfloor \log_2 n \rfloor + 1$	--
MISE (Modified ISE)	1	$\lfloor \log_2 n \rfloor + 1$	--

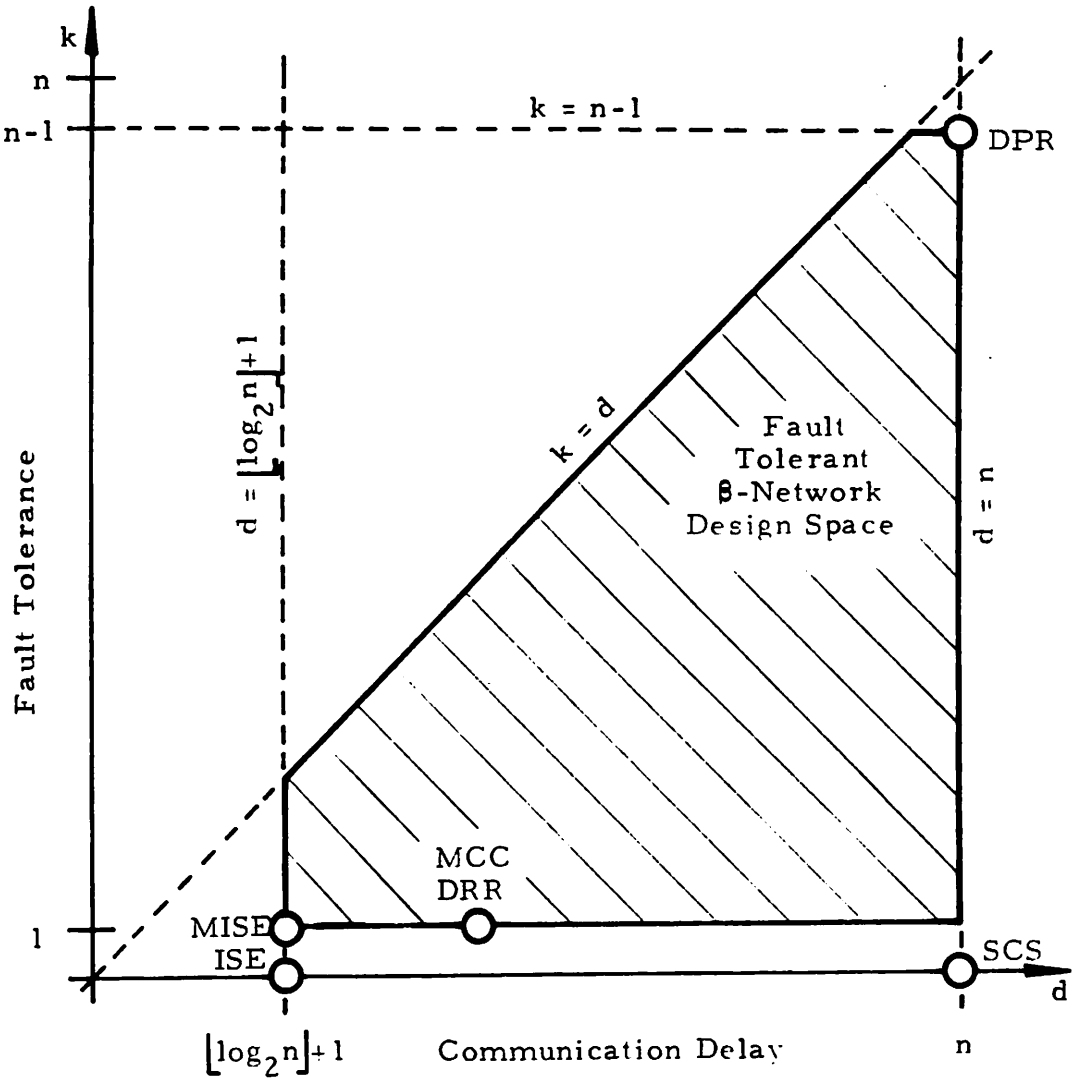


Figure 4.24. β-network design space with respect to k and d; comparison among networks in Table 4.1.

CHAPTER 5

CASE STUDIES

Several families of β-networks have been extensively studied. In this chapter we apply our theoretical results to the analysis of these β-networks. The structure of each family is formally described. FT and CD functions are derived for each family. In Section 5.1 a number of analysis and synthesis techniques which will aid the study of complex β-networks are developed. Each of the subsequent sections analyzes a particular family of β-networks.

5.1 GENERAL ANALYSIS AND SYNTHESIS TECHNIQUES

Many practical complex β-networks are constructed by systematically interconnecting a collection of smaller β-networks. By knowing the properties of the subnetworks and their interconnections we often can draw conclusions about the entire network. This is the approach taken in this section. We first review some of the concepts introduced in Chapter 2. Two β-networks N_1 and N_2 are BG-equivalent if they have the same unlabeled β-graph. Two β-networks are isomorphic if they are BG-equivalent and have the same labeling of terminal links. The number of

terminal links is called the dimension of the β-network. Two isomorphic β-networks must have the same dimension, but two BG-equivalent β-networks can have different dimensions.

Theorem 5.1: If two β-networks N_1 and N_2 are BG-equivalent then N_1 has DFA if and only if N_2 has DFA.

Proof: Let G_1 (G_2) be the unlabeled β-graph of N_1 (N_2). A β-network has DFA if and only if its β-graph is strongly connected. Hence N_1 (N_2) has DFA if and only if G_1 (G_2) is strongly connected. Since N_1 and N_2 are BG-equivalent, G_1 must be identical to G_2, and hence N_1 has DFA if and only if N_2 has DFA. □

This theorem tells us that DFA is independent of the specific labeling of terminal links, i.e., all links can be considered as intermediate or terminal links. The DFA property of a β-network can be checked by inspecting any other β-network in its BG-equivalence class.

Theorem 5.2: If N_1 is a residual network of N with respect to a state s, i.e., $N_1 = N/s$, and N_1 has DFA, then N must also have DFA.

Proof: All the terminal links of N still exist in N_1. Since N_1 has DFA, there exists a connecting path between

any pair of terminal links of N_1. Since N_1 is a residual network of N, all connecting paths in N_1 exist in N. Hence there exists a connecting path between any pair of terminal links of N. Therefore N must have DFA. □

Frequently, a residual network of a β-network has an obvious structure which facilitates fault-tolerance analysis. Theorem 5.2 allows us to analyze the residual network and draw certain conclusions about the original network. Clearly, the converse of Theorem 5.2 is not true. Every faulty β-network is a residual network of the fault-free β-network. A critical fault produces a residual β-network which does not have DFA.

Many complex multi-stage β-networks are cascades, as defined in Section 2.2. The network N shown in Figure 5.1 is a cascade of the subnetworks $N_1, N_2, \ldots, N_g$ and is denoted by $N = N_1 \circ N_2 \circ \cdots \circ N_g$.

Theorem 5.3: Let the β-network N be a cascade of g subnetworks $N_1, N_2, \ldots, N_g$. The network N has full access if any one of the subnetworks $N_1, N_2, \ldots, N_g$ has full access.

Proof: Assume that some subnetwork N_i has full access. Because a β-network realizes some permutation in every state, every output terminal t of N_g must be reachable by at least one output terminal v of N_i. N_i has full access, therefore every input terminal u of N_i must be able

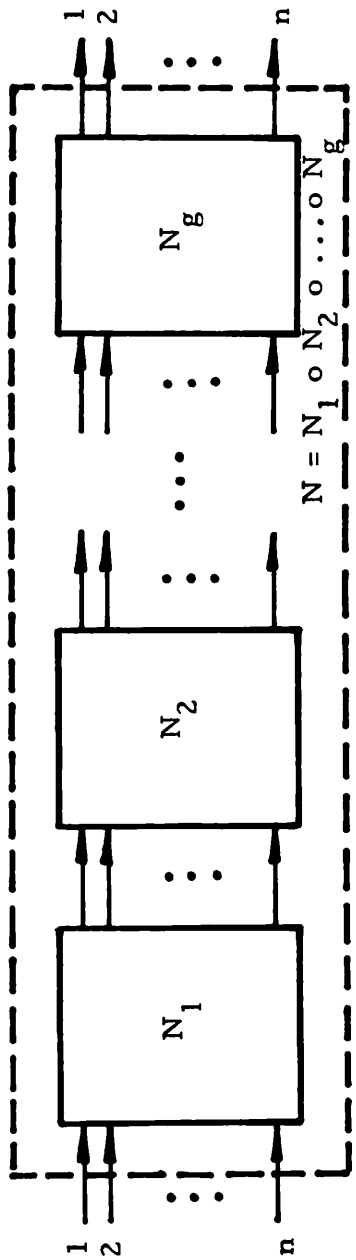


Figure 5.1. A cascaded network N of g subnetworks $N_1, N_2, \ldots, N_g$.

to reach, or be connected to, any output terminal of N_i. Hence every input terminal of N_i can reach any output terminal of N_i, implying that every input terminal of N_i can reach any output terminal of N_g. Since every input terminal of N_1 must be able to reach at least one input terminal of N_i, every input terminal of N_1 must be able to reach every output terminal of N_g. Therefore N must have full access. □

Clearly any β-network N having full access must have DFA. The above theorem says that if N is a cascade of subnetworks then any one of the subnetworks having full access will guarantee full access and DFA for N. A direct corollary to this theorem is the following.

Corollary 5.1: If $N = N_1 \circ N_2 \circ \cdots \circ N_g$ and some N_i for $1 \leq i \leq g$ has full access, then the β-elements in all the other subnetworks constitute an R-set of N.

For example, if N_1 has full access, then the β-elements in $N_2, N_3, \ldots, N_g$ can all be faulty and still not destroy DFA in N. Since all faulty β-networks are residual β-networks, any faulty subnetwork N′ of N has DFA if N′ contains a subnetwork which still possesses full access.

We use I to denote the *identity connection* $\begin{pmatrix} t_1 & t_2 \cdots & t_n \\ t_1 & t_2 \cdots & t_n \end{pmatrix}$ in which every terminal t_i is connected to itself via the network N. We say a network N *contains* the identity connection I if there exists a complete state s

of N that realizes the identity connection. I will also be used to represent the residual network N/s.

Theorem 5.4: Let $N = N_1 \circ N_2 \circ \ldots \circ N_g$. If some N_i has DFA and all the other N_i's contain the identity connection I, then N must have DFA.

Proof: Assume that N_i has DFA. Let $N' = N_1 \circ N_2 \circ \ldots \circ N_{i-1}$ and $N'' = N_{i+1} \circ \ldots \circ N_g$ so that $N = N' \circ N_i \circ N''$. Since each N_i for $i = 1,2,\ldots,g$, contains I, both N' and N'' must also contain I. Let s' and s'' be complete states of N' and N'', respectively, such that $N'/s' = I$ and $N''/s'' = I$. Let s be a partial state of N that results from setting N' and N'' to the states s' and s'', respectively. Clearly $N/s = I \circ N_i \circ I = N_i$. Since N_i has DFA, N/s must have DFA. According to Theorem 5.2, N must also have DFA. □

The above theorem implies that if a β-network N has DFA and contains the identity connection I, then any cascade of multiple copies of N must also have DFA.

The theorems proven in this section will be applied to the analysis of some well-known β-networks in subsequent sections. All the networks studied in this chapter have multiple stages of β-elements. In most cases the number of stages is proportional to $\log_2 n$, where n is the dimension of the network. Hence, for convenience we

assume that all β-networks in this chapter have dimension 2^m where m is an integer. In other words, only $2^m \times 2^m$ β-networks are considered.

5.2 DOUBLE-TREE NETWORKS

The first β-networks to be considered here are called double-tree networks [Levitt et al. 1968]. A double-tree network consists of a right and a left "half." Each half of the network resembles a binary tree. The left and right trees are mirror images of each other, and each pair of mirror-image β-elements is connected by a link. For example, Figure 5.2 illustrates the $2^3 \times 2^3$ double-tree network. The double-tree network has been investigated by three different research teams for three very different applications. Before we formally define the structure of the double-tree network, we briefly summarize these three applications.

5.2.1 Applications

The double-tree network was first proposed by Levitt, Green and Goldberg in their study of a class of β-networks called CPCU (complete permutation-complete utilization) networks [Levitt et al. 1968]. CPCU networks are capable of realizing all n! permutations, where n is the dimension of the network, and are very similar to Benes rearrangeable connecting networks. Levitt et al. were concerned with

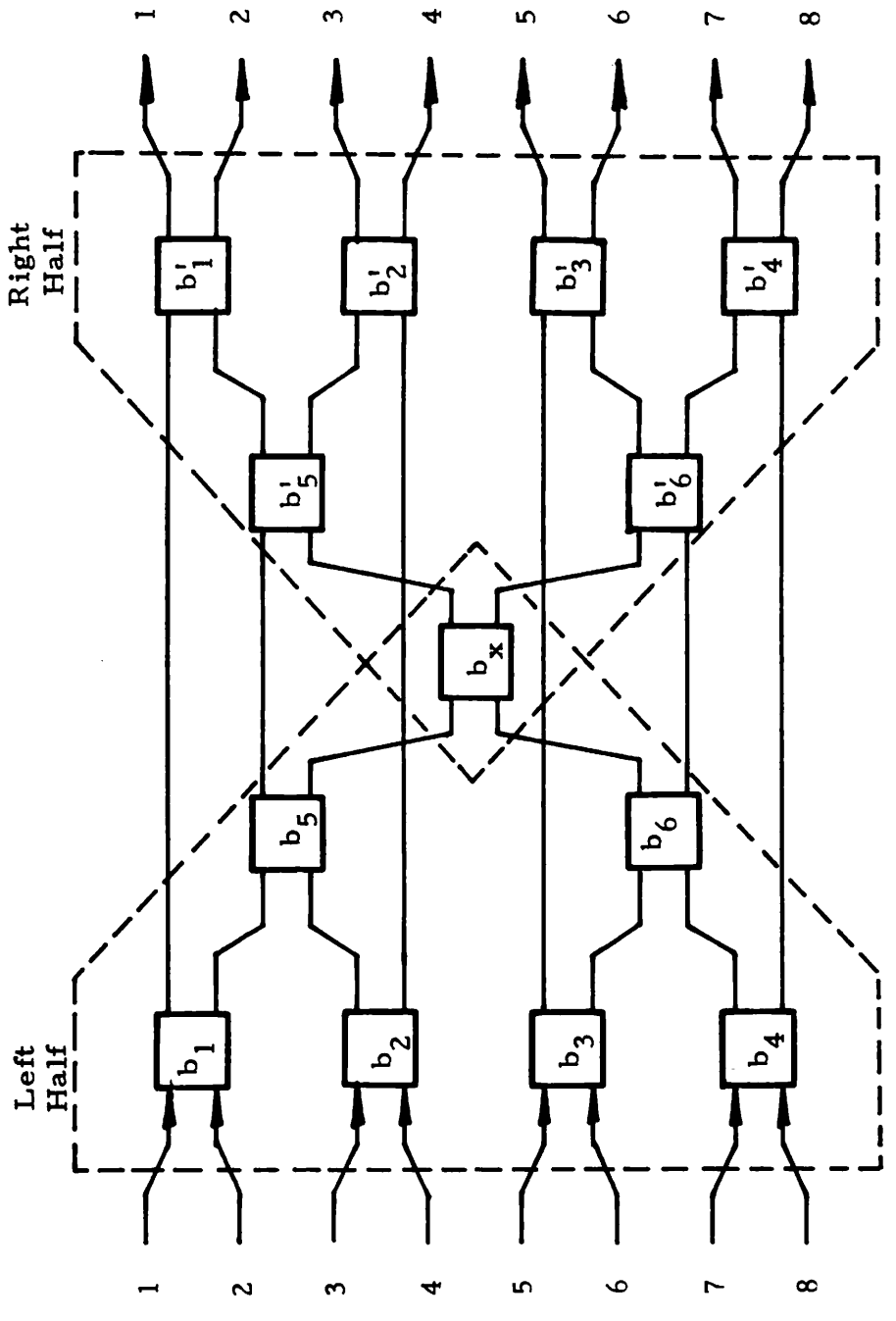


Figure 5.2. The $2^3 \times 2^3$ double-tree network, or DOT-network, $\mathcal{D}_3$.

finding ways of detecting and correcting β-element stuck-at faults in CPCU networks. The double-tree network was designed as a fault correcting network. By cascading a $2^m \times 2^m$ double-tree network with a $2^m \times 2^m$ CPCU network, a composite network similar to Figure 1.4 is obtained. This network is able to correct the effect of any single β-element stuck-at fault. A multiple-fault-correcting CPCU network can be obtained by cascading multiple copies of the double-tree network with the original CPCU network.

As discussed in Section 4.2, local communication networks for computer systems are often designed as shared bus systems or ring systems. A third possibility is to use β-networks to route data between computing units. This approach appears to require simpler hardware and have better performance than either buses or rings. One of the two classes of β-networks studied by Hopper and Wheeler for this application is the double-tree network [Hopper and Wheeler 1979]. Here the double-tree network is implemented as a packet switching network. The network accepts fixed-sized data packets at its inputs and routes them to its outputs.

The third application was suggested by Leung and Dennis for the MIT Data Flow Processor [Leung and Dennis 1980]. Unlike a conventional computer which executes a sequence of operations specified by a program, a data flow processor activates operations as soon as the needed oper-

ands become available. Figure 2.10 illustrates an implementation of the Data Flow Processor in which information is also sent as packets. The instruction cells (IC) in the activity store hold the data flow program to be executed. There are two types of packets: operation packets and result packets. Result packets containing operand values arrive at instruction cells from the distribution network. Each instruction cell sends an operation packet to the operation section when all operands have been received. The operation section, containing a set of functional units, executes the instruction and forwards the result packet to another target instruction via the distribution network. The double-tree network is one of two β-networks being considered for implementing the distribution network of the Data Flow Processor.

5.2.2 Network Structure

The structure of a $2^m \times 2^m$ *double-tree network*, or *DOT-network*, $\mathcal{D}_m$ can be defined recursively as follows. The β-element is defined as the $2^1 \times 2^1$ DOT-network. The $2^m \times 2^m$ DOT-network $\mathcal{D}_m$ is obtained by cascading a stack of 2^{m-1} β-elements to the input side and a stack of 2^{m-1} β-elements to the output side of the $2^{m-1} \times 2^{m-1}$ DOT-network $\mathcal{D}_{m-1}$ according to the following construction rules; see Figure 5.3.

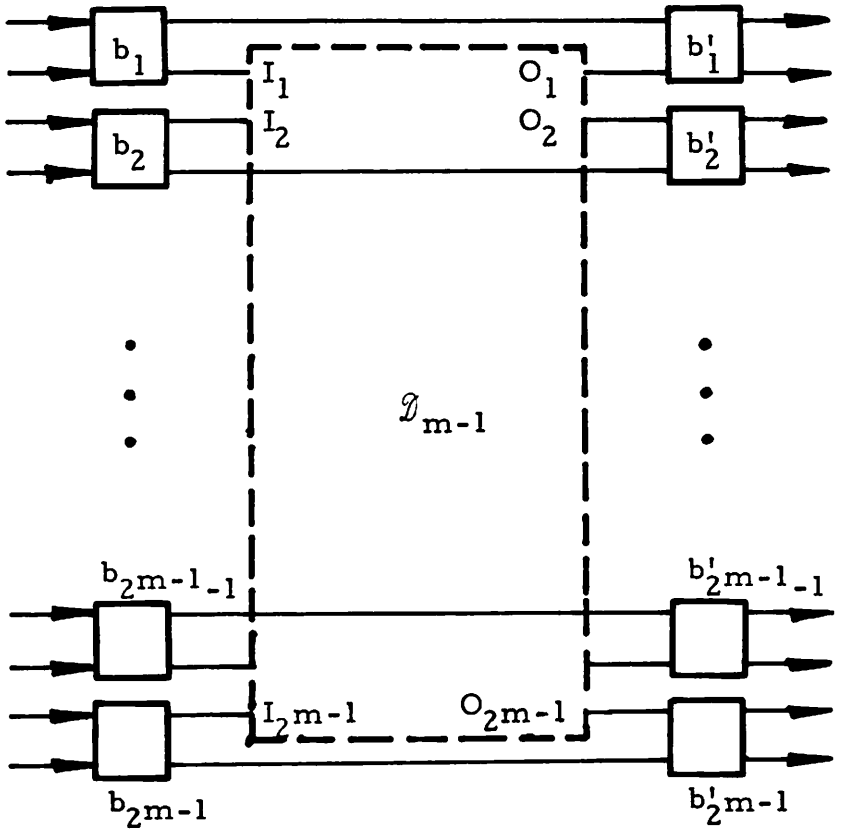


Figure 5.3. The general structure of the $2^m \times 2^m$ DOT-network $\mathcal{D}_m$.

1. Assign the labels $b_1, b_2, \ldots, b_{2^m-1}$ ($b'_1, b'_2, \ldots, b'_{2^m-1}$) to the β-elements to be cascaded to the input (output) side of $\mathcal{D}_{m-1}$.

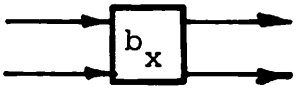
2. Connect one output of the new β-element b_j to input link I_j of $\mathcal{D}_{m-1}$ for $j = 1,2,\ldots,2^{m-1}$.

3. Connect one input of the new β-element b'_j to output line O_j of $\mathcal{D}_{m-1}$ for $j = 1,2,\ldots,2^{m-1}$.

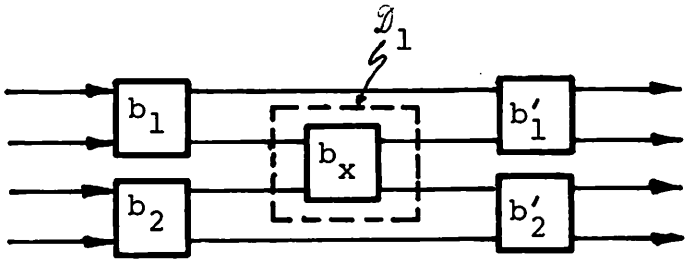
4. Connect the remaining output of b_j to the remaining input of b'_j.

The inputs of the b_j's and the outputs of the b'_j's are the inputs and outputs, respectively, of $\mathcal{D}_m$. Figure 5.4 depicts $\mathcal{D}_1$ and $\mathcal{D}_2$.

In general, $\mathcal{D}_m$ has $2^{m+1}-3$ β-elements, and has 2m-1 stages of β-elements, with stage i and stage 2m-i each having exactly 2^{m-i} β-elements, for $i = 1,\ldots,m$. For example, $\mathcal{D}_3$ as shown in Figure 5.2 has 5 stages and stages 1, 2, 3, 4 and 5 have 4, 2, 1, 2 and 4 β-elements respectively. The single β-element in the middle, i.e., stage m of $\mathcal{D}_m$, is called the center β-element, and is denoted b_x. Based on the construction of $\mathcal{D}_m$, a vertical symmetry and a horizontal symmetry can be identified in the network structure of $\mathcal{D}_m$. The left half and the right half of $\mathcal{D}_m$ are symmetrical with respect to a vertical axis passing



(a)



(b)

Figure 5.4. Examples of DOT-networks: (a) $2^1 \times 2^1$ DOT-network, $\mathcal{D}_1$; (b) $2^2 \times 2^2$ DOT-network, $\mathcal{D}_2$.

through b_x. Furthermore, the upper half and the lower half of $\mathcal{D}_m$ are symmetrical with respect to a horizontal axis passing through b_x. The symmetrical and recursive structure of the DOT-network is illustrated in Figure 5.5.

5.2.3 FT and CD Functions

In this subsection we obtain the FT and CD functions for the DOT family. Since DOT-networks are multiple-stage β-networks, their terminal delay (TD) parameters are not equal to their CD parameters. Hence we also derive the TD function for the DOT family. Recall from Chapter 4, that the CD parameter of a network is equal to the longest distance between any two edges in the β-graph, whereas the TD parameter is equal to the longest distance between any two terminal edges. Hence the TD parameter is always less than or equal to the CD parameter. Clearly, the terminal delay of a full-access β-network with t stages is equal to t. This fact is very useful in the derivation of TD functions.

Lemma 5.1: The $2^m \times 2^m$ DOT-network $\mathcal{D}_m$ has TD parameter $t = 2m-1$ and CD parameter $d = 4m-3$.

Proof: Every input terminal of $\mathcal{D}_m$ can reach the center β-element b_x, and b_x can reach every output terminal. Hence $\mathcal{D}_m$ has full access. Since $\mathcal{D}_m$ has 2m-1

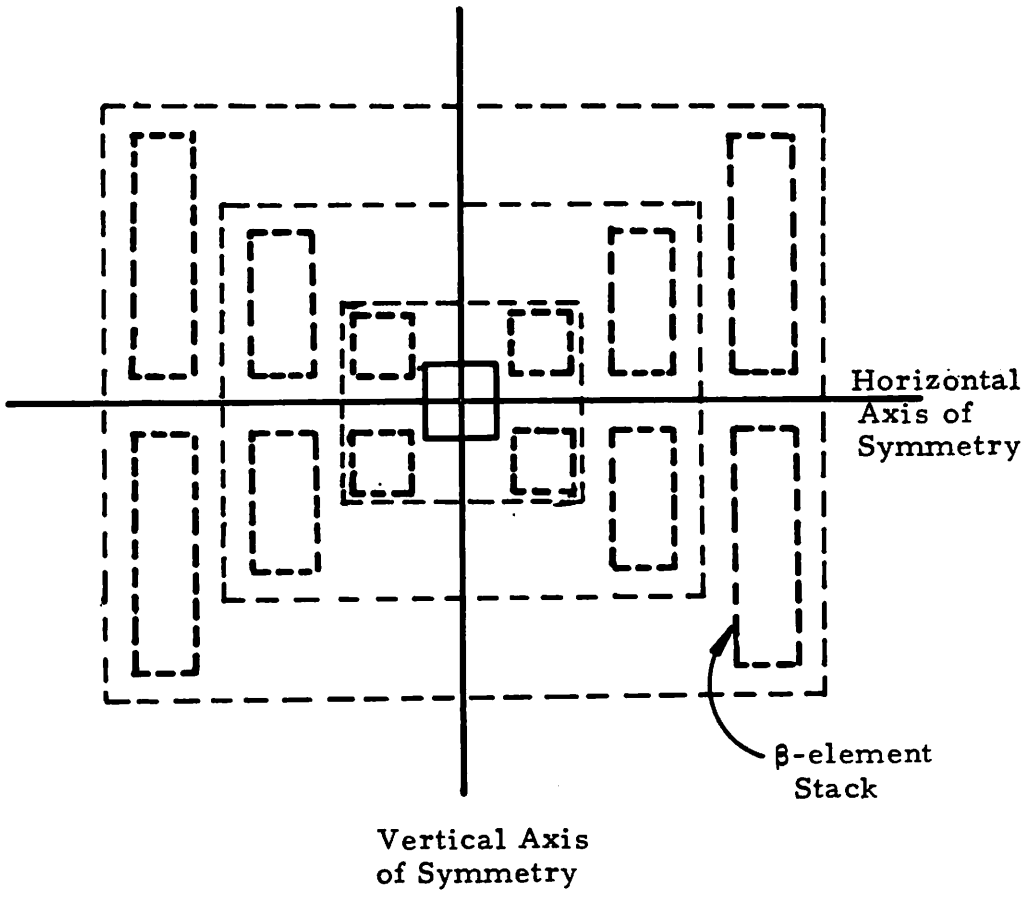


Figure 5.5. Horizontal and vertical symmetry of the DOT-network.

stages, the TD parameter must be $t = 2m-1$.

Let the two input links of b_x be a_1 and a_2, and the two output links of b_x be e_1 and e_2, as shown in Figure 5.6. The links e_1 and e_2 can reach every terminal link of $\mathcal{D}_m$ via exactly m-1 β-elements, and can reach every link of $\mathcal{D}_m$ via 2(m-1) or fewer β-elements. On the other hand, every link in $\mathcal{D}_m$ can reach e_1 and e_2 via 2m-1 or fewer β-elements. Hence the distance between any two links is at most $2(m-1) + 2m-1 = 4m-3$.

To prove that $d = 4m-3$, we must show that there exist two links in $\mathcal{D}_m$ separated by the distance 4m-3. The distance from e_1 back to a_1 is equal to 2m-2, and the distance from a_1 to a_2 is equal to 2m-1. Since the upper and lower halves of $\mathcal{D}_m$ are joined by the center β-element b_x, the shortest path from e_1 to a_2 must include a_1. Hence the distance from e_1 to a_2 is $2m-2+2m-1 = 4m-3$. Similarly the distance from e_2 to a_1 is also 4m-3. Therefore the CD parameter of $\mathcal{D}_m$ is $d = 4m-3$. □

It is easily seen that the center β-element b_x of $\mathcal{D}_m$ is critical. If b_x is s-a-T, then the network will be split into disjoint upper and lower halves. Hence the FT parameter of $\mathcal{D}_m$ is $k = 0$. To summarize the foregoing results we have the following theorem.

Theorem 5.5: Let $\mathcal{D}_m$ denote the $2^m \times 2^m$ DOT-network. The FT parameter of $\mathcal{D}_m$ is $k = 0$. The CD parameter of $\mathcal{D}_m$ is

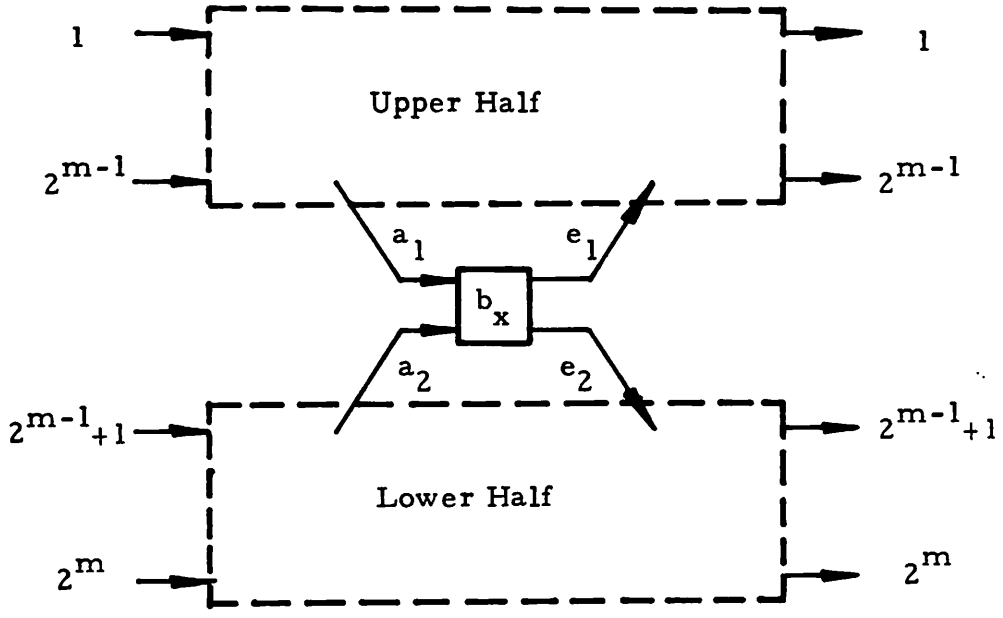


Figure 5.6. The upper and lower halves of a DOT-network.

$d = 4m-3$. The TD parameter of $\mathcal{D}_m$ is $t = 2m-1$. Since the number of β-elements in $\mathcal{D}_m$ is $n = 2^{m+1}-3$, the DOT-networks are $(2^{m+1}-3, 0, 4m-3)$-networks.

5.2.4 Modified DOT-networks

We now propose a modification of the DOT-network to make it fault tolerant. We know that the center β-element b_x is critical; it can easily be shown to be the only critical β-element. We can remove the only single critical fault of b_x s-a-T by simple deleting the center β-element b_x. The *modified DOT-network*, or *MDOT-network*, $\mathcal{X}_m$ is identical to the DOT-network $\mathcal{D}_m$ except that the center β-element b_x is permanently set to the X-state, i.e., link a_1 is connected to e_2 and link a_2 is connected to e_1, as shown in Figure 5.7 for $\mathcal{D}_3$. We can now replace the former links e_2 and e_1 by a_1 and a_2 respectively.

Like the original DOT-network, the MDOT-network has full-access. However, the number of stages, and hence the TD parameter, has been reduced by one to $t = 2m-2$. The number of β-elements has also been reduced by one to $n = 2^{m+1}-4$. Using the argument given in the proof of Lemma 5.1, it can be shown that the CD parameter for the $2^m \times 2^m$ MDOT-network is $d = 4m-4$, which is also one less than the CD parameter of the DOT-network. Thus the MDOT-network actually has better delay characteristics than the DOT-network.

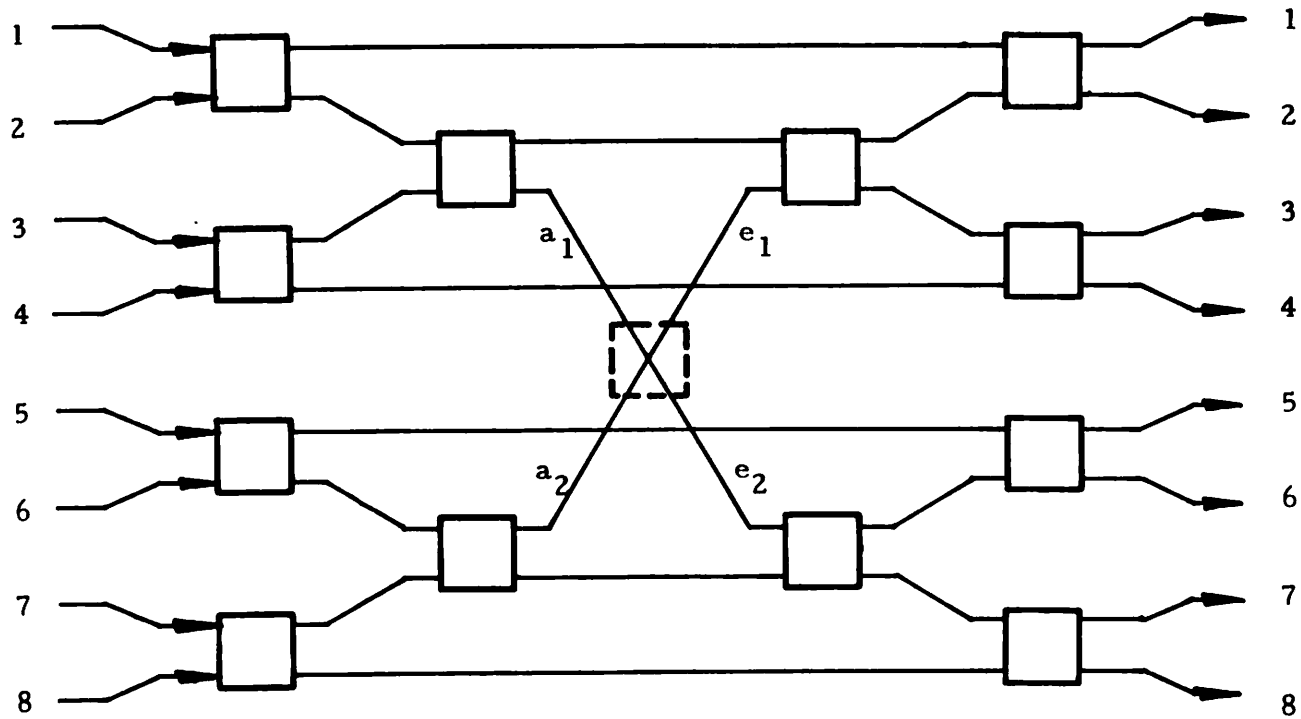
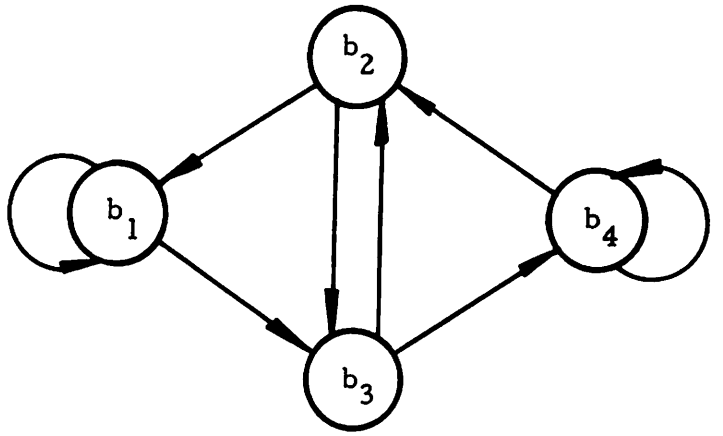


Figure 5.7. The $2^3 \times 2^3$ modified DOT-network, or MDOT-network, $\mathcal{X}_3$.

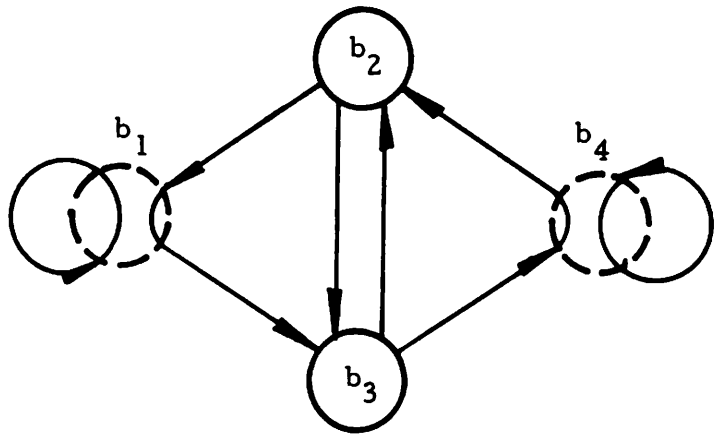
Lemma 5.2: The $2^m \times 2^m$ MDOT-network $\mathcal{X}_m$ has TD parameter $t = 2m-2$ and CD parameter $d = 4m-4$.

We next prove that the MDOT-network is indeed 1-FT. First some preliminary results must be derived. It has been shown in Chapter 4 that the most fault tolerant β-network structure is that of the DPR-network (double parallel ring β-network). There are networks which are not DPR-networks but contain subnetworks that have the DPR-network structure. Given a β-network N with a set B of n β-elements, a *DPR-set* D of order p of N is a subset of B consisting of p β-elements such that the following is true. There exists a residual network N′ of N which contains a component comprising only the β-elements of D, and whose β-graph is BG-equivalent to that of the DPR-network of order p. For example, the set of β-elements b_2 and b_3 of the 8 × 8 ISE-network is a DPR-set of order two. By setting both b_1 and b_4 to the T-state, a DPR-network of order two consists of b_2 and b_3 is formed, as depicted in Figure 5.8. Since a DPR-set of order p exhibits the DPR-network structure, the failure of any p-1 of its β-elements does not destroy DFA. As a direct consequence of the fault tolerance property of the DPR-network, we have the following theorem.

Theorem 5.6: Every proper subset of a DPR-set of a β-network N constitutes an R-set of N.



(a)



(b)

Figure 5.8. Example of a DPR-set: (a) β-graph of the 8 × 8 ISE-network; (b) a residual β-graph containing the DPR-set b_2 and b_3.

Corollary 5.2: A β-network N is k-FT if every subset of k β-elements belongs to a DPR-set of order greater than k.

To prove that $\mathcal{X}_m$ is 1-FT, it suffices to show that every β-element in $\mathcal{X}_m$ belongs to a DPR-set of order two or more. We know that the MDOT-network is symmetrical with respect to an imaginary vertical axis through the center of the network. Each β-element b_i has a mirror image, denoted b_i'. In an MDOT-network, every pair of β-elements $\{b_i, b_i'\}$ is called an image pair. The following lemma shows that an image pair constitutes a DPR-set of order two.

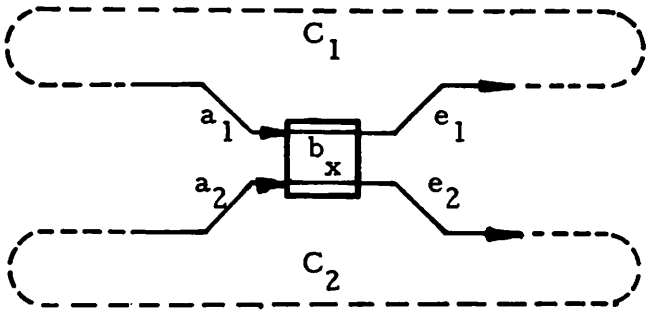
Lemma 5.3: The $2^m \times 2^m$ MDOT-network $\mathcal{X}_m$ is 1-FT.

Proof: From Corollary 5.2, it suffices to show that every image pair $\{b_i, b_i'\}$ constitutes a DPR-set of order two. Let b_i be any β-element in the left half of the network. We want to show that there exist two edge-disjoint paths from b_i to b_i', and two edge-disjoint paths from b_i' to b_i. If all the β-elements in a DOT-network $\mathcal{D}_m$ are set to the T-state, then the resultant network consists of 2^m elementary circuits. Each elementary circuit contains exactly one terminal link. All the upper or lower links of an image pair belong to the same elementary circuit. The elementary circuit containing the upper links and the elementary circuit containing the lower links constitute

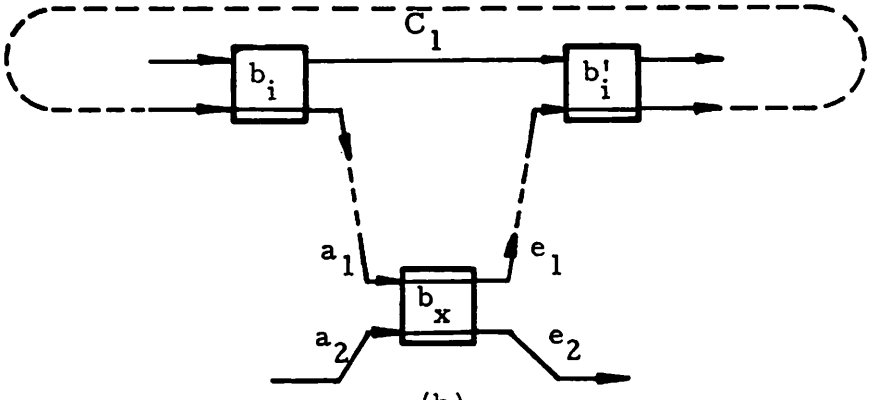
the four edge disjoint paths needed to ensure that an image pair constitutes a DPR-set of order two. In an MDOT-network $\mathcal{X}_m$ all these elementary circuits remain the same, except those containing the center β-element b_x.

An elementary circuit in $\mathcal{D}_m$ containing b_x must contain links a_1 and e_1, or links a_2 and e_2, shown in Figure 5.9a. Let C_1 and C_2 denote the elementary circuits in $\mathcal{D}_m$ containing links a_1, e_1 and a_2, e_2 respectively. The elementary circuits C_1 and C_2 do not exist in the corresponding MDOT-network $\mathcal{X}_m$. Hence the original path from b_i to b_i' via a_1 and e_1, or a_2 and e_2, no longer exists. However, in $\mathcal{X}_m$ a_1 is directly connected to e_2, and a_2 is connected to e_1, so C_2 constitutes a path from e_2 to a_2, while C_1 constitutes a path from e_1 to a_1. Hence if the original path in $\mathcal{D}_m$ from b_i to b_i' is part of C_1 as shown in Figure 5.9b, then a new path from b_i to b_i' can be formed in $\mathcal{X}_m$ from the links of the original path and the elementary circuit C_2, as shown in Figure 5.9c. Therefore for each image pair b_i and b_i', in the MDOT-network $\mathcal{X}_m$, there exist two edge-disjoint paths from b_i to b_i' and two edge-disjoint path from b_i' to b_i. Each image pair constitutes a DPR-set of order two. Hence every β-element in the MDOT-network is an R-set and the network is 1-FT. □

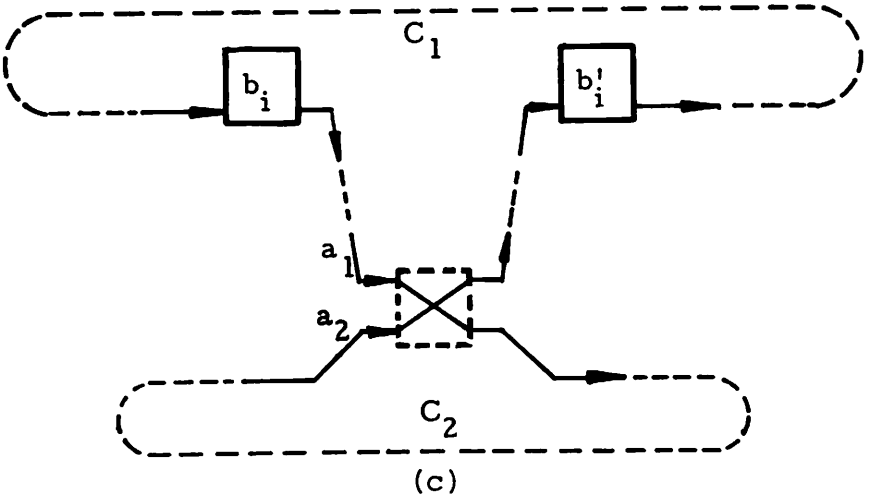
Since an MDOT-network contains many double critical faults, its FT parameter k must be one. Combining Lemmas 5.2 and 5.3 yields the following theorem.



(a)



(b)



(c)

Figure 5.9. Elementary circuits in a DOT-network: (a) elementary circuits adjacent to b_x; (b) a path from b_i to b'_i via b_x; (c) a path from b_i to b'_i in an MDOT-network.

Theorem 5.7: Let $\mathcal{X}_m$ denote the $2^m \times 2^m$ MDOT-network. The FT parameter of $\mathcal{X}_m$ is $k = 1$. The CD parameter of $\mathcal{X}_m$ is $d = 4m-4$. The TD parameter of $\mathcal{X}_m$ is $t = 2m-2$. Since the number of β-elements in $\mathcal{X}_m$ is $2^{m+1}-4$, the MDOT-networks are $(2^{m+1}-4, 1, 4m-4)$-networks.

Interestingly, we have succeeded in modifying a non-fault-tolerant β-network to make it 1-FT, while decreasing its communication delay. As might be expected, the connecting capability of the MDOT-network, in terms of the number of permutations that can be realized, is slightly less than that of the corresponding DOT-netwrok due to the absence of the center β-element. An alternate modification which does not sacrifice any connecting capability is simply to add a redundant β-element b'_x to correct the s-a-T fault of b_x, cf. Figure 4.18. We can call this design the *redundant DOT-network* or the *RDOT-network*. The $2^3 \times 2^3$ RDOT-network is illustrated in Figure 5.10. It is not difficult to verify the following result.

Theorem 5.8: The $2^m \times 2^m$ RDOT-network has $n = 2^{m+1}-2$ β-elements, FT parameter $k = 1$, CD parameter $d = 4m-2$, and TD parameter $t = 2m$.

5.3 INDIRECT BINARY m-CUBE NETWORKS

Various "cube" structures have been proposed for interconnecting large numbers of processors in computer sys-

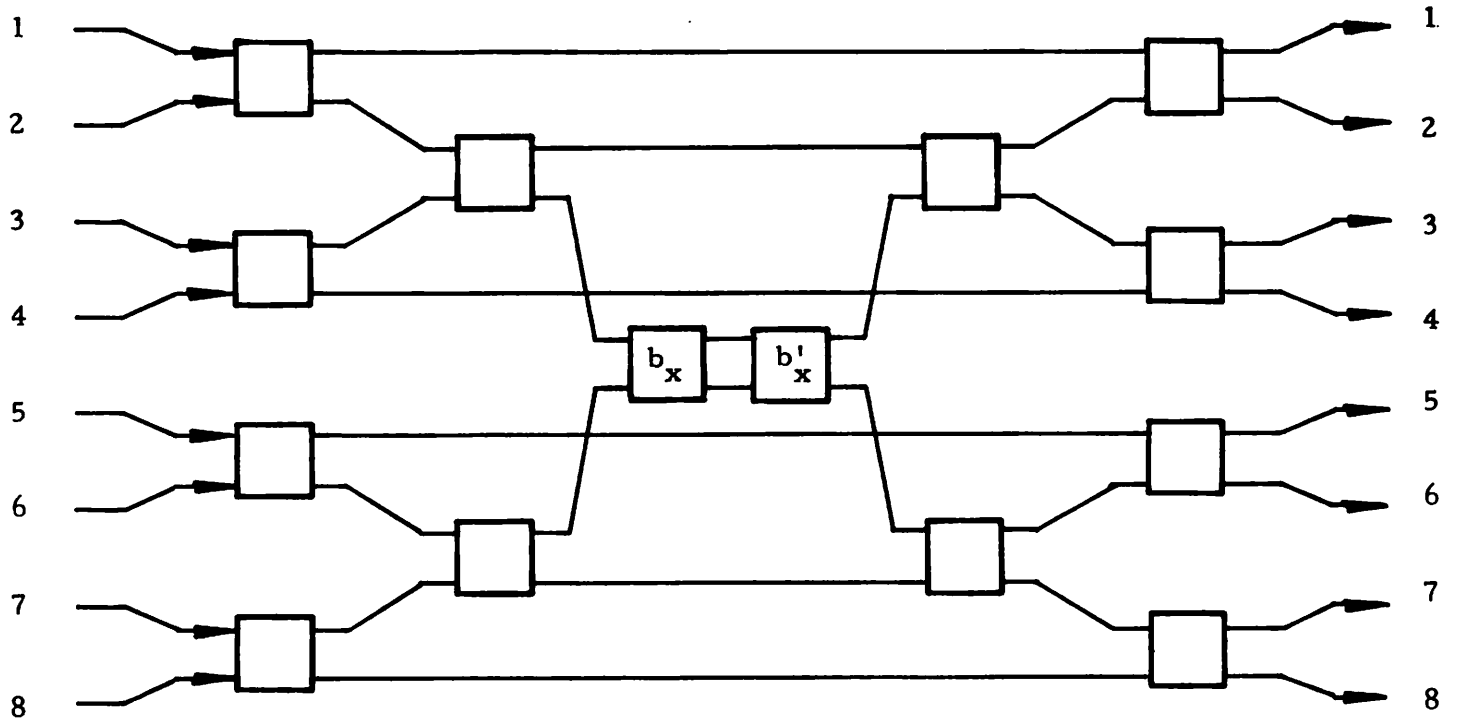
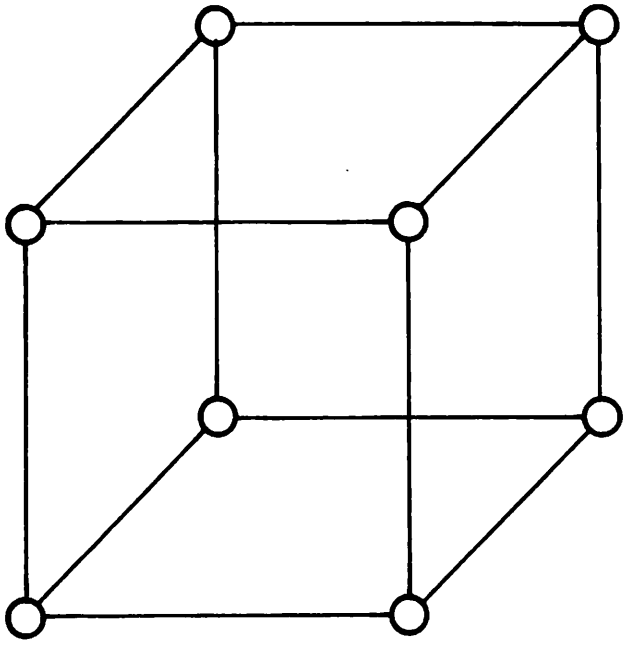


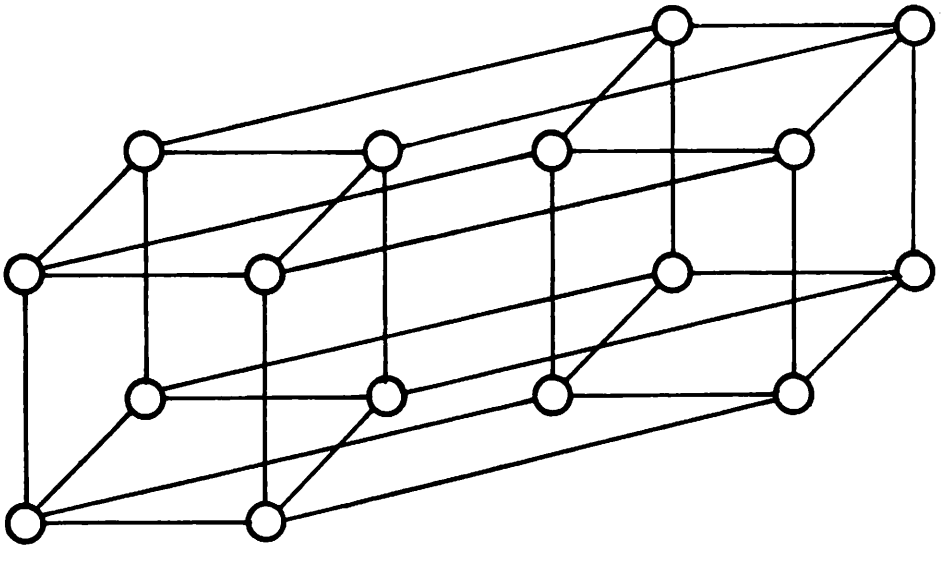
Figure 5.10. The $2^3 \times 2^3$ redundant DOT-network, or RDOT-network.

tems. The binary m-cube structure has been most frequently considered [Squire and Palais 1963, Sullivan and Bashkow 1977]. This network may be thought of as interconnecting 2^m processors which are placed at the vertices of an m-dimensional cube. Each edge of the cube represents a link connecting two processors, hence the name "binary" m-cube. Figure 5.11 depicts the binary 3-cube and 4-cube. One processor can be designated as the origin with an m-bit binary address 00...0. Other processors can then be identified by their corresponding coordinates in the m-dimensional space.

The binary m-cube structure possesses several advantages over other interconnection structures such as buses, rings, crossbars, and trees. When the number of processors is large, the bus and ring structures become infeasible due to contention and excessive communication delays, crossbars become prohibitively expensive, and the tree structure requires complex network control and may have unmanageable bottlenecks. The binary m-cube, on the other hand, has a regular structure and is relatively simple to control. The average communication delay between any pair of processors is small. High levels of parallelism can be achieved without excessive cost.



(a)



(b)

Figure 5.11. Binary cube structures: (a) the binary 3-cube; (b) the binary 4-cube.

5.3.1 Network Structure

The *indirect binary m-cube*, or *m-IBC-network*, denoted by $\mathcal{C}_m$, is a $2^m \times 2^m$ β-network which is defined recursively as follows. A β-element is a 1-IBC-network. An m-IBC-network for $m \geq 2$ is constructed from two (m-1)-IBC-networks and a $2^m \times 2^m$ β-stack according to the following equation

$$\mathcal{C}_m = (\mathcal{C}_{m-1} + \mathcal{C}_{m-1}) \circ \sigma \circ \mathcal{S} \circ \sigma^{-1},$$

where σ is the perfect shuffle permuter, $\mathcal{S}$ is a β-stack, and σ^{-1} is the inverse perfect shuffle permuter. As before, the operator $\circ$ denotes cascade or composition of permutations.

Figure 5.12 shows the general structure of $\mathcal{C}_m$. Two specific examples of the indirect binary 2-cube $\mathcal{C}_2$, and the indirect binary 3-cube $\mathcal{C}_3$ are constructed as shown in Figure 5.13. Redrawing the diagram in Figure 5.13b, we obtain the more familiar layout of the indirect binary 3-cube of Figure 5.13c, which was used as an example earlier (Figure 2.6).

The interconnections provided by the m-IBC-network emulate those of an m-dimensional binary cube. The m-IBC-network has m stages of β-elements with 2^{m-1} β-elements per stage. The total number of β-elements in this network is $m2^{m-1}$. There are 2^m terminal links which correspond to the corners of the binary m-cube, while the β-elements

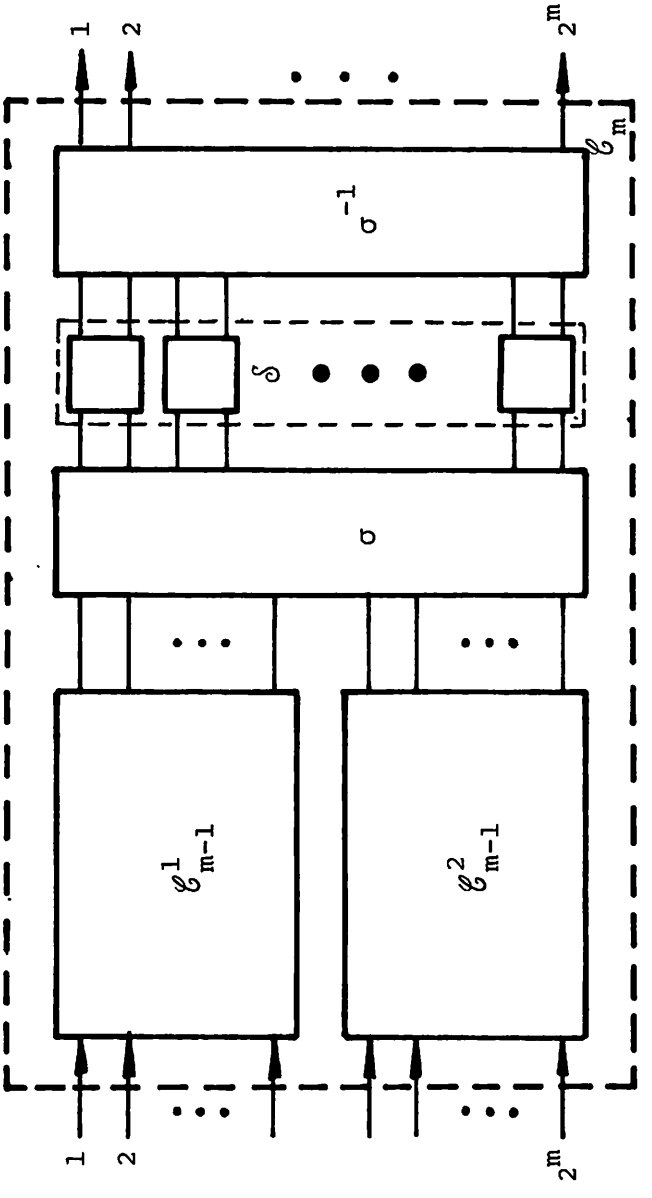
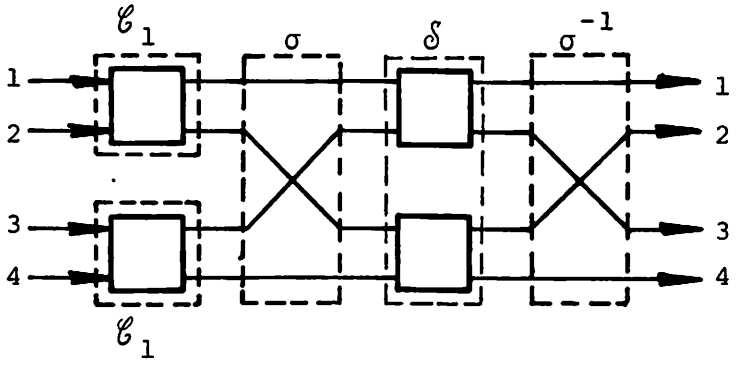
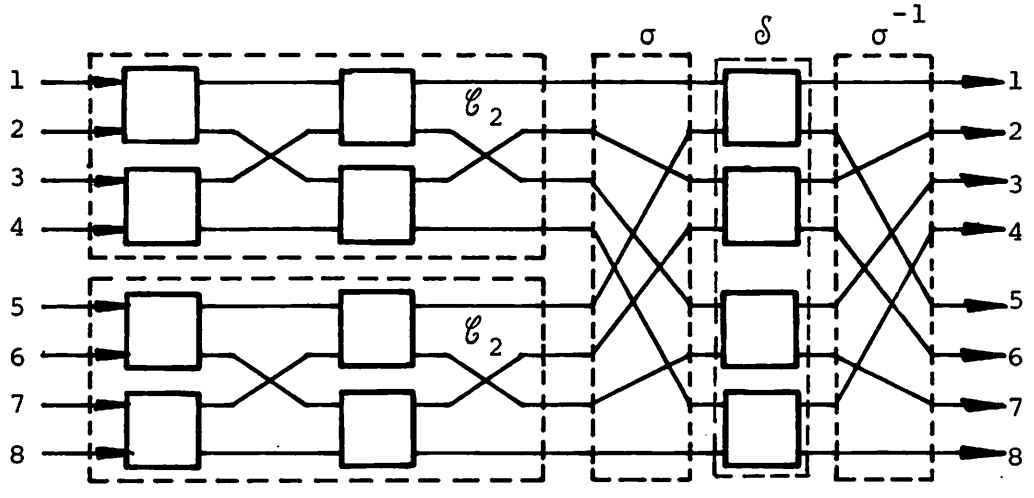


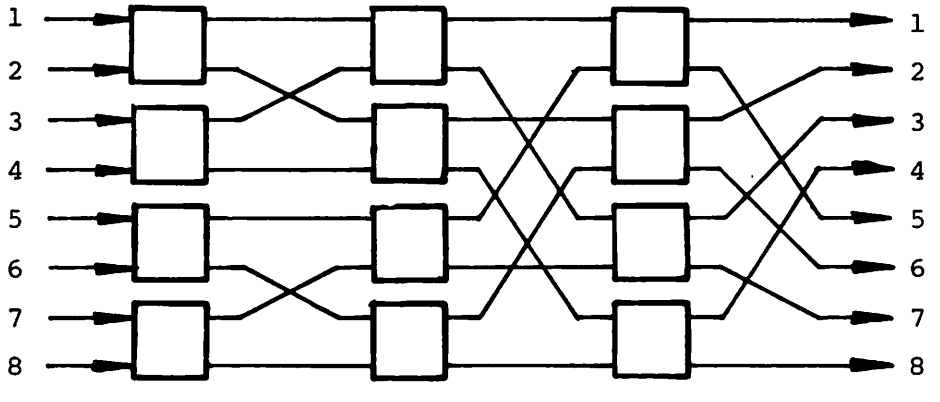
Figure 5.12. The general structure of the m-IBC-network $\mathscr{C}_m$.



(a)



(b)

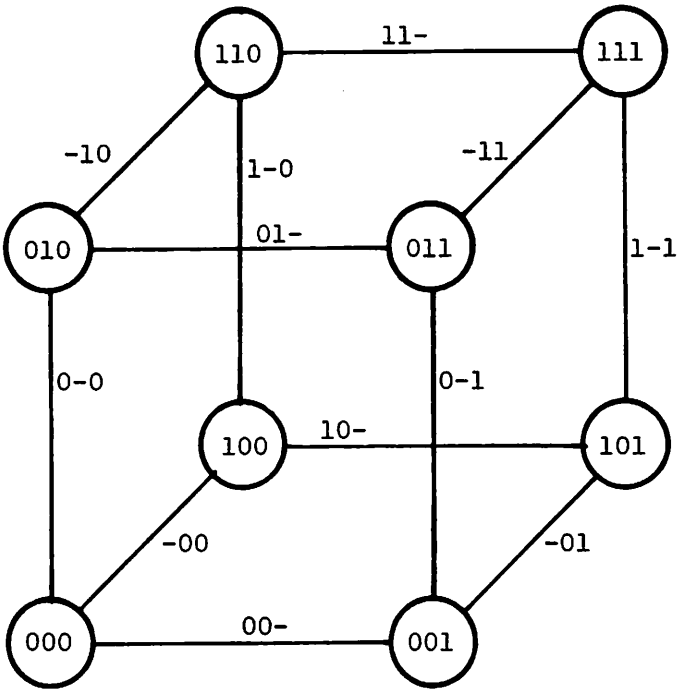


(c)

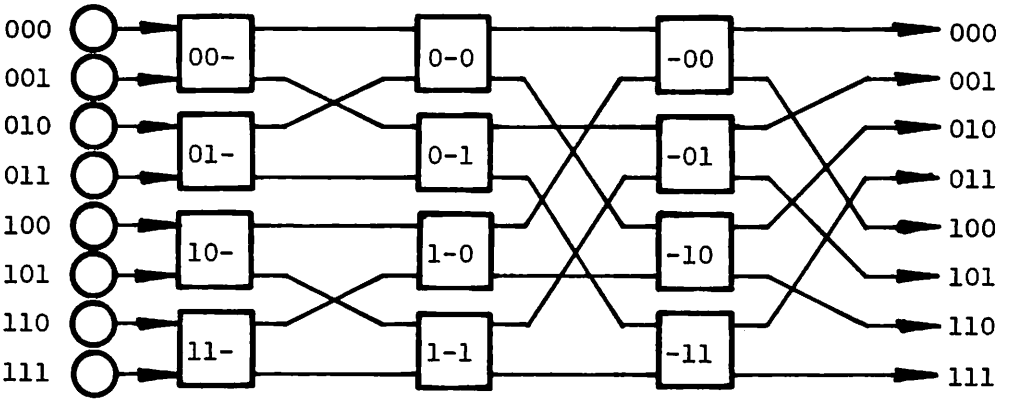
Figure 5.13. m-IBC-networks: (a) the 2-IBC-network $\mathcal{C}_2$; (b) the 3-IBC-network $\mathcal{C}_3$; (c) redrawing of the 3-IBC-network $\mathcal{C}_3$.

correspond to the edges of the binary m-cube, as illustrated in Figure 5.14. The β-elements in any stage correspond to all the edges parallel to the axis in some dimension. A β-element set to the X-state corresponds to a traversal of that edge in the binary m-cube. A simple algorithm, similar to those of the ISE- and MISE-networks discussed in Section 4.4 exists for the control of the IBC-networks [Pease 1977]. The indirect binary m-cube possesses several other unique and useful properties [Pease 1977,Sullivan and Bashkow 1977].

Two other well-known β-networks are actually isomorphisms of the m-IBC-network. The $2^m \times 2^m$ *flip network* used in the Staran SIMD parallel processor manufactured by Goodyear Aerospace is structurally isomorphic to the m-IBC-network [Batcher 1974]. The two networks differ only in their control schemes. Unlike the m-IBC-network, in which each β-element can be individually controlled, there is only one control line for each stage of β-elements in the flip network. All the β-elements in the same stage are set to either the T-state or the X-state simultaneously, to accomplish either the "shift" or the "flip" operation [Batcher 1974]. Lawrie devised a β-network called the *omega network* for accessing and aligning data in an array processor [Lawrie 1975]. The omega network is typically placed between a set of processors and a set of memory modules. Processors access data in the memory via



(a)



(b)

Figure 5.14. Example of binary and indirect binary cubes: (a) the binary 3-cube; (b) the indirect binary 3-cube.

the omega network. A $2^m \times 2^m$ omega network consists of m stages of 2^{m-1} β-elements, each preceded by a perfect shuffle permuter as shown in Figure 5.15. An *inverse omega network* is an omega network in which the direction of all the links are reversed. It has been shown that the $2^m \times 2^m$ inverse omega network and the indirect binary m-cube are isomorphic [Parker 1980]. Figure 2.21 illustrates the 8×8 inverse omega network which is isomorphic to the indirect binary 3-cube of Figure 2.6.

5.3.2 Communication Delay

It is well-known that the m-IBC-network is a full access m-stage β-network, hence its terminal delay t is m. In fact there exists a unique path of length m from each terminal link to any other terminal link. Since a terminal link a_i can reach any other terminal link in distance m, a_i must be able to reach any link, terminal or intermediate, within the distance 2m-1. We now show that the CD parameter d of the m-IBC-network is 2m-1, i.e., the distance between any two links is at most 2m-1.

Two BG-equivalent β-networks have the same communication delay. A BG-equivalent of the m-IBC-network can be obtained by cyclically rotating the stages, i.e., by replacing stage 1 by stage m, stage 2 by stage 1, etc. Since the m-IBC-network is isomorphic to the inverse omega network, which is a cascade of identical stages, any

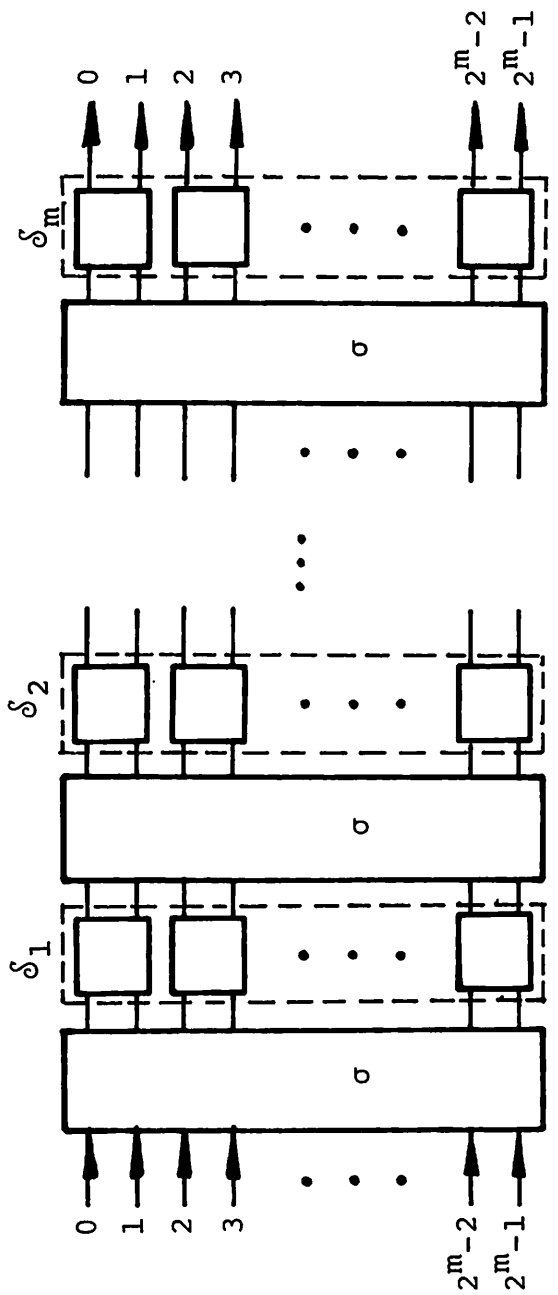


Figure 5.15. The general structure of the $2^m \times 2^m$ omega network.

cyclic rotation will produce an isomorphic β-network. Consequently, the links in any stage can be made the terminal links by the cyclic rotation. Hence any link whether terminal or intermediate can reach any other link within the distance 2m-1. In one pass an input terminal link can reach all the output links of the m^{th} stage,but only half of the input links to the m^{th} stage. The unreached links can be reached in a second pass. Consequently, there exist links separated by the distance 2m-1, hence the following lemma.

Lemma 5.4: The TD parameter t of the m-IBC-network is m and the CD parameter d is 2m-1.

5.3.3 Fault Tolerance

The recursive structure of the m-IBC-network suggests that we can determine its fault-tolerance parameter by induction. For this purpose we first develop several lemmas.

Lemma 5.5: Let $\mathscr{C}_m$ be an m-IBC-network. By setting all the β-elements in the m^{th} stage to the X-state we obtain a residual network $\mathscr{C}'_m$ having the structure illustrated in Figure 5.16.

Proof: The structure of $\mathscr{C}_m$ is defined in Figure 5.12. We need to show the permuters σ and σ′ and the β-elements

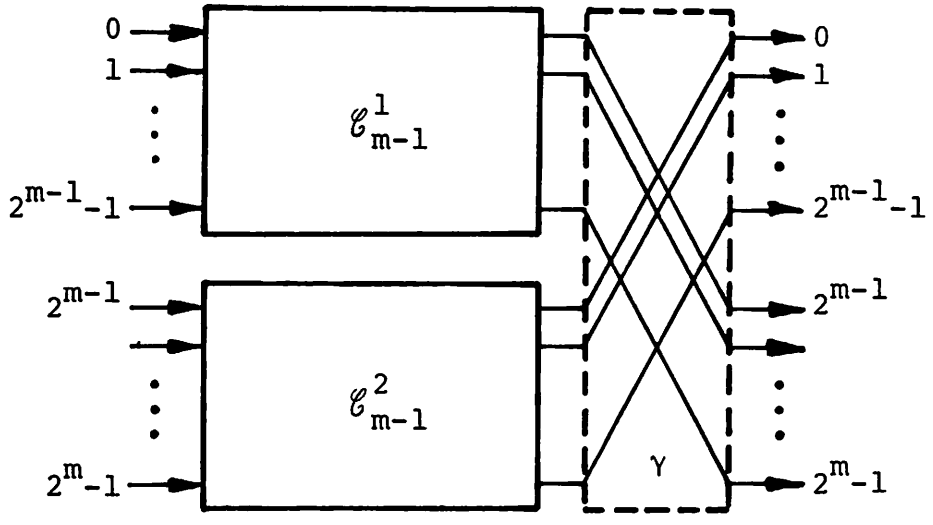


Figure 5.16. A residual network $\mathscr{C}'_m$ of the m-IBC-network $\mathscr{C}_m$.

in the m^{th} stage of $\mathscr{C}_m$ combine to form the permuter γ of Figure 5.16. If we label the links in a stage from top to bottom with the address numbers $0,1,\ldots,2^m-1$, then the permuter γ can be defined by the following permutation:

$$\gamma(a) = a+2^{m-1} \pmod{2^m}$$

where $a \in \{0,1,\ldots,2^m-1\}$ is the address of a link. If we represent a by the equivalent binary number $a_m a_{m-1}\ldots a_1$, then we can write

$$\gamma(a_m a_{m-1}\ldots a_1) = \bar{a}_m a_{m-1}\ldots a_1$$

where $\bar{a}_i = 0$ (1) if $a_i = 1$ (0). Effectively, the permuter γ connects the i^{th} output of the upper $\mathscr{C}_{m-1}$ to the i^{th} input of the lower $\mathscr{C}_{m-1}$ and vice versa. The perfect shuffle σ and the inverse perfect shuffle σ^{-1} can be similarly defined as follows:

$$\sigma(a_m a_{m-1}\ldots a_1) = a_{m-1}\ldots a_1 a_m$$

and

$$\sigma^{-1}(a_m a_{m-1}\ldots a_1) = a_1 a_m\ldots a_2.$$

σ effectively corresponds to a cyclic left shift of the binary address $a_m a_{m-1}\ldots a_1$, and σ^{-1} corresponds to a cyclic right shift of this address.

The permuting effect of a β-element can also be defined in this way. Let the permutation realized by a β-element in the T-state be denoted by $\mathcal{S}^T$ and that realized by a β-element in the X-state be denoted by $\mathcal{S}^X$. Then

$$\mathcal{S}^T(a_m a_{m-1}\cdots a_1) = a_m a_{m-1}\cdots a_1$$

and

$$\mathcal{S}^X(a_m a_{m-1}\cdots a_1) = a_m a_{m-1}\cdots \bar{a}_1.$$

Now

$$\begin{aligned}\sigma\circ\mathcal{S}^X\circ\sigma^{-1}(a_m a_{m-1}\cdots a_1) &= \mathcal{S}^X\circ\sigma^{-1}(a_{m-1}\cdots a_1 a_m)\\ &= \sigma^{-1}(a_{m-1}\cdots a_1\bar{a}_m)\\ &= \bar{a}_m a_{m-1}\cdots a_1\\ &= \gamma(a_m a_{m-1}\cdots a_1)\end{aligned}$$

from which it follows that $\sigma\circ\mathcal{S}^X\circ\sigma^{-1} = \gamma$. □

An obvious corollary to Lemma 5.5 is the following result.

Corollary 5.3: The residual β-network $\mathcal{C}'_m$ of Lemma 5.5 is BG-equivalent to a β-network $\mathcal{C}''_m$ obtained by cascading two (m-1)-IBC-networks $\mathcal{C}^1_{m-1}$ and $\mathcal{C}^2_{m-1}$.

If, instead of setting all the β-elements in the m^{th} stage to the X-state, we set them to the T-state, another interesting residual network is obtained.

$$\sigma \circ \delta^T \circ \sigma^{-1}(a_m a_{m-1} \cdots a_1) = \delta^T \circ \sigma^{-1}(a_{m-1} \cdots a_1 a_m)$$

$$= \sigma^{-1}(a_{m-1} \cdots a_1 a_m)$$

$$= a_m a_{m-1} \cdots a_1$$

Hence $\sigma \circ \delta^T \circ \sigma^{-1} = I$, where I is the identify permuter. This leads to the following lemma.

Lemma 5.6: Let $\mathcal{C}_m$ be an m-IBC-network. By setting all the β-elements in the m^{th} stage to the T-state we obtain a residual network $\mathcal{C}_m^*$ which is a stack of two (m-1)-IBC-networks $\mathcal{C}_{m-1}^1$ and $\mathcal{C}_{m-1}^2$.

The two residual networks $\mathcal{C}_m'$ and $\mathcal{C}_m^*$ will be useful in computing the fault tolerance of $\mathcal{C}_m$. As shown in Corollary 5.3, $\mathcal{C}_m'$ is essentially a cascade of two (m-1)-IBC-networks, while $\mathcal{C}_m^*$ is a stack of two (m-1)-IBC-networks. Since the m-IBC-network has loops of length m, and hence has elementary circuits of length m in its β-graph, its fault tolerance k must be less than m. We show next by induction, that k is indeed m-1.

Lemma 5.7: The FT parameter k of the m-IBC-network $\mathcal{C}_m$ is m-1, for $m \geq 2$.

Proof: The 2-IBC-network and its β-graph are depicted in Figure 2.18. This network is clearly BG-equivalent to

the cyclic β-network or order 4 with generator h = 3. It is 1-FT because its β-graph contains a Hamiltonian circuit and no self-loop. It remains to be shown that for m > 2, if $\mathcal{C}_{m-1}$ is (m-2)-FT, then $\mathcal{C}_m$ must be (m-1)-FT.

Let f be any set of m-1 faulty β-elements in $\mathcal{C}_m$. Since $\mathcal{C}_m$ has m stages, there must exist a stage which does not contain any faulty β-element. We can assume without losing generality, that this fault-free stage is the m^{th} stage. (If the m^{th} stage is not fault-free, then the stages of $\mathcal{C}_m$ can be cyclically rotated to obtain an isomorphic network which has a fault-free m^{th} stage.) Hence all the faulty β-elements of f are in the first m-1 stages, i.e., they are contained in the two subnetworks $\mathcal{C}^1_{m-1}$ and $\mathcal{C}^2_{m-1}$ of $\mathcal{C}_m$. Since all the β-elements in the m^{th} stage are fault-free, they can all be set to the X-state or the T-state to obtain the residual networks $\mathcal{C}'_m$ or $\mathcal{C}^*_m$ respectively. We want to show that the $\mathcal{C}_m$ network containing the fault f or, equivalently, that the residual network $\mathcal{C}_m/s_f$, where s_f is the partial state representing the fault f, still has DFA.

Case 1: (One subnetwork is fault-free) All m-1 faulty β-elements of f are in one subnetwork, say $\mathcal{C}^1_{m-1}$. The other subnetwork $\mathcal{C}^2_{m-1}$ must then be fault-free. We know that a fault-free IBC-network has full access. Hence $\mathcal{C}^2_{m-1}$ must have full access. Since $\mathcal{C}'_m$ is BG-equivalent to a cascade of the two subnetworks (Corollary 5.3), it can be concluded

from Theorem 5.3 that $\mathcal{C}'_m/s_f$ must still have full access. Since $\mathcal{C}'_m/s_f$ has DFA and is a residual network of $\mathcal{C}_m/s_f$ by Theorem 5.2, $\mathcal{C}_m/s_f$ must also have DFA. Therefore $\mathcal{C}_m$ is (m-1)-FT.

Case 2: (Both subnetworks are faulty) The m-1 faulty β-elements of f are distributed in both $\mathcal{C}^1_{m-1}$ and $\mathcal{C}^2_{m-1}$. In this case each subnetwork can contain at most m-2 faulty β-elements. Since by assumption $\mathcal{C}^1_{m-1}$ and $\mathcal{C}^2_{m-1}$ are (m-2)-FT, then both $\mathcal{C}^1_{m-1}/s_f$ and $\mathcal{C}^2_{m-1}/s_f$ must still have DFA. The upper and lower halves, $\mathcal{C}^1_{m-1}$ and $\mathcal{C}^2_{m-1}$ of the residual network $\mathcal{C}^*_m$ are disjoint. Since both $\mathcal{C}^1_{m-1}/s_f$ and $\mathcal{C}^2_{m-1}/s_f$ have DFA, a link in the upper (lower) half of $\mathcal{C}^*_m/s_f$ can reach all the links in the upper (lower) half of $\mathcal{C}^*_m/s_f$. For a link in the upper (lower) half to reach all the links in the lower (upper) half, we can set the fault-free β-elements in the m^{th} stage to the X-state to obtain $\mathcal{C}'_m/s_f$. Consequently any link in $\mathcal{C}_m/s_f$ can reach any other link in $\mathcal{C}_m/s_f$. Hence $\mathcal{C}_m/s_f$ must have DFA and $\mathcal{C}_m$ must be (m-1)-FT. □

Combining Lemmas 5.4 and 5.7, yields the following theorem.

Theorem 5.9: Let $\mathcal{C}_m$ denote the m-IBC-network. The FT parameter of $\mathcal{C}_m$ is $k = m-1$. The CD parameter of $\mathcal{C}_m$ is $d = 2m-1$. The TD parameter of $\mathcal{C}_m$ is $t = m$. Thus m-IBC-networks are $(m2^{m-1}, m-1, 2m-1)$-networks.

Unlike many of the families of β-networks presented earlier, m-IBC-networks have an FT-function which is not a constant, but is a function of the size parameter m. m-IBC-networks exhibit the best FT and CD combination of the β-networks considered so far. If the number of β-elements of $\mathcal{C}_m$ is denoted by n, then the fault tolerance of $\mathcal{C}_m$ is approximately $\log_2 n$ and the communication delay is approximately $2\log_2 n$.

5.4 BENES REARRANGEABLE NETWORKS

One of the earliest studies of connecting networks was performed by Clos on non blocking switching networks [Clos 1953]. Clos presented a class of non blocking networks, now called *Clos networks*, consisting of three stages of crossbar switches, as shown in Figure 2.6. The first stage contains r $n \times m$ crossbar switches, and the third stage contains r $m \times n$ crossbar switches. The center stage has m $r \times r$ square crossbar switches. The Clos network has $N = n \times r$ input and N output terminals. Clos has shown that for $m \geq 2n-1$, the three-stage Clos network is non blocking in the strict sense. The classical paper by Clos has inspired a considerable amount of research into non blocking and rearrangeable networks. The well-known Slepian-Duguid theorem states that the Clos network is rearrangeable if and only if $m \geq n$ [Benes 1965]. Benes presented a special class of the three-stage Clos network, in which $n = m = 2$,

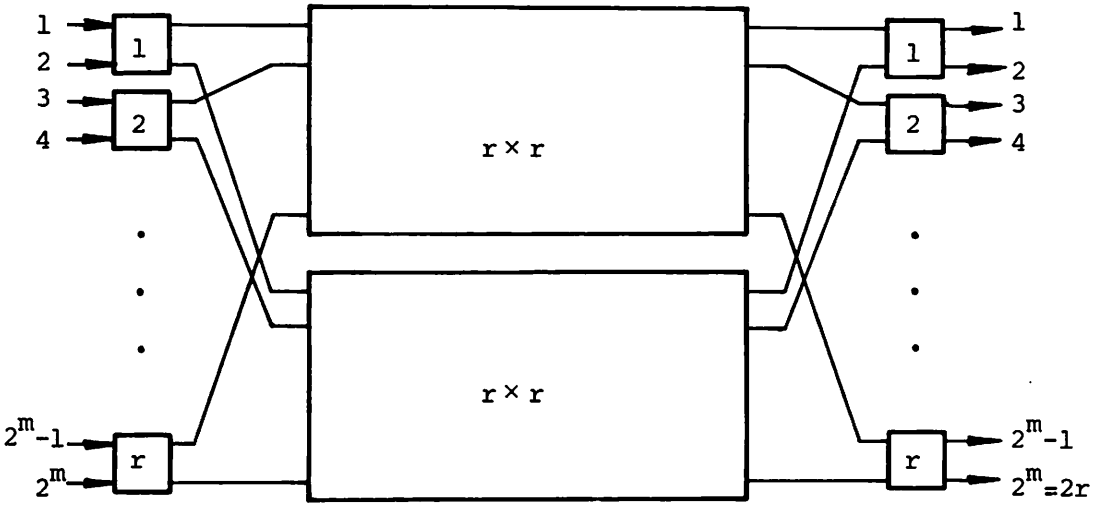
and showed that it is rearrangeable. A network in this class has $2 \times r$ input and $2 \times r$ output terminals and consists of only square crossbar switches. The first and the third stages each contain r 2×2 crossbar switches, or β-elements, while the middle stage contains two $r \times r$ crossbar switches, as shown in Figure 5.17a. It has also been proven that this network can be further decomposed by replacing each of the $r \times r$ crossbar switches in the middle stage by another three-stage rearrangeable network of the same structure, as illustrated in Figure 5.17b. This process can be continued until all the square crossbar switches in the network are β-elements. The resultant network is a rearrangeable $2n \times 2n$ β-network consisting of $2\log_2(2n)-1$ stages of β-elements. We refer to this β-network as the *Benes rearrangeable network* or *BRS-network*. The fault-tolerance properties of these BRS-networks are the topic of this section.

5.4.1 Network Structure

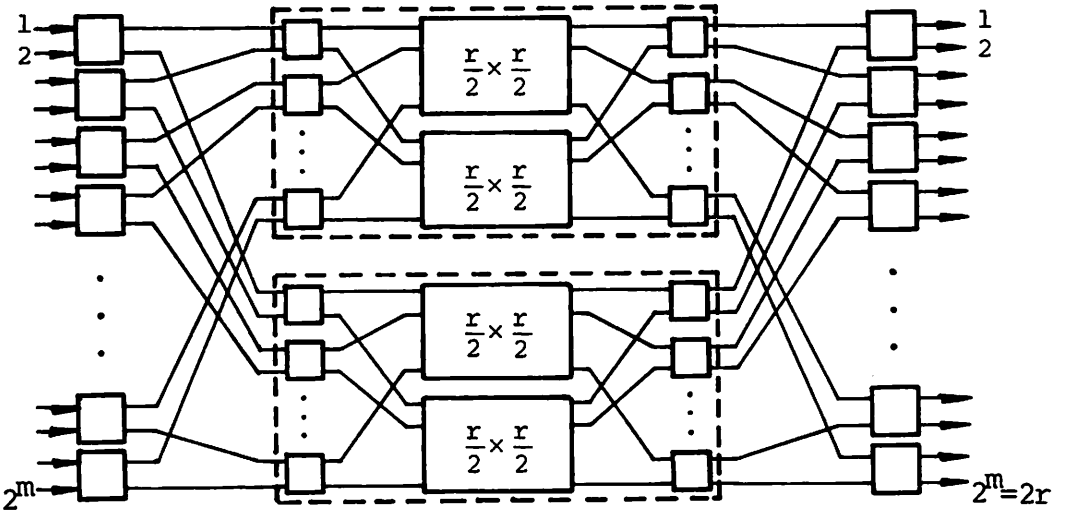
The smallest 2×2 BRS-network is the β-element. The $2^m \times 2^m$ BRS-network $\mathcal{B}_m$ for $m \geq 2$ is defined recursively as follows.

$$\mathcal{B}_m = \mathcal{S}_m \circ \sigma^{-1} \circ (\mathcal{B}_{m-1} + \mathcal{B}_{m-1}) \circ \sigma \circ \mathcal{S}_m$$

where $\mathcal{S}_m$ denotes the $2^m \times 2^m$ β-stack, σ denotes the perfect



(a)



(b)

Figure 5.17. Decomposition of the Clos rearrangeable network: (a) the base-2 network; (b) the partially decomposed network.

shuffle permuter, and $\mathcal{B}_{m-1}$ denotes the $2^{m-1} \times 2^{m-1}$ BRS-network. Figure 5.18 depicts the structure of the $2^m \times 2^m$ BRS-network. The $2^3 \times 2^3$ BRS-network $\mathcal{B}_3$ is illustrated in Figure 5.19. The general BRS-network $\mathcal{B}_m$ has 2m-1 stages of β-elements and is symmetrical with respect to the middle stage. By the definition of rearrangeability, this network is capable of realizing all $2^m!$ possible input-output connections. The BRS-network is then the most powerful β-network in terms of the connecting capability that we have considered so far. It is also the most difficult to analyze.

BRS-networks have been extensively studied by many researchers, including Joel [Joel 1968] and Opferman and Tsao-Wu [Opferman and Tsao-Wu 1971]. Previous work has focused on analysis of the network complexity, and implementation of efficient algorithms for network control. Opferman and Tsao-Wu have also studied the diagnosis of faulty β-elements which are stuck at the T- or the X-states. They assume that the state of each individual β-element is not accessible, and have shown how to derive a very small set of connections, or permutations of the terminals, which correspond to an efficient set of test patterns.

As shown in Figure 5.18, the $2^m \times 2^m$ BRS-network $\mathcal{B}_m$ contains a stack of two $2^{m-1} \times 2^{m-1}$ BRS-networks. We denote the upper one by $\mathcal{B}^1_{m-1}$ and the lower one by $\mathcal{B}^2_{m-1}$. Let

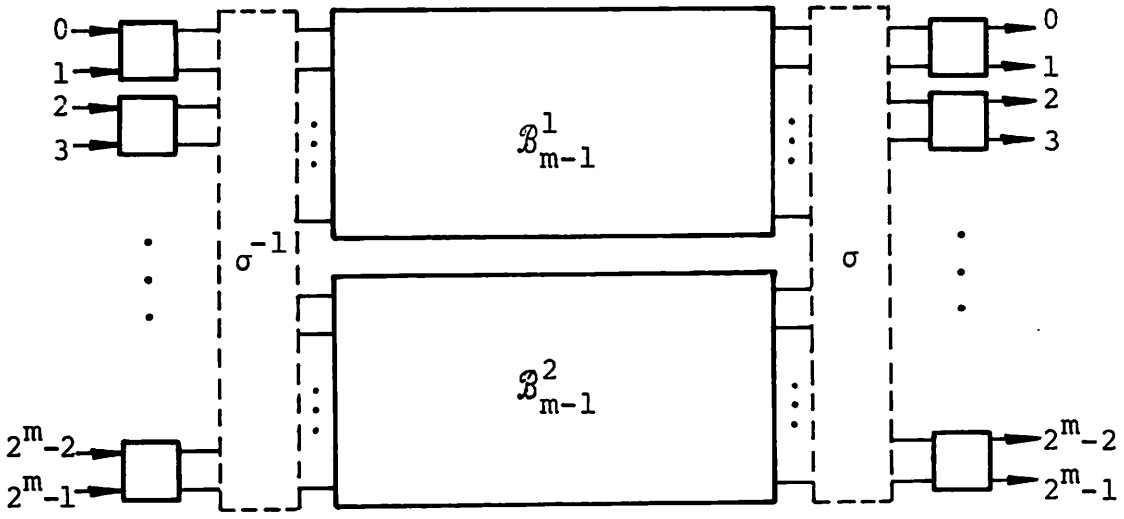


Figure 5.18. The general structure of the $2^m \times 2^m$ BRS-network $\mathcal{B}_m$.

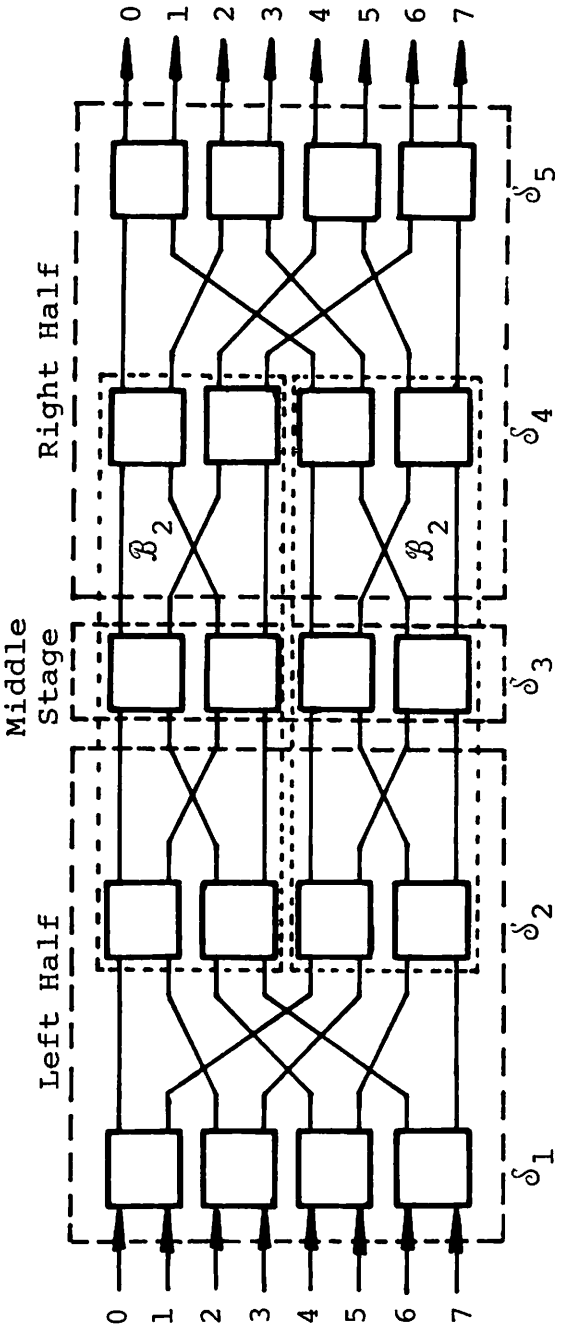


Figure 5.19. $2^3 \times 2^3$ BRS-network, $\mathcal{B}_3$.

S_i denote the stack of β-elements in the i^{th} stage of $\mathcal{B}_m$ whose stages are numbered 1,2,...,2m-1, from left to right. Consequently, S_m denotes the middle stage, and the network $\mathcal{B}_m$ is symmetrical with respect to S_m. We call the subnetwork to the left (right) of S_m the left (right) "half" of $\mathcal{B}_m$. Hence the network $\mathcal{B}_m$ is the cascade of the left half, the middle stage S_m, and the right half as illustrated in Figure 5.19. Wu and Feng designed a full-access β-network called the *baseline* network [Wu and Feng 1978] which is isomorphic to the cascade of the left half of $\mathcal{B}_m$ and the middle stage S_m.

5.4.2 Communication Delay

Since the BRS-network is rearrangeable, it clearly has full access. Hence its terminal delay t is the number of stages in the network, which is 2m-1. The communication delay parameter t is much more difficult to derive than the terminal delay.

Lemma 5.8: The CD parameter of the $2^m \times 2^m$ BRS-network $\mathcal{B}_m$ is $d = 4m-3$.

Proof: The baseline network has full access, hence a terminal link of $\mathcal{B}_m$ can reach all the links in the right half of $\mathcal{B}_m$ in one pass. It can reach any link of $\mathcal{B}_m$ in a second pass, or within the distance $d_1 = (2m-1)+(m-1) = 3m-2$.

The middle stage S_m cascaded with the right half of $\mathcal{B}_m$ is isomorphic to the inverse baseline network. It has been shown that the inverse baseline network is isomorphic to the baseline network [Wu and Feng 1980], so S_m cascaded with the right half of $\mathcal{B}_m$ also possesses full access. Hence any non-terminal link in the left half of $\mathcal{B}_m$ can reach all the terminal links within the distance $d_2 = 2m-2$, and can reach any link of $\mathcal{B}_m$ within the total distance $d_3 = d_2 + (2m-2) = 4m-4$. Similarly, any non-terminal link in the right half of $\mathcal{B}_m$ can reach at least two terminal links within the distance $d_4 = m-1$, and hence can reach any link of $\mathcal{B}_m$ within the total distance $d_5 = d_4 + d_1 = 4m-3$. Since $d_5 > d_3 > d_1$, every link must be able to reach any other link within the distance $d_5 = 4m-3$. Hence the communication delay d of the BRS-network $\mathcal{B}_m$ is at most d_5. If we can show there exist two links separated by the distance 4m-3, then the communication delay must be exactly $d_5 = 4m-3$.

Let b denote a β-element in the upper half of the middle stage S_m. Let k_o denote the upper outgoing link of b and k_i denote the lower incoming link of b. Because the structure of the BRS-network, an upper outgoing link of a β-element in S_m can only reach the upper half terminal links of $\mathcal{B}_m$ in one pass through the network. Hence after the first pass, k_o can only reach all the upper terminal links. Furthermore, in a second pass k_o cannot reach any of the lower incoming links to the β-elements of S_m; see

Figure 5.19. Only in a third pass can k_o reach k_i. Hence the distance from k_o to k_i is equal to $(m-1)+(2m-1)+(m-1) = 4m-3 = d_5$. Therefore the communication delay d of the $2^m \times 2^m$ BRS-network is 4m-3. □

5.4.3 Fault Tolerance

Some upper bounds on the fault tolerance parameter k of $\mathcal{B}_m$ can be easily obtained. Let the 2^m terminals of $\mathcal{B}_m$ be numbered $0,1,\ldots,2^m-1$ from top to bottom. If the top (bottom) β-element in each stage is set to the T-state, a critical fault consisting of 2m-1 β-elements results, which isolates the terminal 0 (2^m-1). Hence the fault tolerance k of the $2^m \times 2^m$ BRS-netwrok must be less than or equal to 2m-2.

The $2^m \times 2^m$ BRS-network contains two major subnetworks $\mathcal{B}^1_{m-1}$ and $\mathcal{B}^2_{m-1}$. Each of these $2^{m-1} \times 2^{m-1}$ BRS-networks in turn contains two $2^{m-2} \times 2^{m-2}$ BRS-networks, etc. We call the terminal links $0,1,\ldots,2^{m-1}-1$ of $\mathcal{B}_m$ its upper terminal links, and call the remaining links $2^{m-1},\ldots,2^m-1$ the lower terminal links. In the left half of $\mathcal{B}_m$ any upper (lower) terminal link can only reach those links which are upper (lower) input links of the subnetworks of $\mathcal{B}_m$. Consequently, if all 2^{m-1} β-elements of the middle stage S_m are set to T, the upper and lower terminal links of $\mathcal{B}_m$ are disconnected, and $\mathcal{B}_m$ is decomposed into two identical subnetworks.

Hence another upper bound for k is $2^{m-1}-1$. Combining the two upper bounds we obtain the following result.

Theorem 5.10: The FT parameter k of the $2^m \times 2^m$ BRS-network $\mathcal{B}_m$ is bounded above by $\min\{2m-2, 2^{m-1}-1\}$.

It is conjectured that k is actually equal to this upper bound, i.e., $k = \min\{2m-2, 2^{m-1}-1\}$.

Conjecture 5.1: The $2^m \times 2^m$ BRS-network $\mathcal{B}_m$ has FT parameter $k = \min\{2m-2, 2^{m-1}-1\}$.

For $m \leq 3$ the above conjecture is known to be true. If it is true for $m = 4$, then the conjecture can be proven by using a similar approach to that of Theorem 5.9.

BRS-networks have fault tolerance and communication delay similar to those of IBC-networks. If the number of β-elements of $\mathcal{B}_m$ is n, then the fault tolerance of $\mathcal{B}_m$ is approximately $2\log_2 n$, and the communication delay is approximately $4\log_2 n$. BRS-networks appear to be fault tolerant with respect to full access as well.

CHAPTER 6

CONCLUSIONS

This thesis documents a study of the fault tolerance of a class of connecting networks called β-networks. These β-networks are intended for use as interconnection and communication networks (ICNs) in large interconnected computer systems, e.g., munticomputer systems. This chapter summarizes the key results obtained, and suggests some further research topics.

6.1 SUMMARY OF RESULTS

The major contributions of this study can be summarized as follows.

1. We have developed a framework for analyzing fault tolerance in β-networks based on β-element s-a-T/X faults and the DFA connectivity property.
2. We have obtained general graph-theoretical characterizations of minimal critical and noncritical faults in β-networks.

3. We have applied the theoretical results to the analysis and synthesis of several important fault tolerant β-networks.

6.1.1 Theory of Fault Tolerance

β-networks are connecting networks composed of 2 × 2 switches called β-elements. A fault model was used which allows β-elements to be stuck in either of their two normal states: the through (T) state or the cross (X) state. Thus a β-element can fail in two ways; it can be either stuck at through (s-a-T) or stuck at cross (s-a-X). Several connectivity properties of connecting networks are well understood, e.g., the non-blocking, rearrangeable, and full-access properties. A β-network has the full-access property if each of its input terminals can be directly connected to any one of its output terminals. A new connectivity property called dynamic full-access (DFA) was introduced here which serves as the criterion for fault tolerance. A β-network has the DFA property if each of its inputs can be connected to any one of its outputs in a finite number of passes through the β-network. A fault in a β-network is a collection of β-element stuck-at faults. A fault is critical if it destroys the DFA property. A minimal critical fault (MCF) is a critical fault none of whose proper subsets constitutes a critical fault.

The set of all MCFs, called the critical fault set, of a β-network is a measure of the vulnerability of the β-network to β-element stuck-at faults.

In our analysis of the nature of faults and fault tolerance in β-networks we made extensive use of a graph model of a β-network called a β-graph. The β-graph of a β-network is a directed graph with vertices representing the β-elements, and edges representing the links in the β-network. Computing units supported by a β-network are not explicitly represented, but are implicitly modeled by the edges representing the terminal links of a β-network. Two graph-theoretical characterizations of the minimal critical faults and the noncritical faults of a β-network were presented. It was shown that there is a one-to-one correspondence between minimal critical faults and the cutsets of the circuit adjacency graphs derived from the β-graph. It was further shown that a fault is non-critical if and only if it is compatible with an Eulerian circuit of the β-graph. Using the first characterization all the minimal critical faults of a β-network can be computed. The second characterization enables one to determine whether any given fault is critical or not.

The fault tolerance of a β-network is measured by its ability to maintain DFA in spite of the presence of stuck-at faults in its β-elements. A β-network is more fault tolerant if it is able to tolerate a greater number of β-

element stuck-at faults. A new numerical measure of fault tolerance was devised in this study. We defined a β-network with DFA as k-fault tolerant or k-FT if the failure, either s-a-T or s-a-X, of any k or fewer β-elements does not destroy DFA. The largest k for which a β-network is k-FT is called the fault tolerance (FT) parameter of the β-network.

In this study the links in a β-network were implicitly assumed to be unidirectional. This assumption gives rise to the directed edges of a β-graph. However, it is worth noting that the T and the X states of a β-element are symmetrical in the following sense. A β-element in the T or the X state is isomorphic to its inverse in the same state, as shown in Figure 6.1. Consequently, whether the links are unidirectional or bidirectional, the same stuck-at T/X fault model of β-networks with unidirectional links is equally applicable to β-networks with bidirectional links.

6.1.2 Synthesis of Fault Tolerant β-networks

We have shown that the foregoing theoretical results can be applied to the synthesis of fault tolerant β-networks, taking into account the need in practice to strike a balance between performance and fault tolerance. The communication delay between computing units has been chosen as a measure of the performance of a β-network.

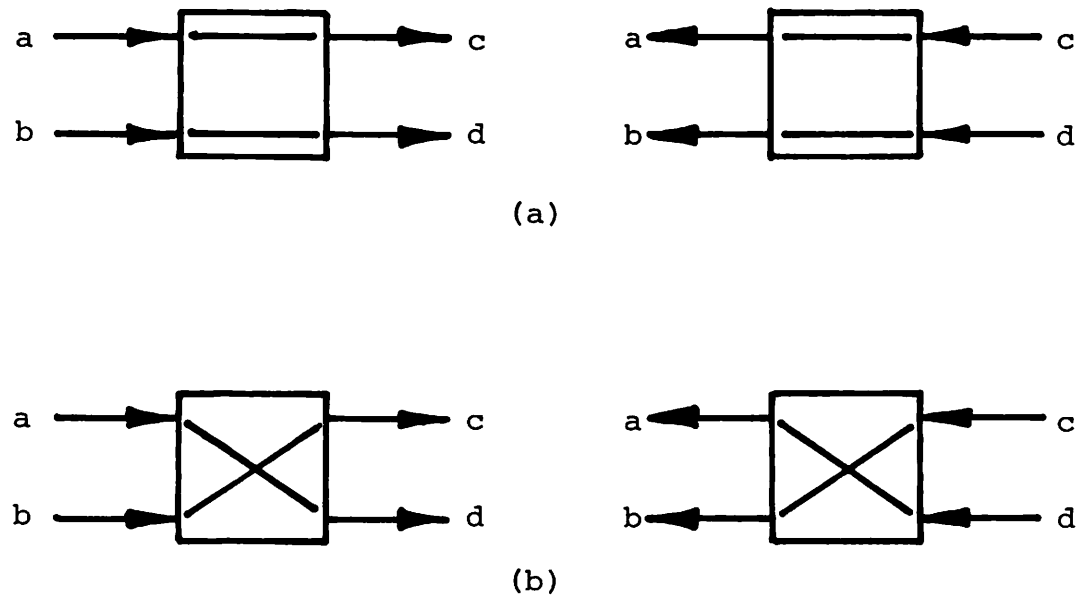


Figure 6.1. (a) Symmetry of the T-state. (b) Symmetry of the X-state.

This delay can be represented numerically by the distance between the two units, i.e., the minimum number of β-elements separating the two units. Since computing units are implicitly modeled by edges in a β-graph, we formally defined the communication delay (CD) parameter of a β-network to be the diameter of its β-graph, which is the longest distance between any two edges of the β-graph. Hence the CD parameter represents the worst possible communication delay between any two computing units. The values of both the FT and CD parameters are expressed in terms of numbers of β-elements. It has been shown here that the FT parameter k and CD parameter d of any β-network with n β-elements must satisfy the following tight bounds

$$0 \leqq k \leqq n-1$$

$$\lfloor \log_2 n \rfloor + 1 \leqq d \leqq n.$$

Furthermore, k must always be less than or equal to d.

The FT and CD parameters for several useful β-networks were computed. The $2n \times 2n$ inverse shuffle-exchange (ISE) network of n β-elements was shown to have FT parameter $k = 0$ and CD parameter $d = \lfloor \log_2 n \rfloor + 1$. A cyclic β-network called the double parallel ring (DPR) network was shown to have FT parameter $k = n-1$ and CD parameter $d = n$. The ISE-network is 0-FT and can be referred to as zero fault tolerant. A

modified design of the ISE-network, called the modified ISE (MISE) network, was devised which has FT parameter $k = 1$, and the same CD parameter $d = \lfloor \log_2 n \rfloor + 1$ as the ISE-network. Consequently, the MISE-network has the minimal communication delay and is minimally fault tolerant. The DPR-network, on the other hand, is maximally fault tolerant but has the worst possible communication delay. It was further shown that the DPR-network is unique in achieving maximal fault tolerance. The MISE-network and the DPR-network represent extreme cases of fault tolerant β-networks with reasonable communication delays.

6.1.3 Analysis of Well-known β-networks

Many useful β-networks are complex multiple-stage β-networks. In Section 5.1, we developed several general techniques for the analysis and synthesis of complex β-networks. These techniques utilize the symmetry and regularity in the network structure to ease the analysis of complex structures.

Three complex β-networks were analyzed in Chapter 5. The first is called the double-tree network, or DOT-network. The DOT-network was originally proposed as a fault detecting and correcting network, [Levitt et al. 1968] and has also been considered in the design of the MIT Data Flow Processor. We have shown in this thesis that the

$2^m \times 2^m$ DOT-network containing $2^{m+1}-3$ β-elements has fault tolerance $k = 0$ and communication delay $4m-4$.

The indirect binary m-cube, or the m-IBC-network, has been studied by many researchers. It has been proposed for SIMD/MIMD systems and special purpose signal processors. It was shown in this study that the $2^m \times 2^m$ m-IBC-network contains $m2^{m-1}$ β-elements, and has fault tolerance $k = m-1$ and communication delay $d = 2m-1$. The fault tolerance of the m-IBC-network is not a constant but a function of the size of the network. If the number of β-elements of the m-IBC-network, is denoted by n, then the fault tolerance of the network is approximately $\log_2 n$, and the communication delay is approximately $2\log_2 n$.

The third network is the Benes rearrangeable network, or BRS-network, it is a special case of a Clos three-stage network, which was one of the earliest connecting networks to be studied. The BRS-network is the most powerful β-network we have considered, and its analysis is quite complex. It was shown that the $2^m \times 2^m$ BRS-network contains $2^{m-1}(2m-1)$ β-elements and has communication delay $d = 4m-3$. It was conjectured that the BRS-network has fault tolerance $k = \min\{2m-2, 2^{m-1}-1\}$. If the conjecture is true, then the fault tolerance of the BRS-network is approximately $2\log_2 n$, where n is the number of β-elements. Like the indirect binary m-cube, the BRS-network possesses FT and CD parameters which are logarithmic functions of the total

number of β-elements.

We have also determined the FT and CD parameters for various families of β-networks. These networks form a representative sample of β-network design options. Figure 6.2 shows graphically the tradeoffs between fault tolerance and communication delay for β-networks in general. The shaded area is the feasible "design space" for fault-tolerant β-networks. The positions of the networks studied in this thesis are shown as data points. Clearly, the overall goal in β-network design is to maximize k and minimize d. Hence good designs are those in the upper left corner of the shaded area of Figure 6.2.

6.2 FURTHER RESEARCH

There are several promising extensions to this study.

1. In the development of our fault tolerance theory, some restrictive assumptions were made to facilitate the analysis. Both the fault model and the fault tolerance criterion can be generalized to obtain broader, although possibly weaker, results. For example, more than two distinct fault states can be allowed in the β-elements. Other connecting capabilities, such as full-access or rearrangeability, can be used as the fault tolerance criterion.

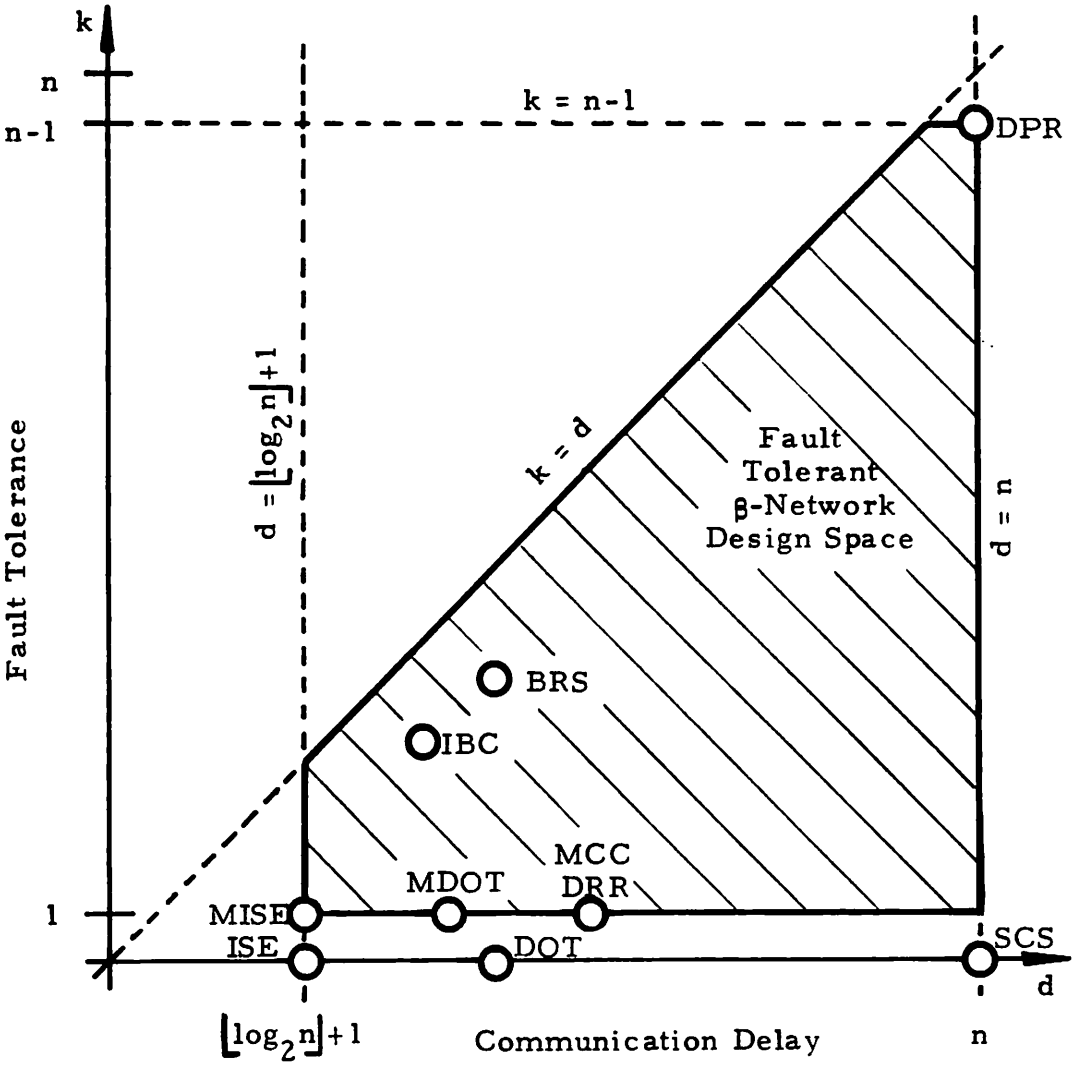


Figure 6.2. β-network design space.

2. In order to make the MCF and noncritical fault characterizations practical, efficient algorithms for identifying circuit partitions and computing Eulerian circuits in β-graphs are needed. Exhaustive search for circuit partitions and Eulerian circuits in an arbitrary β-graph may not be practical. However, linear algorithms are known for obtaining one Eulerian circuit. It appears that efficient procedures for obtaining all Eulerian circuits can be developed by exploiting the symmetry found in most useful β-network structures.

3. We have only made an initial attempt at identifying good designs for fault tolerant β-networks. It may be possible to develop a comprehensive procedure for synthesizing β-networks with specified fault tolerance and communication delay. A large inventory of designs with various combinations of fault tolerance and communication delay could then be prepared and used as a source of off-the-shelf designs.

4. There are still many implementation issues to be addressed, including the design of

procedures for fault diagnosis and fault recovery. Problems associated with the implementation of β-networks using VLSI technology also deserve further study.

Because of the continuing decrease of integrated circuit cost and the increase of chip complexity, there is likely to be a proliferation of multicomputer systems, many of which will use β-networks for intercomputer communication. Consequently, the study of β-networks is expected to increase in importance in the future.

BIBLIOGRAPHY

[Aardenne-Ehrenfest, T. and N.G. de Bruijn, 1951] "Circuits and trees in oriented linear graphs," *Simon Stevin*, vol. 28, pp. 203-217, 1951.

[Anderson, G.A. and E.D. Jensen, 1975] "Computer interconnection structures: taxonomy, characteristics, and examples," *Computing Surveys*, vol. 7, no. 4, pp. 197-213, December 1975.

[Avizienis, A., 1967] "Design of fault-tolerant computers," *Proc. AFIPS Conf.*, vol. 31, (1967 FJCC), pp. 733-743.

[Avizienis, A., et al. 1971] "The STAR computer: an investigation of the theory and practice of fault-tolerant computer design," *IEEE Trans. Computers*, vol. C-20, pp. 1312-1321, November 1971.

[Batcher, K.E., 1968] "Sorting networks and their applications," *Proc. AFIPS Conf.*, vol. 32, (1968 SJCC), pp. 307-314.

[Batcher, K.E., 1974] "STARAN parallel processor system hardware," *Proc. AFIPS Conf.*, vol. 43, (1974 NCC), pp. 405-410.

[Batcher, K.E., 1976] "The flip network in STARAN," *Proc. Parallel Processing Conf.*, pp. 65-71, August 1976.

[Benes, V.E., 1965] *Mathematical Theory of Connecting Networks and Telephone Traffic*, New York: Academic Press, 1965.

[Berge, C., 1973] *Graphs and Hypergraphs*, Amsterdam, North-Holland, 1973.

[Bott, R. and J.P. Mayberry, 1954] "Matrices and trees," *Economic Activity Analysis*, pp. 391-400, New York, Wiley, 1954.

[Cayley, A., 1895] "The theory of groups, graphical representation," *Mathematical Papers*, vol. 10, pp. 26-28, 1895.

[Clark, D.C., K.T. Pogran, and D.P. Reed, 1978] "An introduction to local networks," *Proc. IEEE*, vol. 66, pp. 1497-1517, November 1978.

[Clos, C., 1953] "A study of non-blocking switching networks," *Bell Sys. Tech. J.*, vol. 32, no. 2, pp. 406-424, March 1953.

[Davis, A., 1979] "A data flow evaluation system based on the concept of recursive locality," *Proc. ACM. Conf.*, pp. 1079-1086, June 1979.

[de Bruijn, N.G., 1946] "A combinatorial problem." *Nederl. Akad. Wetensch. Proc.*, vol. 49, pp. 758-764, 1946.

[Dennis, J.B., 1979] "The varieties of data flow computers," *Proc. First Int. Conf. Distributed Computing Sys.*, pp. 430-439, 1979.

[Despain, A.M. and D.A. Patterson, 1978] "X-tree: a structured multiprocessor computer architecture," *Proc. 5th Ann. Symp. Computer Architecture*, pp. 144-151, 1978.

[Deo, N., 1974] *Graph Theory with Applications to Engineering and Computer Science*, Englewood Cliffs, New Jersey: Prentice-Hall, 1974.

[Edmonds, J. and E.L. Johnson, 1973] "Matching, Euler tours and the Chinese postman," *Mathematical Programming*, vol. 5, pp. 88-124, 1973.

[Farber, D.J., 1975] "A ring network," *Datamation*, pp. 45-46, February 1975.

[Flynn, M.J., 1972] "Some computer organizations and their effectiveness," *IEEE Trans. Computers*, vol. C-21, pp. 948-960, September 1972.

[Friedman, A.D. and P.R. Menon, 1975] *Theory and Design of Switching Circuits*, Woodland Hills, California: Computer Science Press, 1975.

[Goke, L.R. and G.J. Lipovski, 1973] "Banyan networks for partitioning multiprocessor systems," *Proc. 1st Ann. Symp. Computer Architecture*, pp. 21-28, 1973.

[Good, I.J., 1946] "Normal recurring decimals," *J. London Math. Soc.*, vol. 21, pp. 167-169, 1946.

[Hafner, E.R., Z. Nendal, and M. Tschantz, 1974] "A digital loop communications system," *IEEE Trans. Commun.*, pp. 877-881, June 1974.

[Harary, F., 1969] *Graph Theory*, Reading, Massachusetts: Addison-Wesley, 1969.

[Hayes, J.P., 1978] *Computer Architecture and Organization*, New York: McGraw-Hill, 1978.

[Hopper, A. and D.J. Wheeler, 1979] "Binary routing networks," *IEEE Trans. Computers*, vol. C-28, pp. 669-703, October, 1979.

[Jensen, E.D., K.J. Thurber, and G.M. Schneider, 1976] "A review of systematic methods in distributed processor interconnection," *Proc. ICC*, pp. 7.17-17.22, 1976.

[Joel, A., Jr., 1968] "On permutation switching networks," *Bell Sys. Tech. J.*, vol. 47, no. 5, pp. 813-822, May-June 1968.

[Kapur, R.N., U.V. Premkumar, and G.J. Lipovski, 1980] "Organization of the TRAC processor-memory subsystem," *Proc. AFIPS Conf.*, vol. 49, (1980 NCC), pp. 623-629, May 1980.

[Lawrie, D.H., 1975] "Access and alignment of data in an array processor," *IEEE Trans. Computers*, vol. C-24, pp. 1145-1155, December 1975.

[Lee, S.C., 1976] "Vector boolean algebra and calculus," *IEEE Trans. Computers*, vol. C-25, pp. 865-874, September 1976.

[Leung, C.K.C. and J.B. Dennis, 1980] "Design of a fault-tolerant packet communication computer architecture," *Proc. Int. Symp. Fault-Tolerant Computing*, pp. 328-335, August 1980.

[Levitt, K.N., M.W. Green, and J. Goldberg, 1968] "A study of the data commutation problems in a self-repairable multiprocessor," *Proc. AFIPS Conf.*, vol. 32, (1968 SJCC), pp. 515-527.

[Lipovski, G.J. and K.L. Doty, 1978] "Developments and directions in computer architecture," *Computer*, pp. 54-67, August 1978.

[Lipovski, G.J. and M. Malek, 1981] "A theory for multicomputer interconnection networks," University of Texas, Austin, Report to be published.

[Lundstrom, S.F. and G.H. Barnes, 1980] "A controllable MIMD architecture," *Proc. Parallel Processing Conf.*, pp. 19-27, 1980.

[Masson, G.M., G.C. Gingher, and S. Nakamura, 1979] "A sampler of circuit switching networks," *Computer*, vol. 12, No. 6, pp. 32-48, June 1979.

[Mayeda, W., 1972] *Graph Theory*, New York: Wiley-Interscience, 1972.

[Opfermann, D.C. and N.T. Tsao-Wu, 1971] "On a class of rearrangeable switching networks, part II: enumeration studies and fault diagnosis," *Bell Sys. Tech. J.*, pp. 1601-1618, May-June 1971.

[Parker, D.S., Jr., 1980] "Notes on shuffle/exchange-type switching networks," *IEEE Trans. Computers*, vol. C-29, pp. 213-222, March 1980.

[Patel, J.H., 1979] "Processor-memory interconnections for microprocessors," *Proc. 6th Ann. Symp. Computer Architecture*, pp. 168-177, April 1979.

[Pease, M.C., 1977] "The indirect binary n-cube microprocessor array," *IEEE Trans. Computers*, vol. C-26, pp. 458-473, May 1977.

[Pierce, R.A. and D.H. Moore, 1977] "Network operating system functions and microprocessor front-ends," *Proc. COMPCON*, pp. 325-328 Spring 1977.

[Premkumar, U.V., et al, 1980] "Design and implementation of the Banyan interconnection network in TRAC," *Proc. AFIPS Conf.*, vol. 49, (1980 NCC), pp. 643-653.

[Rattner, J.R., 1977] "Microprocessor architecture-where do we go from here?" *Proc.COMPCON*, pp.223-224,Spring 1977.

[Russo, P.M., 1977] "Interprocessor communication for multi-microcomputer systems," *Computer*, vol. 10, No. 4, pp. 67-76, April 1977.

[Senjowski, M.C., et al., 1980] "An overview of the Texas reconfigurable array computer," *Proc. AFIPS Conf.*, vol. 49 (1980 NCC), pp. 631-641, May 1980.

[Shen, J.P. and J.P. Hayes, 1980] "Fault tolerance of a class of connecting networks," *Proc. 7th Ann. Symp. Computer Architecture*, pp. 61-71, 1980.

[Siegel, H.J. and S.D. Smith, 1978] "Study of multistage SIMD interconnection networks," *Proc. 5th Ann. Symp. Computer Architecture*, pp. 223-229, 1978.

[Siegel, H.J., 1979] "Interconnection networks for SIMD machine," *Computer*, vol. 12, No. 6, pp. 57-65, June 1979.

[Sowrirajan, S. and S.M. Reddy, 1980] "A design for fault-tolerant full connection networks," to be published.

[Squire, J. and S. Palais, 1963] "Programming and design considerations of a highly parallel computer," *Proc. AFIPS Conf.*, vol. 23, (1963 SJCC), pp. 395-400.

[Stone, H.S., 1971] ""Parallel processing with the perfect shuffle," *IEEE Trans. Computers*, vol. C-20, pp. 153-161, February 1971.

[Sullivan, H. and T.R. Bashkow, 1977] "A large scale, homogeneous, fully distributed parallel machine, I," *Proc. 4th Ann. Symp. Computer Architecture*, pp. 105-117, March 1977.

[Swan, R.J., S.H. Fuller, and D.P. Siewiorek, 1977] "CM* - a modular, multi-microprocessor," *Proc. AFIPS Conf.*, vol. 46, (1977 NCC), pp. 637-644.

[Tarjan, R., 1972] "Sorting using networks of queues and stacks," *JACM*, vol. 19, pp. 341-346, April 1972.

[Thanawastien, S. and V.P. Nelson, 1981] "A generalized control algorithm for the omega connection network to support fault tolerant operations in a random access environment." submitted to *IEEE Trans. Computers.*

[Thurber, K.J. and G.M. Masson, 1977] "Recent advances in microprocessor technology and their impact on interconnection design in computer systems," *Proc. ICC*, pp. 46.2.216-46.2.220, 1977.

[Tripathi, A.R. and G.J. Lipovski, 1979] "Packet switching in Banyan networks," *Proc. 6th Ann. Symp. Computer Architecture*, pp. 160-167, April 1979.

[Waksman, A., 1968] "A permutation network," *JACM*, vol. 15, pp. 159-163, January 1968.

[Watson, I. and J. Gurd, 1979] "A prototype data flow computer with token labeling," *Proc. ACM Conf.*, pp. 623-628, 1979.

[Wilkes, M.V., 1975] "Communication using a digital ring," *Proc. Pacific Area Computer Communication Network Sys. Symp.*, pp. 47-56, August 1975.

[Wu, C.L. and T.Y. Feng, 1978] "Routing techniques for a class of multistage interconnection networks," *Proc. Parallel Processing Conf.*, pp. 197-205, August 1978.

[Wu, C.L. and T.Y. Feng, 1980] "The reverse-exchange interconnection network," *IEEE Trans. Computers*, vol. C-29, pp. 801-811, September 1980.

[Zafiropulo, P., 1974] "Performance evaluation of reliability improvement techniques for single-loop communications systems," *IEEE Trans. Communications*, pp. 742-751, June 1974.