Self Routing in the Fault Tolerant Modification of the Benes Network

Maliha Mahboob 442 5416

ASSUMPTION UNIVERSITY
FACULTY OF ENGINEERING

COMPUTER ENGINEERING PROJECT I

CE 4901

PROJECT I REPORT

PROJECT TITLE:

SELF ROUTING IN THE FAULT TOLERANT MODIFICATION OF THE BENES NETWORK

SUBMITTED TO:

DR. GENNADY VESELOVSKY

STUDENT NAMES:	IDs:
MALIHA MAHBOOB	4425416
MOHAMMAD HASANUL KARIM	4425417

DATE OF SUBMISSION: 19.09.03

ABSTRACT

This project is about self-routing in the fault tolerant modification of the Benes network. Benes network is a non-blocking multistage interconnection network constructed recursively from exchange elements. Fault tolerance is defined as the ability of a system to execute specified algorithms correctly regardless of hardware failures and program errors by having multiple copies of critical hardware components or subsystems. In the Benes network, there is more than one independent path for each input-output pair, so that in principle it is possible to bypass single faults and to realize any given permutation in two passes without extra stages. If the first and last stages are augmented with multiplexors and demultiplexors, then the modified Benes network will be fault tolerant to single faults. A self-routing permutation network is a connector, which can set its own switches to realize any one-to-one mapping of its inputs onto its outputs. A self-routing switch routes a message to its destination using only the information contained in the message without requiring knowledge about other connections. In this project we will implement the self-routing algorithm in fault tolerant Benes network for certain permutations.

INTRODUCTION

The term parallel processing refers to a large class of methods that attempt to increase computing speed by performing more than one computation concurrently. A parallel processor is a computer than implements some parallel processing technique. All modern computers involve some degree of parallelism. Two major types of parallel computers are considered: pipeline (vector) processors and multiprocessors.

MULTIPROCESSOR SYSTEMS

There has been a debate on whether one fast processor would be faster and more cost effective than a system with more than one slower but less expensive processor. However now it is apparent that multiple processors must be used to obtain improvements in speed even though the difficulties of using more than one processor on a single problem have not been resolved. Multiprocessor systems are also designed to gain fault tolerance, i.e. to be able to continue operating in the presence of hardware (and possibly software) faults. Hardware fault tolerance is achieved by the addition of circuits that are not necessary for normal operation but that enable the system to continue if the faults occur. The number of faults that can be present is limited and dependent upon the system design. Specific applications in which computer systems must continue operating for as long as possible or over an initial period of time. For example, the computer system in an aircraft must continue working while operating with aircraft, and a faulty system could lead to losses of life. Fault tolerance is also extremely important in the military areas, commercial field and manufacturing plants.

MULTIPROCESSOR CLASSIFICATIONS

A number of taxonomies of parallelism have been proposed. The earliest is probably Flynn's originally proposed in 1966. A normal single processor stored program computer (von Neumann computer) generates a single stream of instructions, which acts upon single data items. Flynn called this type of computer a **single instruction stream-single data stream (SISD)** computer. In a general-purpose multiprocessor system, one instruction stream is generated for each processor. Each instruction acts upon different data. Flynn called this type of computer a **multiple instruction stream-multiple data stream (MIMD)** computer. Apart from these two, there are computers in which a single instruction stream is generated by a single control unit and the instructions are broadcast to more than one processor. Each processor executes the same instruction, but using different data. The data items form a vector and the instructions act upon the complete vector in one instruction cycle. Flynn called this type of computers **a single instruction stream-multiple data stream (SIMD)** computer. The fourth combination, **multiple instruction stream-single data stream (MISD)** computer does not exist, unless one specifically classifies pipelined architectures in this group or possibly some fault tolerant systems.

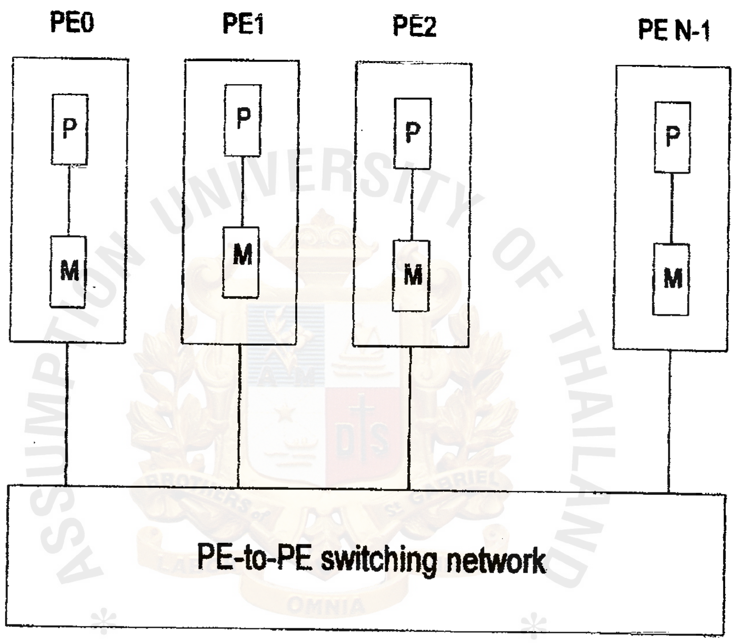


Figure: Distributed Memory System

Single instruction stream-multiple data stream (SIMD) multiprocessor computing systems occupy a central place among parallel computing system architectures. In these systems, N processors or "processor elements" (PE) execute the same instruction with different operands. SIMD computers are vector machines with speed currently in the range of billions and tens of billions floating point arithmetic operations per second; the number of processors in the system may reach several thousands. One of the main difficulties in SIMD computers, as in other parallel systems is timely delivery of data to all processors. Efficient choice of switching networks helps to overcome this difficulty.

Two basic switching structures are available for SIMD computers. In the first structure, each processor has its own individual memory M and is linked by a switching network to other identical processors. In the second structure, the switching network is interposed between the processors and the shared memory modules M of all the processors.

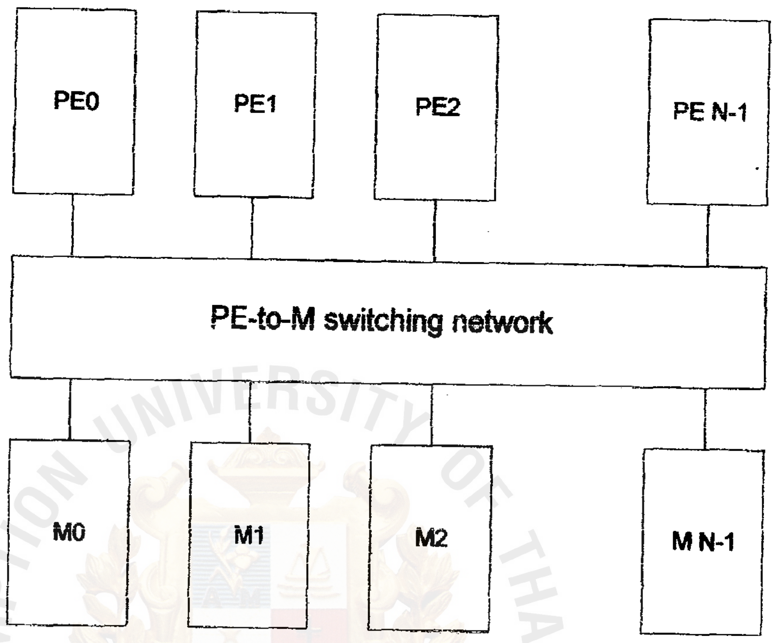


Figure: Shared Memory Systems

The advantage of the second switching structure compared with the first is that each processor has access to the entire addressing space of the shared memory. The main shortcomings are additional delay introduced by the switching network when accessing the memory and the possibility of collision when two or more processors try to access the same memory module.

INTERCONNECTION NETWORK

Interconnection networks (also known as permutation networks) are used for regular interconnections of processors in a parallel computer. Such a network consists of N nodes, also known as switches, each of which has k input and output lines.

Interconnection networks can be divided into two categories: dynamic and static. Dynamic connection means the path between two entities (PE to memory or PE to PE) may change from one communication to next; static connection means the network topology stays the same all the time.

Interconnection networks are critical to parallel systems because their performances are closely related to network latency and throughput.

An interconnection network of size N, where $N = 2^n$, for some n, $n \geq 0$, has n + 1 stages numbered 0 through n (the stages will appear in increasing order from left to right in the figures). Each stage has N nodes. Nodes in stage 0 are called initial nodes and those in stage n are final nodes. Each non-initial node has two input ports, known as top and bottom. Each non-final node has two output ports known as top and bottom. Each output port of a node in stage i is connected to a distinct input port of a node in stage i + 1, $0 < i < n$.

CLASSIFICATION OF INTERCONNECTION NETWORKS

BY CONNECTIVITY

1. Blocking
2. Non Blocking
3. Rearrangable

1. Blocking Network

A blocking network is always formed by some specified permutation and do not allow nay permutation. Examples are Crossbar, Clos network and Benes network

2. NonBlocking Network

A non-blocking network is capable of providing N parallel paths between pairs of nodes forming any arbitrary permutation. Examples are Shuffle Exchange, Omega network and Hypercube network.

- "Strictly non-blocking": If it is always possible to connect an idle inlet to an idle outlet, regardless of the state of the network.
- "Non-blocking in wide sense": If by suitable choosing routes for new calls it is possible to avoid all the blocking states and still to satisfy all demands for connection as they arise, without having to rearrange existing calls.

3. Rearrangable Network

A rearrangeable network is an interconnection network, which can achieve all possible permutations of its inputs' connections to its outputs. In a rearrangable network, a blocked call can always be unblocked by assigning new routes to the call in progress. One class of rearrangeable networks, are Clos three-stage networks. Earlier procedures to route such networks rapidly require an excessive amount of hardware, either in the network itself or in the device required to compute the routing.

BY TOPOLOGY

Topology is the structure of the interconnection network, which determines

–Degree: number of links from a node
–Diameter: max number of links crossed between nodes
–Average distance: number of hops to random destination
–Bisection: minimum number of links that separate the network into two halves.

Interconnection networks can be divided into two categories by topology:

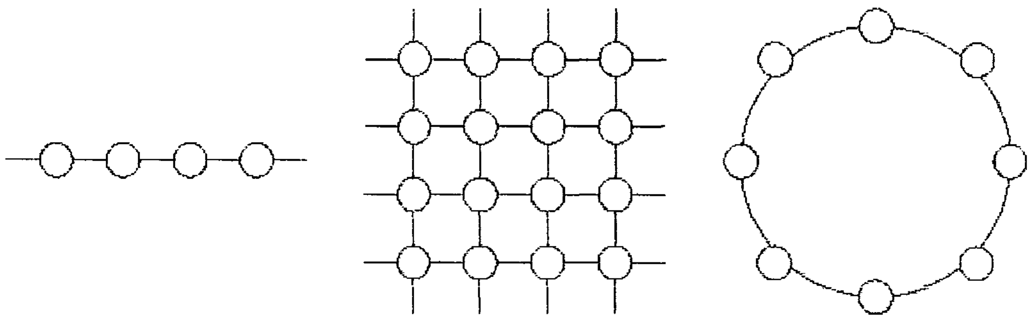
1. Static Topologies
2. Dynamic Topologies

STATIC TOPOLOGIES

Static connection means that the network topology stays the same all the time. The static network is characterized by node degree, number of links (edges) connected to the node.

Meshes and Rings

The simplest - and cheapest - way to connect the nodes of a parallel computer is to use a one-dimensional mesh. Each node has two connections, boundary nodes have one. If the boundary nodes are connected to each other, we have a ring, and all nodes have two connections. The one-dimensional mesh can be generalized to a *k*-dimensional mesh, where each node (except boundary nodes) has 2*k* connections. Again, boundary nodes can be connected, but there is no general consensus on what to do on boundary nodes.



However, this type of topology is not suitable to build large-scale computers, since the maximum message latency, that is, the maximum delay of a message from one of the N processors to another, is $\sqrt{N}$; this is bad for two reasons: firstly, there is a wide range of latencies (the latency between neighboring processors is much lower than between not-neighbors), and secondly the maximum latency grows with the number of processors.

Stars

In a star topology there is one central node, to which all other nodes are connected; each node has one connection, except the center node, which has N-1 connections.

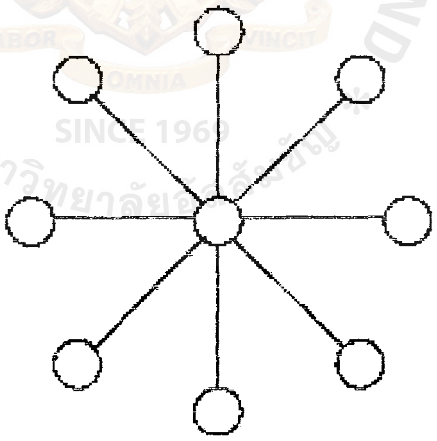


Figure: Star Topology

Stars are also not suitable for large systems, since the center node will become a bottleneck with increasing number of processors.

Hypercubes

The hypercube topology is one of the most popular and used in many large-scale systems. A k-dimensional hypercube has 2^k nodes, each with k connections.

Hypercubes scale very well, the maximum latency in a k-dimensional hypercube is $\log_2 N$, with $N = 2^k$. An important property of hypercubes is the relationship between node-number and which nodes are connected together. The rule is, that any two nodes in the hypercube, whose binary representations differ in exactly one bit, are connected together. For example in a four-dimensional hypercube, node 0 (0000) is connected to node 1 (0001), node 2 (0010), node 4 (0100) and node 8 (1000). This numbering scheme is called Gray code scheme.

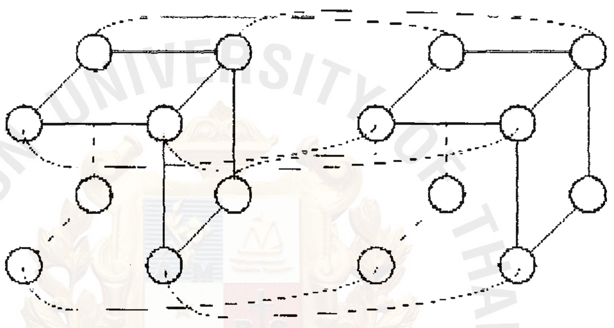


Figure: 4-D Hypercube Topology

ROUTING ALGORITHMS

Routing algorithm determines

- which of the possible paths are used as routes
- how the route is determined

Routing Delay

Routing delay is a function of the number of channels (hops) on the route, i.e. routing distance h, delay D incurred at each switch as part of the output port selection. Routing distance depends on network topology, routing algorithm, and source and destination node.

Meshes and Rings

Typically, in meshes the so-called dimension-order routing technique is used. That is, routing is performed in one dimension at a time. In a three-dimensional mesh for example, a message's path from node (a, b, c) to the node (x, y, z) would be moved along the first dimension to node (x, b, c), then, along the second dimension to node (x, y, c), and finally, in the third dimension to the destination-node (x, y, z).

Stars

Routing in stars is trivial. If one of the communicating nodes is the center node, then the path is just the edge connecting them. If not, the message is routed from the source node to the center node, and from there to the destination node.

Hypercubes

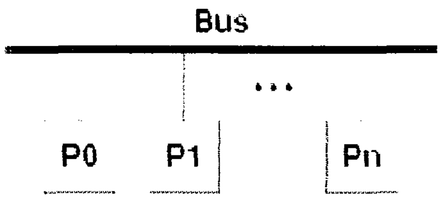
A k-dimensional hypercube is nothing else than a k-dimensional mesh with only two nodes in each dimension, and thus the routing algorithm is the same as for meshes; apart from one difference: the path from node A to node B is calculated by simply calculating the exclusive-or $X = A \oplus B$ from the binary representations for node A and B. If the i-th bit in X is '1' the message is moved to the neighboring node in the i-th dimension. If the i-th bit is '0' the message is not moved anyway. This means, that it takes at most $\log_2 N$ steps for a message to reach its destination (where N is the number of nodes in the hypercube).

DYNAMIC TOPOLOGIES

Dynamic connection means the path between two entities (PE to memory or PE to PE) may change from one communication to next.

SINGLE-STAGE NETWORKS

Buses and crossbars are the two main representatives of this class. A bus is the simplest way to connect a number of processors with each other: all processors are simply connected to one wire. This makes communication and especially message routing very simple. The drawback of this type of network is, that the available bandwidth is inversely proportional to the number of connected processors. This means, that buses are good only for small networks with a maximum of about 10 processors.



The other extreme in terms of complexity is the crossbar network. With a crossbar full connectivity is given, i.e. all processors can communicate with each other simultaneously without reduction of bandwidth. In the figure, the connection of n processors with m memory modules (as in a shared memory system) is shown. Certainly crossbars can also be used to connect processors with each other. In that case the memory modules are connected directly to the processors (which results in a distributed memory system), and the lines that were connected to the memory modules M_i are now connected to the processors P_i.

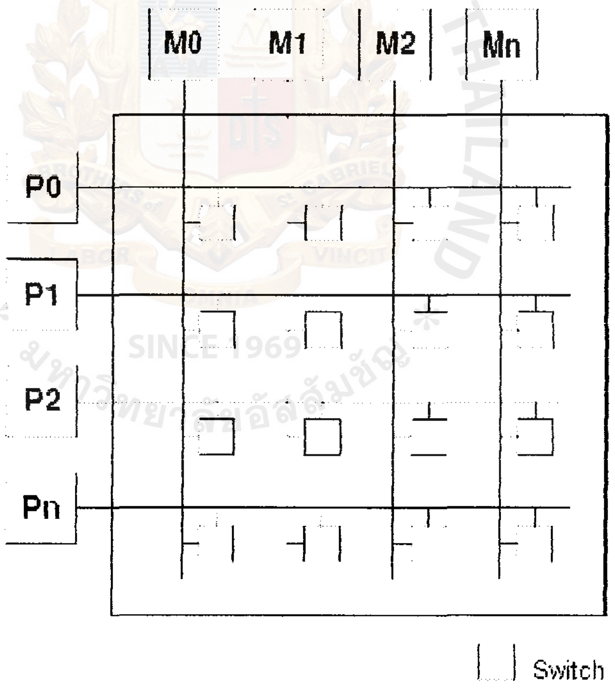
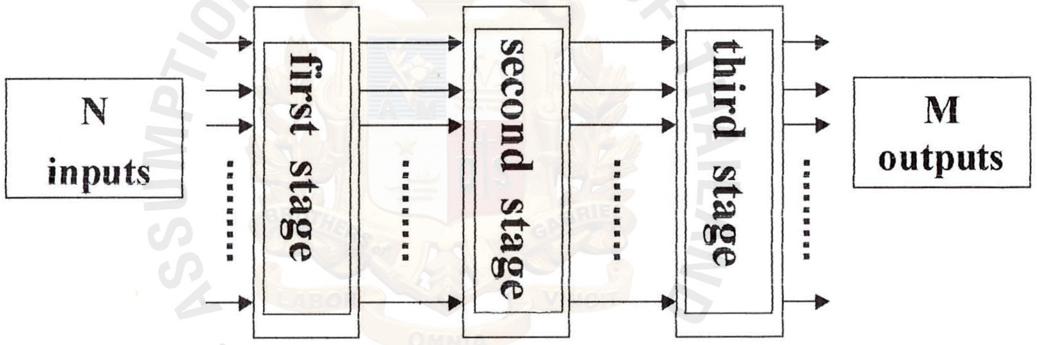


Figure: Crossbar Network

To connect n processors to n memory modules n^2 switches are needed. Consequently, crossbar networks cannot be scaled to any arbitrary size. Today's commercially available crossbars can connect up to 256 units.

MULTISTAGE INTERCONNECTION NETWORKS

Multistage interconnection networks (MINs) provide more cost-effective communication than crossbar networks and higher-bandwidth communication than bus systems. Multistage switching networks are designed for both synchronous and asynchronous parallel systems with a large number of PEs (hundreds or more). An N x N multistage switching network is a communication network with N input terminals (sources) and N output terminals (destinations) composed of a certain number of stages of switching elements (with 2 X 2 switching elements, the number of stages is usually not less than $\log_2 N$). Each stage is connected to the next stage by at least N data transmission lines between any source-destination pair. Each switching element may select from two or more output lines when establishing a connection with an input line.



Three-stage MIN

An optimal switching element has two inputs and two outputs. It has been shown theoretically that such an element ensures the least number of connection points. Moreover control algorithms for networks of two input elements.

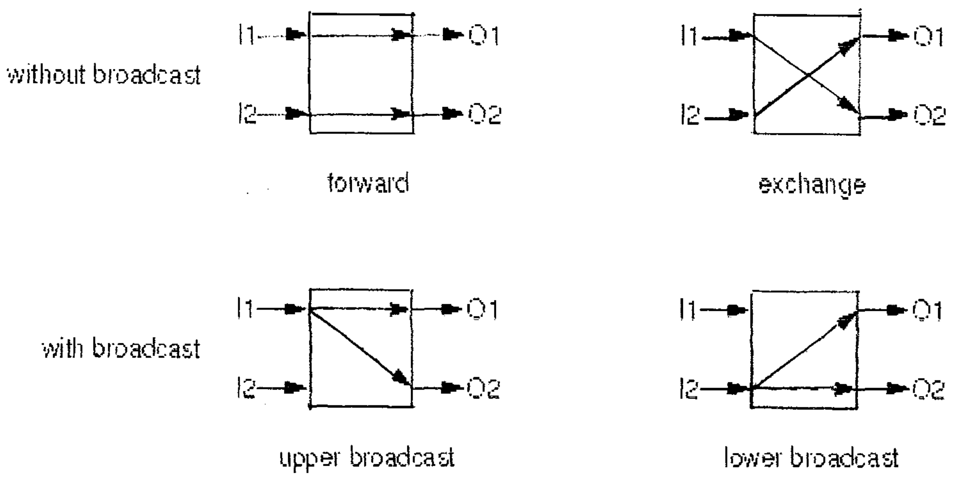


Figure: Shuffle-Exchange Switching Elements

The figure shows possible states for a 2 X 2 switch. With integrated circuit implementation of such switches, their input length may vary from one bit to one word.

Important network properties include blocking of information in the network; speed, i.e. the rate of transmission of a message from the source to the destination; ease of use, i.e. the degree to which the connections are automatically established in the network; partitionability, i.e. the possibility of partitioning the system into subsystems of different size; modularity; i.e. the possibility of constructing the system from a limited number of LSI chip; extensibility; i.e. the possibility of extending a given system to a larger size or in other words the amount of changes required to make the system work with great number of inputs and outputs; fault tolerance; i.e. the ability of the system to remain functional even when some components are faulty.

One of the most important parameters of a switching network is the connectivity or combinatorial power C. this is defined as the ratio of the number of permutations r realized by a network to the total number of permutations of N elements, which is equal to N!

$$C = r/N! \text{ where } 1 \leq r \leq N!$$

Other conditions being equal, the greater the parameter C, the higher is the MSN throughput. Another network parameter is Q, the average number of source-destination pairs that can simultaneously exchange information, assuming that all the N! permutations are equiprobable. An important technological parameter of the network is the switch bandwidth B, which depends on the interconnection of the switch components. Another parameter is the message delay D in the switching network. T-the broadcast scope characterizes the ability to transmit information from one source simultaneously to a group of destinations.

BASIC TYPES OF DATA TRANSFER

The choice of the switching network depends on the frequently used types of messages. The main types of permutations belong either to BPC or to Omega class. Such permutations usually have names and they are listed together with their names as below where each equation shows the mapping of a source to the destination.

BPC Permutations

A permutation is called a BPC permutation if the destination tag can be obtained from the source tag by permuting the bits in the source address ($s_{n-1}\ s_{n-2}\ \ldots\ s_1\ s_0$) and/or complementing some or all of its bits positions. The class BPC (n), $N=2^n$, only contains N(logN)! of the possible N! permutations. Nevertheless, many of the permutations encountered in parallel algorithms are included in this class. For example, Lenfant identifies five families of "frequently used bijections" (FUB). Three of these FUB families (namely $\alpha^{(n)}$, $\beta^{(n)}$, $\gamma^{(n)}$) are included in this BPC (n).

Shuffle:

$$\Pi_{\text{shuffle}} = (s_0\ s_{n-1}\ \ldots\ s_2\ s_1)$$

Vector Reversal:

$$\Pi_{\text{vector reversal}} = (\overline{s}_{n-1}\ \overline{s}_{n-2}\ \ldots\ \overline{s}_1\ \overline{s}_0)$$

Butterfly:

$$\Pi_{\text{butterfly}} = (s_0\ s_{n-2}\ \ldots\ s_1\ s_{n-1})$$

Exchange:

$$\Pi_{\text{exchange}} = (s_{n-1}\ s_{n-2}\ \ldots\ s_{i+1}\ \overline{s}_i\ s_{i-1}\ \ldots\ s_1\ s_0)$$

Flip:

$\Pi_{flip} = s \oplus \delta$ where $1 \leq \delta \leq 2^n - 1$

Unshuffle Permutation:

$\Pi_{unshuffle} = (s_{n-2}\, s_{n-3} \ldots s_0\, s_{n-1})$

Bit Reversal:

$\Pi_{bit\ reversal} = (s_0\, s_1 \ldots s_{n-2}\, s_{n-1})$

Matrix Transposition:

$$\Pi_{matrix\ transposition} = \begin{cases} s_{l-1} \ldots s_1\, s_0\, s_{2l-1} \ldots s_{l+1}\, s_l & \text{if } n = 2l \\ s_{l-1} \ldots s_1\, s_0\, s_{2l} \ldots s_{l+1}\, s_l & \text{if } n = 2l+1 \end{cases}$$

Bit Shuffle:

$$\Pi_{bit\ shuffle} = \begin{cases} s_{n-1} \ldots s_3\, s_1\, s_{n-2} \ldots s_2\, s_0 & \text{if } n = 2l \\ s_{n-2} \ldots s_3\, s_1\, s_{n-1} \ldots s_2\, s_0 & \text{if } n = 2l+1 \end{cases}$$

Shuffle Row Major:

$$\Pi_{\text{shuffle row major}} = \begin{cases} s_{2l-1}s_{l-1} \ldots s_{l+1}\, s_l\, s_l\, s_0 & \text{if } n = 2l \\ s_l\, s_{2l}\, s_{l-1} \ldots s_{l+2}\, s_l\, s_{l+1}\, s_0 & \text{if } n = 2l+1 \end{cases}$$

Omega Permutations

Lawrie has defined the class of omega permutations to consist of exactly those permutations realizable by omega network. Inverse omega permutations are realizable using an omega network backwards. The followings are some examples of inverse Omega permutations:

Cyclic Shift Of Amplitude k:

$\Pi_{\text{cyclic shift of amplitude k}} = (js + k) \bmod 2^n$ where $1 \leq k \leq 2^n$, and j is odd

Cyclic Shift Within Segments:

$\Pi_{\text{cyclic shift within segments}} = \delta \oplus (s+k) \bmod 2^{n-j}$ where $1 \leq k \leq 2^n$, and δ is the number equivalent to the j most significant bits in the binary representation of s.

Unscrambling j-Ordered Vectors:

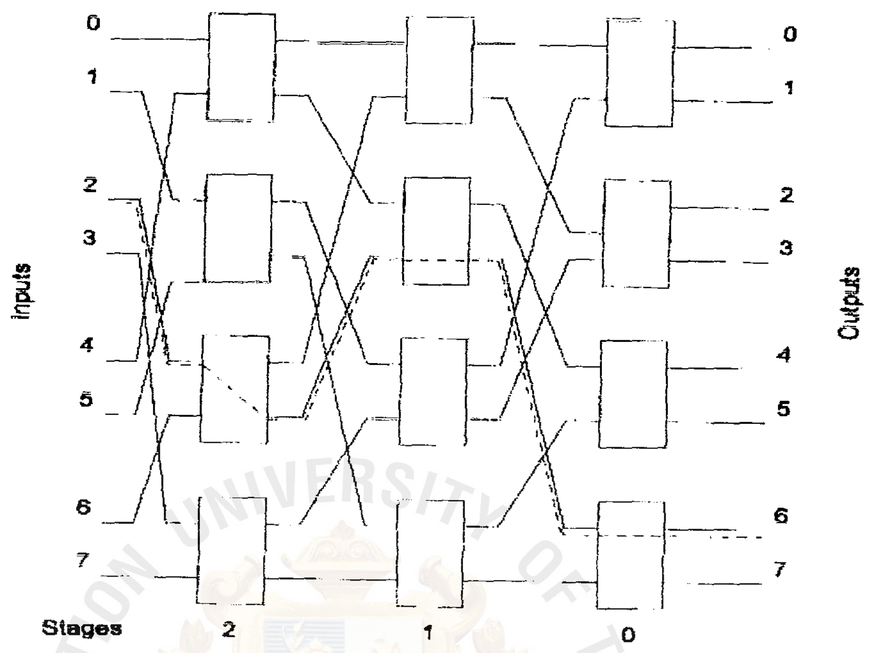
$\Pi_{\text{unscrambling j-ordered vectors}} = (js) \bmod 2^k \oplus (s_k \ldots s_{n-2}\, s_{n-1})\, 2^k$ where $1 \leq k \leq n$, and j is odd

MULTISTAGE SWITCHING NETWORKS WITH BLOCKING

Multistage switching networks for SIMD computers can be divided into two main categories: blocking networks, in which certain permutations may lead to collision and non-blocking networks. For blocking networks we obviously have C<1, while for non-blocking networks C=1.

Blocking networks for SIMD computers usually allow self-tuning or distributed control, while non-blocking networks require centralized control, and the setup of such network is very time consuming. This accounts for the considerable popularity of blocking networks in SIMD computers.

Typical representatives for blocking networks are Omega network and the n-cube network. An 8 X 8 omega network is shown in the figure.



The 8x8 omega network

Each stage in this network implements the interconnections by the ideal shuffle scheme. This network is defined by two functions: the shuffle function and the exchange function. The shuffle function is

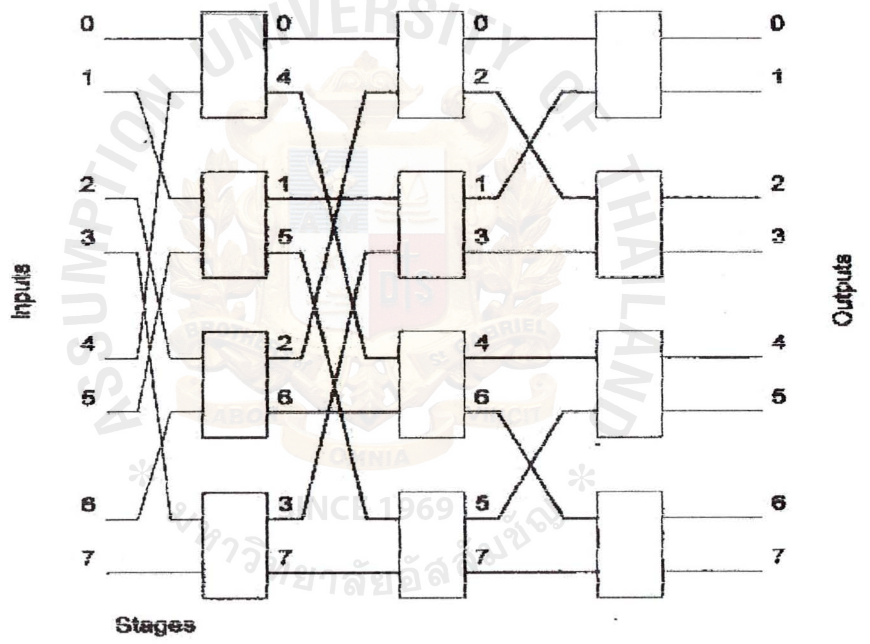
$$\text{shuffle}\,(s_{n-1}\, s_{n-2} \ldots s_1\, s_0) = s_{n-2}\, s_{n-3} \ldots s_1\, s_0\, s_{n-1}$$

The exchange function is defined as:

$$\text{exchange}\,(s_{n-1}\, s_{n-2} \ldots s_1\, s_0) = (s_{n-1}\, s_{n-2} \ldots s_1\, s_0)$$

This network connects 2^N nodes using N stages, with each stage containing 2^{N-1} switches. Successive stages are interconnected in a pattern called the perfect shuffle. The Omega network is a blocking network, because some messages may be blocked by the settings required for other messages. In addition, it is not fault tolerant; if a single switch fails; some connections are no longer possible. By adding additional stages, we may increase fault tolerance and reduce or eliminate blocking. The omega network has the problems with output port contention and path contention. Path contention and output port contention can seriously degrade the achievable throughput of the switch.

The n-cube network belongs to the same network as omega network. It derives it names from its geometrical interpretation as n-dimensional cube whose vertices are the processor addresses in binary representation. An 8 X 8 cube is shown in the figure. The synthesis algorithm for the cube network is quite simple. If the upper and lower outputs for each switch have the same addresses as the upper and lower inputs, then the i-th stage pairs on the switch inputs those input lines whose addresses differ only in the i-th position(in the binary representation of the terminal addresses). Functionally the cube is equivalent to the omega network, i.e. it realizes without collision the same classes of permutations and has the same control algorithm.



The 8x8 n-cube network

For a network with N= 2^n inputs and binary representation of the input address in the form $g_{n-1}g_{n-2} \dots g_1g_0$, the cube network may be regarded as consisting of n functions:

$$\text{cube}_i(g_{n-1} \dots g_{i-1}g_ig_{i+1} \dots g_1g_0) = g_{n-1} \dots g_{i-1}\bar{g}_ig_{i+1} \dots g_1g_0$$

Cube and Omega are essentially topological modifications of the same network. The same class of networks also includes the so-called indirect cube (the full name is "indirect binary n-cube"). In the indirect binary n-cube network, the sequence of stages is reversed compared to the direct cube. The indirect cube is functionally equivalent to the direct cube with all input and output addresses transformed from $g_{n-1}g_{n-2} \ldots g_1g_0$ to $g_0g_1 \ldots g_{n-1}$.

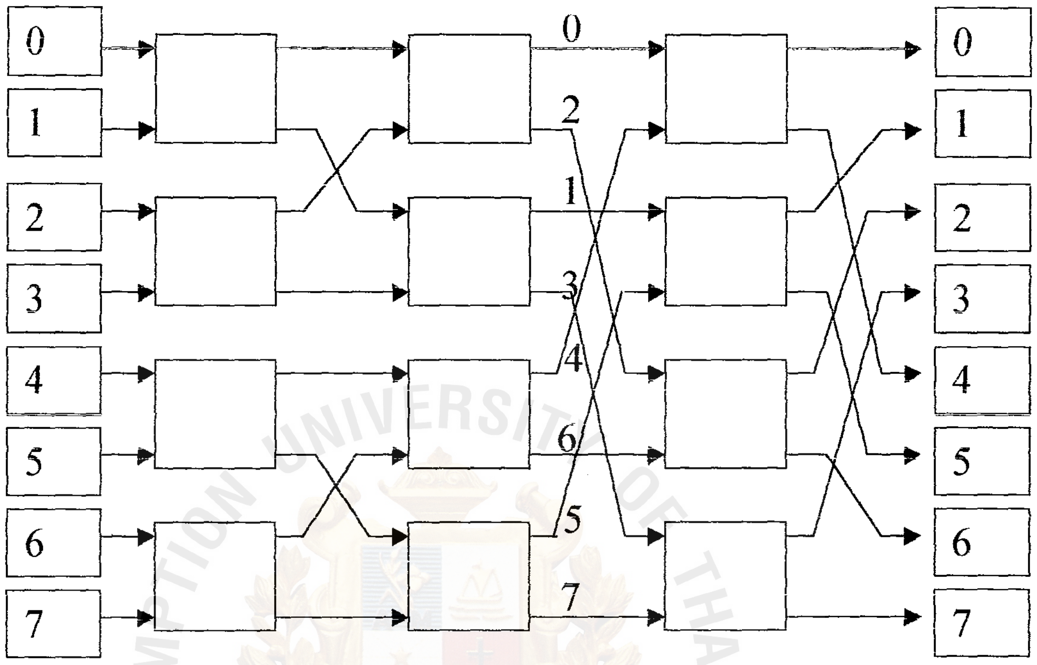


Figure: The 8 X 8 Indirect Cube Network

From the list of frequently used permutations, the cube network and its modifications realize without collision all the arbitrary shifts, including shifts within segments, but neither ideal shuffle nor reversal can be realized in one pass through the network, i.e. these networks fail.

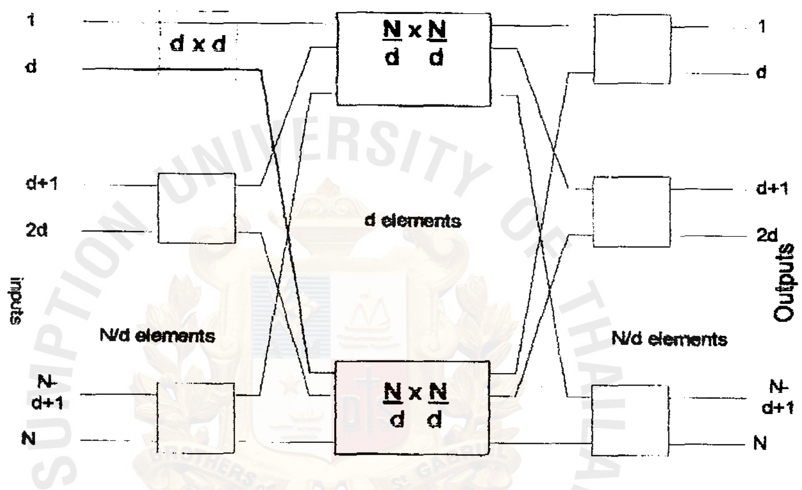
Banyan networks are topologically equivalent to cube networks. However banyan networks differ from the cube by their use of the switching elements, which have six, and not four, admissible states.

A fundamental property of different networks considered above is that they all lay a unique path from each input to output. This produces intersecting routes and may cause collision in the network.

THE BENES NETWORK AND ITS MODIFICATIONS

A second class of multistage switching networks that are of considerable practical interest is so called Clos-Benes non-blocking network representing a fairly wide class of network topologies. The objective was to develop a multistage network realizing crossbar switching functions, i.e. no blocking, yet having fewer connection points than in crossbar networks.

Let N=dq, where d and q are integers. Then the NXN Clos network is representable by three stages.

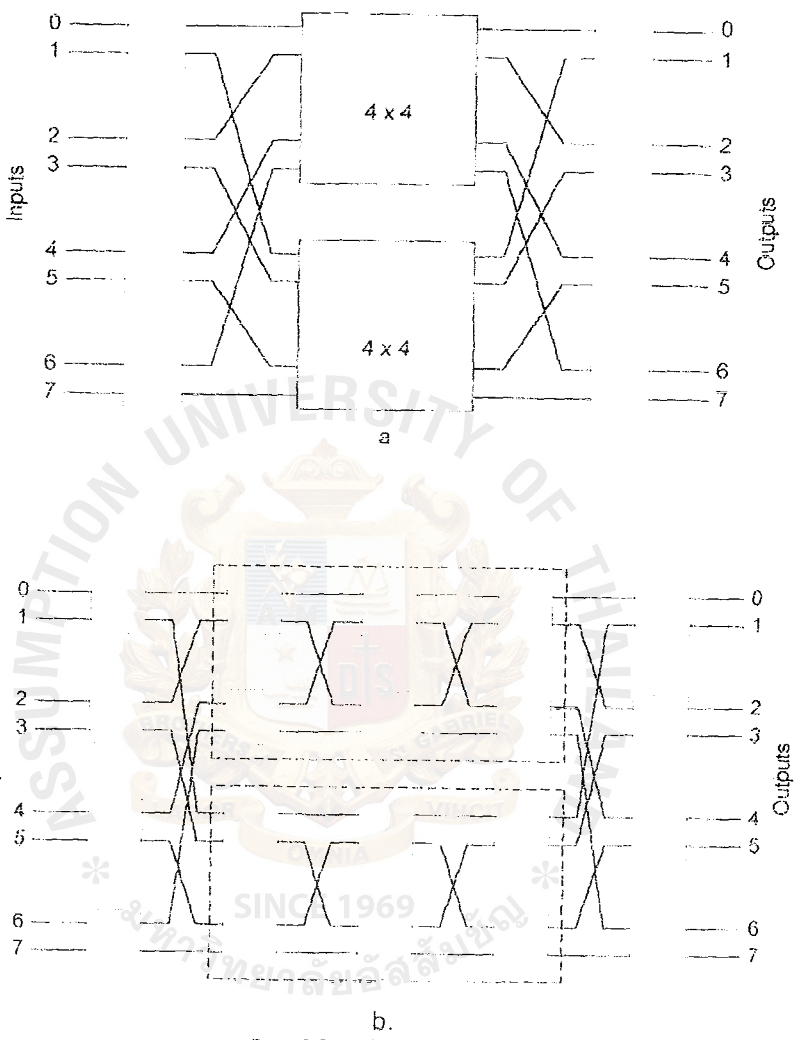


Base-d N x N Clos network.

The input and output switches contain N/d dxd subnetwork each. The intermediate stage consists of d identical N/d x N/d subnetwork. Such a network is usually termed as base-d network.

Such networks are fairly efficient in terms of hardware costs and are known in the literature as rearrangable because in case of blocking there is always a possibility of realizing the desired connection by rearranging some of the previously established connections. For these networks there are algorithms for collision free realization of arbitrary permutations with simultaneous establishment of all the specified connections in this network. This is typically of the class of synchronous systems, which includes the SIMD computer systems as a particular case. Thus for rearrangable networks C=1.

For d=2, successive decomposition of the middle stage into three stage structures produces a network implemented using only two input elements. The network constructed in such a way is called Benes network. The figure shows a base-2 8 X 8 Benes network.



Base-2 8 x 8 Benes network.

With N= 2^n inputs, the Benes network has $2\log_2 N - 1$ stages. Its main shortcomings as compared with cube networks is the need for centralized control; moreover it uses a complex and essentially sequential algorithm in order to computer the collection of signals controlling subnetwork switching (the "control vector" in what follows) during the realization of an arbitrary permutation.

Characteristics Of Benes Network

- This is a multistage network constructed recursively from exchange elements.
- The Benes network is non-blocking.
- It has $2\log_2 N-1$ columns and N/2 exchange elements.
- The Benes network is harder to control than a crossbar.
- The Benes network can perform any permutation of N inputs.

Construction Of The Benes Network

- A general permutation can be described as a set of N (source, destination) pair.
- Let us consider the set (0,0), (1,1),... as the permutation in order to construct the Benes network.
- For each level of recursion, we construct one column of N/2 (input) exchange elements followed by two Benes networks of size N/2 inputs and output (upper and lower N/2 permutations), followed by one column of N/2 (output) exchange elements.
- We pick one source and set the input exchange box to send it to the upper N/2 permutations.
- Using the upper permutation, we connect its output to the output exchange box, which is associated with the correct destination.
- We set the output exchange box to connect to the correct destination.
- Now working backwards, we connect the paired destination (the second destination on the output exchange box) to the lower N/2 permutation.
- We find the source pair matching this destination and set the lower permutation to connect it to the correct input exchange box (for the source).
- We set the input exchange box to connect to the correct input.

LOOPING ALGORITHM:

The so-called looping algorithm is the basic for interconnection control in Benes network. We assume that the Benes is required to realize the permutation o: 0 →3, 1→7, 2→4, 3→0, 4→2, 5→6, 6 →1,7→5 which is usually represented in the compact form:

$$p = \begin{pmatrix} 0\,1\,2\,3\,4\,5\,6\,7 \\ 3\,7\,4\,0\,2\,6\,1\,5 \end{pmatrix}$$

First this permutation should be described as an interconnection map. The pair of network inputs that belong to the same switching element of the output stage defines the row address and the pair of network input that belong to the same switching element of the input stage defines the column address. Crosses in the corresponding positions mark pairs of input-output connections of the given permutations. Then the input into

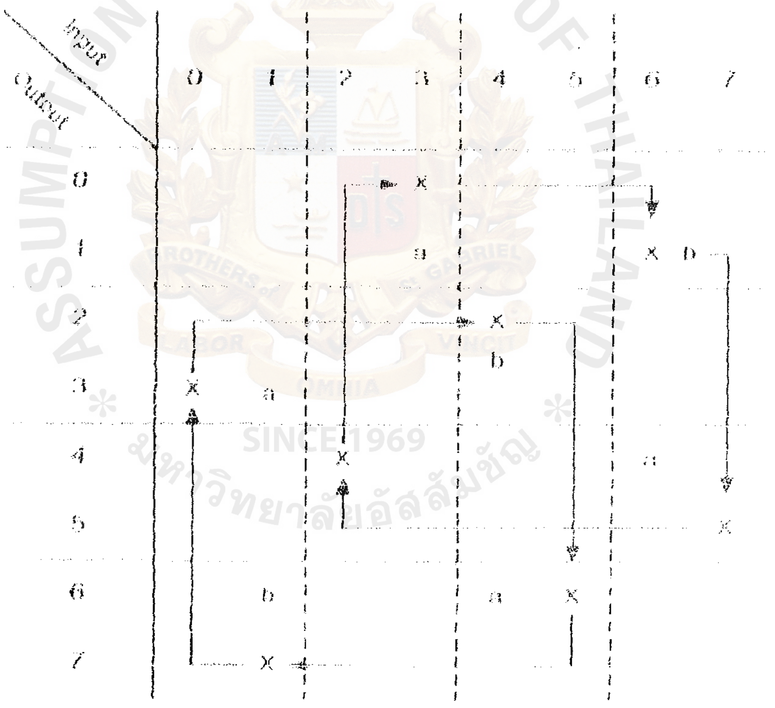


Figure: Looping Algorithm

the map is chosen arbitrarily. For example, choosing as the starting point the cross in row 23 and column 01, we look for the next input in the same row or same column so as to form a loop. In figure, the input in row 23 and column 45 is chosen. The process is continues until no loop can be formed because we return to row 23 and column 01. The loop inputs are then alternately named as a and b. a second loop is formed similarly. Then the input and output lines named a are assigned to the subnetwork a, while the lines named b are assigned to the subnetwork b. The resulting control of the input and output switching elements are shown in the figure.

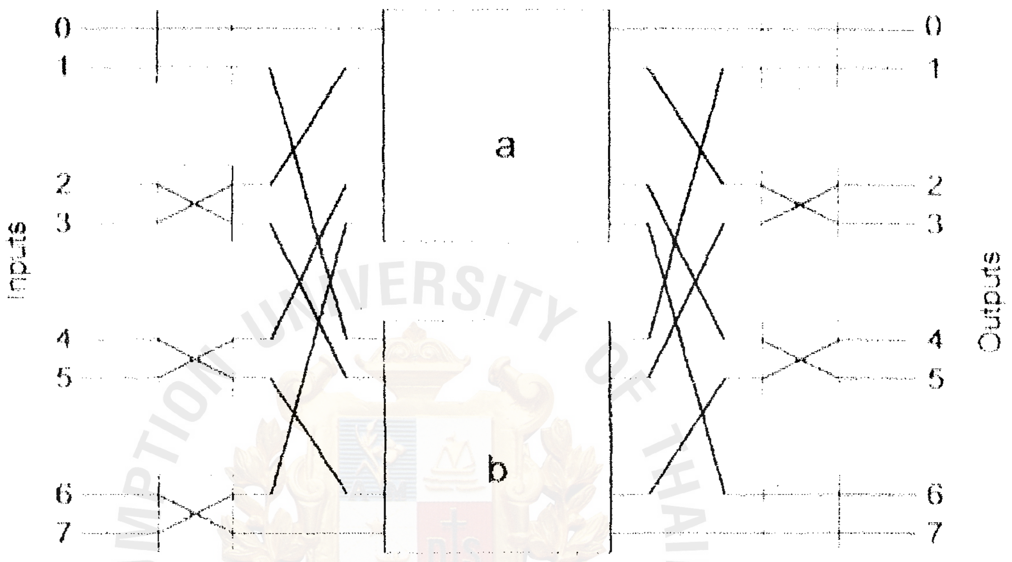
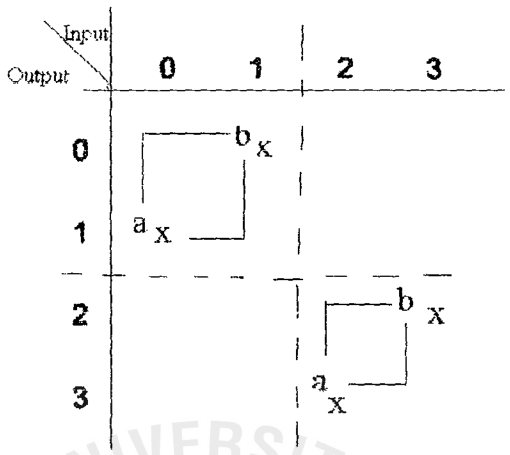
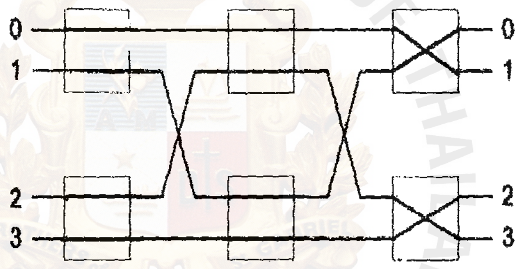


Figure: States Of Input And Output Switches

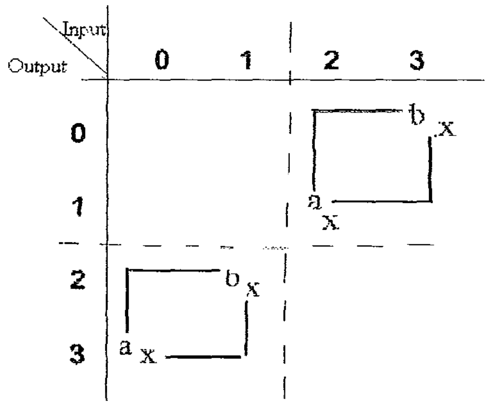
Using the given permutations and the established connections in the input and output stages, we can easily identify the permutations that should be realized by the sub networks a and b. Then the looping algorithm is recursively applied to both sub networks.

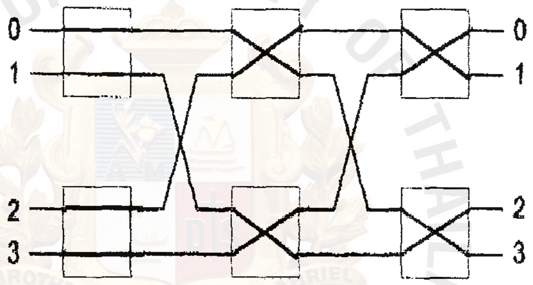
$$p_a = \begin{pmatrix} 0\,1\,2\,3 \\ 1\,0\,3\,2 \end{pmatrix}$$





$$p_b = \begin{pmatrix} 0\,1\,2\,3 \\ 3\,2\,1\,0 \end{pmatrix}$$





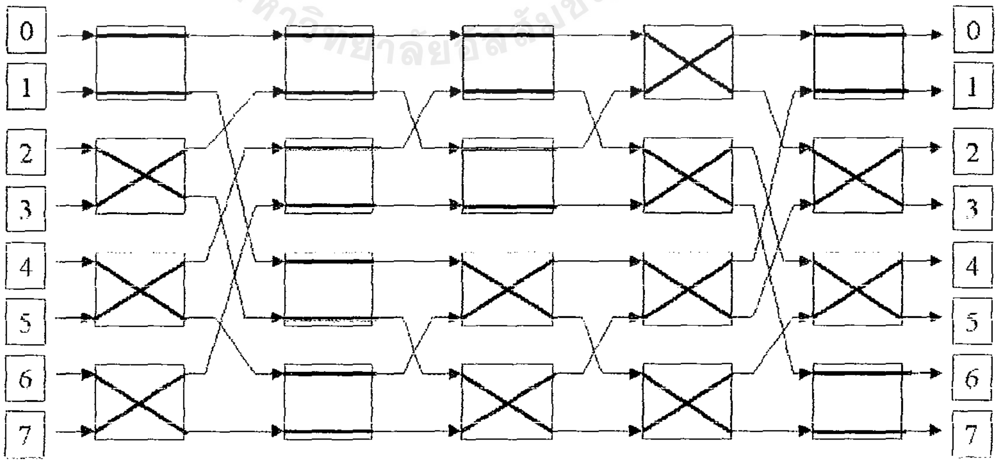


Figure: The Resulting Benes Network

Such algorithms run in time O ($N\log_2 N$). These time requirements to compute the control vector are obviously very large and may be totally unacceptable in some cases. To avoid this difficulty, we can compute the control vectors in advance (" e.g., in the compiling stage) and store them in a "control vector table". Such tables take up a lot of memory space-not less than ($2\log_2 N-1$) N/2 bits per control vector for one permutation.

<u>FAULT TOLERANT SWITCHING NETWORKS</u>

With a large number of processors (hundreds and thousands), the networks are fairly complex devices. The switching control algorithm is also fairly complex-essentially for Benes network. Since the reliability of a switching network has a direct impact on the reliability of the entire system, reliability improvement can be achieved by designing networks that are fault tolerant with respect to single faults (usually). The problems of designing of a fault tolerant network is formulated as follows:

1. For single faults in the switching elements and their interconnections, the network should realize without collision the original classes of admissible permutations.
2. A fault tolerant network should have a sufficiently simple algorithm that will rearrange it in case of a fault.
3. Network control should be capable of partitioning the network into sub networks of equal size, coupled both electrically and structurally. The last requirement makes it possible to disconnect the faulty half of the network for repair, while the non-faulty half continues functioning, emulating the complete switching network with performance reduced by a factor of 4.

Fault tolerance implies neutralization of the effect of faults in the network. This is possible only if provide static redundancy, e.g. masking in a three-fold majority voting structure, or dynamic redundancy, which pinpoints the fault and reconfigures the network or the data flows. In what follows, we consider dynamic redundancy techniques.

The data transmitted in a single clock cycle through the switching networks may be one bit, one byte or one processor word. In the last two cases, faults may be detected by appending a parity bit or several bits in the Hamming code. However these methods, while signaling the existence of a fault, do not pinpoint the fault location in the switching network. It is better to use for this purpose special diagnostic tests which are run periodically as part of network testing or after a fault has been detected.

Simple and efficient diagnostic tests available for the class of single struck- at faults const ≡ 1 (or ≡ 0) on one of the information inputs (outputs) or on the control input of the switching element in a base-2 switching network. The main idea is to apply an combination to the N network inputs such that each switching element receives the signal 01 or 10 on its two inputs, with the same control signal on all the elements. As all the input values are inverted in a non-faulty network, each output terminal should successively produce the two values 01 and 10. if this sequence is not observed on one fo the output terminals, there is a fault in the information channel connected with this element. Repeating the same test for inverted values of the control signal on each switching element, we may detect another information channel passing through the same faulty component. The fault is in the component, which is common to both information channels.

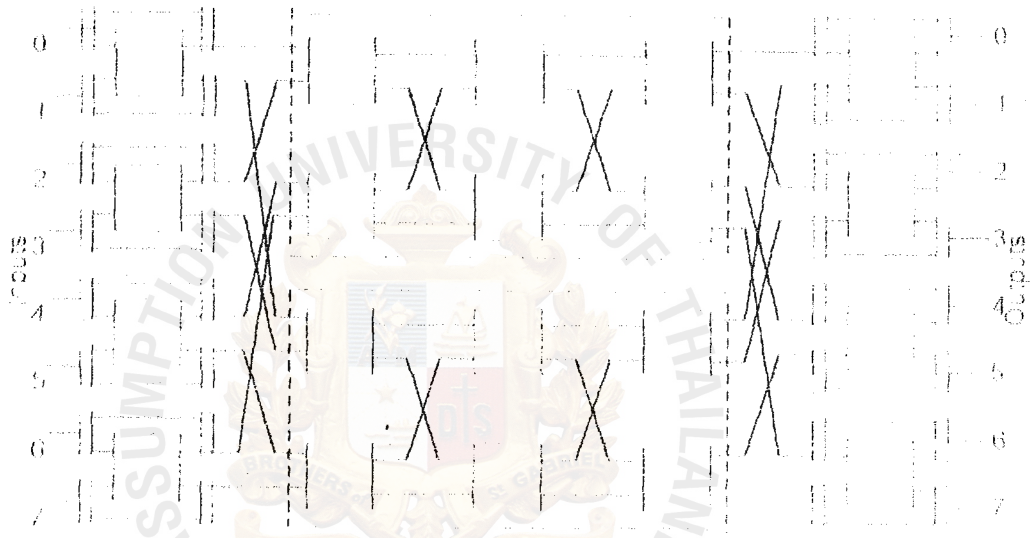
These tests are similarly applied to check for control signal faults. An error in the output signal is detected by comparing the output values with a reference. A reference signal is needed because if there is a fault in the control signal of the switching elements in the last stage, a 1-out-of-2 code will not be violated on the network output terminals (unlike the case of such a fault in all the other stages), and the output therefore have to be tested by comparison with reference values.

All the above remains valid for cube and omega networks, their generalizations and also for Benes networks. Thus a base-2 switching network can be diagnosed by four test combinations, which identify the faulty switching element or the faulty component in the transmission channel.

The diagnostic results make it possible to move to the next stage of recovery in a faulty switching network neutralizing the effect of the fault. Here two approaches are available. The first relies on the introduction of an extra stage in cube networks. Because of the extra stage, there are now two paths from any source to any destination in the switching network.

The second approach designs a switching network with a given constant number of nonintersecting routes from any source to the destination. For blocking networks, the existence of alternative routes is ensured by a special choice of the switching element base in the network.

In an 8 X 8 cube network the main idea that achieves the fault tolerance is the introduction of an extra stage from the side of the data source (stage 3). As a result each source can access through a switching element two inputs of a cube subnetwork. In the cube network there is a unique path from each source to any destination, and therefore in a non-faulty network with an extra stage, two different routes lead them from the extra stage inputs to the same destination. These routes are non-intersecting, and therefore in case of a fault on one of the routes, the "conjugate" route may be used.



A fault-tolerant modification of the 8 x 8 Benes network.

In the Benes network, there are more than one independent paths from one input-output pair, so that in principle it is possible to bypass single faults and to realize any given permutation in two passes without extra stages. If the first and last stages are augmented with multiplexors-demultiplexors, then the modified Benes network will be fault tolerant to single faults. In this case, fault tolerance is also accomplished in two passes through the network. In the first pass, the data are switched through the non-faulty paths. Because of the perfect symmetry of the two alternative paths between the same pair of input-output terminals, the data are switched to the alternative path in the second pass by passing the control signal from the switching element of one "storey" of intermediate stages to another "storey". The control signals are generated in advance for the given interconnection list by one of the known algorithms, assuming anon-faulty network. The control for the switching elements of the first and the last stages participating in the unrealized connection is reversed, i.e. if one element was initially connected "straight", it is now switched to "exchange" and conversely.

If the fault is detected in the first or the last stage, then the network control is somewhat different. Specifically if the fault is detected in a switching element of the first (last) stage connected "straight", then this element is disconnected, the bypass multiplexors-demultiplexors are activated, and the required switching can be realized even in a single pass, because all the other control signals are not altered. If the faulty element was realizing exchange function, then the faulty element is disconnected, all the input-output elements, except those connected to the faulty element, are switched in the first pass, and the bypass multiplexors-demultiplexors of the faulty element are activated in the second pass, when the control signals from the first storey of intermediate stages are transferred to the second storey and conversely, thus exciting two alternative paths relative to the initial paths in the faulty network.

The modified Benes network allows decomposition of the entire network into two sub networks of equal size, so that the non-faulty half can emulate the entire network. This decomposition is useful during the recovery of the faulty half of the network.

A SELF-ROUTING PERMUTATION NETWORK

A self-routing permutation network is a connector, which can set its own switches to realize any one-to-one mapping of its inputs onto its outputs. A self-routing switch routes a message to its destination using only the information contained in the message without requiring knowledge about other connections. Self-routing has several advantages over a global routing scheme: the routing time of a self-routing network is the same as the propagation delay in the network; if the address decoding logic at each switch can be kept simple then the hardware cost of a self-routing will in general be less than that of a network with global routing scheme. Consequently, self-routing scheme reduces the connection complexity for the control lines. Again, as the control is distributed in self-routing network to each switch, it is less susceptible to the faults of a switch.

An Example Of Self-Routing Algorithm For Omega Network

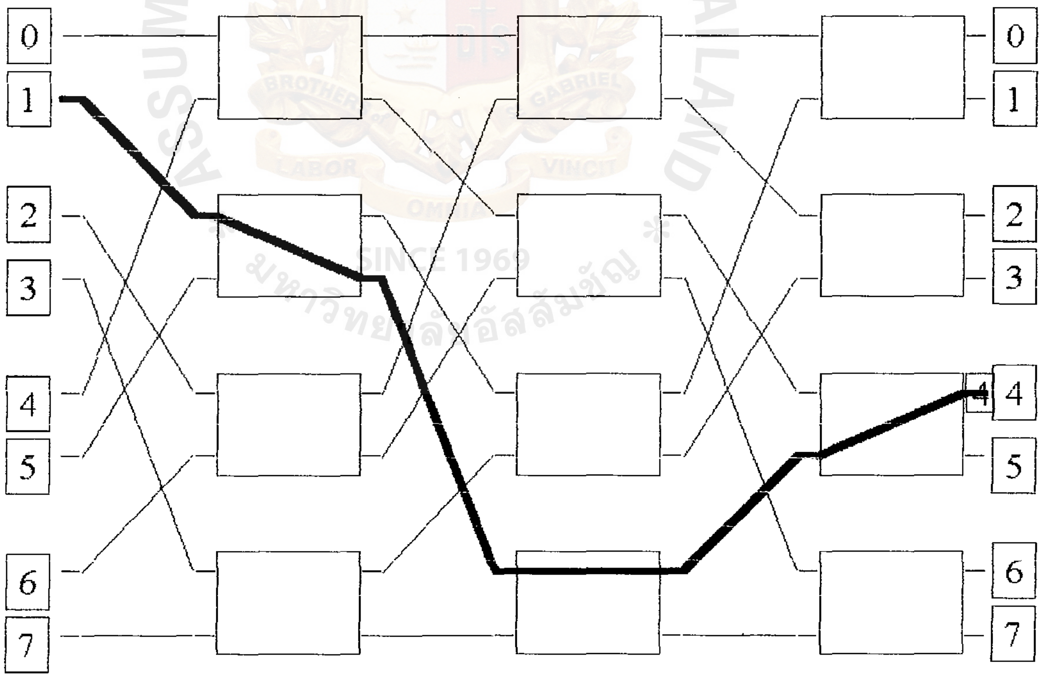
The omega network is another example of a banyan multistage interconnection network that can be used as a switch fabric. The omega MIN uses the "perfect shuffle". The interconnections between stages are defined by the logical "rotate left" of the bits used in the port ids

•Example: 000 ---> 000 ---> 000 ---> 000
•Example: 001 ---> 010 ---> 100 ---> 001
•Example: 011 ---> 110 ---> 101 ---> 011
•Example: 111 ---> 111 ---> 111 ---> 111

Omega network has self-routing property. The path for a cell to take to reach its destination can be determined directly from its routing tag (i.e., destination port id).

- Stage k of the MIN looks at bit k of the tag.
- If bit k is 0, then MIN sends cell out to the upper port.
- If bit k is 1, then MIN sends cell out to the lower port.

Cell destined for output port 4 (= 100_2) from port 1



ALGORITHM FOR SELF-ROUTING IN BENES NETWORK

Here we investigate the possibility of rapidly obtaining the switch settings of the Benes network for certain classes of permutations. It is shown that by providing a "destination tag" with each signal and by adding some simple logic to each switch in the Benes network, it is possible for each switches to determine its own setting dynamically (i.e., when it receives the incoming signal). The resulting network can perform certain permutations in O (logN) time (including the setup time). It is demonstrated that the richness of the set F (the set of permutations realizable on "self-routing" Benes network) includes most classes of permutations studied in the parallel processing literature.

Let D_i be the "destination tag" on input terminal i, $0 \le i \le N-1$. Here (D_{n-1} D_{n-2} … D_1 D_0) is a permutation of (N-1, …, 1,0). The data at input terminal i is to be routed to output terminal D_i. The switch settings are determined from the binary representation of D_i. If an $N= 2^n$ input / output Benes network is being used, then there are 2n-1 stages of switches whose settings need to be used. Let the stages be numbered 0 through 2n-2. the state of a switch in stage b or stage 2n-2-b, $0\le b \le n-1$, is determined by bit b of the destination tag of its upper input. If bit b is 0, the switch is set to state 0, otherwise to state 1.

Figure shows the switch settings obtained by this procedure for bit reversal. The destination for each switch input is given on the respective input line.

$$p = \begin{pmatrix} 0\,1\,2\,3\,4\,5\,6\,7 \\ 0\,4\,2\,6\,1\,5\,3\,7 \end{pmatrix}$$

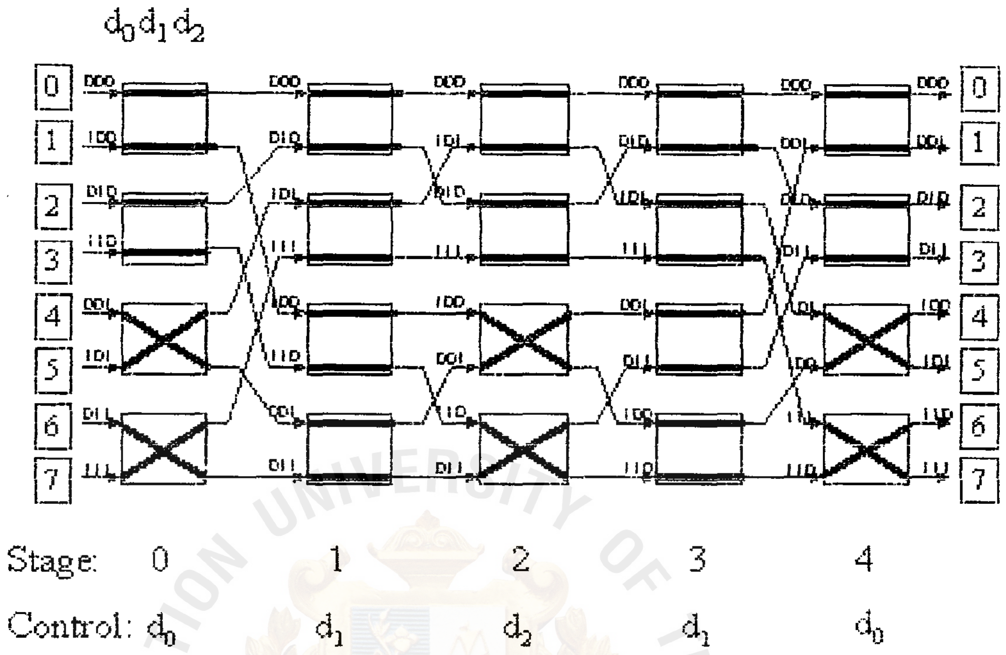


Figure: Bit Reversal Permutation Using Self-Routing Algorithm

The number of switches and delay in this self-routing network are both about twice the corresponding figures in a self-routing Omega network. However the number of permutations realizable on Benes networks (i.e. the cardinality of the set F) is much larger than that of an omega network.

F is the set of permutations realizable on "self-routing" Benes network. David Nassimi and Sartaj Sahni concluded that BPC (n) $\subseteq$ F (n) and I$\Omega \subseteq$ F(n), i.e., all of Lenfant's five families of "frequently used bijections" are included in F(n). This is because as commented earlier, three of the FUB families are in BPC and the remaining two in IΩ.

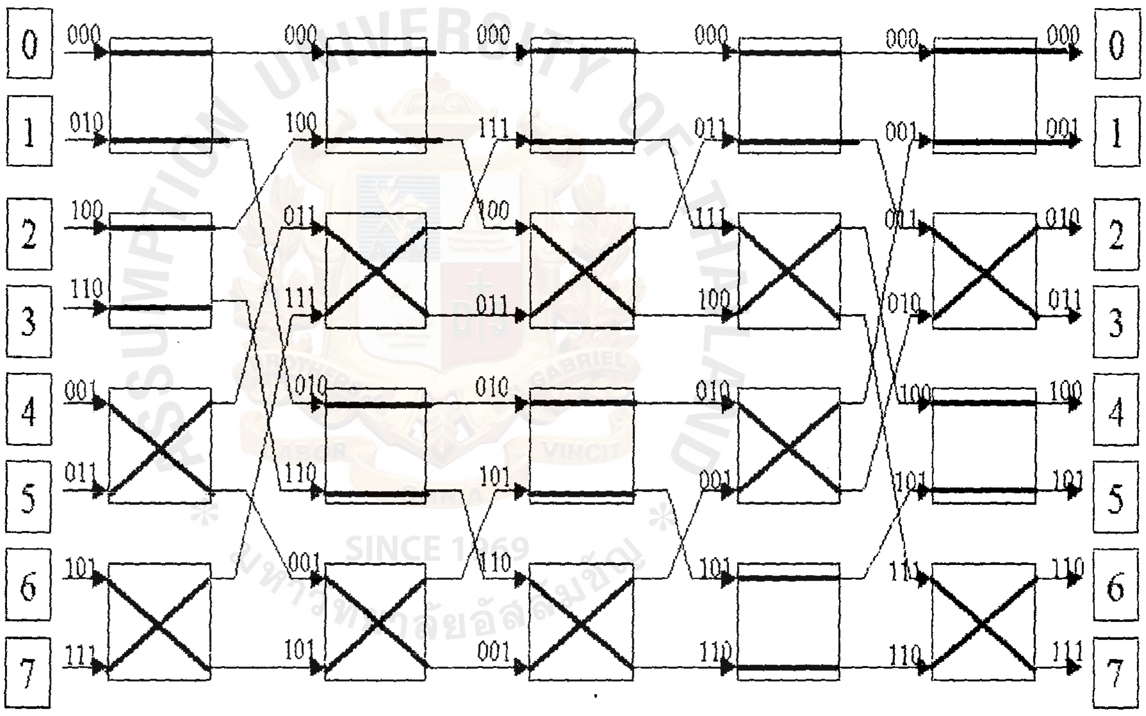
Unfortunately, not all Ω(n) permutations are in F (n). However, as mentioned earlier, many Ω(n) permutations of interest (e.g., cyclic shift and p-ordering) are also in IΩ(n), hence in F (n). By providing the self-routing network with some additional simple logic, the network can handle all Ω(n) permutations as well. Since the last n stages of B(n) correspond to an omega network, it is easy to see that an Ω(n) permutation can be realized on Benes network if the switches in stages 0 through n-2 are all placed in state 0, while the remaining n stages obey the self-routing scheme described earlier. One way to implement this on the Benes network is to provide an additional "omega" bit with each "destination tag". (This bit will be equal to 1 if and only if we are performing an omega permutation.) Each switch in stages 0 through n-2 places itself in state 0 if it finds the "omega" bit = 1, otherwise the switch determines its state as before. The logic of switches in the last n stages of the self-routing network is not altered.

SOME ADMISSIBLE PERMUTATIONS USING SELF-ROUTING ALGORITHM

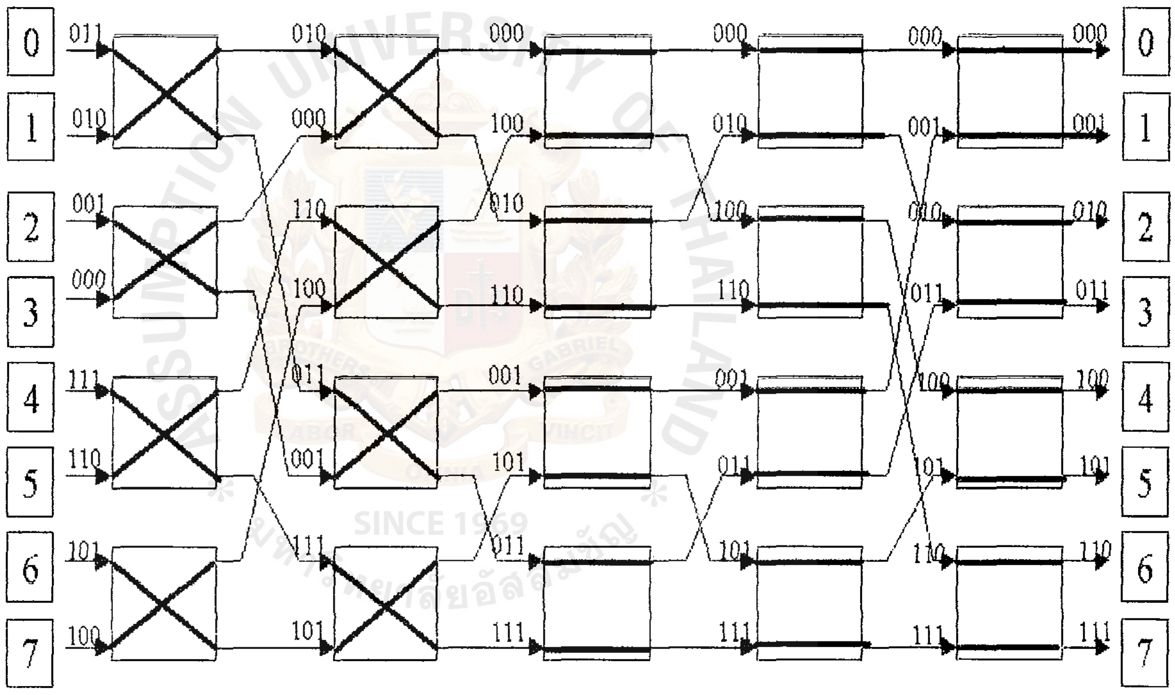
Perfect Shuffle

0 1 2 3 4 5 6 7

0 2 4 6 1 3 5 7



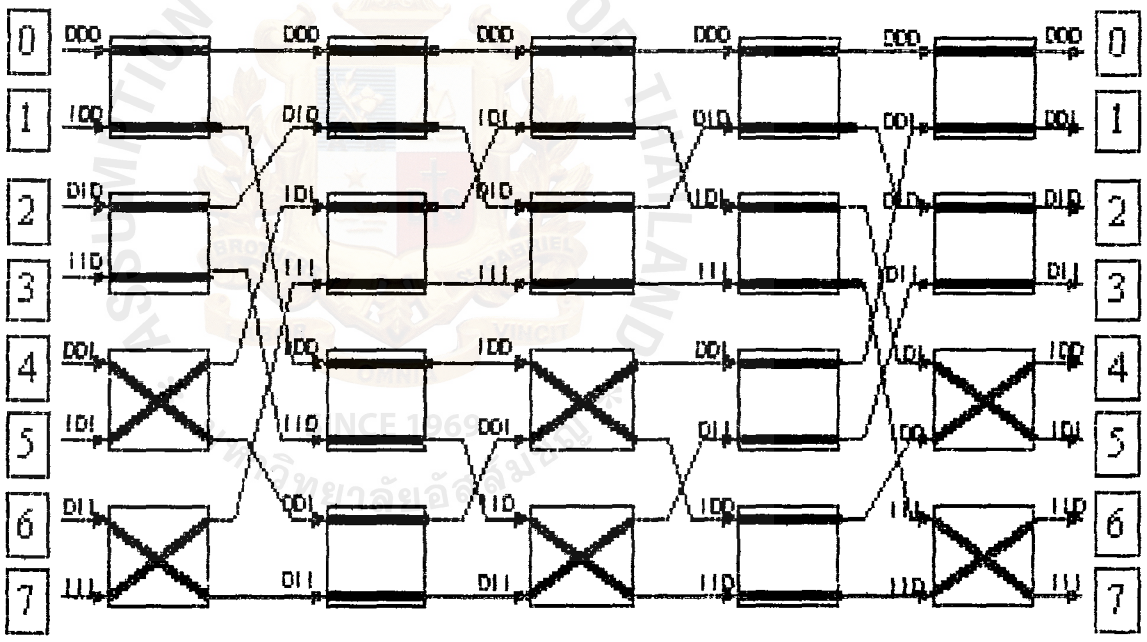
FLIP (011)
0 1 2 3 4 5 6 7
3 2 1 0 7 6 5 4



Bit Reversal

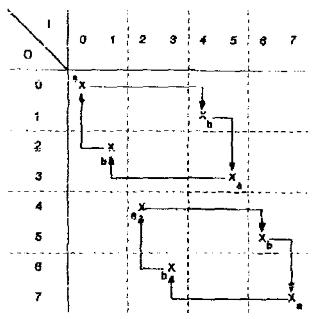
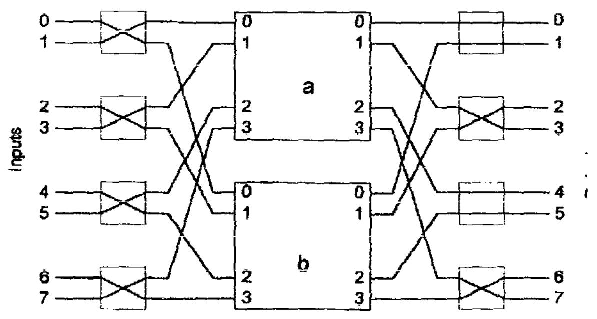
0 1 2 3 4 5 6 7

0 4 2 6 1 5 3 7

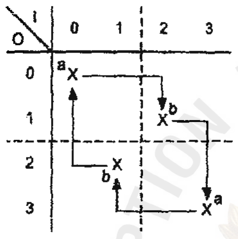
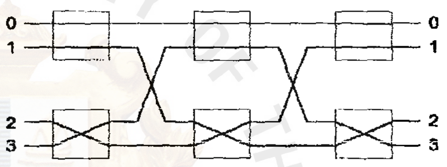


SOME ADMISSIBLE PERMUTATIONS USING LOOPING ALGORITHM

Perfect Shuffle
0 1 2 3 4 5 6 7
0 2 4 6 1 3 5 7

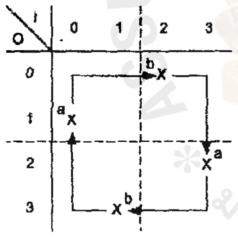
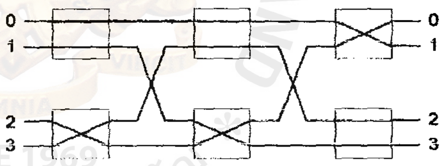



0 1 2 3
0 2 1 3

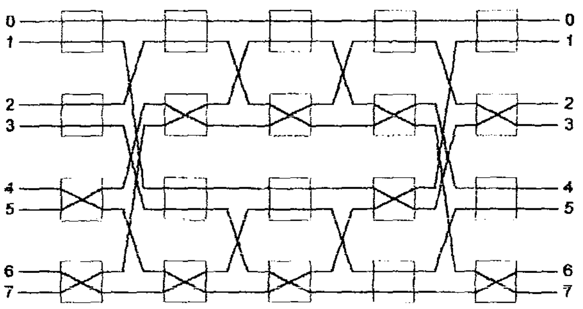
0	1
0	1
0	1
1	0

0 1 2 3
1 3 0 2

0	1
0	1
0	1
1	0

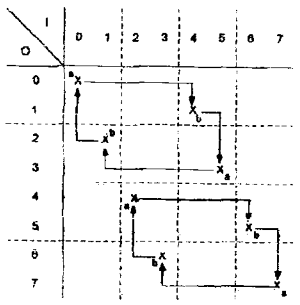
0	0	0	0	0
0	1	1	1	1
1	0	0	1	0
1	1	1	0	1

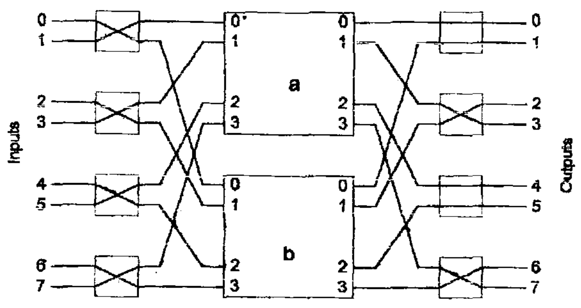


SHIFT 1

0 1 2 3 4 5 6 7

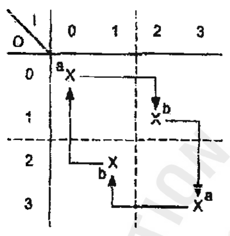
0 2 4 6 1 3 5 7

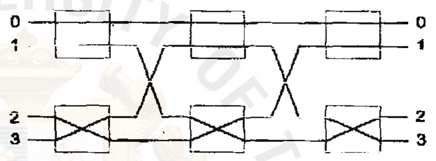




0 1 2 3

0 2 1 3

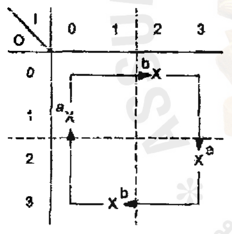


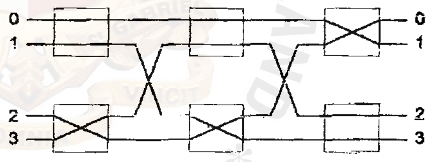


0	1
0	1
0	1
1	0

0 1 2 3

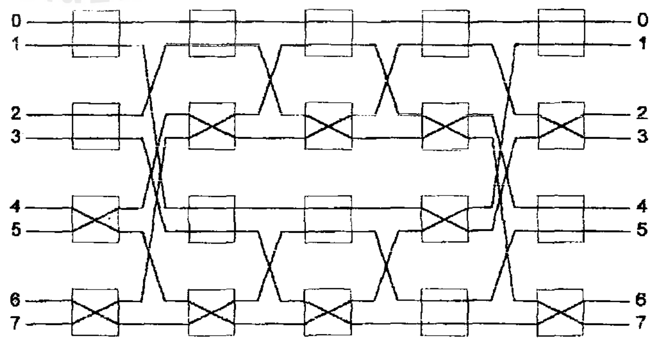
1 3 0 2



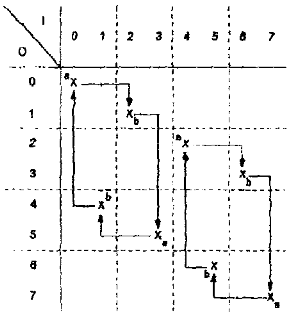


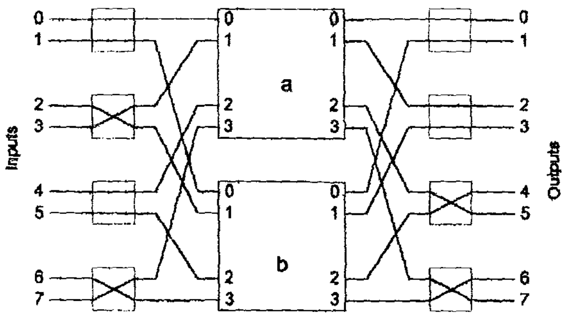
0	1
0	1
0	1
1	0

0	0	0	0	0
0	1	1	1	1
1	0	0	1	0
1	1	1	0	1

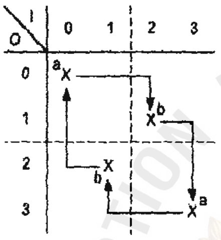


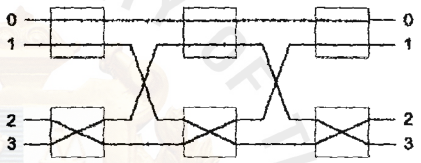
SHIFT 2
0 1 2 3 4 5 6 7
0 4 1 5 2 6 3 7





0 1 2 3
0 2 1 3

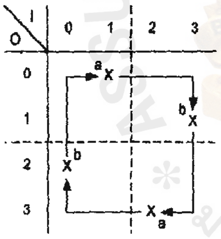


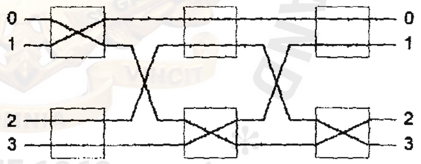


0 1
0 1

0 1
1 0

0 1 2 3
2 0 3 1

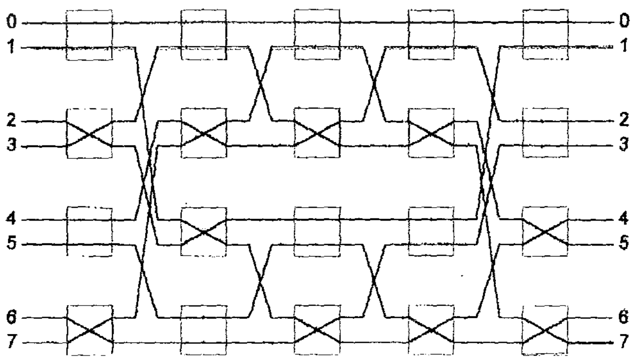




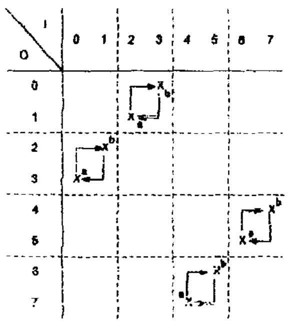
0 1
0 1

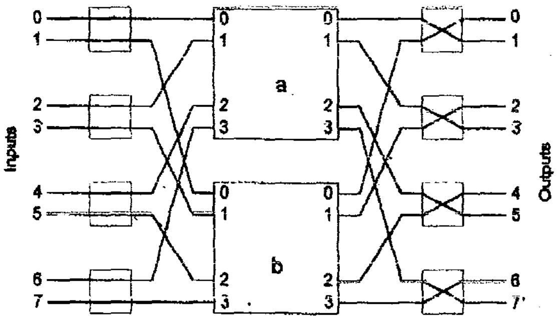
0 1
1 0

0	0	0	0	0
1	1	1	1	0
0	1	0	0	1
1	0	1	1	1

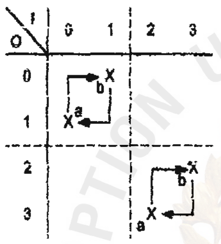


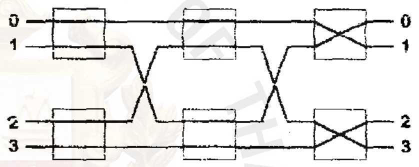
FLIP (011)
0 1 2 3 4 5 6 7
3 2 1 0 7 6 5 4





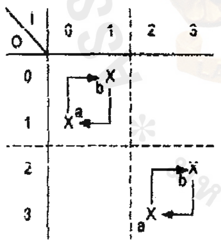
0 1 2 3
1 0 3 2

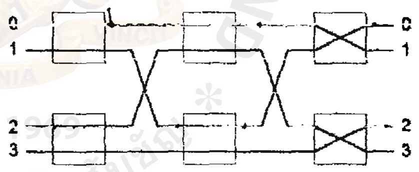




0	1
0	1
0	1
0	1

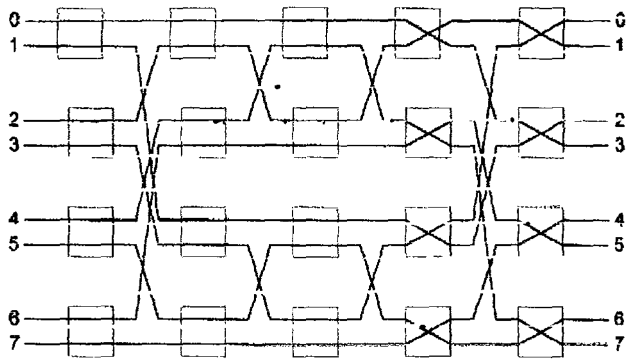
0 1 2 3
1 0 3 2





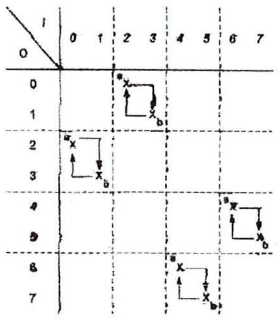
0	1
0	1
0	1
0	1

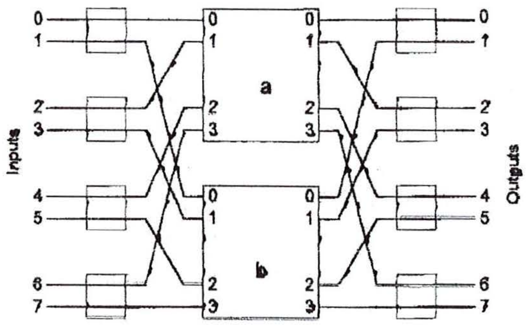
0	0	0	1	1
0	0	0	1	1
0	0	0	1	1
0	0	0	1	1



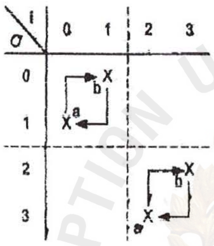
FLIP (010)

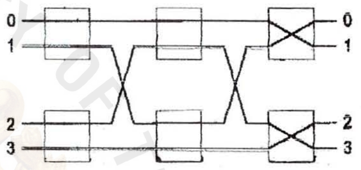
0 1 2 3 4 5 6 7
2 3 0 1 6 7 4 5





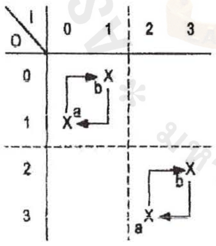
0 1 2 3
1 0 3 2

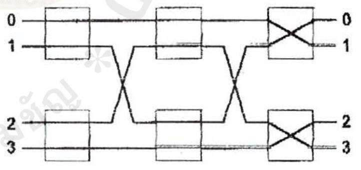




0	1
0	1
0	1
0	1

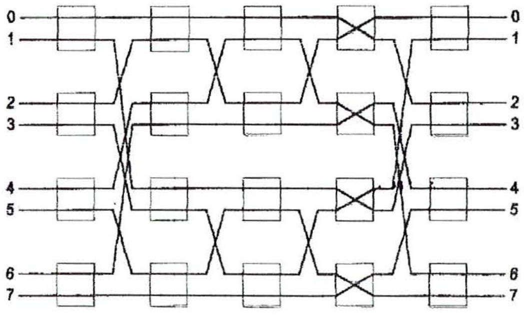
0 1 2 3
1 0 3 2



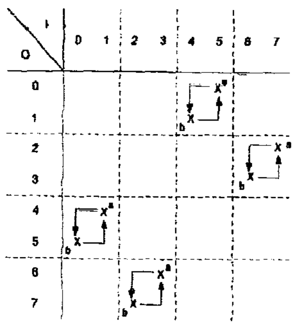


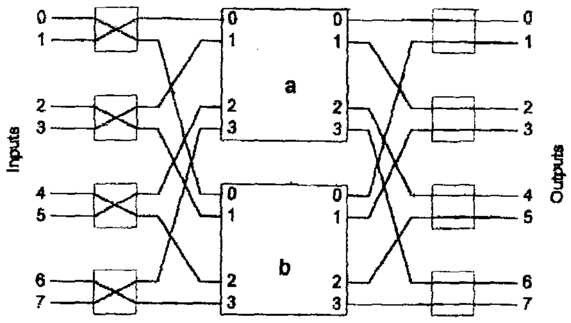
0	1
0	1
0	1
0	1

0	0	0	1	0
0	0	0	1	0
0	0	0	1	0
0	0	0	1	0

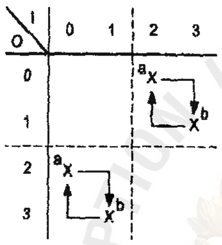


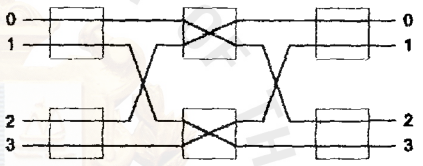
FLIP (101)
0 1 2 3 4 5 6 7
5 4 7 6 1 0 3 2





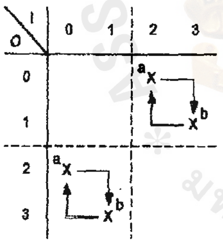
0 1 2 3
2 3 0 1

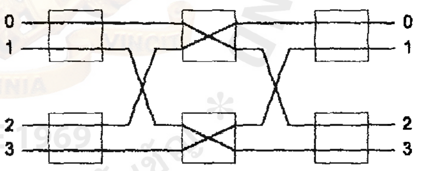




0	1
1	0
0	1
1	0

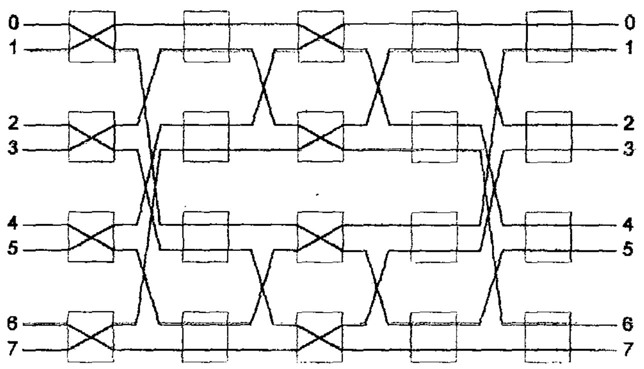
0 1 2 3
2 3 0 1



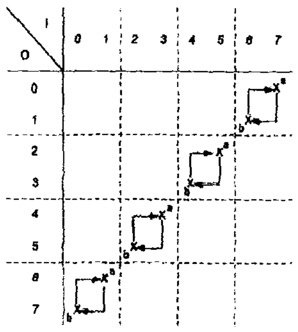


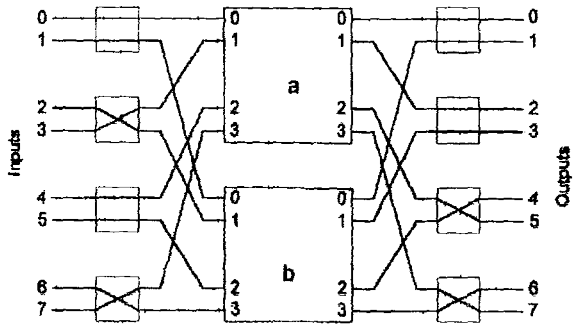
0	1
1	0
0	1
1	0

1	0	1	0	0
1	0	1	0	0
1	0	1	0	0
1	0	1	0	0

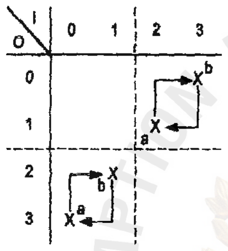


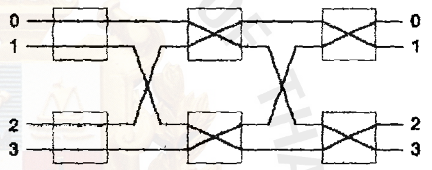
FLIP (111)
0 1 2 3 4 5 6 7
7 6 5 4 3 2 1 0





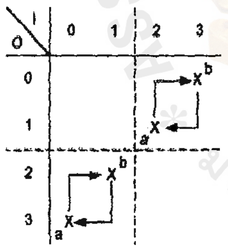
0 1 2 3
3 2 1 0

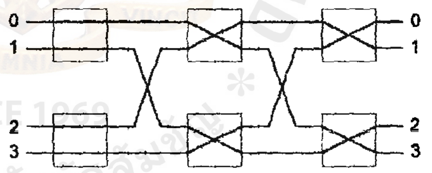




0	1
1	0
0	1
1	0

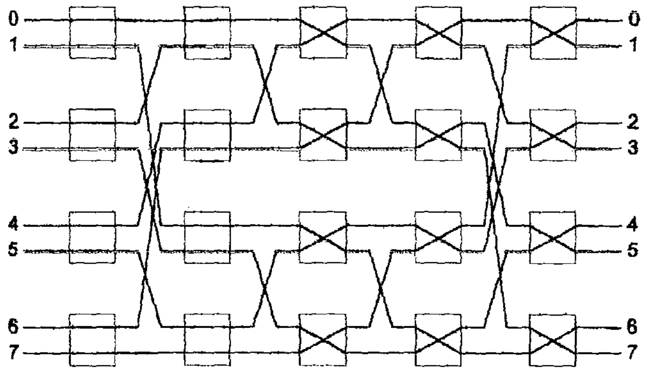
0 1 2 3
3 2 1 0





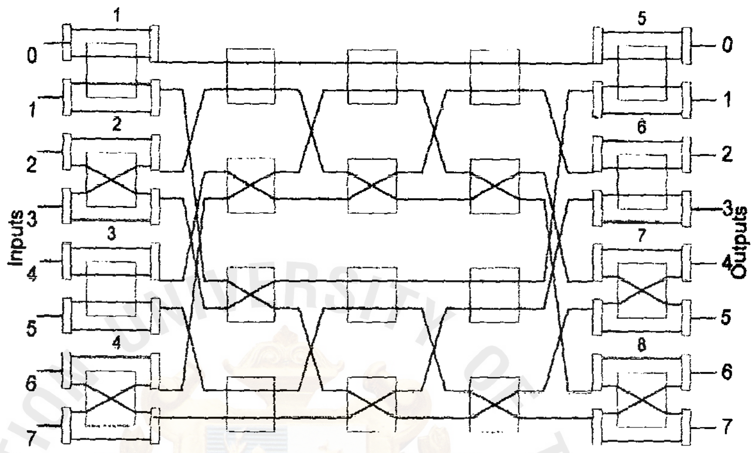
0	1
1	0
0	1
1	0

0	0	1	1	1
0	0	1	1	1
0	0	1	1	1
0	0	1	1	1



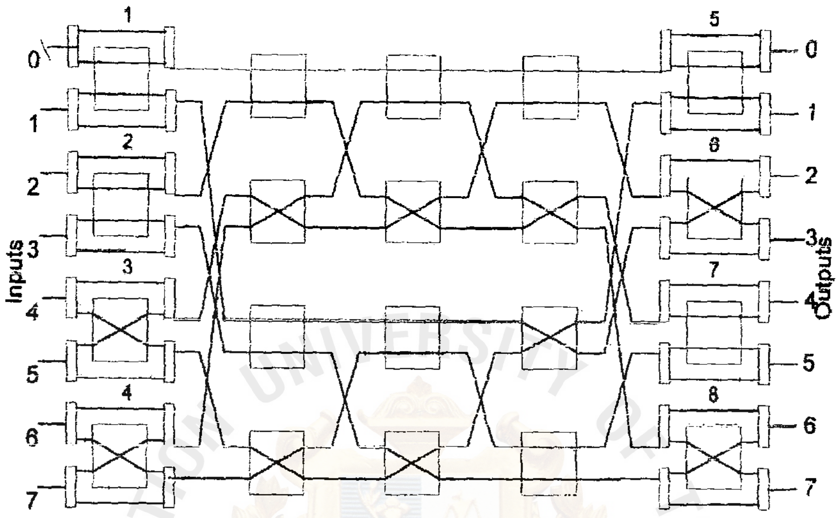
FAULT TOLERANCE IN THE FIRST AND THE LAST STAGE

SHIFT 2
0 1 2 3 4 5 6 7
0 4 1 5 2 6 3 7



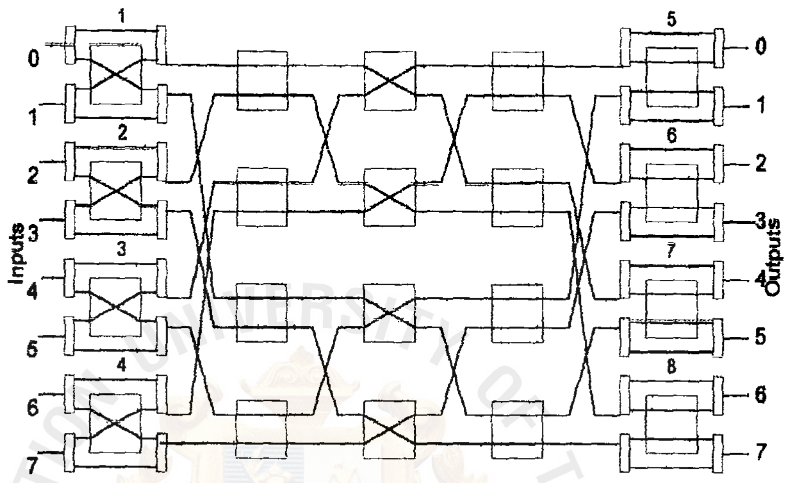
Switch No.	State
1	Straight
2	Exchange
3	Straight
4	Exchange
5	Straight
6	Straight
7	Exchange
8	Exchange

Perfect Shuffle
0 1 2 3 4 5 6 7
0 2 4 6 1 3 5 7



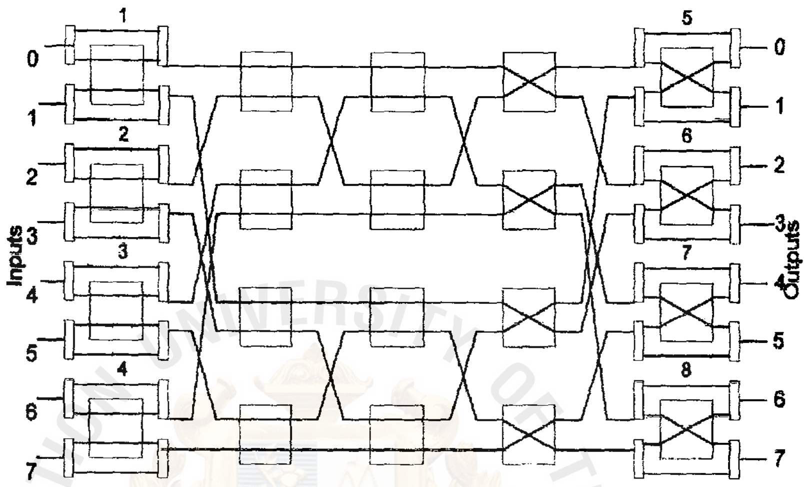
Switch No.	State
1	Straight
2	Straight
3	Exchange
4	Exchange
5	Straight
6	Exchange
7	Straight
8	exchange

FLIP (101)
0 1 2 3 4 5 6 7
5 4 7 6 1 0 3 2



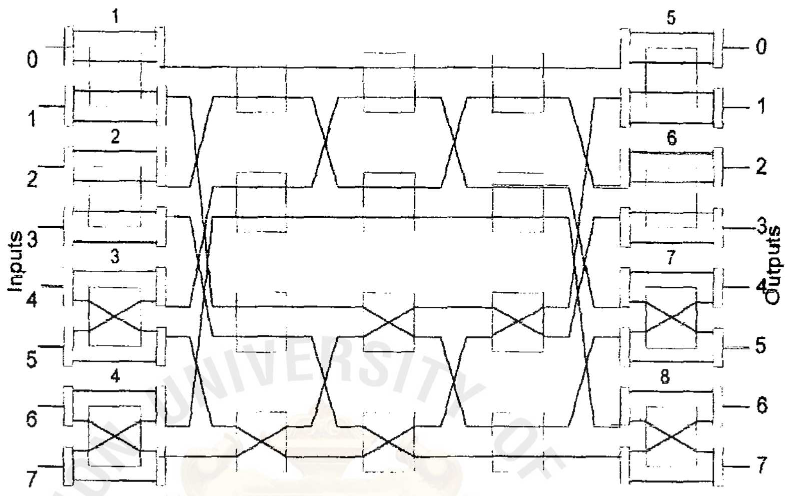
Switch No.	State
1	Exchange
2	Exchange
3	Exchange
4	Exchange
5	Straight
6	Straight
7	Straight
8	Straight

FLIP (011)
0 1 2 3 4 5 6 7
3 2 1 0 7 6 5 4



Switch No.	State
1	Straight
2	Straight
3	Straight
4	Straight
5	Exchange
6	Exchange
7	Exchange
8	Exchange

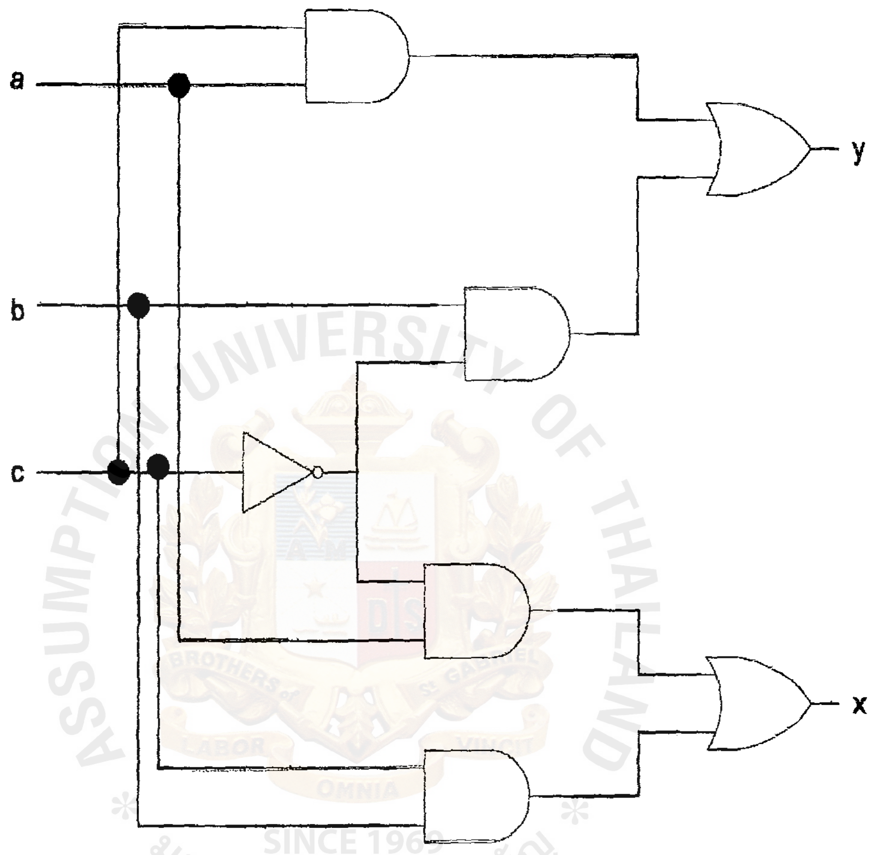
Bit Reversal
0 1 2 3 4 5 6 7
0 4 2 6 1 5 3 7



Switch No.	State
1	Straight
2	Straight
3	Exchange
4	Exchange
5	Straight
6	Straight
7	Exchange
8	Exchange

HARDWARE IMPLEMENTATION OF FAULT TOLERANT BENES NETWORK

Circuit Diagram of the switch



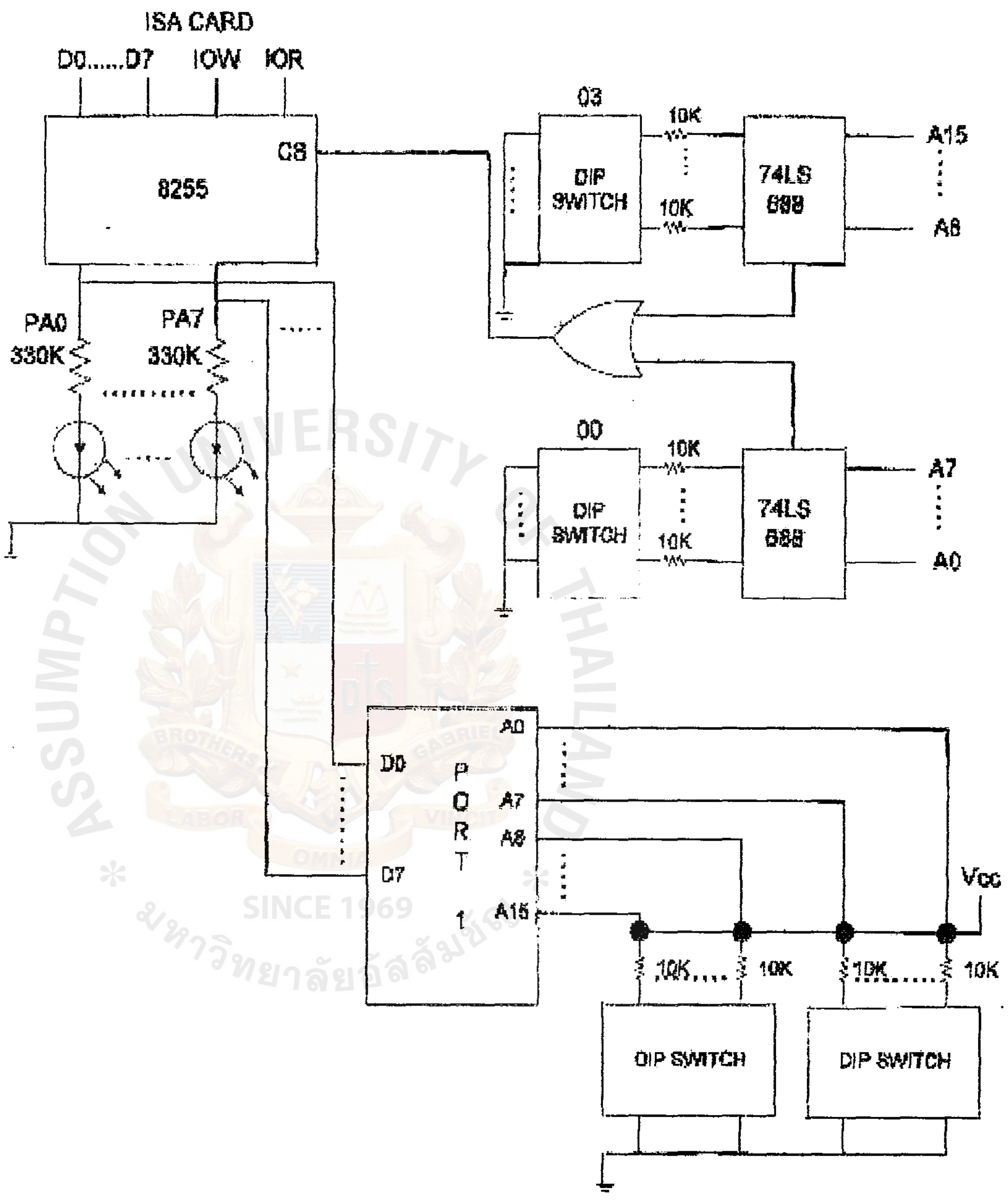
State 0:

If c= 0
x=a
y=b

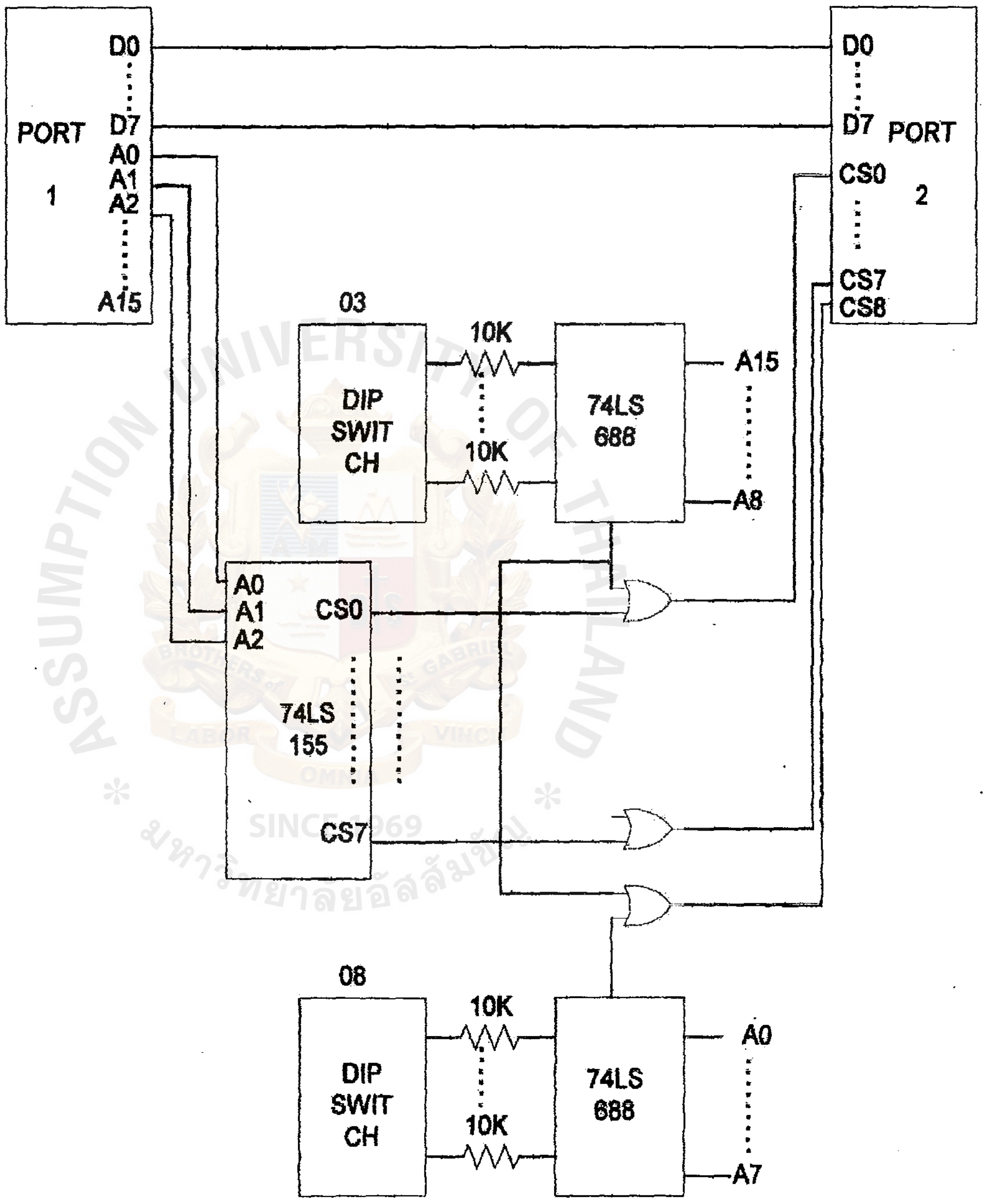
State 1:

If c=1
x=b
y=a

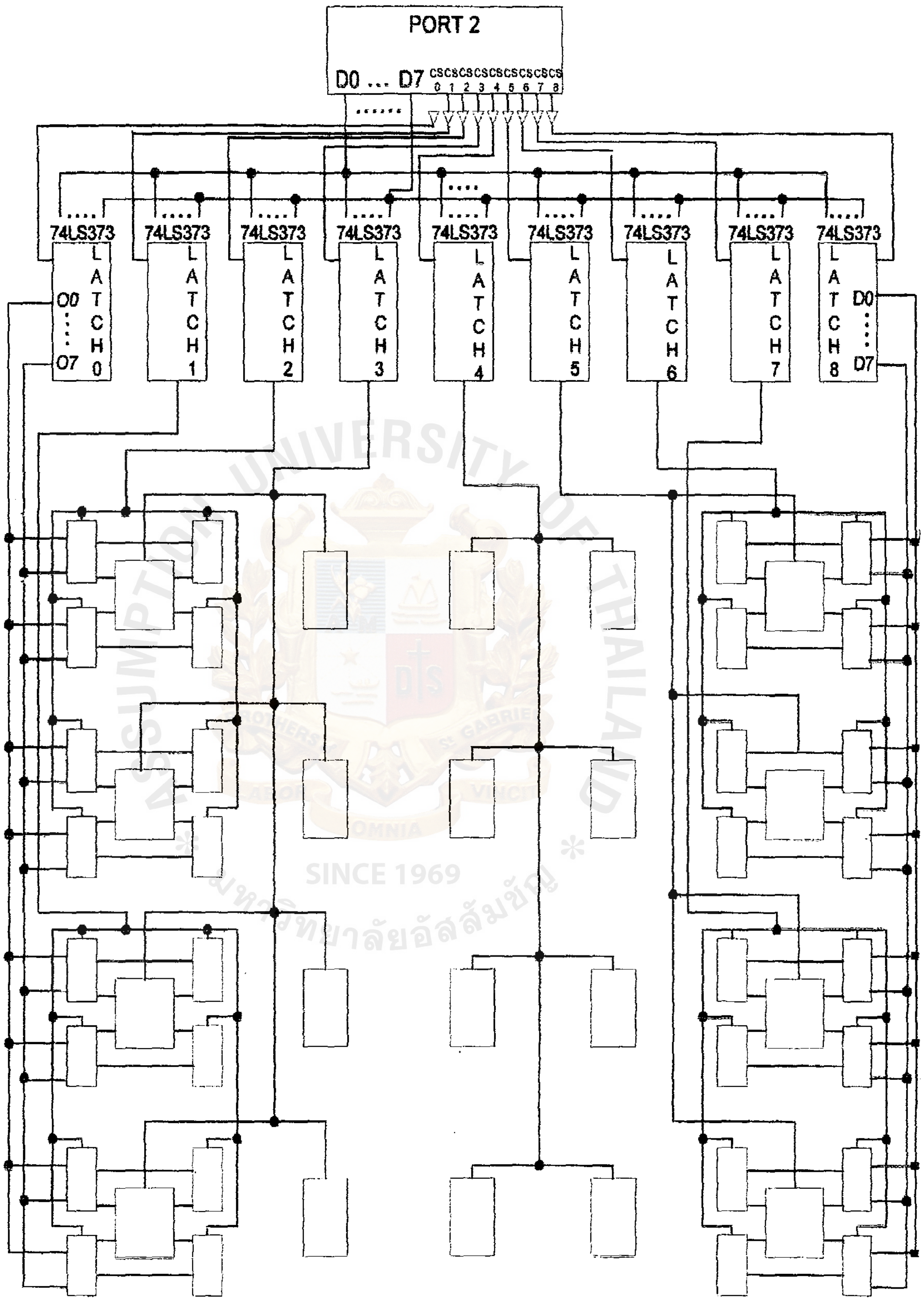
Block Diagram Of The Address Decoder



Circuit Diagram of the address decoder 2



Circuit Diagram of the Benes Network



CONCLUSION

Here we have presented the structure of a fault-tolerant self-routing Benes Network. The destination tag (logN bits) is passed through the network along with each input. A very simple logic is required in each switch. This logic determines the binary state of the switch from a particular bit of the destination tag of its upper input. The resulting network is capable of realizing a rich class of permutations, F(n), $N = 2^n$, in O (logN) time. The class F(n) includes Lawrie's IΩ(n) permutations, Nassimi and Shahni's BPC(n) permutations and consequently Lenfant's FUB families. Any Ω(n) permutation can also be realized on this network if the switches in the first n-1 stages are forced into state 0. The fault tolerant self-routing Benes network promises an effective interconnection network for SIMD computers.

REFERENCES

1. Switching Networks For SIMD Multiprocessor Computing Systems (State-Of-The-Art-Review);Gennady Veselovsky, M.F. Karavai, And S.M. Kuznechik.

2. A Study Of The Permutation Capability Of A Binary Hypercube Under Deterministic Dimension-Order Routing; Gennady Veselovsky, Dobri Atanassov Batovski.

3. Private Communication With Dr. Gennady Veselovsky.

4. A Self Routing Benes Network And Parallel Permutation Algorithms; David Nassimi And Sartaj Sahni.