# Analysis of Various Crosstalk Avoidance Techniques in Optical Multistage Interconnection Networks

Thesis submitted in partial fulfillment of the requirements for the award of degree of

**Master of Engineering**
**in**
**Computer Science & Engineering**

**By:**
**Shruti Chopra**
**(80732021)**

Under the supervision of:
**Rinkle Aggarwal**
Lecturer (SS), CSED
Thapar University, Patiala

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT
THAPAR UNIVERSITY
PATIALA – 147004

**JUNE 2009**

## Certificate

I hereby certify that the work which is being presented in the thesis entitled, **"Analysis of various cross avoidance techniques in Optical Multistage Interconnection Networks"**, in partial fulfillment of the requirements for the award of degree of Master of Engineering in Computer Science Engineering submitted in Computer Science and Engineering Department of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of *Ms. Rinkle Aggarwal* and refers other researcher's works which are duly listed in the reference section.

The matter presented in this thesis has not been submitted for the award of any other degree of this or any other university.

Shruti Chopra

(**Shruti Chopra**)

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

Rinkle 18/06/09

**(Rinkle Aggarwal)**
Lecturer(SS),
Computer Science and Engineering Department,
Thapar University, Patiala.

**Countersigned by**

Seema Bawa 18/06/2009

**(SEEMA BAWA)**
Professor & Head,
Computer Science & Engineering Department,
Thapar University,
Patiala.

R.K.Sharma 25/6/09

**(R.K.SHARMA)**
Dean (Academic Affairs),
Thapar University,
Patiala.

## Acknowledgement

It is a great pleasure for me to acknowledge the guidance, assistance and help I have received from Ms. Rinkle Aggarwal, Lecturer (SS), Computer Science and Engineering Department. I am thankful for her continual support, encouragement and invaluable suggestions. She not only provided me help whenever needed, but also the resources required to complete this thesis report on time.

I am also thankful to Dr. Seema Bawa, Head, Computer Science and Engineering Department for her kind help and cooperation. I would also like to thank all the staff members of Computer Science and Engineering Department for providing me all the facilities required for the completion of my thesis work.

I would like to say thanks for support of my Classmates. I want to express my appreciation to every person who contributed with either inspirational or actual work to this thesis.

I am deeply grateful to my family for the inspiration and ever encouraging moral support, which enabled me to pursue my studies.

Shruti Chopra.

(**Shruti Chopra**)

# Table of Contents

# List of Figures

# List of Tables

# Chapter-1 Introduction

Any system which incorporates two or more microprocessors working together to perform a task is commonly referred to as a multiprocessor system. Multiprocessing is a type of processing in which two or more processors work together to process more than one program simultaneously [1]. It allows the system to do more work in a shorter period of time. Multiprocessor system is also known as parallel system or tightly-coupled system. It means that multiple processors are tied together in some manner. Generally, the processors are in close communication with each other. They share common data structures and a common system clock.

## 1.1 Parallel Computer Structures

Parallel processing is a process to increase the computer's performance by executing many instructions simultaneously or in "parallel". Parallel computers are those systems that emphasize parallel processing. Parallel computer structure can be divided into three structural classes: Pipelined computers, Array Processors and Multiprocessor system.

### 1.1.1 Pipelined Computers

A pipeline computer performs overlapped communication to exploit temporal parallelism. A pipeline computer is a synchronous parallel computer. The flow of data from one stage to another stage is triggered by the help of a common clock of the pipeline. Once the pipeline is filled up, an output result is produced from the pipeline on each cycle. Due to overlapped instruction and arithmetic execution, it is obvious that pipeline machines are better to perform the same operations repeatedly through the pipeline.

### 1.1.2 Array Processors

An array processor is a synchronous parallel computer which consists of multiple processing elements, a control unit and an interconnection network. The control unit broadcasts instructions to the processing elements (PE's) and all the PE's execute these instructions in the parallel on different data. Array processors are also known as Single Instruction Multiple Data Stream (SIMD) machines [1].

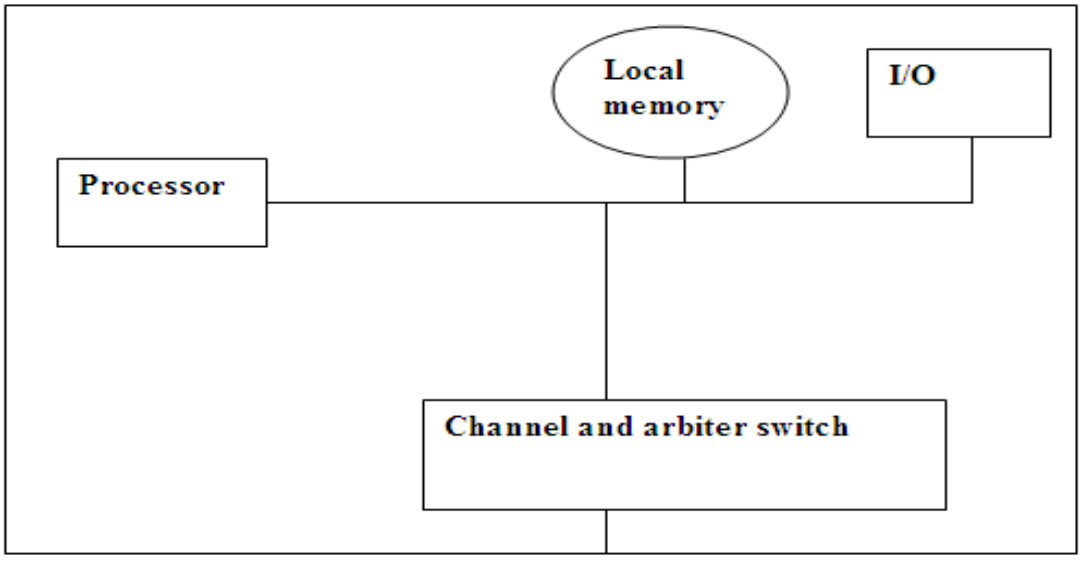
### 1.1.3 Multiprocessor Systems

A Multiprocessor System consists of two or more processors of approximately comparable capability. All processors have access to memory modules, I/O channels and peripheral devices. A multiprocessor system is controlled by a single integrated operating system.

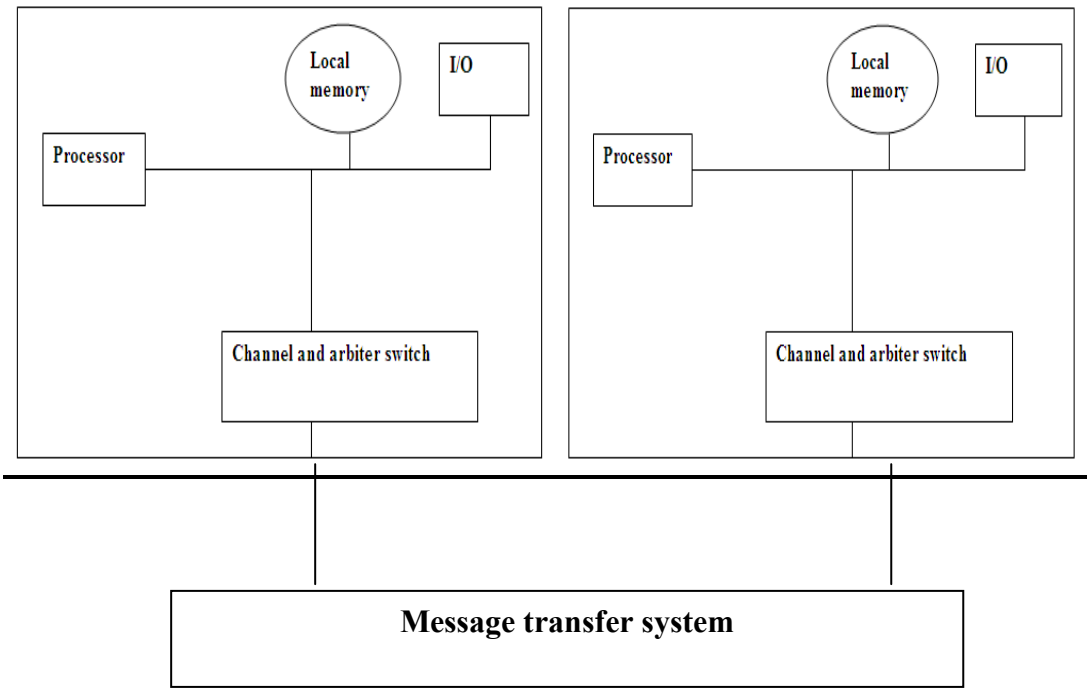
## 1.2 Processor Coupling

### 1.2.1 Loosely Coupled multiprocessor system

Loosely Coupled multiprocessor systems are based on the multiple standalone single or dual processor computers interconnected via a high speed communication system. In such a system, each processor has a set of I/O devices and a large memory [1]. A computer module can be referred to a processor, local memory and its I/O devices. Computer modules communicate through an interconnection network. Since the processor accesses most of its instructions and data from local memory, a loosely coupled system is referred to as distributed system. The Basic organization of a loosely coupled multiprocessor system is shown in figure 1.1:




**(a) Computer module**




**(b)Loose coupling of computer modules**

**Figure 1.1: A Loosely Coupled Multiprocessor System**

### 1.2.2 Tightly Coupled Multiprocessor System

Tightly Coupled multiprocessor systems contain multiple CPUs that are connected at the bus level. Processors communicate through shared memory modules. Each processor may have its own local memory and buffer cache.

## 1.3 Interconnection Networks

Interconnection network [1,2] make a major factor to differentiate modern multiprocessor architectures. They can be categorized such as

- Switching methodology
- Network topology
- Routing Strategy

Interconnection network is complex connection of switches and links permitting processors in multiprocessor system to communicate among themselves or with memory modules.

Performance criteria for Interconnection network are

- Fast Communication
- Low cost
- Reliability during failures
- Efficiency
- Bandwidth

## 1.4 Switching Methodology

Two major switching methodologies are [2]

- Circuit switching
- Packet switching

In Circuit switching, a physical path is actually established between source and destination. This path exists as long as transmission of data.

In Packet switching, data is broken into packets and routed through interconnection network without establishing physical path. These packets are transmitted from source to destination in store and forward manner.

## 1.5 Network Topology

Interconnection network are built up of switching elements. Topology is the pattern in which individual switches are connected to other elements; like processor, memories and other switches [2].

Topologies are categorized in two groups

- Static
- Dynamic

In static topology, links between two processors are passive and dedicated buses cannot be reconfigured for direct connection to other processors. They are mainly used in message passing architectures.

Some important networks are

- Fully connected
- Mesh
- Ring
- Hypercube
- Shuffle exchange

### 1.5.1 Static Topologies

**(i) Linear Array(1-D Mesh)**

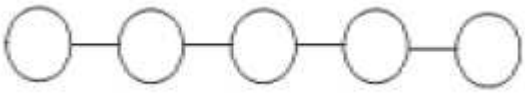


**Figure: 1.3: Linear array**

In this topology, all nodes are connected through single bus. Each node has two connections.

**(ii) Star**

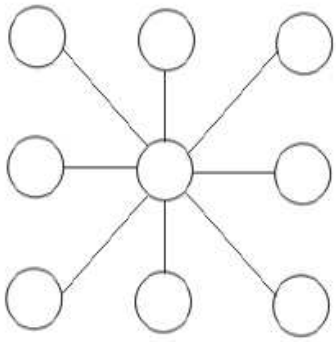


**Figure: 1.4: Star**

In this topology, there is one central node to which all other nodes are connected.

**(iii) Binary Tree**

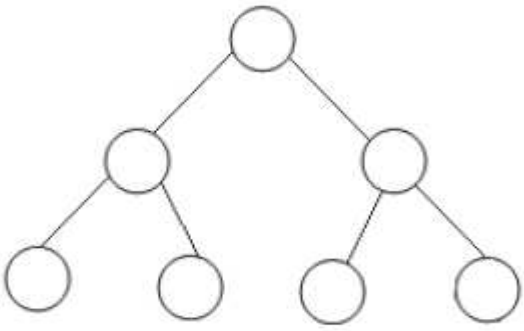


**Figure: 1.5: Binary Tree**

Each node is connected to atmost two nodes.

**(iv) 2-DMesh**

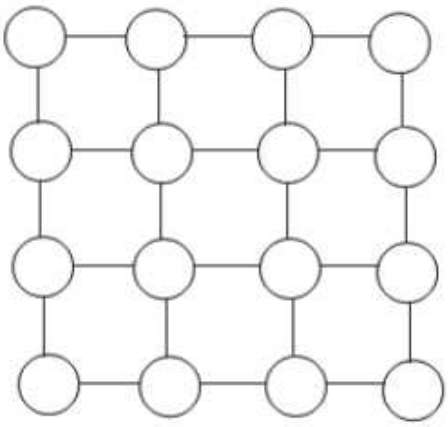


**Figure: 1.6: 2-D Mesh**

The simplest and easiest way to connect the nodes of a parallel computer is to use mesh. Each node can be connected to 2, 3 or 4 nodes. The cost incurred is 2(N x N). A normal sized mesh can be extended to any size.

**(v) Three-D Cube**

In this topology, a node can be connected to 3, 4, 5 or 6 other nodes to communicate.

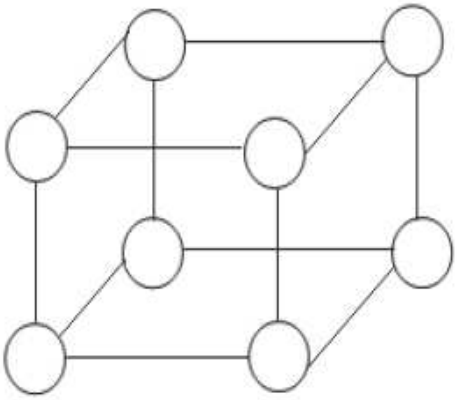


**Figure: 1.7: Three-D Cube**

**(vii) Fully connected or all-to-all**

This is the most powerful interconnection network (topology). Each node is directly connected to all other nodes. Each node has N-1 connections (N-1 nearest neighbors) giving a total of N (N-1) /2 connections for the network implemented for small values of N.

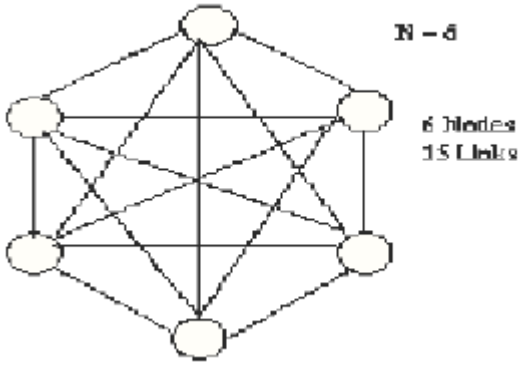



**Figure: 1.8: Fully connected**

## Table 1.1: Static Topologies

| Topology | Node Degree | Diameter | Bisection Width | Arc Connectivity | Cost |
|---|---|---|---|---|---|
| Linear Array | 1 or 2 | N-1 | 1 | 1 | N-1 |
| Ring | 2 | N/2 | 2 | 2 | N |
| Star | 1 or N-1 | 2 | 1 | 1 | N-1 |
| Binary Tree | 1,2 or 3 | 2log((N+1)/2) | 1 | 1 | N+1 |
| 2-D Mesh | 2,3 or 4 | 2(N1/2-1) | N1/2 | 2 | 2(N-N1/2) |

**1. "Approaches to avoid crosstalk in optical omega network", Monir Abdullah:**

Routing in the degree descending of the message conflicts gave the best performance among them. Simulated annealing (SA) algorithm was also used to improve the performance of solving the problem and gave the best results. This paper presents an efficient approach that combines the SA algorithm with the best heuristic algorithms gave much better result in a very minimal time.

**2. "Message Routing and Scheduling in Optical Multistage Networks Using Simulated Annealing", Ajay K Katangur, Yi Pan, Martin D Fraser:**

In this paper, we focus on efficient solution to avoid crosstalk, which is routing traffic through an N*N optical network to avoid coupling two signals within each switching element. Many heuristic algorithms are designed by many researchers to perform this routing such as sequential algorithm, degree descending algorithm, etc. The Simulated annealing algorithm is used in this research to improve the performance and optimizing the result.

**3. "Parallel computing using optical interconnections", Yi Pan, University of Dayton:**

Optical interconnections for communication networks and multiprocessor systems have been studied extensively. A basic element of optical switching networks is a directional coupler with two inputs and two outputs or switching element. Depending on the control voltage applied to it, an input optical signal is either of the two outputs, setting the SE to either the cross or straight state.

**4. "Parallel processing architecture", Jungsun Kim:**

This paper provides a performance study of multistage interconnection networks in packet switching environment. In comparison to earlier work, the model is more extensive- it includes several parameters such as multiple- packet messages, variable buffer size and wait delay at a source. The model is also uniformly applied to several representative networks and thus provides a basis for fair comparison as well as selection of optimal values for parameters.

**5. "A Hybrid Genetic algorithm using Probabilistic selection", M. K. Pakharia:**

In this paper an attempt is made to intermix the search properties of GA, in order to develop a hybrid algorithm which is equally applicable and has a better searching ability and a power to reach near optimal solution. The author has incorporated an SA like selection criteria in a GA framework. The selection process need not have to wait for the generation of a complete pool of chromosomes. This leads to the development of a very fast hardware - software ensemble, designed with a pipelined architecture, to solve the most complicated types of optimization problems.

**6. "Isomorphism of Conflict Graphs in Multistage Interconnection Networks", Nabanita Das, B. Bharghab Bhattacharya:**

In a hybrid optical multistage interconnection networks (MINs), optical signals are routed by electronically controlled switches using directional couplers. A relevant design problem is to minimize the path-dependent loss of the optical signal, which is directly proportional to the number of couplers, i.e., the number of switches through which the signal has to pass. In general, given the network size and the type of the MIN, the number of stages is a constant. Hence, an input signal has to pass through a fixed number of couplers to reach the output. In this paper, it is shown that the routing delay and path-dependent loss in a fixed-stage $N \times N$ MIN, can be significantly reduced on the average by using a variable-stage shuffle-exchange network instead.

**7. "A Comparison Study of Optical MIN Networks with Parallel Planes", Qimin Yang:**

The major challenges in designing optical switched interconnection networks include the lack of optical buffering and the crosstalk originated from the optical switching elements. Vertical Stacking Banyan networks and Data Vortex networks provide two different approaches to solve the problem. Both networks utilize several parallel planes in conjunction with traffic scheduling to eliminate the contention and the crosstalk within the routing nodes. While the two share similar compromise between system cost and switch performance, their traffic control mechanism has fundamental difference due to specific configurations of the parallel routing planes. This paper provides a comparison study of two network approaches, focusing on the overall system complexity and the resulting switch performance in throughput and latency.

**8. "Fast method to find conflicts in optical multistage interconnection networks", F. Abed, M. Othman:**

This paper presents, one undesirable problem introduced by the optical multistage interconnection network is a crosstalk that is caused by coupling two signals within a switching element is considered. To avoid a crosstalk, many approaches have been proposed such as time domain and space domain approaches. Because the messages should be partitioned into several groups to send to the network, window method is used to find out which messages should not be in the same group. In this paper fast window method based on bitwise operations (BWM) is represented. This algorithm reduces the execution time approximately more than ten times compared with previous algorithms.

**9. "Fast ZeroX algorithm for efficient message routing in optical multistage interconnection networks", T. D. Shahida, M. Othman, M. Khazani:**

In this paper, a fast and efficient crosstalk-free routing algorithm is proposed to enhance message routing in optical multistage interconnection networks (OMINs). The new Fast ZeroXY algorithm is designed based on the Zero algorithms, which uses the time dilation approach to eliminate the negative effect of crosstalk associated with optical switching in the optical Omega network. To evaluate the performance of the new algorithm, a crosstalk-free version of the original ZeroXY algorithm is developed extended from the Improved ZeroXY algorithm, called the Modified ZeroXY algorithm.

**10. "A New Algorithm for Routing and Scheduling in Optical Multistage Interconnection Networks", S. C. Chau and T. Xiao:**

Multistage Interconnection Networks (MINs) are popular in switching and communication applications. Recently, there have also been significant advances in electro-optic switches that have made Optical MINs (OMINs) a good choice for the high channel bandwidth and low communication latency of high performance computing and/or communication applications. Many heuristic algorithms, including the sequence increasing algorithm, sequence decreasing algorithm, degree-ascending algorithm, and degree descending algorithm have been proposed to perform this routing problem. Some researchers have adopted the Genetic Algorithm (GA) and the

Simulated Annealing (SA) algorithm to improve the performance of the routing and scheduling for a permutation

**11. "Optical Multistage Interconnection Networks: New Challenges and Approaches", Y. Pan, C. Qiao and Y. Yang:**

In this paper there is an introduction about Optical interconnections for communication networks and multiprocessor systems. It is studied that a basic element of optical switching networks is a directional coupler with two inputs and two outputs or switching elements. Depending on the control voltage applied to it, an input optical signal is coupled to either of the two outputs, setting the SE to either the straight or cross state. A class of topologies that can be used to construct optical networks is multistage interconnection networks, which interconnect their inputs and outputs via several stages of SEs (Switching elements), are also mentioned in this paper. In this paper various ways to deal with the unique problem of avoiding crosstalk in the SEs (Switching elements) are given.

**12. "Permutation Capability of Optical Multistage Interconnection Networks" Yuanyuan Yang, Yi Pan:**

In this paper, optical multistage interconnection network (OMINs) is studied. Advances in electro-optic technologies have made optical communication a promising networking choice to meet the increasing demands for high channel bandwidth and low communication latency of high-performance computing/communication applications. Although optical MINs hold great promise and have demonstrated advantages over their electronic counterpart, they also hold their own challenges. Due to the unique properties of optics, crosstalk in optical switches should be avoided to make them work properly. Most of the research work described in the literature is for electronic MINs, and hence, crosstalk is not considered. In this paper, we introduce a new concept semi permutation to analyze the permutation capability of optical MINs under the constraint of avoiding crosstalk.

# Chapter -3 Problem Definition

The problem undertaken for dissertation is **"Analysis of cross avoidance techniques in Optical Multistage Interconnection Networks".** The major problem called crosstalk is caused by coupling of two signals within a switching element. A lot of work has already been done in designing various algorithms and methods so that good quality solution can be obtained. Window method is the method that is used to find the messages that are not in the same group because it causes crosstalk in the network. Then comes Improved window method in which the first window is eliminated for this we make the conflict matrix initialized to 0, here number of windows is M-1, where $M=\log_2 N$ and N is size of network. It takes less time to find conflicts than the windows method. Then comes the most efficient method called Bitwise Window method. In this method, source and destination address is in decimal format. There is only one decimal number in each row and each window for comparison and finding a conflict. Bitwise window method is the best method as its execution time is very less in comparison to other methods. Four Heuristic algorithms are there that are also used to avoid crosstalk. The Bitwise heuristic techniques can improve the time nearly more than 10 times special when the network size is large.

All the window methods have been implemented using 'C' language and results have been analyzed.

# Chapter-4 Multistage Interconnection Networks

## 4.1 Introduction

Multi-stage interconnection networks (MINs) [3,4] consist of more than one stages of small interconnection elements called switching elements and links interconnecting them. Multi-stage interconnection networks are described by three characteristic features: the switching elements, network topology and control structures. A multi-stage interconnection network is actually a compromise between crossbar and shared bus networks, indicated in the following table; describing the properties of various types of multiprocessor interconnections. MINs are popular in switching and communication applications. It consists of more than one stages of small interconnection elements called switching elements and links interconnecting them. Multistage networks are described by three characteristic features: the switching elements, and network topology and control structures.

**Table 4.1 Comparison of bus, crossbar and MINs [3]**

| Property | Bus | Crossbar | Multi-stage |
|---|---|---|---|
| Speed | Low | High | High |
| Cost | Low | High | Moderate |
| Reliability | Low | High | High |
| Configurability | High | Low | Moderate |
| Complexity | Low | High | Moderate |

Multi-stage interconnection networks:

- Attempt to reduce cost.
- Attempt to decrease diameter where diameter is the longest path between any two nodes.

## 4.2 Classification of Multistage Interconnection Networks

Multi-stage interconnection networks can be classified according to different categories. The main classification categories are path, switches and control.

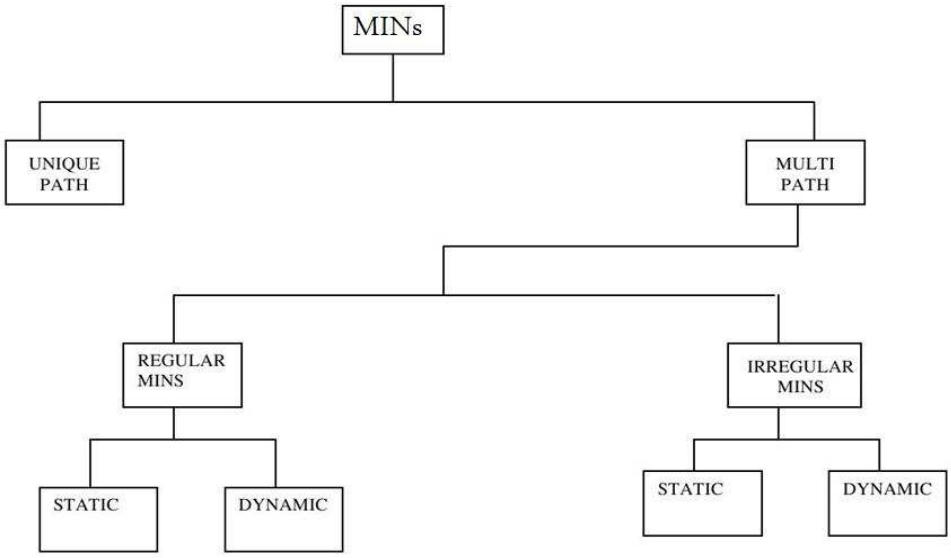



**Figure 4.1: Classification of MINs**

### 4.2.1 Classification according to path

MINs can be classified according to path as unique and multi-path networks, as described below [5]:

- ***Unique path networks:*** These networks provide a unique path between every source and destination. The failure of any switching element along the path disconnects some source-destination pairs, so adversely affecting the capabilities of existing network. These networks are not reliable for a large multiprocessor system, as they cannot tolerate even a single fault. In case of multiple requests, a source destination connection may be blocked by a previously established connection, thus providing a poor performance.
- ***Multi path networks****:* These networks provide more than one paths between a given source and a destination. In case, there is a failure of one switching element in the path, the request is routed through some alternative path. Unique path multi-stage interconnection networks can be made multi path by adding redundancy in the form of extra switching elements, links, stages, sub networks, by increasing the size of switching elements or using multiple networks. Multi path multi-stage interconnection networks can be either static or dynamic. For static networks, if a fault is encountered, then data has to

backtrack, to the source or some fixed point to select an alternative path in the network. The implementation of backtracking is expensive in terms of the hardware. In dynamic networks, if a fault is encountered in a particular stage, a switching element in preceding stage will reroute data through an alternative available path.

### 4.2.2 Classification according to switches

MINs can be classified according to switches as regular and irregular networks, as described below [5]

- ***Regular networks:*** Regular multi-stage interconnection networks have an equal number of switching elements per stage; as a result they may impose equal time delay to all the requests passing through them.
- ***Irregular networks:*** Irregular multi-stage interconnection networks have unequal number of switching elements as each stage and thus they are inherently multi path in nature. For a given source destination pair, different path lengths are available.

### 4.2.3 Classification according to control

MINs can be classified according to control as flip controlled and distributed control networks, as described below [5]

- ***Flip controlled networks****:* Flip controlled multi-stage interconnection networks have a common control signal for switching in various switching elements at a given stage. These networks are less complicated due to lesser number of control signals but have lesser bandwidth.
- ***Distributed control networks****:* Distributed control multi-stage interconnection networks have a separate control signal for every switching element. These have higher bandwidth due to selection of source destination pair at a given time and are quite complex.

### 4.2.4 Other classification [6]

- ***Blocking networks****:* In blocking network, simultaneous connections of more than one terminal may result in conflict in use of network communication links. Example of blocking network is Omega network.

- ***Non blocking networks:*** A network is called non-blocking if it is possible to route data from any source to any destination, in presence of other established source destination routes, provided no two sources have same destination. In other words, a network that can handle all possible connections without blocking is called non-blocking network.

## 4.3 Types of connections in MINs

There are four types of connections which are commonly used in multi-stage interconnection networks [6]. These are

**One to one connection**: a one to one connection passes information from a source to a destination. The exact route taken by the information is determined by the path itself.

**Multi path connection:** Multi path means many one to one connections are active simultaneously.

**Permutation connection**: A set of one to one connections such that no two connections have the same source or destination. Such connections are meaningful only in cases of equal number of sources and destinations.

**Broadcast connection**: Information flows from source to various destinations either some or all. Thus a number of destinations simultaneously receive the information.

## 4.4 Fault Tolerance

A fault tolerant multi-stage interconnection network provides service even under the faults. Fault can be permanent or transient in nature [5,7]. Fault tolerance is a criterion that must be met for the network which has tolerated a given fault or faults. A network is single fault tolerant if it can function as specified by its fault tolerance criteria despite any single fault conforming to its fault models. In general, if any set of i-faults can be tolerated by a network, then network is said to be i-fault tolerant. A network that can tolerate some instances of i-faults is robust although not i-fault tolerant.

## 4.5 Switching elements

These switches are the devices having multiple input and outputs. A two-function switch can assume either the straight or the exchange states [6]:

- Straight
- Exchange
- Upper broadcast
- Lower broadcast

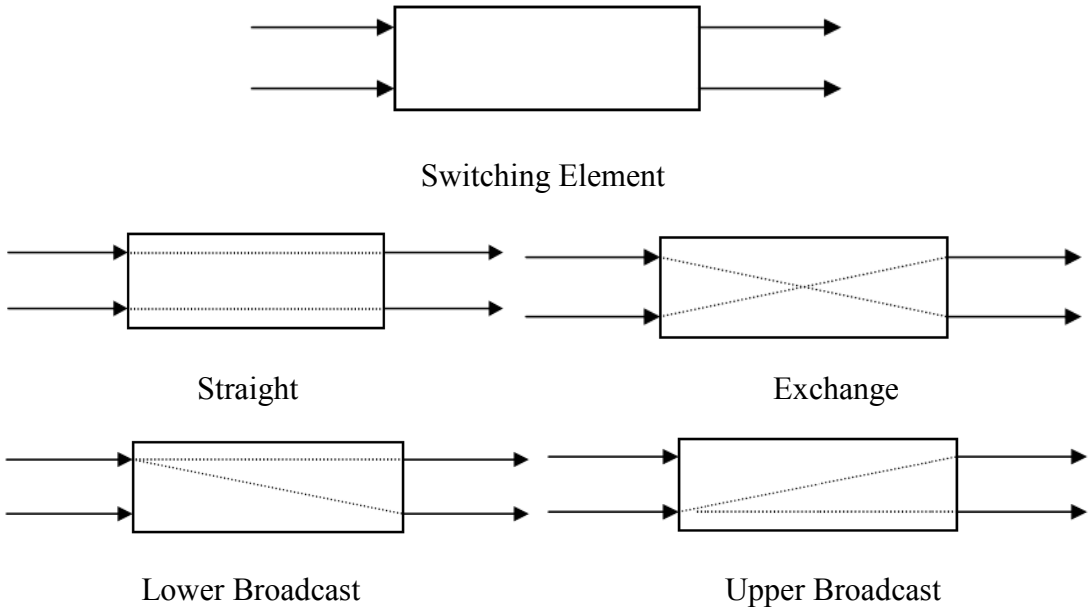



**Figure 4.2: Different states of switching elements**

## 4.6 Type of routing in MINs

There is basically three type of routing that is commonly used in multistage interconnection networks [8].

- **Non – adaptive routing**: In this method, a source leans a fault when a path it is attempting to establish reaches the faulty network component. A notice of fault is sent to the source, which tries next alternative path. This method has poor performance though it requires less hardware.
- **Adaptive routing**: The adaptive routing can be of the following types:
  - Notification on demand: With notification on demand, a source maintains a table of faults it encountered in attempting to establish paths and uses this information to guide the future routing.
  - Broadcast routing: With broadcast notification of fault, all sources are notified of the fault components are diagnosed.
- **Dynamic routing**: a dynamic routing can be accomplished in multistage interconnection networks constructed of switches, which are capable of

performing the necessary tag revision.

The most widely used MINs are electronic MINs. In electronic MINs, electricity is used, whereas in optical MINs, light is used to transmit messages.

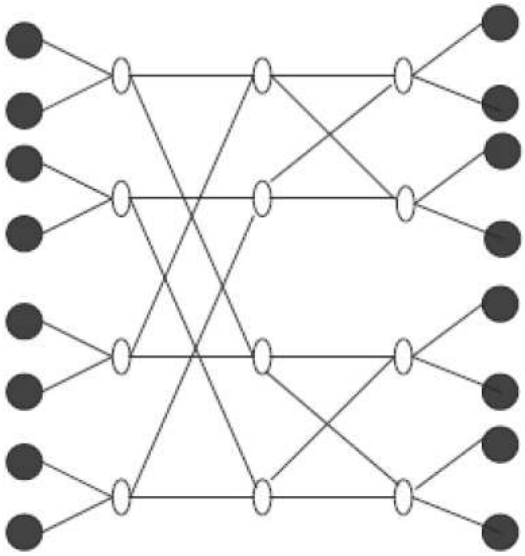


**Figure 4.3: Multistage Interconnection Networks**

## 4.7 Optical Multistage Interconnection Networks

### 4.7.1 Introduction

In OMIN, optical signal is converted to/from electrical signal at the network input/output as shown in figure 4.4. OMINs work in optical domain [11].

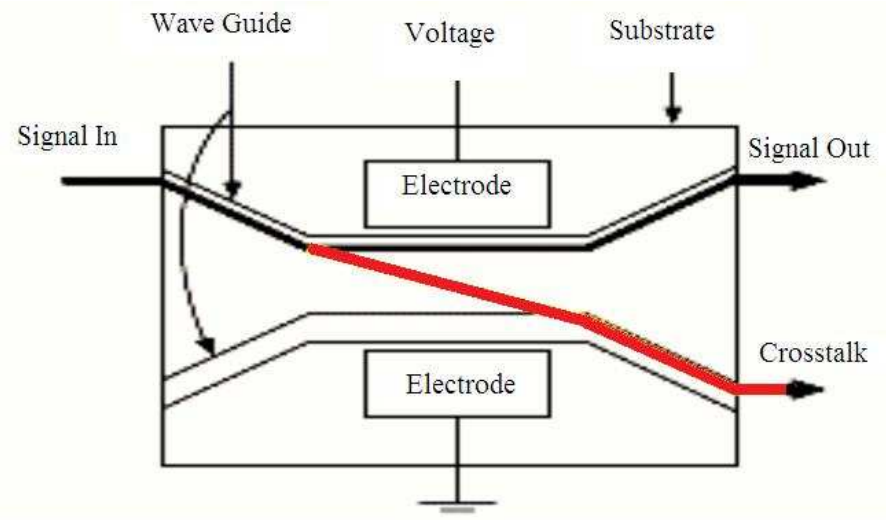



**Figure 4.4: Crosstalk in a switching element [11]**

This advantage makes signal transmission in optical network faster. Optical network provide higher capacity and reduced costs for new applications such as the Internet, video and multimedia interaction, and advanced digital services.

### 4.7.2 Various Optical MINs [12]

- **Crossbar network**: Squared, wide – sense non blocking network without crossover. It has $N^2$ switches.
- **Clos network**: It is a 3 – stage network with 2 stages of r n*m switches and one middle stage of m r*r switches.
- **Double crossbar network**: It consist of two crossbar like structures placed on top of each other. It is non blocking network. It uses $2N^2$ switches.
- **N-stage planar network:** It is rearrangeable non blocking network. It has first order switch crosstalk but no crossover. Its path length is N.
- **Banyan network**: It has best results on number of switch elements ((NlogN)/2).

### 4.7.3 Characteristics of Optical MINs

#### 4.7.3.1 Switching Model [13]

Optical MINs use optical switching, which involves the switching of optical signals, rather than electronic signals as in conventional electronic systems. There are two types of optical systems, which can be identified. First is a hybrid system in which optical signals are switches, but both the switch control and routing decisions are carried out electronically at a speed that can be much lower than the bit rate of optical signals being switched. The second is all optical switching, which would potentially overcome the speed mismatch problem associated with hybrid approach. Packet switching is very common in an electronic MIN. Hybrid optical MINs used electronically controlled optical switching elements. As packet switching requires, conversion between optical signals and electronic ones, which is very costly. When comparing the switching speed of SEs, the process of determining their settings based on the address information in each packet could become a significant overhead. For these reasons, circuit switching is usually preferred in optical MINs, which a direct

connection between the source and the destination is set up by a network controller before data is sent. As neither packet processing nor buffering is needed at each SE, circuit switching can be implemented with SEs that is simpler and faster than those required for packet switching.

#### 4.7.3.2 Shuffle-Exchange Optical MINs

A shuffle-exchange multistage interconnection network (SEN) is one network in a large class of topologically equivalent MINs that include the omega, indirect binary n-cube, baseline, and generalized cube [14]. Figure 4.5 is an example of an 8×8 SEN. Each switching element (SE), the basic building block of a SEN, can be viewed as a 2×2 SEN. The SE can either transmit the inputs straight through itself or has a cross connections.

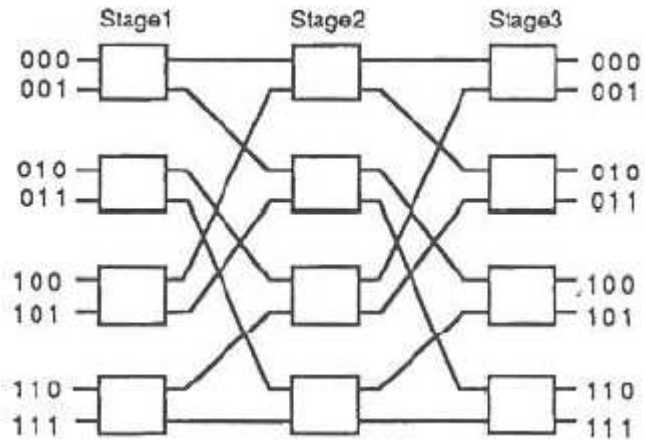



**Figure 4.5: 8x8 Shuffle - exchange multistage interconnection network [14]**

### 4.7.4 Parameters of optical MINs

All MINs are blocking architectures, and thus are limited in the amount of connectivity they can support. The biggest advantage is the substantial reduction in the path length, $O(\log2(n))$, as opposed to $O(2n - 1)$ for the crossbar topology [14,15]. This property has made MINs extremely popular in electrical switching systems in telecommunications and computing applications because network complexity for a given level of connectivity is very low relative to other topologies. In the integrated optical communication networks, MINs can be implemented on a single plane but they cannot be implemented without waveguides crossing. Optical waveguides

crossing are transmission lines with axial symmetry and rectangular cross sections, which provide communication between the electro-optical transmitter and receiver creating optical interconnects. Determining an expression for the number of waveguides crossings in a shuffle-exchange network of arbitrary $n$ ($n$ is a power of two) is relatively simple because the same permutation occurs in each stage. Counting the number of crossings for the first, second, ...$n$/2 waveguides results in 0, 1, …, $n$/2 –1 crossings, respectively. Adding the first and last terms in this series, the second and second to last, the third and third to last, and so on, for all ($n$/4) pairs, clearly results in a total of ($n$/4)($n$/2 – 1) crossings in each stage of the network. Because the shuffle-exchange network has a total of log2($n$) – 1 stages, the total number of waveguide crossings equals ($n$/4)($n$/2 – 1) (log2($n$) – 1).

### 4.7.5 Reliability Evaluation

Reliability of a network is concerned with the ability of a network to carry out its desired network operation successfully. In our case, interconnection networks for processor-processor and processor-memory information exchanges in multiprocessing parallel processing systems, contribute appreciably to the performance as well as the reliability of the overall system. The reliability measures of particular interest are: terminal reliability (TR), broadcast reliability (BR), and network reliability (NR) [15, 32].

***Terminal Reliability* (*TR*):** Generally used as a measure of robustness of a MIN, is the probability of existence of at least one fault free path between a designated pair of input ($s$) and output ($t$) terminals (two terminal). This is denoted by $Rst(G)$, if $G$ is directed, there should at least be one directed path between $s$and $t$ .

***Broadcast Reliability* (*BR*):** Its ability is to broadcast data from a given input terminal to all the output terminals of a network.

***Network Reliability* (*NR*):** It is defined as the probability that there exists a connection between each input to all outputs.

### 4.7.6 Limitations of Optical MINs

Due to difference in speeds of electronic and optical switching elements and nature of optical signals, optical MINs hold their own challenges [11].

**Path dependent loss**: Path dependent loss means optical signals become weak after passing through optical path. In large MIN, a substantial art of this path dependent loss is directly proportional to number of couplers that optical path passes through. Hence, it depends on architecture used and network size.

**Optical crosstalk**: Optical crosstalk occurs when two signal channels interact with each other. There are two ways in which optical paths can interact in a planar switching network. The channels carrying signals could cross each other in order to embed a particular topology.

## 5.1 Window Method

Window method is the method that is used to find the messages that are not in the same group because it causes crosstalk in the network. If we consider the network of size N*N, there are N source and N destination address. Combination matrix is formed by combining source and its destination address. From this, optical window size is M-1, where $M=\log_2 N$ and N is size of network. In window method, number of windows is equal to number of stages [15].

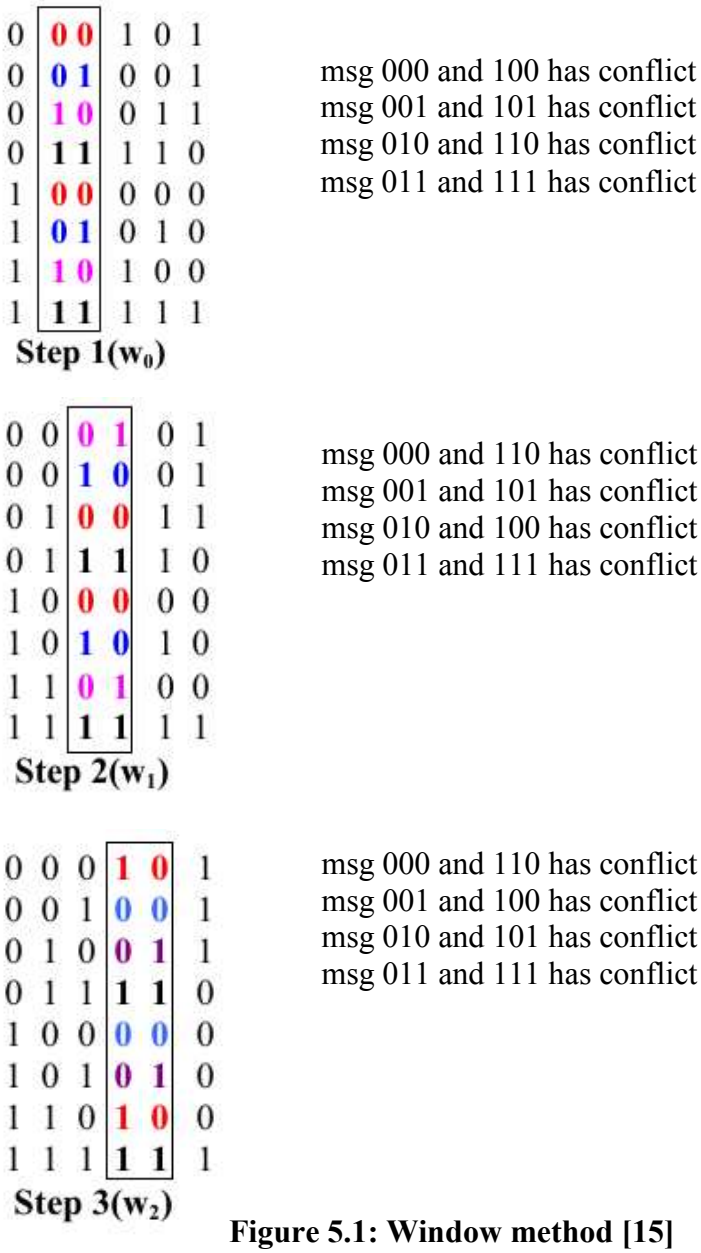



**Figure 5.1: Window method [15]**

### 5.1.1 Conflict Graph

After finding conflicts using window method, conflict graph is generated shown in figure 5.2 [15]. The number of nodes is the size of the network. The nodes that are having conflict are connected through edge. Degree of each message is the number of conflicts to the other message.

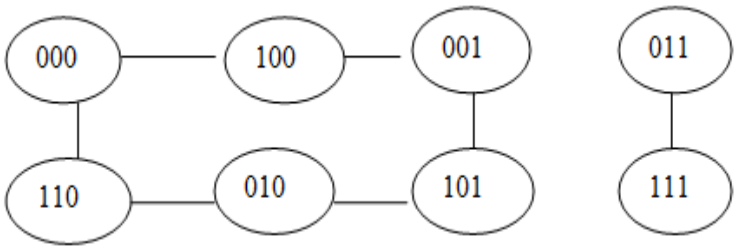



**Figure-5.2 Conflict graph**

### 5.1.2 Conflict Matrix

The conflict matrix is a square matrix with N*N entry, it consists of the output of the window method shown in table 5.1. The definition of Conflict Matrix is the matrix $M_{ij}$ with size N*N. N is the size of the network.

$M_{ij}$ = {1 if conflict

{0 if no conflict

**Table 5.1: Conflict Matrix [15]**

| msg | 000 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## 5.2 Improved Window Method

In this method the first window is eliminated for this we make the conflict matrix initialized to 0, here number of windows is M-1. It takes less time to find conflicts than the windows method. Therefore, it is called improved window method [13,15]. It is shown in figure 5.3.

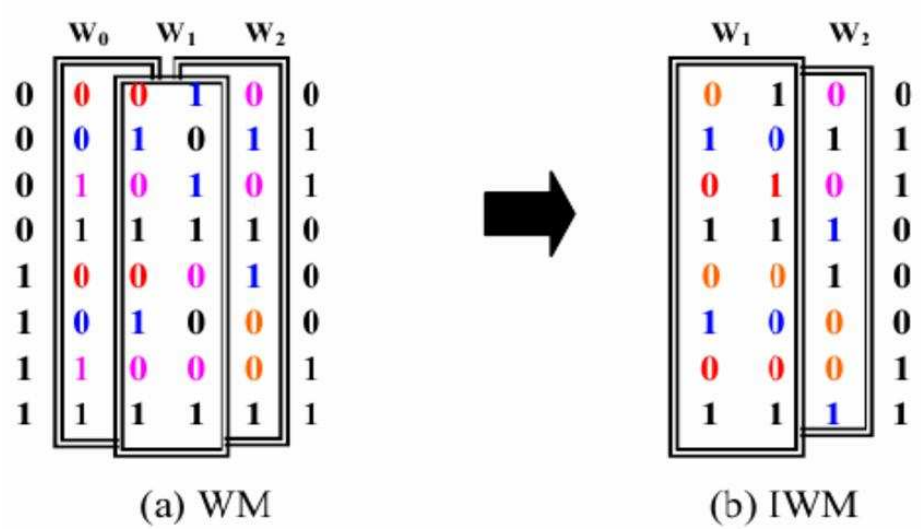



**Figure 5.3: Conversion of Window Method to Improved Window Method [13,15]**

## 5.3 Bitwise Combination Matrix

For Bitwise combination Matrix, all binary bits of single rows in each windows are converted to decimal and number of windows is reduced to n as shown in figure 5.4. By this method, time is reduced approximately by ten times. It is very effective method even when the network is very large.

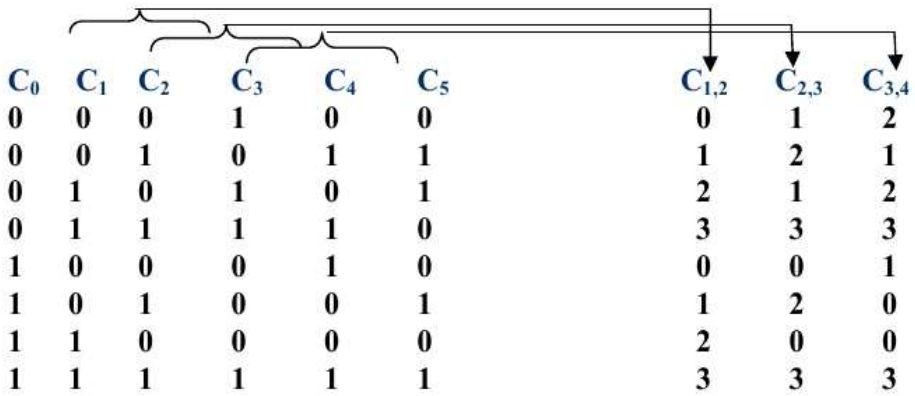



**Figure 5.4: Bitwise Combination Matrix [13]**

## 5.4 Bitwise Window Method

In this method, source and destination address is in decimal format. Number of windows is $\log_2 N$. It is shown in figure 5.5. Thus, from combination matrix, the optical window size is only one for a different network size and the number of window is $\log_2 N$ [13]. In other words, there are only one decimal number in each row and each window for comparison and finding a conflict.

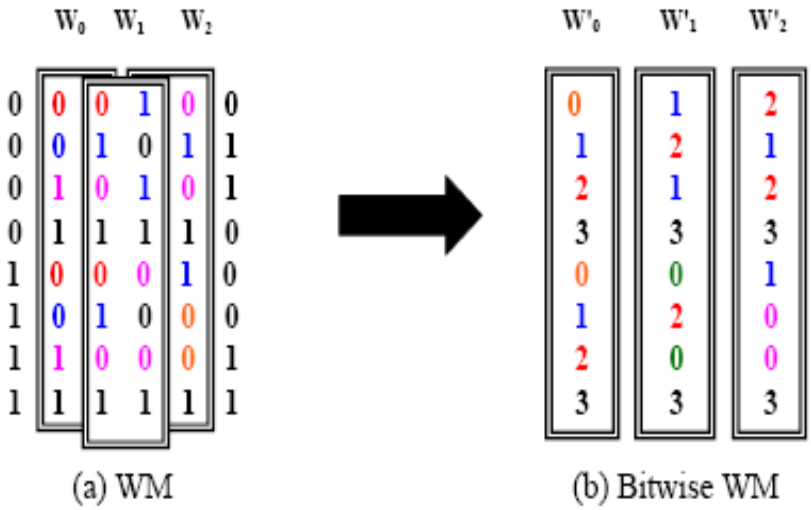



**Figure 5.5: Bitwise Window Method [17]**

## 5.5 Sequential Window Method

SWM is used to find the conflicts among the messages to be sent. It can be described briefly as follows [16]. Given a permutation, we combine each source address and its corresponding destination address to produce a matrix. The optical window size is the m-I where m = $\log_2 N$ and N is the size of the network. We use this window on the produced matrix from left to right except the first column and last column. If two messages have the same bit pattern in any of the optical window, they will cause conflict in the network. That means they cannot be in the same group, hence, they have to be routed in different passes. In Figure 5.6, take the second and third columns as a matrix, messages 000 and 100 in this window have the same bit pattern of 00 inside the window and hence have a conflict. The bit patterns can be any of the four combinations of 00, 01, 10, and 11, and hence are shaded using different colors.

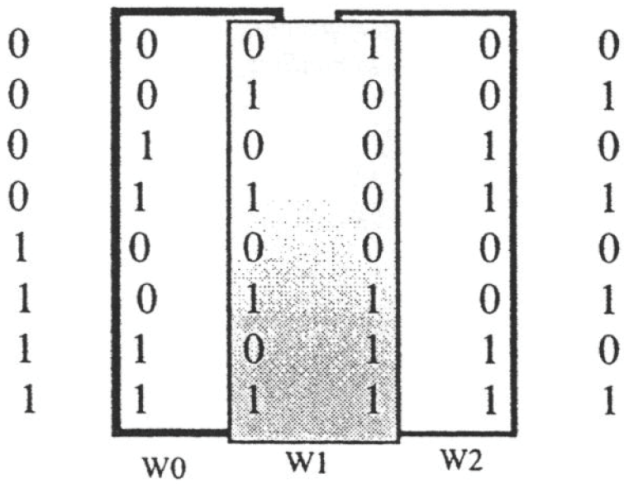



**Figure 5.6: Sequential Window method [16]**

From this window, the messages 000, 001, 010 and 011 have conflict with messages 100, 101, 110 and 111 respectively. Because of the dependence of each window, we can execute this method in parallel to reduce the execution time.

## 5.6 Unbalanced Parallel Window Method

In this algorithm, the master processor will send the whole window to the slave processor which in turn will do the comparisons and then put the result in a buffer [16]. Thereafter, the slave will send the buffer to the master. Finally the master processor will generate the conflict matrix. It starts with description of operations of the master process Po and finishes by the same description of the slave processes Pp. In this algorithm, some processors are not working while the others are working. In another meaning, there is neither throughput nor load balancing. As an example of an 8x8 network size, when the first window is eliminated by IWM algorithm, the number of window will be only two. The master process will send the first window to process 0 while the second window will be sent to process P1 as shown in figure 5.7.

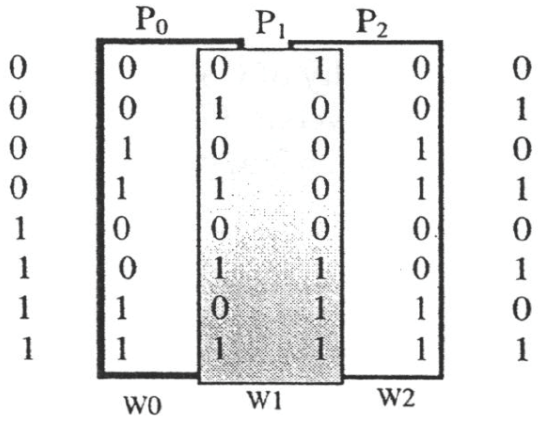



**Figure 5.7: Unbalanced Parallel Window Method [16]**

In this algorithm, the master process works as a control process just in the beginning, when the master is waiting the results from slaves, it does the comparison of one window. After that, it totally works as a control process.

## 5.7 Balanced Parallel Window Method

Load balancing is the task of equally dividing work among the available processors. This can easily be done when the same operations are being performed by all the processes. In this algorithm, an efficient approach is proposed to solve the unbalancing problem [17]. An independent problem is found in window itself. The problem is divided into small sub problems. In other words, each window is divided into many Sub Windows. In each sub window, there is a conflict. Each window in network size 8x8 can be divided into seven independent sub windows. In general, each window in network size N x N can be derived into N-1 sub windows. The division of window is made according to the contents of figure 5.8.

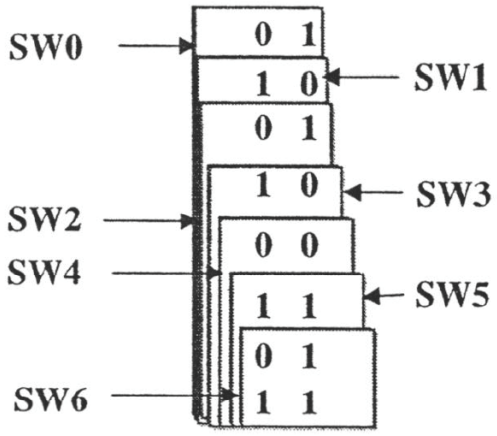



**Figure 5.8: Decompose Window in the BPWM Algorithm [17]**

Now each SW can be sent to the slave processor. In this case, the communication overhead will be increased but a broadcast command is used to decrease this overhead that will give arise to problem in how the slave can determine the sub windows. To solve this problem, the master processor will send a flag to each slave to determine the demand sub windows. When the master processor sends a flag to the slave, as a result the communication overhead will increase.

## 5.8 Fast Zero X Algorithm

The algorithm is developed to optimally minimize the execution time of ZeroX algorithms [18]. The messages are scheduled in three passes including messages 000, 011, 101 and 110 for the first pass, messages 001, 010 and 100 for the second pass, and message 111 for the last pass as shown in figure 5.9. In the first pass, routing input messages 101 and 110 simultaneously will result in crosstalk since destination 101 and 100 for each respective input message is connected to the same switching element in the network. The Fast ZeroX algorithm is shown to eliminate the crosstalk completely.

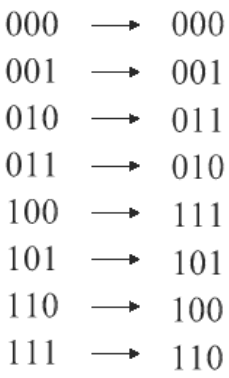



**Figure 5.9: A source to destination permutation [18]**

As indicated by its name, the algorithm is developed to optimally minimize the execution time of ZeroX algorithms. Based on analysis, it is concluded that the original and Improved ZeroX algorithm involves a lot of iterative procedures in the algorithm's functions to find the intersection as well as to ensure no conflicts between the messages in each pass after adding successful intersections. The Fast ZeroX algorithm successfully avoids crosstalk for any given permutation. The Framework of the Fast Zero X algorithm is shown in figure 5.10.

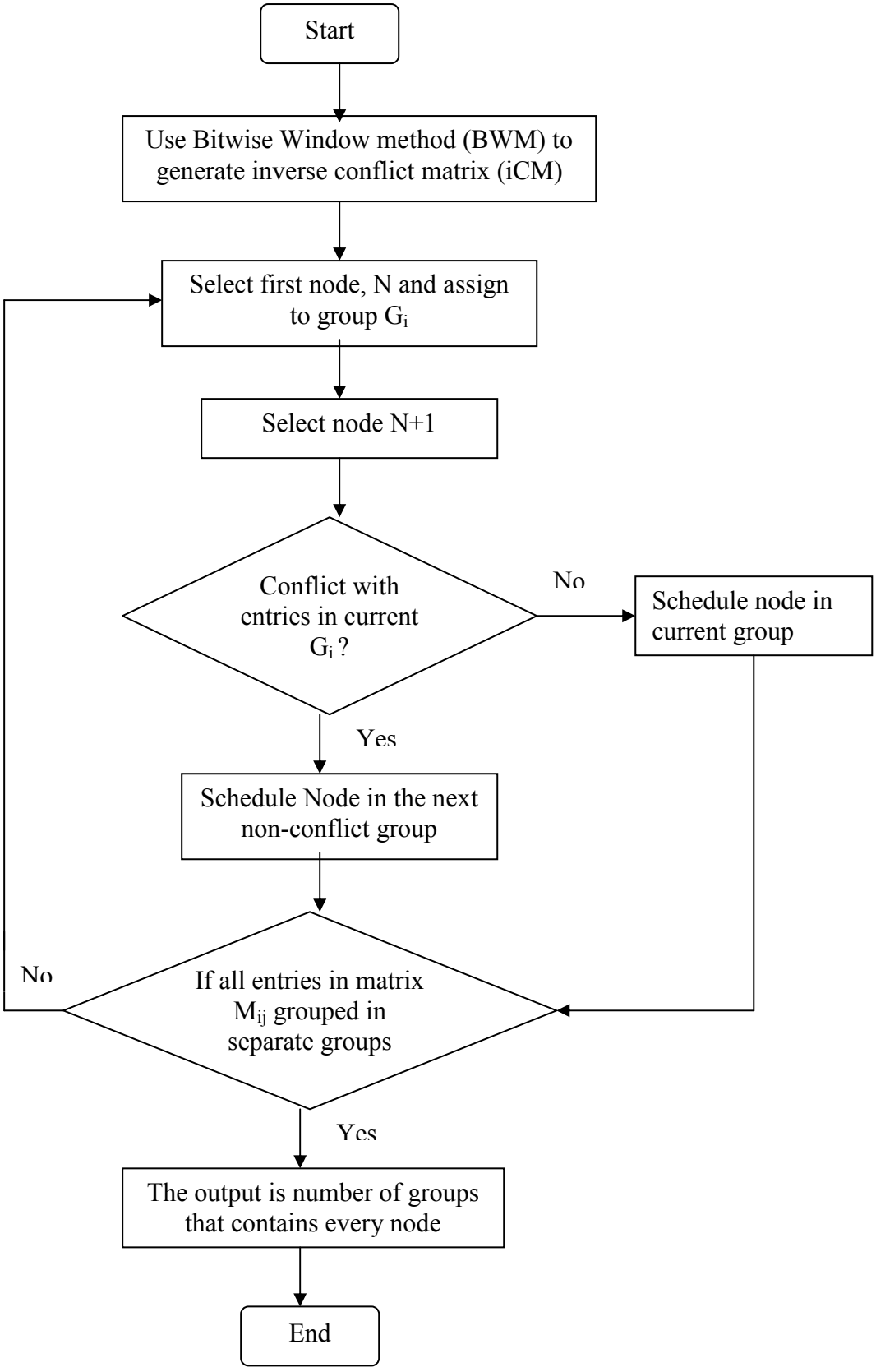



**Figure 5.10: Fast Zero X algorithm framework [18]**

## 5.9 Fast Zero Y Algorithm

The new Fast ZeroY algorithm solved this problem by adopting the inverse conflict matrix shown in table 5.2 [19].

**Table 5.2: An inverse conflict matrix**

| Message | 000 | 001 | 010 | 011 | 100 | 101 | 110 | 111 |
|---|---|---|---|---|---|---|---|---|
| 000 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 |
| 001 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 010 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| 011 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| 100 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| 101 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 |
| 110 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 111 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |

After obtaining the iCM, the first Group 1 is initialized with the message with destination address equals to the last node, *N* of the network. Next, message with address *N-1* is selected and the intersection between this message and the messages in Group 1 is checked in the iCM. If the intersections result in 0 values for all messages in Group 1, this message is directly added to Group 1[19]. Otherwise, it is scheduled in the next non-conflicting pass. The process is repeated for the rest of the other messages. In Fast ZeroY algorithm, the Unique Case and Refine function is removed. Scheduling of messages is more straightforward from the inverse conflict matrix without prior row summation essential in the original ZeroY algorithm that definitely consumes more time. The framework of the Fast ZeroY algorithm is presented in the figure 5.11.

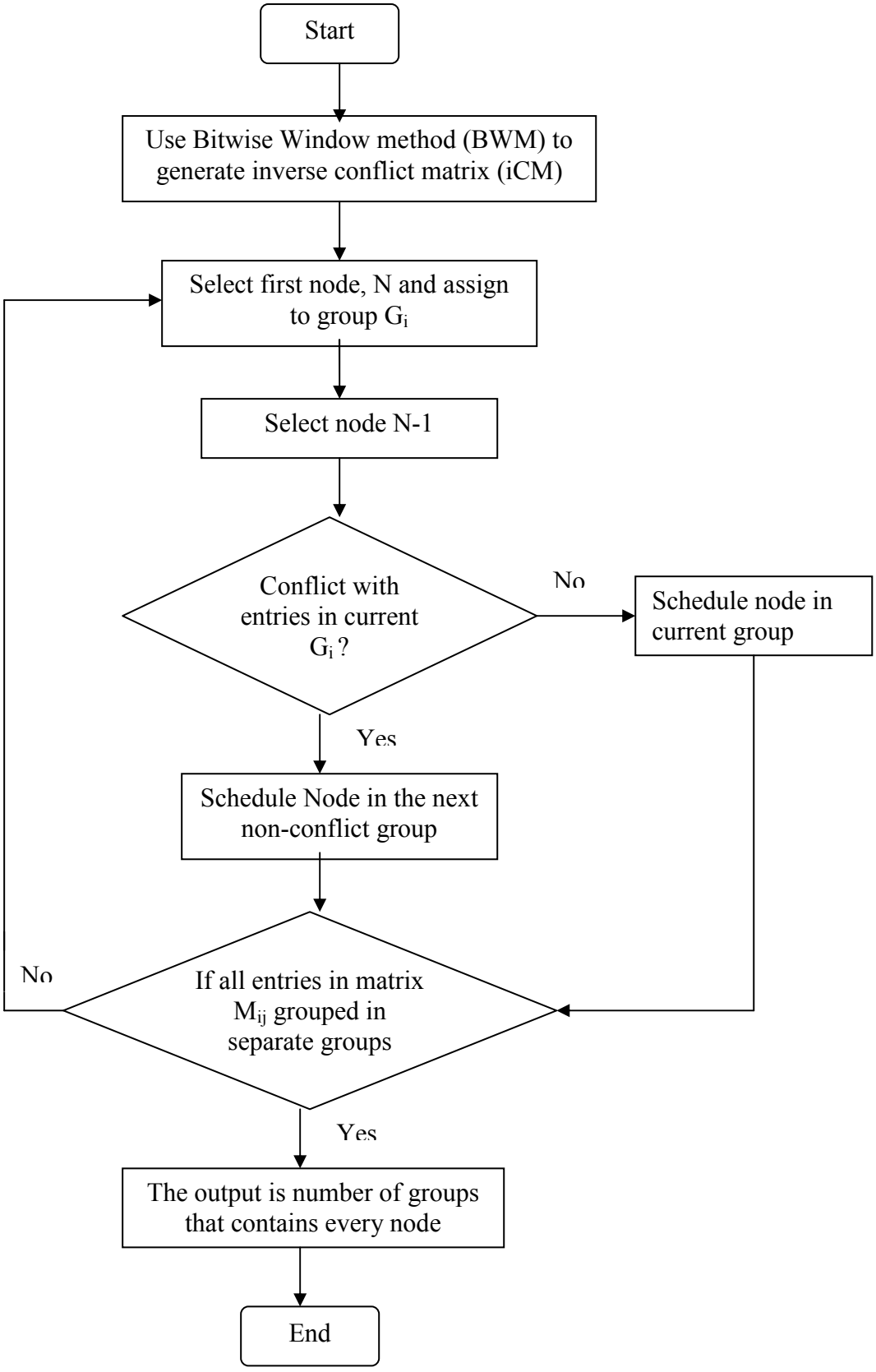



**Figure 5.11: Fast Zero Y algorithm framework [19]**

## 5.10 Fast Zero XY algorithm

The Fast ZeroXY algorithm is designed based on the idea of combining both the Fast ZeroX and Fast ZeroY algorithm [20]. For each permutation set, messages are scheduled first according to Fast ZeroX algorithm followed by the Fast ZeroY algorithm to efficiently partition the messages into crosstalk-free passes as shown in Table 5.3. Result of the Fast ZeroXY algorithm is the result obtained from comparing both the algorithms after execution. The goal of the Fast ZeroXY algorithm is to optimize the average number of passes needed to route a particular permutation as well as the average execution time of the algorithm. The Fast ZeroXY algorithm is presented to efficiently avoid crosstalk in optical MINs. The Fast ZeroXY algorithm successfully improves the execution time especially when the network size is larger. The Fast ZeroXY algorithm is also proven to route a permutation in reduced number of passes in average compared to that of the Fast ZeroX and Fast ZeroY algorithms. Fast Zero XY Algorithm Framework is shown in figure 5.12.

**Table 5.3: The Inverse Conflict Matrix [20]**

| Message | 000 | 001 | 010 | 011 | 100 | 101 | 110 | 111 |
|---|---|---|---|---|---|---|---|---|
| 000 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 |
| 001 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| 010 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 011 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 |
| 100 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 101 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| 110 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 |
| 111 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |

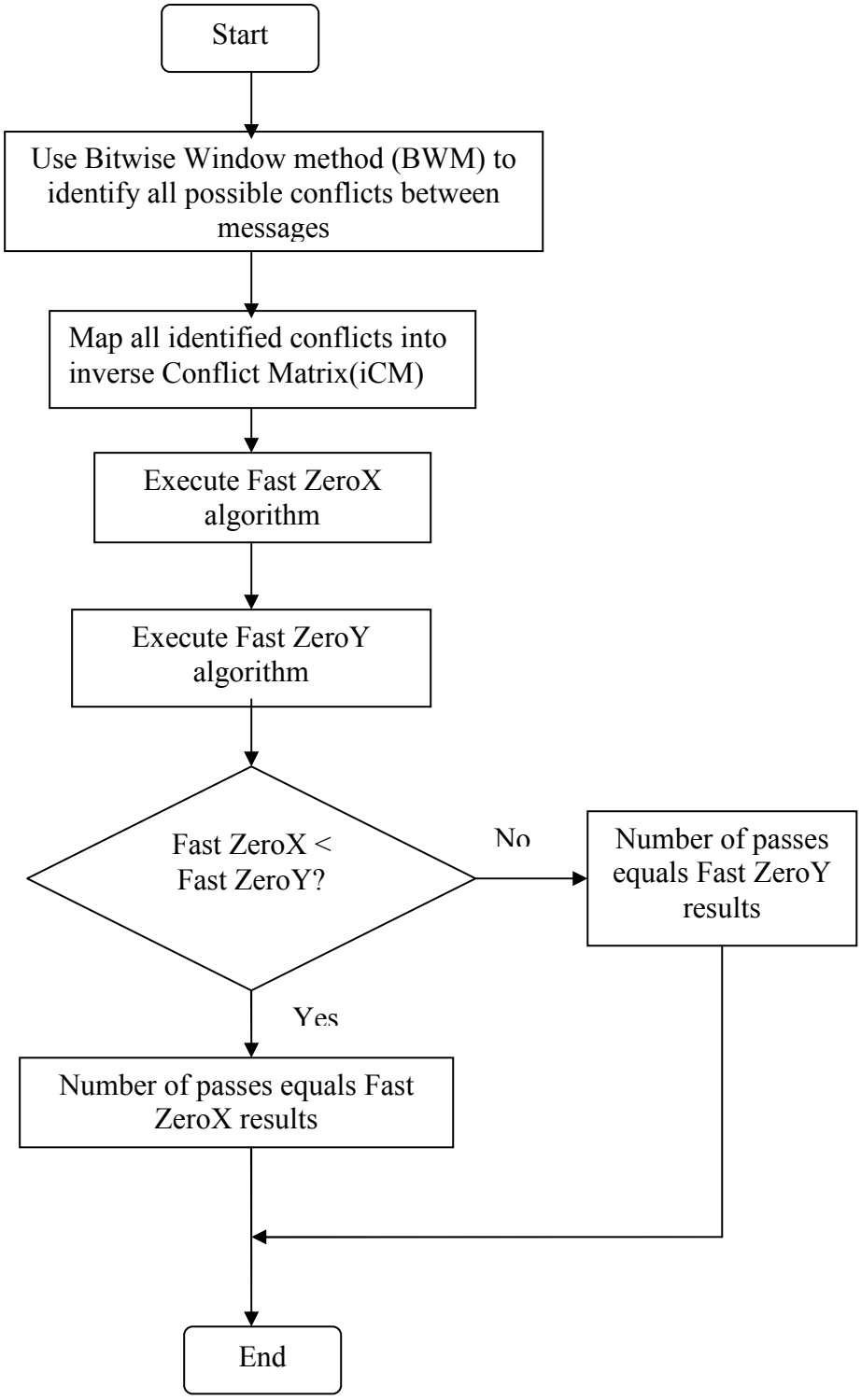



**Figure 5.12: The Fast Zero XY Algorithm Framework [20]**

## 5.11 Time Domain Approach for Avoiding Crosstalk in Optical Multistage Interconnection Network

Another way to solve the problem of crosstalk is the time domain approach. With the time domain approach, the same objective is achieved by treating crosstalk as a conflict; that is, two connections will be established at different times if they use the same SE [21,28]. Whereas we want to distribute the messages to be sent to the network into several groups, a method is used to find out which messages should not be in the same group because they will cause crosstalk in the network. A set of connections is partitioned into several subsets such that the connections in each subset can be established simultaneously in a network as shown in figure 5.13. There is no crosstalk in these subsections. This approach makes importance in optical MINs for various reasons:

1. Most of the multiprocessors use electronic processors and optical MINs. There is a big mismatch between the slow processing speed in processors and the high communication speed in networks carrying optical signals.
2. There is a mismatch between the routing control and the fast signal transmission speed.

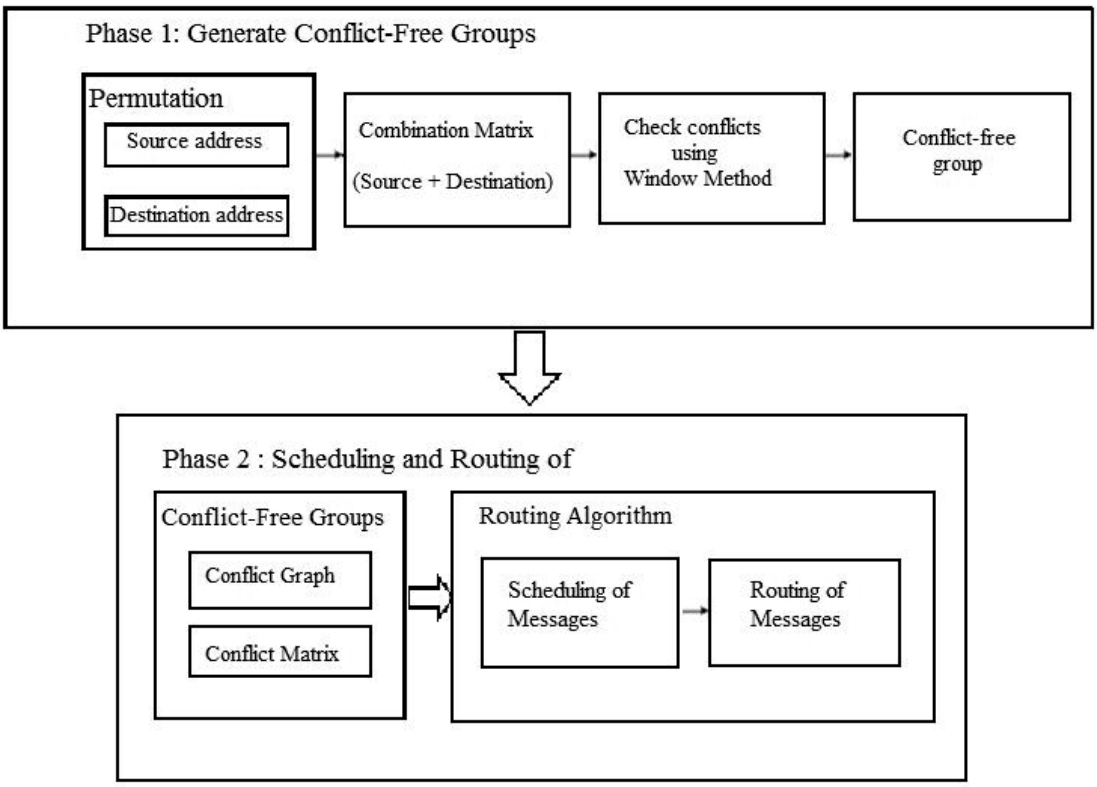



**Figure 5.13: Time Domain Approach Framework [21]**

## 5.12 Space Domain Approach

One way to solve the crosstalk problem is a space domain approach [21], where a MIN is duplicated and combined to avoid crosstalk. The number of switches required for the same connectivity in a network with space domain approach is slightly larger than twice that for the regular network. This approach uses more than double the original network hardware to achieve the same. Thus for the same permutation the hardware or we can say the no of switches will be double. Thus cost will be more with the networks using space domain approach. In all four cases only one input and only one output is active at a given time so that no cross talk occurs. With the space domain approach, extra switching elements (SEs) (and links) are used to ensure that at most one input and one output of every SE will be used at any given time. It is shown in figure 5.14.

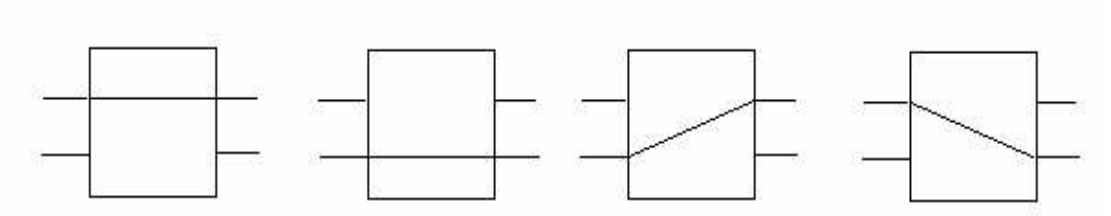


**Figure 5.14: Space Domain Approach [21]**

# Chapter -6 Heuristic Routing Algorithms

## 6.1 Introduction

The purpose of the routing algorithm is to schedule the messages in different time slots in order to avoid path conflict in the network. a routing algorithm has several phases, which are repeated to schedule all messages in the graph. In the first phase, has to be selected in order to be scheduled. In the second phase, the message is scheduled in a time slot.

**Four heuristic routing algorithms include [22]**

- Sequential Increasing
- Sequential Decreasing
- Degree Ascending
- Degree Descending

**Four strategies are used for selecting the message:**

1. Selecting a message sequentially in increasing order of source addresses.
2. Selecting a message sequentially in decreasing order of source addresses.
3. Selecting a message based on the order of increasing degrees in the conflict graph.
4. Selecting a message based on the order of decreasing degrees in the conflict graph.

After selecting the appropriate message, assigning a time slot for the message that can be scheduled by several strategies [22]

1. Schedule a free time slot for a message in increasing order. If the message has a conflict with message in each of the existing time slots, then a new time slot is allocated to the message, otherwise, the first available time slot is selected for the message.

2. Schedule a time slot for a message in the decreasing order. If the message has a conflict with message in each of the existing time slots, then a new time slot is allocated to the message.

3. Schedule a time slot for a message randomly among time slots already scheduled for other messages. If the message has conflict with at least one message in each of the existing time slots, then a new time slot is allocated to the message.

## 6.2 Message routing in heuristic algorithms

Each row consists of a pass and the last column indicates how many messages routed in each pass. In all heuristic algorithms the bottleneck is where the program has to find conflict in each pass with the new routed messages. To improve the heuristic algorithms a new routing matrix was defined. A bitwise matrix with maximum 10 rows and N columns is built. This matrix has only digital element (0 and 1).

## 6.3 Bitwise Four Heuristic Algorithms

In bitwise Four Heuristic (BFH) algorithms, the bitwise operations are used to route the messages [23]. The efficiency of these algorithms is much better than the standard four heuristic algorithms. The algorithm of all heuristic algorithms can be described in short as follows:

- Build a routing matrix by maximum 10 rows and columns (number of network size (10*N)).
- Put the messages in first pass; check the conflicts in the first pass.
- If there is conflict with the other messages in current pass, remove the message from that pass and try next passes until the messages fits into one suitable pass.

### 6.3.1 Bitwise Sequential Increasing and Decreasing Algorithm

- In BSeqInc algorithm the message is selected sequentially in the increasing order of the source address as same as SeqInc. Consequently, applying the bitwise operation in SeqInc reduces the execution time more than 10 times.
- BSeqDec algorithm selects the message sequentially in the decreasing order of the source address to route them in OMIN. Accordingly, applying the bitwise operation in SeqDec reduces the execution time because of bitwise operation essence in speed and simplicity [22,23].

### 6.3.2 Bitwise Degree Ascend and Degree Descending Algorithm

- In BDegAsc a message choose based on the order of the increasing degrees in the conflict graph. The degree of each message is the number of conflicts to other messages.
- BDegree Descending algorithm is a same as BDegree Ascending but this chooses a message based on the order of the decreasing degrees in the conflict graph.

## 6.4 Simulated Annealing

### 6.4.1 Introduction

Simulated Annealing originated in the annealing processes found in thermodynamics and metallurgy [24]. Simulated Annealing was introduced by Metropolis et al. and is used to approximate the solution of very large combinatorial optimization problems [Kirk] (e.g. NP-hard problems). It is based upon the analogy between the annealing of solids and solving optimization problems. When SA was first proposed, it was mostly known for its effectiveness in finding near optimal solutions for large-scale combinatorial optimization problems, such as the traveling salesperson problem, buffer allocation in production lines, and chip placement problems in circuits (finding the layout of a computer chip that minimizes the total area). But recent approaches of SA demonstrated that this class of optimization approaches could be considered competitive with other approaches for solving optimization problems. Simulated Annealing was derived from physical characteristics of spin glasses. The principle behind simulated annealing is analogous to what happens when metals are cooled at a controlled rate. The slowly falling temperature allows the atoms in the molten metal to line them up and form a regular crystalline structure that has high density and low energy as shown in figure 6.1. But if the temperature goes down too quickly, the atoms do not have time to orient themselves into a regular structure and the result is a more amorphous metal with higher energy. From figure 6.2 we can see that, heating the material to a very high temperature gives atoms the energy to move around. Then cool the material very slowly; which gently restricts the range of motion till everything freezes into a low energy configuration.

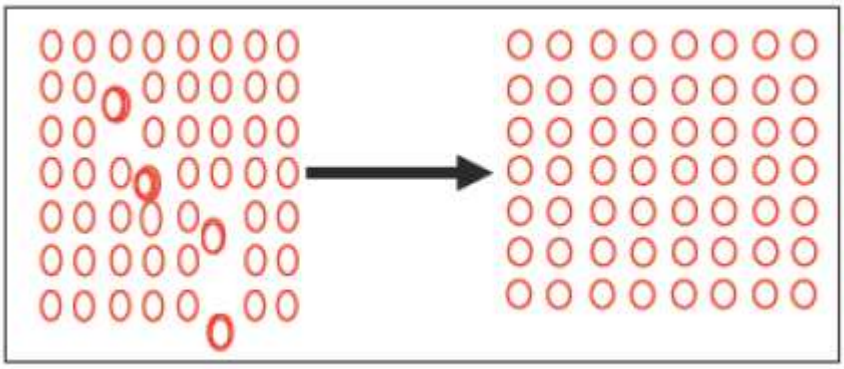


**Figure 6.1: Perfect = all atoms lined up on crystal lattice sites, no defects.**
**Perfect = this is the lowest energy "state" for this set of atoms [24]**

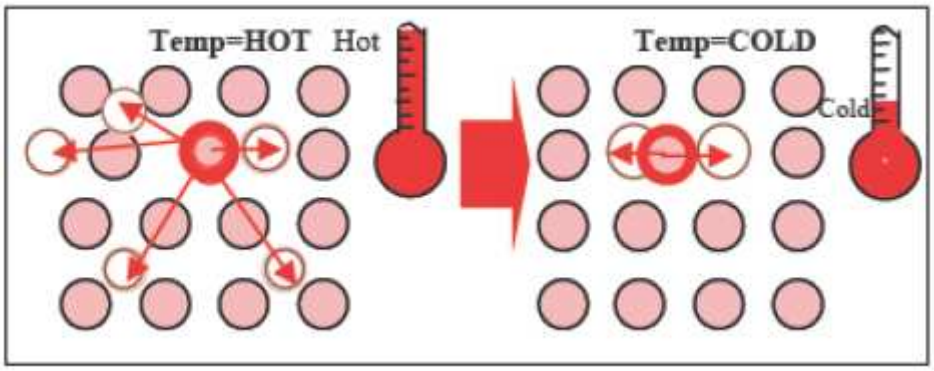



**Figure 6.2: How to reach the "low energy state": "anneal" the material [24]**

In simulated annealing, the value of an objective function that we want to minimize is analogous to the energy in a thermodynamic system. At high temperatures, SA allows function evaluations at faraway points and it is likely to accept a new point with higher energy. This corresponds to the situation in which high-mobility atoms are trying to orient themselves with other non-local atoms and the energy state can occasionally go up. At low temperatures, SA evaluates the objective function only at local points and the likelihood of it accepting a new point with higher energy is much lower. This is analogous to the situation in which the low-mobility atoms can only orient themselves with local atoms and the energy state is not likely to go up again.

Obviously, the most important of SA is the so-called annealing schedule or cooling schedule, which specifies how rapidly the temperature is lowered from high to low values. This is usually application specific and requires some experimentation by trial-and-error.

## 6.5 Fundamental Terminology

The following fundamental terminology about SA is useful before going to a detailed description of SA [25].

***Objective function:*** An objective function f(·) maps an input vector x into a scalar E : E = f (x), Where each x is viewed as a point in an input space. The task of SA is to sample the input space effectively to find an x that minimizes E.

***Generating function:*** A generating function g(·,·) specifies the probability density function of the difference between the current point and the next point to be visited. Specifically, Δx (= $x_{new}$ − x) is a random variable with probability density function g(Δx, T), where T is the temperature. For common SA used in combinatorial optimization applications g(·,·) is a function independent of temperature T. For this problem, move set approach to generate the next solution is used.

***Acceptance function:*** After a new point $x_{new}$ has been evaluated, SA decides whether to accept or reject it based on the value of the acceptance function h(·,·). The most commonly used acceptance function is the Boltzmann distribution function.

h (ΔE, T) = exp (-ΔE / T)

Where T is the temperature and ΔE is the energy difference between $x_{new}$ and x.

ΔE = f ($x_{new}$) – f (x)

The common practice is to accept $x_{new}$ with probability h(ΔE, T). Note that when ΔE is negative, SA tends to accept the new point because it reduces the energy. When ΔE is positive, SA may accept the new point and end up in a higher energy state or it may not accept the point. Lower the temperature; the less likely SA is to accept any significant high-energy states.

***Annealing schedule:*** An annealing schedule regulates how rapidly the temperature T goes from high to low values. The easiest way of setting an annealing schedule is to decrease the temperature T by a certain percentage at each iteration. To find the next legal point, usually a move set is defined, denoted by M(x), as a set of legal points available for exploration after x. Usually the move set M(x) represents a set of neighboring points of the current point x in the sense that the objective function at any point of the move set will not differ too much from the objective function at x.

## 6.6 Simulated Annealing algorithm

The basic Simulated Annealing algorithm includes the following steps [25,26]:

- Select a large starting value for the temperature T, a value for the final temperature $T_{final}$, a value for the stopping condition $t_s$ that is chosen as 10, suitable value for the temperature reducing parameter α. The $t_s$ value is used because it will stop the algorithm if there are no changes to the solution after $t_s$ iterations. Also choose a variable M so that M iterations are performed for each temperature.
- Initialize the current gain to be equal to the gain of the solution, which is used to proceed, and $t_s$ to some variable $t_{stop}$.
  while ($t_{stop} > 0$ and $T > T_{final}$) it proceed,
  for i = 1 to M do
  Generate a new solution using the present solution.
- Calculate the new gain .If the new gain is less than the current gain it is accepted otherwise use the following strategy to decide.
  R = random number (0<R<1)
  $Y = \exp(-\Delta E / T)$
  If (R < Y) then accept the solution,
  else reject it.
- If move is accepted then the current gain will be set to the new gain and keep the new solution, otherwise keep the old solution.
- If solution is accepted then assign $t_{stop} = ts$ else decrement $t_{stop}$ by 1 ($t_{stop} = t_{stop} –1$)
  Then change the temperature by a factor α, T = T * α.
  End.

## 6.7 Operators in Simulated Annealing

There are many type of operators in simulated annealing. Lets discuss briefly about the various type of operators, the initial solution and the parameters used in simulated annealing [25,26].

### 6.7.1 Move Sets in Simulated Annealing

Move sets are the most important operators in SA approach. We used three types of Move sets for SA in this research. They are Inversion, Translation and Switching.

***Inversion:*** There is a sorted vertex and inversion is applied to this sorted vertice to generate a new solution. For example if there is a sorted vertex as shown in Figure 6.3:

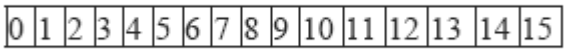



Figure 6.3: Sorted Vertex

Then generate two random points and then replace that section in the opposite order. For example if the random points as 4 and 11 are generated, then invert the numbers from 11 to 4 and store them in this particular order as shown in Figure 6.4.

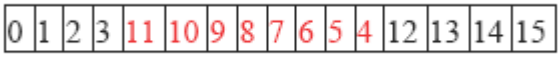



Figure 6.4: Inversion

***Translation:*** Randomly generate two points and then that section of the vertice is stored in between two randomly generated points. For example if 1 and 3 are generated for the section to be replaced, and then 14 and 15, the section 1-2-3 is placed in between 14 and 15. This is applied on the sorted vertex after Inversion is applied as shown in Figure 6.5.

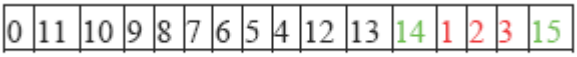



Figure 6.5: Translation

***Switching:*** Randomly generate two points and then switch the points at those positions. Suppose if 0 and 15 are generated, then 0 and 15 are interchanged in the above sorted vertices as shown in Figure 6.6.

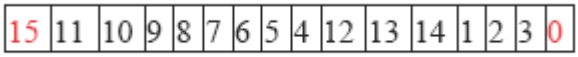



Figure 6.6: Switching

These three techniques are used and found out that Inversion is mostly suitable for application in Simulated Annealing to generate a new solution. This is because, translation needed more time when compared to inversion because of the random number generations and switching move set tends to rupture the solution.

## 6.8 Initial solution for the Simulated Annealing Algorithm

Initially a solution is required to proceed with the SA algorithm. This is because SA cannot proceed without having any arbitrary solution to generate the next solution. For this purpose the sorted vertices is formed with vertices taken in the increasing order. Then the number of passes needed with this sorted vertice is calculated. Then assign the current gain to the number of passes that are initially found and proceed to the SA algorithm.

## 6.9 Parameters of SA Algorithm

The parameters used in the algorithm are the starting temperature, the final temperature, the temperature-cooling rate (α), the number of iterations (M) for a particular temperature value and the stopping value $t_s$ [26]. The starting temperature is used for the algorithm to start at a particular temperature. The cooling rate (α) is chosen in between 0 and 1. This value regulates on how rapidly the temperature goes from high to low. The final temperature is chosen because after reaching this point the algorithm terminates and the solution is returned. It is required to generate solutions at a particular temperature and see whether they are accepted or not. For this purpose, iterate at a particular temperature depending on the M value chosen. The $t_s$ is chosen as 10 and is fixed. It is used to stop the algorithm when there have been no changes to the solution after $t_s$ iterations. In this way the algorithm terminates either on reaching the final temperature or after $t_s$ iterations. Random number generator is used to generate random numbers between 0 and 1. This is used for comparing the random number value to $\exp(-\Delta E / T)$.

## 6.10 Application of SA Algorithm to the problem [27]

1. First sorted vertices in ascending order are used as the initial solution to the problem. Then, calculate the number of passes using these sorted vertices and then assign the number of passes to the current gain.
2. Start at the initial temperature. Generate a new solution using one of the move set approaches.
3. Then calculate the number of passes needed with this sorted vertices and assign it to a variable called new gain.
4. If the new gain is less than or equal to the current gain then the solution is accepted.
5. If the new gain is more than the current gain then calculate exp (-ΔE / T) and then generate a random number.
6. If the random number generated is less than exp (-ΔE / T) then the solution is accepted, else the solution is rejected.
7. If the solution is accepted then the new gain is assigned to current gain and the new sorted vertices are used for generating the next solution.
8. If the solution is not accepted then the current gain does not change and the sorted vertices will remain the same. M iterations are performed for each temperature value and after these M iterations the value of the temperature is changed to $T = \alpha * T$. Then, repeat the process if T is greater than the final temperature and if $t_{stop}$ is greater than 0 ($t_{stop} > 0$). Finally the algorithm stops having met any of the two conditions. At this point the number of passes is returned which corresponds to the final solution obtained after applying the SA algorithm.

# Conclusion

The major problem introduced by optical multistage interconnection networks is crosstalk which is caused by the coupling of two signals within same switching element. The messages are partitioned into several groups. Various techniques are used to find conflicts between the messages. In this thesis, various methods have been compared and all are implemented in C. It has been observed that the improved window method take lesser time to find conflicts as compared to window method. The bitwise Window Method reduces the execution time ten times than the other algorithms even when the network size is large.

In this thesis, bitwise operations are used to improve the performance of routing in OMINs. The Bitwise heuristic techniques can improve the time nearly more than 10 times special when the network size is large. Efficient message routing algorithms directly affect the performance of communication networks.

# Future Scope

1. New algorithms can be developed to find conflicts in OMINs. Routing algorithms can be developed that will take very less execution time.

2. For future research, the Bitwise Window Method can be implemented on the distributed parallel computers.

3. The study can be extended to irregular OMINs.

# References


[1] Anujan Varma and C. S. Raghavendra, "Interconnection Networks for Multiprocessors and Multicomputers, Theory and Practice", IEEE Computer Press, 1994.

[2] J.Sengupta, "Interconnection networks for parallel processing", Nutech Books, Deep and Deep publications, New Delhi, (ISBN-81-7629-736-4), pp. 7-8, 15-19, 42-45, 69-70, 2005.

[3] C. P. Kruskal, M. Shir, "Performance of multistage interconnection networks for multiprocessors", IEEE transactions on computers, vol. C 32, December 1983.

[4] P. Dharma Aggarwal, "Testing and Fault Tolerance of Multistage Interconnection Networks", IEEE Transactions on Computers, pp. 41-53, 1982.

[5] P.K Bansal, Kuldeep Singh, R.C Joshi , "On Fault tolerant Multistage Interconnection Network", Conference on Computer Electrical Engineering, vol. 20, no. 4, pp. 335-345, 1994.

[6] N. Laxmi Bhuyan, Qing Yang, P. Dharma Aggarwal, "Performance of Multiprocessor Interconnection Networks", Proceeding of IEEE, pp. 25-37, 1989.

[7] N. Tzeng, P. Yew and C. Zhu, "A fault-tolerant for multistage interconnection networks", 12th International Symposium on Computer Architecture, pp. 368–375, 1985.

[8] A. Varma and C.S. Raghavendra, "Fault-tolerant routing in Multistage interconnection networks", IEEE Transactions on Computers 385–393, 1989.

[9] C. Wu and T. Y. Feng, "On a class of multistage interconnection networks", IEEE transactions on Computers pp. 1145–1155, 1980.

[10] K. Suresh Bhogavilli, Abu-Amara Hosame, "Design and Analysis of High Performance Multistage Interconnection Networks", IEEE Transactions on Computers, vol. 46, no. 1, pp. 110 -117,1997.

[11] Y. Pan, C. Qiao, and Y. Yang, "Optical Multistage Interconnection Networks: New Challenges and Approaches", IEEE Communications Magazine, Feature Topic on Optical Networks, Communication Systems and Devices, vol. 37, no. 2, pp. 50-56, 1999.

[12] Y. Yang, J. Wang, and Y. Pan, "Permutation Capability of Optical Multistage Interconnection Networks", Journal of Parallel and Distributed System,vol. 60, pp. 72-91, 2000.

[13] Siu-Cheung Chau and Tiehong Xiao, "A New Algorithm for Routing and Scheduling in Optical Multistage Interconnection Networks", Proceedings of the 4th IASTED International Multi-Conference on Wireless and Optical Communications, pp. 749-755, 2004.

[14] Pan, Y. Ji, C. Lin, X. Jia, X, "Evolutionary Approach for Message Scheduling in Optical Omega Networks", Fifth International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP), pp. 1-9, 2002.

[15] Mohammed Ali Al-Shabi, Mohamed Othman, Rozita Johari and Shamala Subramaniam, "New Algorithm to Avoid Crosstalk in Optical Multistage Interconnection Networks", Proceeding of IEEE International Conference on Network (MICC-ICON 2005), pp. 501-504, 2005.

[16] H. Miao, "A Java Visual Simulation Study of Four Routing Algorithms on Optical Multistage Omega Network", A Master project, Computer Science Department, University of Dayton, May 2000.

[17] Enyue Lu and S. Q. Zheng, "High-Speed Crosstalk-Free Routing for Optical Multistage Interconnection Networks", Proceedings of IEEE International Conference on Computer Communications and Networks (ICCCN 2003), pp. 249-254, 2003.

[18] T. D. Shahida, M. Othman and M. Khazani, "Fast ZeroX Algorithm for Efficient Message Routing in Optical Multistage Interconnection Networks", Proceedings of International Conference on Computer and Communication Engineering, pp. 275-279, 2008.

[19] T. D. Shahida, M. Othman and M. Khazani, "Fast ZeroY Algorithm for Efficient Message Routing in Optical Multistage Interconnection Networks", International Conference on Networking, vol. 4, pp. 1-6, August 2008.

[20] Siu-Cheung Chau, T. Xiao, and Ada Wai-Chee Fu, "Routing and Scheduling for a Novel Optical Multistage Interconnection Network", Euro-Par 2005 Parallel Processing, LNCS, Springer-Verlag Berlin Heidelberg, vol. 3648, pp. 984-993, 2005.

[21] C. Qiao, R. Melhem, D. Chiarulli and S. Levitan, "A time domain approach for avoiding crosstalk in optical blocking multistage interconnection networks," J. Lightwave Technology, vol. 12, no. 10, pp. 1 854-1862, 1994.

[22] T. D. Shahida, M. Othman, M. Khazani, "Routing Algorithms in Optical Multistage Interconnection Networks: Revisited", World Engineering Congress 2007, August 2007.

[23] F. Abed and M. Othman, "Bitwise-Based Routing Algorithms in Optical Multistage Interconnection Networks", Master Thesis, University Putra Malaysia, 2007.

[24] A. K. Katangur, Y. Pan, and M. D. Fraser, "Message Routing and Scheduling in Optical Multistage Networks Using Simulated Annealing", Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS 2002), pp. 201-208, 2002.

[25] V. Cerny, "Thermodynamically Approach to the Traveling Salesman Problem: An Efficient Simulation Algorithm". Journal of Optimization Theory and Applications, vol. 45, no. 1, pp. 41-45, 1985.

[26] C. Koulamas, S. R. Antony, and R. Jaen, "A survey of simulated annealing applications to operations research problems". Omega International Journal of Management Science, vol. 22, no. 1, pp. 41-56, 1994.

[27] Theodore W. Manikas, James T.Cain, "Genetic Algorithms vs. Simulated Annealing: A Comparison of Approaches for Solving the Circuit Partitioning Problem", Technical report, pp. 96-101, May 1996.

[28] A. Munir, O. Mohamed and J. Rozita "An efficient approach to avoid crosstalk in optical Omega Network". International Journal of Computer, Internet and Management, vol. 14, no. 1, pp. 50-60, 2005.

[29] N. Das, B. B. Bhattacharya, R. Menon, A. Sarkar, "Permutation Routing in Optical MINs with Minimum Number of Stages," Journal of Systems Architecture, vol. 48, pp. 311-323, 2003.

[30] P. Dharma Aggarwal, "Testing and Fault Tolerance of Multistage Interconnection Networks", IEEE Transactions on Computers, pp. 41-53, 1982.

[31] P. K. Bansal, Kuldeep Singh, R. C. Joshi, "On Fault tolerant Multistage Interconnection Network", Conference on Computer Electrical Engineering, vol. 20, no.4, pp. 335-345, 1994.

[32] T. Blaket James, S. Kishore Trivedi , “Reliabilities of Two Fault-Tolerant Interconnection Networks”, Proceeding of IEEE, pp. 300-305, 1988.

[33] B. George Adams, P. Dharma Agrawal and Howard Jay Siegel, “A Survey and Comparison of Fault-Tolerant Interconnection Networks”, IEEE Transactions on Computers, pp. 14-27, 1987.

# Papers Published

1) Shruti Chopra and Rinkle Aggarwal, "Analysis of various Heuristic algorithms in Optical Multistage Interconnection Networks" National conference on Emerging trends in Software and Networking Technologies (ETSNT'09), Amity University, Noida during 17-18 April 2009.

2) Shruti Chopra and Rinkle Aggarwal, "Comparative Analysis of various Crosstalk Avoidance Techniques in Optical Multistage Interconnection Networks" National Conference on Recent Advances and Future Trends In Information Technology (RAFIT 2009), Punjabi University, Patiala during 9-10 April 2009.

   Paper has been selected for publication of the Italy based **International Journal "Atti della Fondazione Giorgio Ronchi".**

## 8.1 Pseudo-code for window method of conflicts

**Global Parameters Passed:** Nothing

**Function name:** main ()

**Called by:** operating system

**Calling:** Convert ()

**Parameters passed:** no

**Purpose:** This function does all the necessary initializations and check whether there is conflict in data or not.

**Function Body:**

1. Initialize the conflict matrix with value 0.If there is conflict between data sent then corresponding value is changed to 1 in conflict matrix.
2. Input or initialize the number of source-destination pairs from the user.
3. Calculate the no of windows and window size.
4. Make the window to see where the conflict in the data is.
5. Access the window to see where the conflict in the sent data is.
6. If any conflict occurs change that particular value in conflict matrix to 1.
7. Go to step 4 if any more window is there else go to step 8.
8. Use convert () function to show where the conflict is.
9. Finally, display the source-destination and conflict matrix as output. Output is displayed by using graphics also.

**Function name:** convert ()

**Called by:** main ()

**Calling:** nothing

**Parameters passed:** one integer variable and one string.

**Purpose:** This function only converts integer number into binary number to show the conflict.

**Function Body:**

1. This function receives the integer variable and one string.
2. It checks it in switch-case.
3. Then copy the particular code in passed string.
4. Finally, return to main.

## 8.2 Pseudo-code for Improved window method

**Global Parameters Passed:** Nothing

**Function name:** main ()

**Called by:** operating system

**Calling:** Convert ()

**Parameters passed:** no

**Purpose:** This function does all the necessary initializations and check whether there is conflict in data or not.

**Function Body:**

1. Initialize the conflict matrix with value 0.If there is conflict between data sent then corresponding value is changed to 1 in conflict matrix.
2. Input or initialize the number of source-destination pairs from the user.
3. Calculate the no of windows and window size. Decrement window size 1 as there is one window less as was in window method.
4. Make the window to see where the conflict is there in data.
5. Access the window to see where the conflict is in sent data.
6. If any conflict occurs change that particular value in conflict matrix to 1.
7. Go to step 4 if any more window is there else go to step 8.

8. Use convert () function to show where the conflict is.
9. Finally, display the source-destination and conflict matrix as output. Output is displayed by using graphics also.

**Function name**: convert ()

**Called by:** main ()

**Calling:** nothing

**Parameters passed:** one integer variable and one string.

**Purpose:** This function only converts integer number into binary number to show the conflict.

**Function Body:**

1. This function receives the integer variable and one string.
2. It checks it in switch-case.
3. Then copy the particular code in passed string.
4. Finally, return to main.

## 8.3 Pseudo-code for Bitwise Window method

**Global Parameters Passed:** Nothing

**Function name:** main ()

**Called by:** operating system

**Calling:** nothing

**Parameters passed:** no

**Purpose:** This function does all the necessary initializations and check whether there is conflict in data or not.

**Function Body:**

1. Initialize the conflict matrix with value 0.If there is conflict between data sent then corresponding value is changed to 1 in conflict matrix.
2. Input or initialize the number of source-destination pairs from the user.
3. Calculate the no of windows and window size. Decrement window size 1 as there is one window less as was in window method.
4. Here the window size will be smallest i.e. window size is of one column size.
5. Access the window to see where the conflict in the sent data is.
6. If any conflict occurs change that particular value in conflict matrix to 1.
7. Go to step 4 if any more window is there else go to step 8.
8. Finally, display the source-destination and conflict matrix as output. Output is displayed by using graphics also.

## 8.4 Pseudo-code for all Window methods

**Global Parameters Passed:** Include files bit1.c, wm1.c and newwm.c

**Function name:** main ()

**Called by:** operating system

**Calling:**

1. Window1()
2. Newwindow()
3. Bit()

**Parameters passed:** no

**Purpose:** This function does all the necessary initializations.

**Function Body:**

1. Get the choice of method you want to execute for conflicts.
2. According to the choice call the appropriate function either window () or new window () or bit () or all three.

**Function name:** window ()

**Called by:** main ()

**Calling:** Convert ()

**Parameters passed:** no

**Purpose:** This function does all the necessary initializations and check whether there is conflict in data or not.

**Function Body:**

1. Initialize the conflict matrix with value 0.If there is conflict between data sent then corresponding value is changed to 1 in conflict matrix.
2. Input or initialize the number of source-destination pairs from the user.
3. Calculate the no of windows and window size.
4. Make the window to see where the conflict in the data is.
5. Access the window to see where the conflict in the sent data is.
6. If any conflict occurs change that particular value in conflict matrix to 1.
7. Go to step 4 if any more window is there else go to step 8.
8. Use convert () function to show where the conflict is.
9. Finally, display the source-destination and conflict matrix as output. Output is displayed by using graphics also. Return to main.

**Function name**: convert ()

**Called by:** window () or newwindow ()

**Calling:** nothing

**Parameters passed:** one integer variable and one string.

**Purpose:** This function only converts integer number into binary number to show the conflict.

**Function Body:**

1. This function receives the integer variable and one string.
2. It checks it in switch-case.
3. Then copy the particular code in passed string.
4. Finally, return to calling function.

**Function name:** newwindow ()

**Called by:** main ()

**Calling:** Convert ()

**Parameters passed:** no

**Purpose:** This function does all the necessary initializations and check whether there is conflict in data or not.

**Function Body:**

1. Initialize the conflict matrix with value 0.If there is conflict between data sent then corresponding value is changed to 1 in conflict matrix.
2. Input or initialize the number of source-destination pairs from the user.
3. Calculate the no of windows and window size. Decrement window size 1 as there is one window less as was in window method.
4. Make the window to see where the conflict in the data is.
5. Access the window to see where the conflict in the sent data is.
6. If any conflict occurs change that particular value in conflict matrix to 1.
7. Go to step 4 if any more window is there else go to step 8.
8. Use convert () function to show where the conflict is.
9. Finally, display the source. It is displayed by using graphics also. Destination and conflict matrix are outputs. Output is also displayed by using graphics. Return to main.

**Function name**: bit ()

**Called by:** main ()

**Calling:** Nothing

**Parameters passed:** no

**Purpose:** This function does all the necessary initializations and check whether there is conflict in data or not.

**Function Body:**

1. Initialize the conflict matrix with value 0.If there is conflict between data sent then corresponding value is changed to 1 in conflict matrix.
2. Input or initialize the number of source-destination pairs from the user. Then calculate the number of windows and window size. Decrement window size 1 as there is one window less as was in window method.
3. Here the window size will be smallest i.e. window size is of one column size.
4. Access the window to see where the conflict in the sent data is.
5. If any conflict occurs change that particular value in conflict matrix to 1.
6. Go to step 4 if any more window is there else go to step 8.
7. Finally, display the source-destination and conflict matrix as output. Output is displayed by using graphics also. Return to main