# GUI Based Simulation of Interconnection Networks

Project Report submitted in partial fulfillment of the requirement for the degree of

Bachelor of Technology

in

**Computer Science & Engineering**

under the Supervision of

***Dr. Nitin Chanderwal***

By

***Simran Setia***
***111204***

to

# Certificate

This is to certify that project report entitled “GUI Based Simulation of Interconnection Networks”, submitted by Simran Setia(111204) in partial fulfillment for the award of degree of Bachelor of Technology in Computer Science & Engineering to Jaypee University of Information Technology, Waknaghat, Solan has been carried out under my supervision.

This work has not been submitted partially or fully to any other University or Institute for the award of this or any other degree or diploma.

**Date:**

**Supervisor’s Name: Dr. Nitin**

**Designation: Associate Professor**

# Acknowledgement

I am highly indebted to Jaypee University of Information Technology for their guidance and constant supervision as well as for providing necessary information regarding the project & also for their support in completing the project.

I would like to express my gratitude towards my parents & Project Guide for their kind co-operation and encouragement which help me in completion of this project.

I would like to express my special gratitude and thanks to industry persons for giving me such attention and time.

My thanks and appreciations also go to my colleague in developing the project and people who have willingly helped me out with their abilities

Date: Name of the student: Simran Setia

# Table of Content


| S. No. | Topic | Page No. |
|---|---|---|
| 1. | Abstract | 5 |
| 2. | Motivation | 6 |
| 3. | Introduction | 7 |
| 4. | Problem Definition | 9 |
| | 4.1 Topology | 10 |
| | 4.2 Routing Algorithm | 12 |
| | 4.3Flow control mechanism | 20 |
| 5. | Implementation | 21 |
| | 5.1 Auto-Align method | 22 |
| | 5.2 Draw Method | 24 |
| 6. | Tools and Technologies | 26 |
| | 6.1 Java Swings | 30 |
| | 6.2 Net Beans IDE | 32 |
| 7. | Screen Shots | 35 |
| 8. | Code | 40 |
| 9. | References | 56 |

# List of Figures


| S.No. | Title | Page No. |
|---|---|---|
| 1. | Interconnection network | 8 |
| 2. | Ring interconnection | 9 |
| 3. | Mesh interconnection | 10 |
| 4. | Hypercube interconnection | 12 |
| 5. | Omega interconnection | 14 |
| 6. | Various topologies | 20 |
| 7. | XY routing | 26 |
| 8. | Stop N wait | 31 |
| 9. | Go back N | 33 |
| 10. | Selective Repeat ARQ | 35 |

# 1. Abstract

Multi-stage Interconnection Networks (MINs) are widely used for broadband switching technology and for multiprocessor systems. Besides this, they offer an enthusiastic way of implementing switches used in data communication networks. With the performance requirement of the switches exceeding several terabits/sec and teraflops/sec, it becomes imperative to make them dynamic and fault-tolerant. A number of techniques have been used to increase the reliability and fault-tolerance of the MINs. MINs are typically used in high-performance or parallel computing as a low-latency interconnection (as opposed to traditional packet switching networks), though they could be implemented on top of a packet switching network. Though the network is typically used for routing purposes, it could also be used as a co-processor to the actual processors for such uses as sorting; cyclic shifting, as in a perfect shuffle network; and bitonic sorting

The typical modern day application of the MINs includes fault-tolerant packet switches, designing multicast, broadcast router fabrics while system on-chip and networks on-chip are hottest now days. Normally the following aspects are always considered while deigning the fault-tolerant MINs: the topology chosen, the routing algorithm used, and the flow control mechanism adhered. The topology helps in selecting the characteristics of the present chip technology in order to get the higher bandwidth, throughput, processing power, processor utilization, and probability of acceptance from the MIN based applications, at an optimum hardware cost. Soon, as the topology is freeze, the analytical bounds, which helps for measuring reliability and availability can be examined. The topology helps in determining the throughput and latency of the MINs whereas the routing algorithm and flow control encourage in achieving the performance bounds. Whenever we want to design a interconnection network, we used to design them manually using the windows word. At present, we do not have any tool through which we can develop the interconnection networks tool or this remains out of limelight therefore in this paper; we have discussed a tool designed for developing fault-tolerant multi-stage interconnection networks. The designed tool is one of its own kind and will help the user in developing 2 and 3-disjoint path networks.

# 2. Motivation

There are several reasons why computer architects should devote attention to interconnection networks. In addition to providing external connectivity, networks are commonly used to interconnect the components within a single computer at many levels, including the processor microarchitecture. Networks have Long been used in mainframes, but today such designs can be found in personal Computers as well, given the high demand on communication bandwidth needed to enable increased computing power and storage capacity. Solutions in order to more effectively design and evaluate computer systems.

Interconnection networks cover a wide range of application domains, very much like memory hierarchy covers a wide range of speeds and sizes. Networks implemented within processor chips and systems tend to share characteristics much in common with processors and memory, relying more on high-speed hardware solutions and less on a flexible software stack. Networks implemented across systems tend to share much in common with storage and I/O, relying more on the operating system and software protocols than high-speed hardware—though we are seeing a convergence these days network-on-chip (NoC), this type of network is used for interconnecting microarchitecture functional units, register files, caches, compute tiles, processor and IP cores within chips or multichip modules. Currently, OCNs support the connection of up to only a few tens of such devices with a maximum interconnection distance on the order of centimeters. Most OCNs used in high-performance chips are custom designed to mitigate chip-crossing wire delay problems caused by increased technology scaling and transistor integration, though some proprietary designs are gaining wider use.

Multistage interconnection networks are widely used for high speed computer networks , basically used to design system on-chip and network on-chip. They are composed of processing elements at one end and memory elements at the other end , connected by switching elements. These types of networks are typically used for routing purposes. With the performance requirement being several terabytes, MINs are required to be fault tolerant. This simulator will help in designing fault tolerant MINs.

# 3. Introduction

**Multistage interconnection networks** (MINs) are a class of high-speed computer networks usually composed of processing elements (PEs) on one end of the network and memory elements (MEs) on the other end, connected by switching elements (SEs). The switching elements themselves are usually connected to each other in stages, hence the name.

Such networks include networks, omega networks, delta networks and many other types. MINs are typically used in high-performance or parallel computing as a low-latency interconnection (as opposed to traditional packet switching networks), though they could be implemented on top of a packet switching network. Though the network is typically used for routing purposes, it could also be used as a co-processor to the actual processors for such uses as sorting; cyclic shifting, as in a perfect shuffle network; and bitonic sorting.

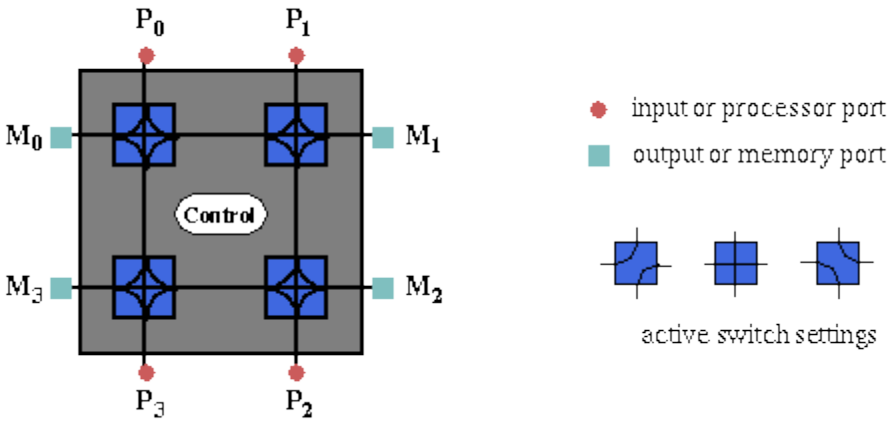



Fig. 1

## 3.1 Interconnection topologies

An idealized interconnection structure takes a set of n input ports labeled 0, … , n–1 and sets up connections between them and a set of m output ports 0, … , m–1 with the connections determined by control signals.

### 3.1.1 Ring

### 1. Ring.

Processor i directly connected to processors i+1(mod N) and i–1 (mod N). Data can be moved from any processor to any other by a sequence of cyclic shifts. Many parallel algorithms include calculations of the form X[i] := X[i–1] + X[i].Usually every item of an array except the first and last is updated in this way. The data routing can be defined by routing functions:

C+1(i ) ≡ (i+ 1) mod N

C–1(i ) ≡ (i – 1) mod N

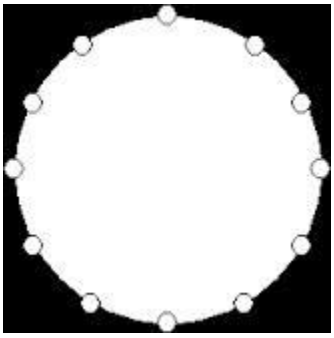


Fig 2.

### 3.1.2 Mesh

A mesh is similar to having "row & column" cyclic shifts. Four-point iteration is common in the solution of partial differential equations. Calculations of the form

X [i, j ] := (X [i +1, j ] + X [i–1, j ] + X [i , j–1] + X [i , j +1]) ÷ 4)

are performed frequently. In a 64-element mesh, any node can be reached from any other in no more than 7 of these shifts. Without the end-around connections (a "pure" 2D mesh), the diameter is 2( N –1).It is also possible to have a multidimensional mesh. The diameter of a d-dimensional mesh is d(N 1/d )–1 and its bisection width is N(d–1)/d. The average distance is d × 2(N 1/d )/3 (without end-around connections).

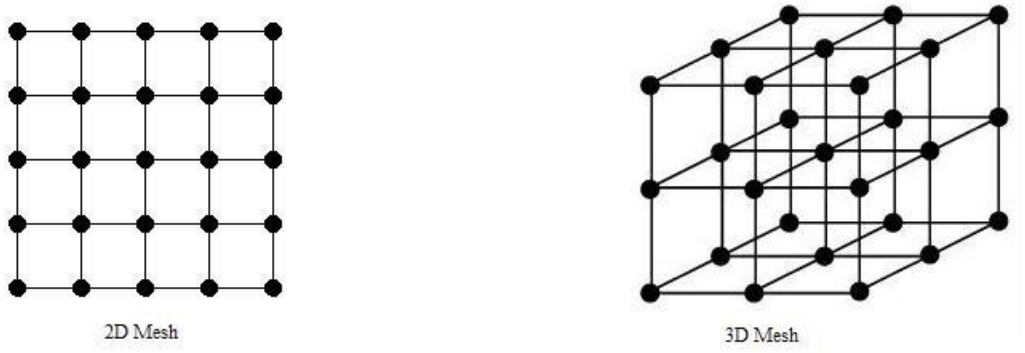



Fig 3.

### 3.1.3 Hypercube

For a NxN network, we have m = log2N stages each stage has N/2 two-input/two-output interchange boxes. The connection pattern among the boxes is different for the different multistage interconnection networks (MINs). A hypercube is a generalized cube.

In a hypercube, there are 2n nodes, for some n. Each node is connected to all other nodes whose numbers differ from it in only one bit position.

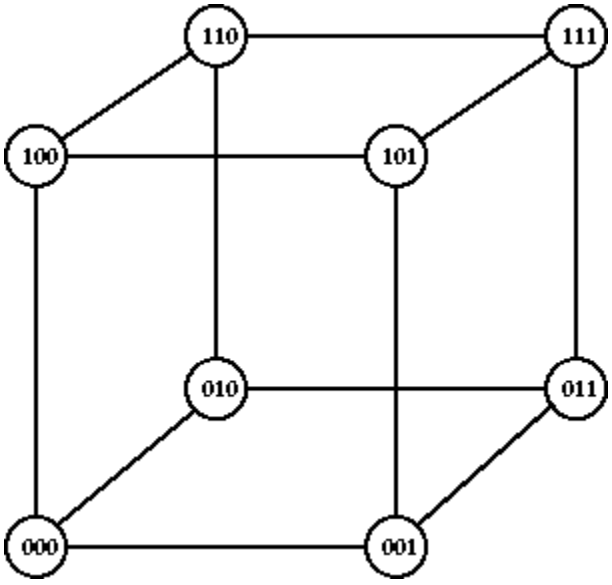



Fig 4.

### 3.1.4 Omega

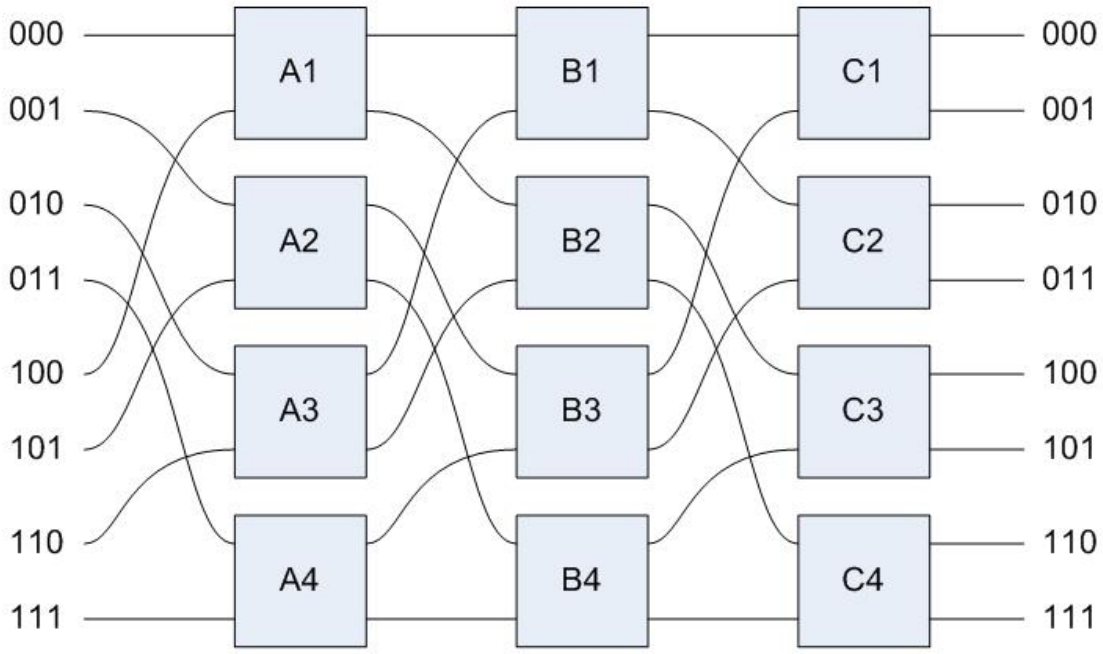



Fig. 5

An 8x8 Omega network is a multistage interconnection network. Processing elements (PEs) are connected using multiple stages of switches. The outputs from each stage are connected to the inputs of the next stage. For N processing element, an Omega network contains N/2

switches at each stage, and $\log_2 N$ stages. Switches used in most of MINs are Crossbar switches. Crossbar switch is a switch connecting multiple inputs to multiple outputs in a matrix manner.

# 4. Problem Definition

The following aspects are always considered while designing the fault-tolerant MINs:

- the topology chosen
- the routing algorithm used
- the flow control mechanism

First of all , all routers are connected in a fashion which gives rise to a particular topology . Based on that topology and xy routing algorithm a fault tolerant interconnection network will be designed which will ensure load balancing and reliable delivery of packets.

## 4.1 Topology

**Network topology** is the arrangement of the various elements (links, nodes, etc.) of a computer network Essentially, it is the topological structure of a network and may be depicted physically or logically. *Physical topology* is the placement of the various components of a network, including device location and cable installation, while *logical topology* illustrates how data flows within a network, regardless of its physical design. Distances between nodes, physical interconnections, transmission rates, or signal types may differ between two networks, yet their topologies may be identical.

There are two basic categories of network topologies: physical topologies and logical topologies.

The cabling layout used to link devices is the physical topology of the network. This refers to the layout of cabling, the locations of nodes, and the interconnections between the nodes and the cabling. The physical topology of a network is determined by the capabilities of the network access devices and media, the level of control or fault tolerance desired, and the cost associated with cabling or telecommunications circuits.

The logical topology in contrast, is the way that the signals act on the network media, or the way that the data passes through the network from one device to the next without regard to the physical interconnection of the devices. A network's logical topology is not necessarily the same as its physical topology. For example, the original twisted pair Ethernet using repeater hubs was a logical bus topology with a physical star topology layout. Token Ring is a logical ring topology, but is wired a physical star from the Media Access Unit.

The logical classification of network topologies generally follows the same classifications as those in the physical classifications of network topologies but describes the path that the *data* takes between nodes being used as opposed to the actual *physical* connections between nodes. The logical topologies are generally determined by network protocols as opposed to being determined by the physical layout of cables, wires, and network devices or by the flow of the electrical signals, although in many cases the paths that the electrical signals take between nodes may closely match the logical flow of data, hence the convention of using the terms *logical topology* and *signal topology* interchangeably.

Logical topologies are often closely associated with Media Access Control methods and protocols. Logical topologies are able to be dynamically reconfigured by special types of equipment such as routers and switches.

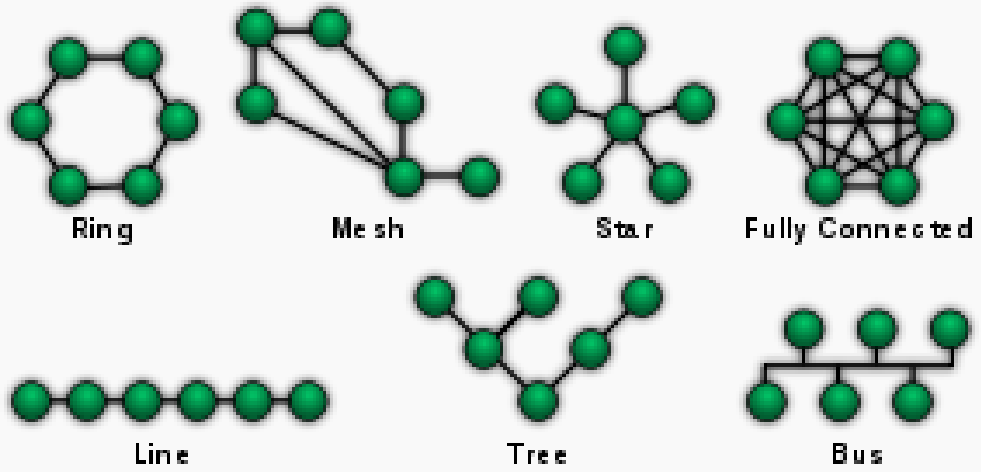



Fig 6. Diagram of different network topologies.

### 4.1.2 Routing algorithm

**Routing** is the process of selecting best paths in a network. In the past, the term routing was also used to mean forwarding network traffic among networks. However this latter function is much better described as simply forwarding. Routing is performed for many kinds of networks, including the telephone network (circuit switching), electronic data networks (such as the Internet), and transportation networks. This article is concerned primarily with routing in electronic data networks using packet switching technology.

### XY Routing Algorithm

The most popular routing algorithm that is XY routing is proposed by Wang Zhang and Ligang Hou in which packet have header flit, tail flit and data flit where the header flit having the destination node address ,For implementation2-Dimesion mesh topology and wormhole

switching mechanism are used .Each router having co-ordinates as (x,y),for the routing current address (Cx,Cy) is compare with the destination router address (Dx,Dy) of the packet ,depends up on the comparison output of routing algorithm route routes the packets. if (Dx>Cx ) head flit moves to East else it take West turn upto (Dx Cx) become equal this passion is called as horizontal alignment .Now (Dy,Cy) undergoes compression , if it is found that (Dy < Cy) then packet's header flit moves towered South else North upto (Dy=Cy.For simulation an environment maintained as the packet size is 8 bytes with a random destination mode , the percentage load is 50% which mean that 50% of maximum bandwidth is used, the interval between successive flits is 2 clock cycles, the simulation runs 1000 clock cycles and the clock frequency is 1 GHz, Synthetic traffic generators generate traffic in the first 300 clock cycles with warm-up period of 5 clock cycles. After simulation authors define an average performance parameter P to evaluate the average performance of the algorithm that is P where P= Average Throughput of Network/ Average Latency per packet of Network which is 0.86 for the XY algorithm. So they come to conclusion that the implement of XY routing algorithm is simple as well as the X-direction channel latency is averagely lager than Y-direction channel latency and Xdirection channel throughput is averagely all square with Y direction throughput.

## IX/Y

For less collision Ahmad M. Shafiee, Mehrdad Montazeri, and Mahdi Nikdast developed new routing algorithm called as Intermittent Routing Algorithm in which both XY and YX routing algorithm are used and name it as IX/Y routing algorithm. For study the performance of this routing algorithm,Ahmad Shafiee et al.used 4 X 4 mesh topology with NOXIM Simulator. In IX/Y Boolean variable 0 or 1 is set circular manner that is for first packet 0 is set and XY routing algorithm takes place, for the second packet Boolean variable 1 is set and YX routing algorithm is used. For the next packet again 0 is set and XY routing take place and same processer follow for next packets. this will give the help to have fewer collisions. While comparing the result , delay with respect to injection rate with other routing algorithm like XY, Dyad T Negative first, North last, west first and odd even algorithm Author found that intermittent algorithm is roughly remained stable below 30 cycles, and also notice that this statistics is approximately the same in a 10*10 mesh-based and is an enduring algorithm in higher injection rates.

## DyXY

A novel routing algorithm, namely dynamic XY (DyXY) routing algorithm which provides adaptive routing based on congestion condition in the proximity, and ensures deadlock free and livelock-free routing at same time was proposed by Ming Li, Qing-An Zeng, Wen-Ben Jone .Using DyXY routing algorithm packet only route a shortest path between source and destination .If multiple shortest paths are available then router choose the path for the packet depends upon the congestion condition of the network. The detailed routing algorithm is summarized as follows:

1 Read the destination of an incoming packet.

2 Compare addresses of the destination and the current router.

2.1 If
the destination is the local core of the current router,
send the packet to the local core;

2.2 Else

2.2.1 If
the destination has the same x (or y-axis) address as
the current router, send the packet to the neighbouring
router on the y-axis (or x-axis) towards the
destination;

2.2.2 Else
check the stress values of current router's neighbours
towards the destination , and send the packet to the
neighbour with smallest stress value.

Where the stress value is a parameter representing the congestion condition of a router, that is the number of occupied cells in all input buffers and each stress value is updated based on an event-driven mechanism. As With static XY routing, the length of a path traveled by packets for a give pair of source and destination is a constant, which equals to shortest path length. For DyXY routing, although the routing path is not static, it is always a shortest path and hence the length is still the shortest path length. Therefore, the average packet path length is only affected by the communication pattern. For the performance calculation author used an event-driven simulator using C++. By injecting more than 140,000 packets into the network in each simulation, and the NOC was warmed up for 20,000 packets before measuring latencies. After that they compared results of XY and DyXY and they found that

3.3.1 By size varied from 3X3 to 9X9 with average packet

injection rate increasing from 0.1 to 0.3

3.3.2 The DyXY routing algorithm achieves better balance in load distribution

3.3.3 The average mean response time for all routers can be effectively estimated using the analytical lower bound and upper bound

## XYX

Ahmad Patooghy and Seyed Ghassem Miremadi proposed deadlock free and Fault-Tolerant routing algorithm for the higher traffic load using XY and YX routing algorithm. The basic idea behind the XYX routing algorithm is that the source node adds some detection code to each packet and the destination node checks the contains of received packets, If error is found then the source node request to resend corrupted packet to destination using NACK signal. Actually At the source node original packet is converted into two types of packet as original and redundant packet this is possible using indicator bit set as “0”and “1” respectively in the header flit of packet. Using the indicator bit router comes to know that current packet is either the original or the redundant .In XYX routing algorithm the original packets follows XY routing algorithm and redundant packets YX routing algorithm which is inversed version of XY routing algorithm. At the destination node when a new original packet is received the parity bit will be regenerated and checked with those detection code embedded in the header of the packet. If two parity sets are equal, the packet will be accepted at the destination node otherwise it will be dropped detection code waits to receive the redundant copy of the packet. For experimental analysis Author create simulation platform as, 6×6 mesh topology a minimum of 20000 packets have been delivered, a fixed length of 32 flits per packet , mean rates of spanned from 0.001 to 0.02 packet per cycle per node, and in the reliability evaluation experiments, a wide range of error rates have been also injected to the network to precisely investigate the reliability improvement of the proposed routing algorithm. In this experiment the network has been simulated with error injection rate approximately 0.5, 1, 2, 5, 10, 20 and 50 percent of all flits experience a bit flip error. In result it found that at lower traffic rate XYX and other algorithms like S2S have correctly delivered more than 90% of the packets and their behavior are almost the same, In high error rate condition, the XYX routing could tolerate more than 60% of errors while it imposes lower performance and power consumption overheads and the XYX routing has the best performance.

## Adaptive XY

The Adaptive XY routing algorithm is a derivative of classic XY routing algorithm. In proposed routing algorithm author used a context aware agent which route packets based on the entire network status and other members of society. This algorithm could operate as a deterministic or adaptive routing depends on neighbor agents'(switches) load condition. When congestion becomes high, agent tries to route packets through less congested path. The agents find less congested paths using 2-bits quantized load value so each agent has four quantized value that are sent to four North, South, East, and
West neighbors'. Configuration of load values for agents is operated using configuration packets. Configuration packets are generated according to information received from all neighbors then all neighbors' collaborate each other to route packets. Emergence of a new route containing two parts: Firstly packets are routed by XY routing, when the destination port determined by XY routing is congested (a threshold is defined for each port of switch which is a criterion in order to realize a port is congested or not. This threshold involves the position of a switch for example switches are located in the center of mesh have threshold0 and the others have threshold1)author do not route packet to the destination where declared by XY routing rather author[19] compare quantized load value of neighbors and select the path that its quantized load value is minimum. This algorithm causes user don't have to wait until the crowded route release. Author also perform study the other routs which do not have any heavy load. Simulation perform on 3X3 mesh topology using SystemC language in which environment keeping base class as SystemC . For results use two mode of load generation random test and test using multimedia applications. When sent packets in a period are changed from 500 to 25000, the improvement of latency is changed from %11 to %25 and the improvement of max latency is changed from 0% to 33%.

A different type of XY routing algorithm is used in different network condition. The selection of XY routing algorithm it totally depends upon the application and the traffic of packet in the network. As simplicity in implementation is important in all architecture so an XY routing algorithm widely used .For fewer collisions Intermittent Routing Algorithm is Preferred. The DyXY routing algorithm achieves better balance in load distribution as well as provide deadlock-free and livelock-free facility. Were we can't compromised with accuracy of received data , we go for the fault-tolerant routing like XYX .If application is focused on network resources utilization the Adaptive XY routing algorithm is best choice.

Finally we can say that choice of XY routing algorithm is totally depends upon environmental condition of NOC architecture.

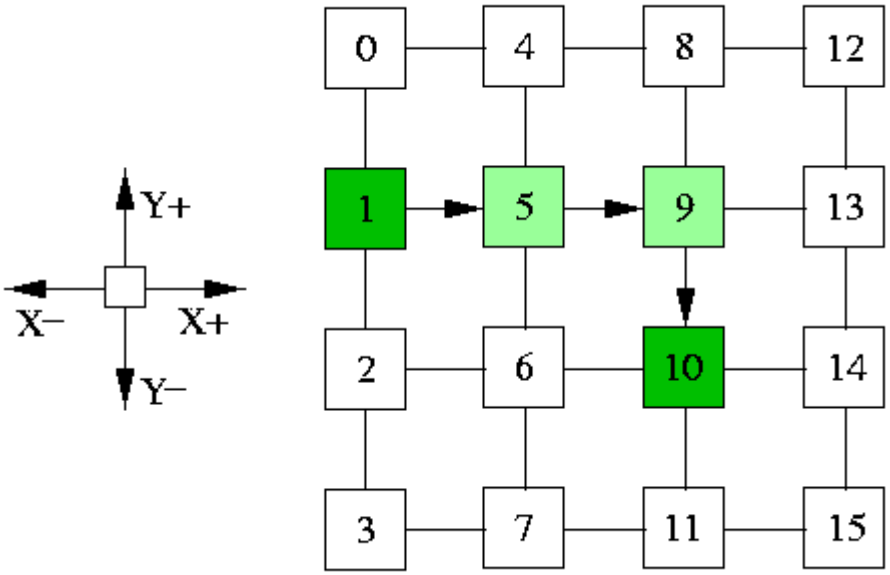



Fig 7. XY routing

## 4.3 Flow Control Mechanism

In data communications, **flow control** is the process of managing the rate of data transmission between two nodes to prevent a fast sender from overwhelming a slow receiver. It provides a mechanism for the receiver to control the transmission speed, so that the receiving node is not overwhelmed with data from transmitting node. Flow control should be distinguished from congestion control, which is used for controlling the flow of data when congestion has actually occurred.[1] Flow control mechanisms can be classified by whether or not the receiving node sends feedback to the sending node.

Flow control is important because it is possible for a sending computer to transmit information at a faster rate than the destination computer can receive and process it. This can happen if the receiving computers have a heavy traffic load in comparison to the sending computer, or if the receiving computer has less processing power than the sending computer

### 4.3.1 Stop and wait

Stop-and-wait flow control is the simplest form of flow control. In this method, the receiver indicates its readiness to receive data for each frame, the message is broken into multiple frames. The sender waits for an ACK (acknowledgement) after every frame for specified time (called time out). It is sent to ensure that the receiver has received the frame correctly. It will then send the next frame only after the ACK has been received.

This is the simplest file control protocol in which the sender transmits a frame and then waits for an acknowledgement, either positive or negative, from the receiver before proceeding. If a positive acknowledgement is received, the sender transmits the next packet; else it retransmits the same frame. However, this protocol has one major flaw in it. If a packet or an acknowledgement is completely destroyed in transit due to a noise burst, a deadlock will occur because the sender cannot proceed until it receives an acknowledgement. This problem may be solved using timers on the sender's side. When the frame is transmitted, the timer is set. If there is no response from the receiver within a certain time interval, the timer goes off and the frame may be retransmitted.

## Operations

1. **Sender:** Transmits a single frame at a time.
2. **Receiver:** Transmits acknowledgement (ACK) as it receives a frame.
3. Sender receives ACK within time out.
4. Go to step 1.

If a frame or ACK is lost during transmission then it has to be transmitted again by sender. This retransmission process is known as ARQ (automatic repeat request).

The problem with Stop-and wait is that only one frame can be transmitted at a time and that often leads to inefficient transmission channel till we get the acknowledgement the sender can not transmit any new packet. During this time both the sender and the channel are unutilised.

## Pros and cons of stop and wait

### Pros

The only advantage of this method of flow control is its simplicity.

### Cons

The sender needs to wait for the ACK after every frame it transmits. This is a source of inefficiency, and is particularly bad when the propagation delay is much longer than the transmission delay.[2]

Stop and wait can also create inefficiencies when sending longer transmissions.[3] When longer transmissions are sent there is more likely chance for error in this protocol. If the messages are short the errors are more likely to be detected early. More inefficiency is

created when single messages are broken into separate frames because it makes the transmission longer.

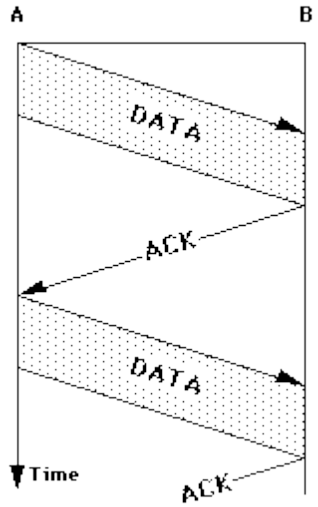



Fig 8. Stop n wait

### 4.3.2 Sliding Window

A method of flow control in which a receiver gives a transmitter permission to transmit data until a window is full. When the window is full, the transmitter must stop transmitting until the receiver advertises a larger window.

Sliding-window flow control is best utilized when the buffer size is limited and pre-established. During a typical communication between a sender and a receiver the receiver allocates buffer space for *n* frames (*n* is the buffer size in frames). The sender can send and the receiver can accept *n* frames without having to wait for an acknowledgement. A sequence number is assigned to frames in order to help keep track of those frames which did receive an acknowledgement. The receiver acknowledges a frame by sending an acknowledgement that includes the sequence number of the next frame expected. This acknowledgement announces that the receiver is ready to receive n frames, beginning with the number specified. Both the sender and receiver maintain what is called a window. The size of the window is less than or equal to the buffer size.

Sliding window flow control has a far better performance than stop-and-wait flow control. For example in a wireless environment data rates are very low and noise level is very high, so waiting for an acknowledgement for every packet that is transferred is not very feasible.

Therefore, transferring data as a bulk would yield a better performance in terms of higher throughput.

Sliding window flow control is a point to point protocol assuming that no other entity tries to communicate until the current data transfer is complete. The window maintained by the sender indicates which frames he can send. The sender sends all the frames in the window and waits for an acknowledgement (as opposed to acknowledging after every frame). The sender then shifts the window to the corresponding sequence number, thus indicating that frames within the window starting from the current sequence number can be sent.

Inspite of the use of timers, the stop and wait protocol still suffers from a few drawbacks. Firstly, if the receiver had the capacity to accept more than one frame, its resources are being underutilized. Secondly, if the receiver was busy and did not wish to receive any more packets, it may delay the acknowledgement. However, the timer on the sender's side may go off and cause an unnecessary retransmission. These drawbacks are overcome by the sliding window protocols.
In sliding window protocols the sender's data link layer maintains a 'sending window' which consists of a set of sequence numbers corresponding to the frames it is permitted to send. Similarly, the receiver maintains a 'receiving window' corresponding to the set of frames it is permitted to accept. The window size is dependent on the retransmission policy and it may differ in values for the receiver's and the sender's window. The sequence numbers within the sender's window represent the frames sent but as yet not acknowledged. Whenever a new packet arrives from the network layer, the upper edge of the window is advanced by one. When an acknowledgement arrives from the receiver the lower edge is advanced by one. The receiver's window corresponds to the frames that the receiver's data link layer may accept. When a frame with sequence number equal to the lower edge of the window is received, it is passed to the network layer, an acknowledgement is generated and the window is rotated by one. If however, a frame falling outside the window is received, the receiver's data link layer has two options. It may either discard this frame and all subsequent frames until the desired frame is received or it may accept these frames or buffer them until the appropriate frame is received and then pass the frames to the network layer in sequence.

## Go Back N

An automatic repeat request (ARQ) algorithm, used for error correction, in which a negative acknowledgement (NAK) causes retransmission of the word in error as well as the previous

N–1 words. The value of N is usually chosen such that the time taken to transmit the N words is less than the round trip delay from transmitter to receiver and back again. Therefore a buffer is not needed at the receiver.

The normalized propagation delay (a) = $^{\text{propagation time (Tp)}}/_{\text{transmission time (Tt)}}$, where Tp = Length (L) over propagation velocity (V) and Tt = bitrate (r) over Framerate (F). So that a = $^{LF}/_{Vr}$.

To get the utilization you must define a window size (N). If N is greater than or equal to 2a + 1 then the utilization is 1 (full utilization) for the transmission channel. If it is less than 2a + 1 then the equation $^{N}/_{1+2a}$ must be used to compute utilization.

If a frame is lost or received in error, the receiver may simply discard all subsequent frames, sending no acknowledgments for the discarded frames. In this case the receive window is of size 1. Since no acknowledgements are being received the sender's window will fill up, the sender will eventually time out and retransmit all the unacknowledged frames in order starting from the damaged or lost frame. The maximum window size for this protocol can be obtained as follows. Assume that the window size of the sender is n. So the window will initially contain the frames with sequence numbers from 0 to (w-1). Consider that the sender transmits all these frames and the receiver's data link layer receives all of them correctly. However, the sender's data link layer does not receive any acknowledgements as all of them are lost. So the sender will retransmit all the frames after its timer goes off. However the receiver window has already advanced to w. Hence to avoid overlap , the sum of the two windows should be less than the sequence number space.

**w-1 + 1 < Sequence Number Space**

**i.e., w < Sequence Number Space**

**Maximum Window Size = Sequence Number Space - 1**

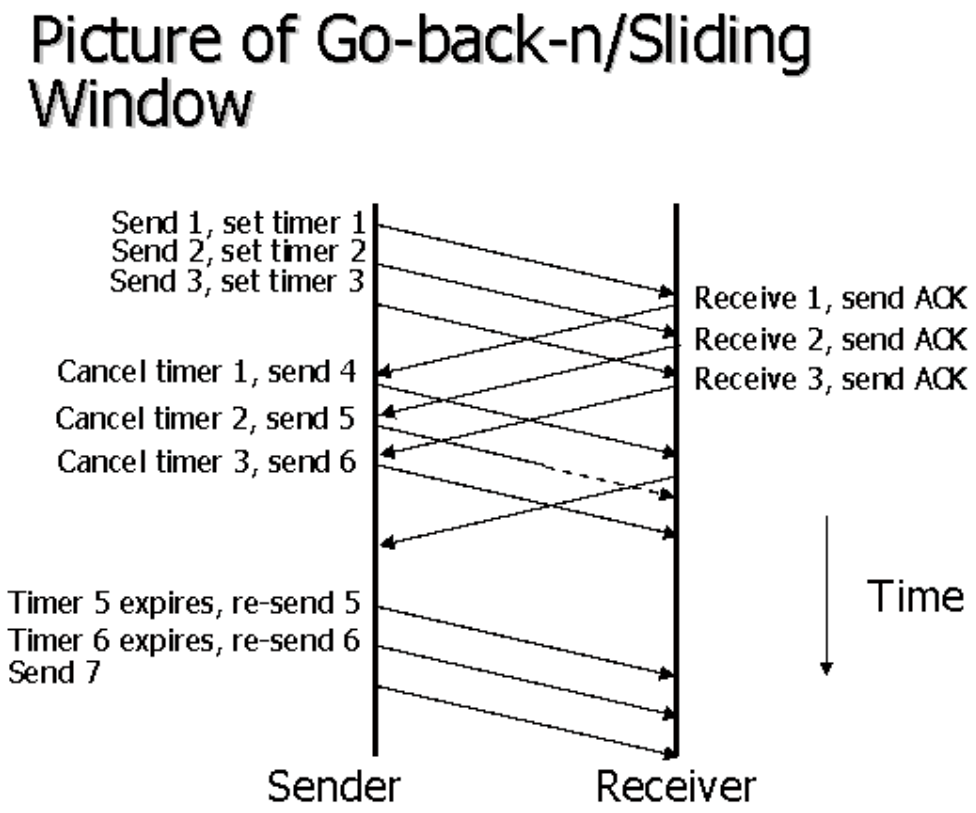



Fig 9.

## Selective Repeat

Selective Repeat is a connection oriented protocol in which both transmitter and receiver have a window of sequence numbers. The protocol has a maximum number of messages that can be sent without acknowledgement. If this window becomes full, the protocol is blocked until an acknowledgement is received for the earliest outstanding message. At this point the transmitter is clear to send more messages.

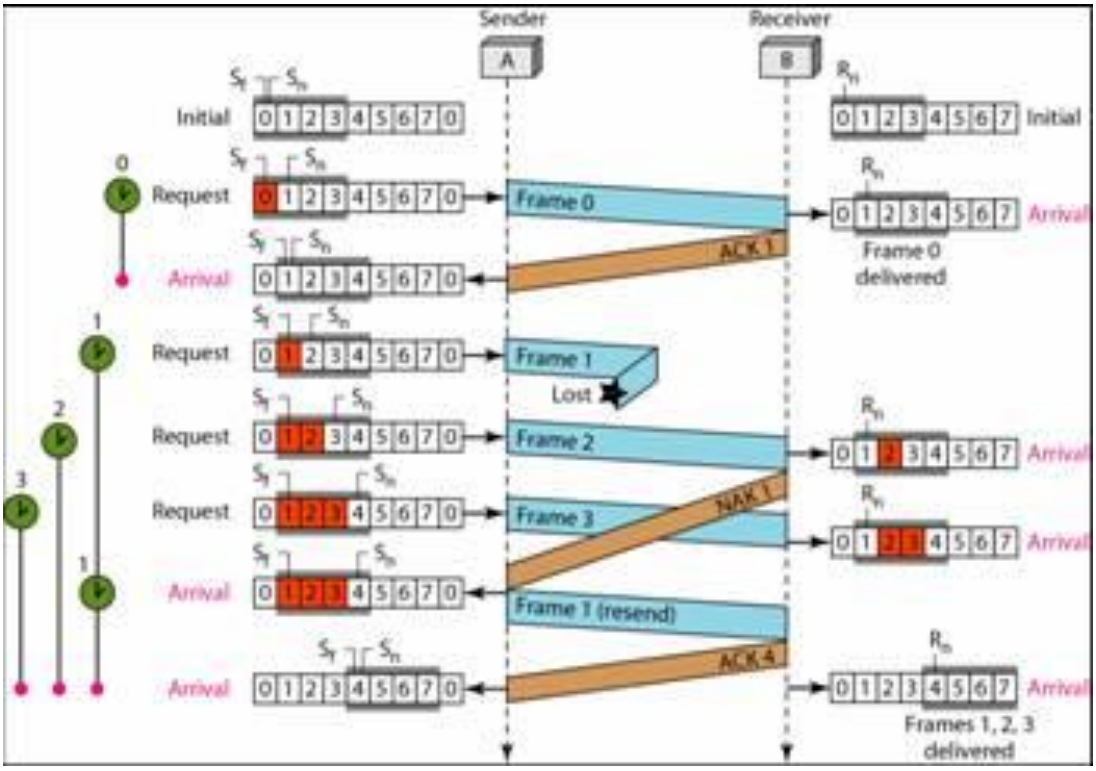



Fig 10-Selective Repeat.

## Comparison

This section is geared towards the idea of comparing Stop-and-wait, Sliding Window with the subsets of Go Back N and Selective Repeat.

Each equation is a general overview. View each page for more in-depth definition.

## Stop-and-wait

Error free: 1/2a+1

With errors: 1-P/1+2a

## Selective Repeat

We define throughput T as the average number of blocks communicated per transmitted block. It is more convenient to calculate the average number of transmissions necessary to communicate a block, a quantity we denote by 0, and then to determine T from the equation.

T=1/b

# 5. Implementation

## 5. 1 Auto Align Method

Function of this Method: This method is used to align the components on the canvas and is called when the auto align button is clicked.

ALGORITHM: AUTO_ALIGN

Step 1: Divide whole of the canvas into a grid.
Step 2: Assign each component to the respective column such that column width is less than twice the width of the component after making a copy of all the current coordinates.
Step 3: Set the x coordinate of the components equal to the average x value of that column.
Step 4: Set the x coordinates of the components in the following columns by adding the horizontal distance specified by the user.
Step 5: Get the y coordinate of the first component in first column and determine the total y height of the first column.
Step 6: Get the y heights of all the columns.
Step 7: Determine the height differences of all the columns.
Step 8: To the aligning y factor of every column add the average height difference.
Step 9: Assign the newly calculated (x, y) coordinates to the respective components.
Step 10: Revert alignment if overlapping occurs.

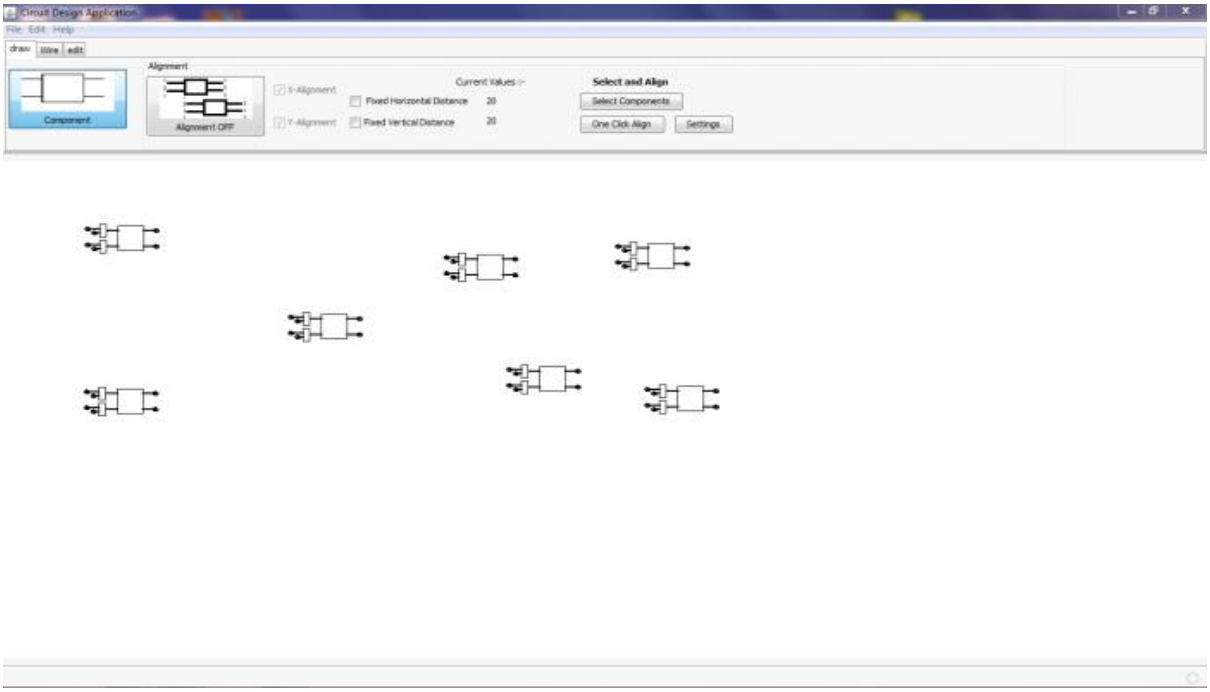


Figure 11. Shows the components to be aligned.

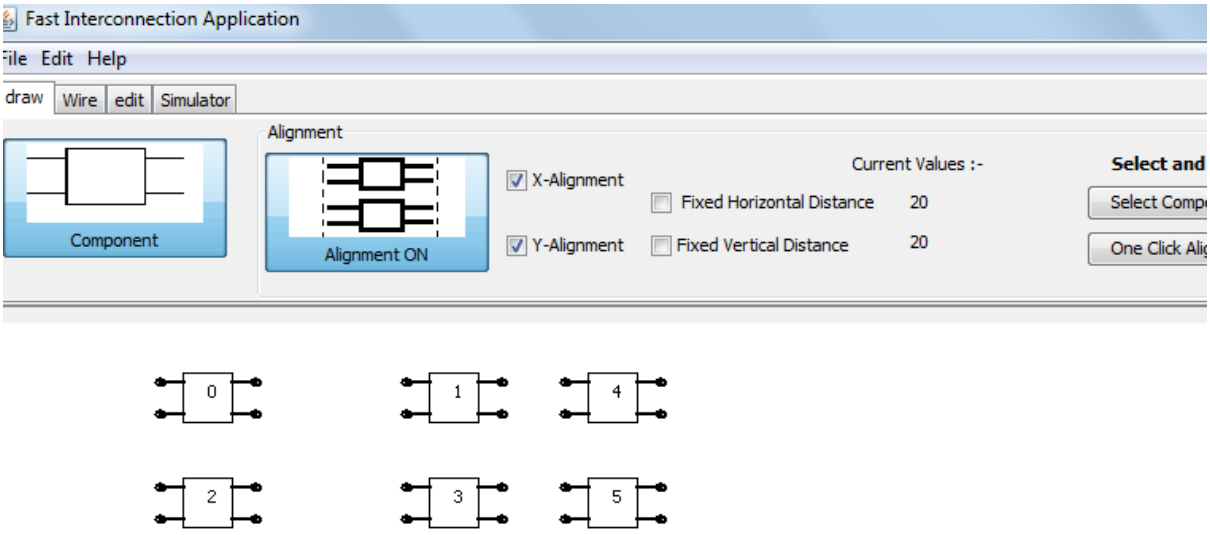



Figure 12. Shows that the elements when alignment is on.

## 5.2 Draw Method

Function of this method: This method is used to draw the components on the canvas. It is called when the canvas is refreshed, a new component is added to the circuit, when any movement occurs, or when a wire connection is made.

ALGORITHM: DRAW
Declare four points;
Get the graphics object for the canvas;
Initialize a for loop from 0 to the total 'number of components' - 1
Begin for loop
Draw each component according to the 'type of the component' variable stored in the component class;
End loop
Initialize a for loop from 0 to the total 'number of lines' - 1
Begin for loop
If line to be drawn is a normal line
get start point;
get end point;
draw line;
Else
retrieve the top and bottom points of the component
If line is to be drawn from upper point
Then
the line clearance will be 1.5 times the width of the component;
Else
the line clearance is 2 times
End if block
If the start point is before the final point draw line 1.5 times of the width from right;
Else
draw line 1.5 times of the width from left;
End if block
End if block
End for loop

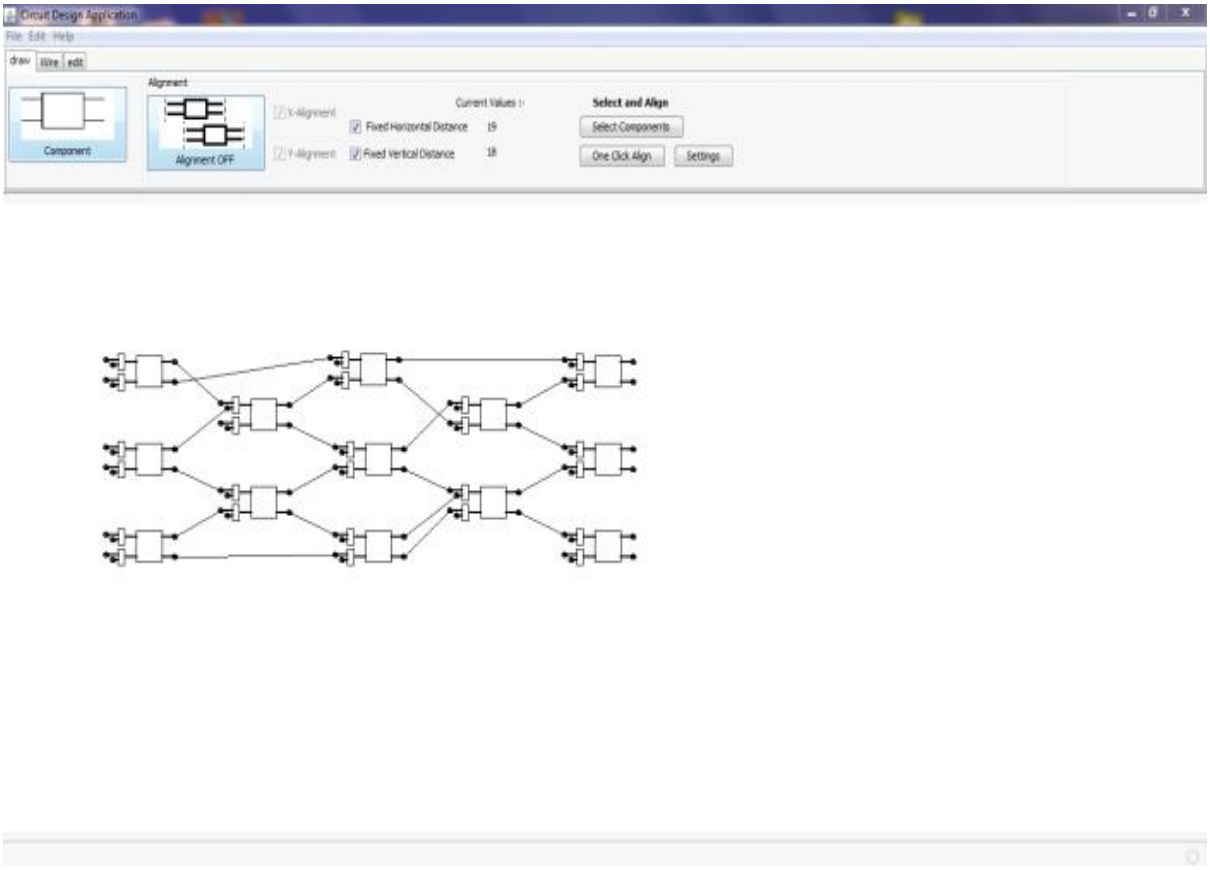



Fig 13 Drawing components

# 6. Tools and Technologies

## 6.1 Java Swings API

**Swing** is a GUI widget toolkit for Java . It is part of Oracle's Java Foundation Classes (JFC) — an API for providing a graphical user interface (GUI) for Java programs.
Swing was developed to provide a more sophisticated set of GUI components than the earlier Abstract Window Toolkit (AWT). Swing provides a native look and feel that emulates the look and feel of several platforms, and also supports a pluggable look and feel that allows applications to have a look and feel unrelated to the underlying platform. It has more powerful and flexible components than AWT. In addition to familiar components such as buttons, check boxes and labels, Swing provides several advanced components such as tabbed panel, scroll panes, trees, tables, and lists.
Unlike AWT components, Swing components are not implemented by platform-specific code. Instead they are written entirely in Java and therefore are platform-independent. The term "lightweight" is used to describe such an element.

## 6.2 Tool-Net Beans IDE

**NetBeans** is an integrated development environment (IDE) for developing primarily with Java, but also with other languages, in particular PHP, C/C++, and HTML5. It is also an application platform framework for Java desktop applications and others.

The NetBeans IDE is written in Java and can run on Windows, OS X, Linux, Solaris and other platforms supporting a compatible JVM.

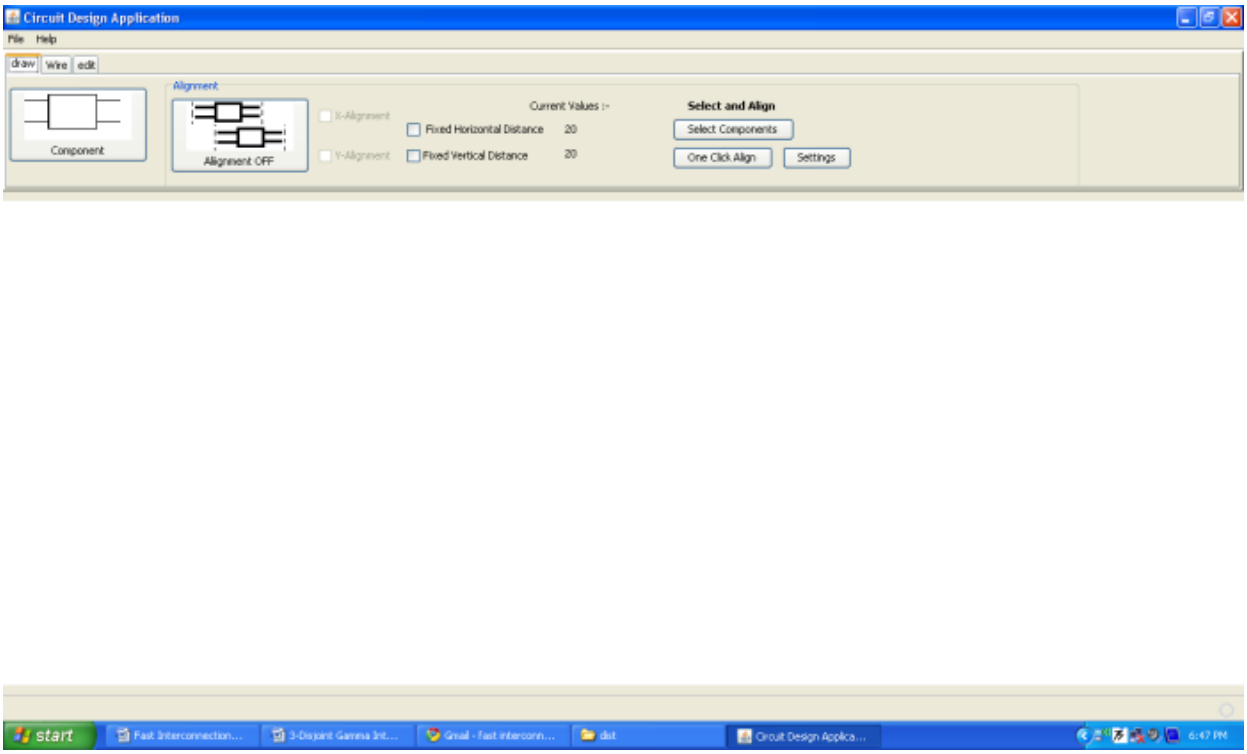
The NetBeans Platform allows applications to be developed from a set of modular software components called *modules*. Applications based on the NetBeans Platform (including the NetBeans IDE itself) can be extended by third party developers.

The NetBeans Team actively support the product and seek feature suggestions from the wider community. Every release is preceded by a time for Community testing and feedback

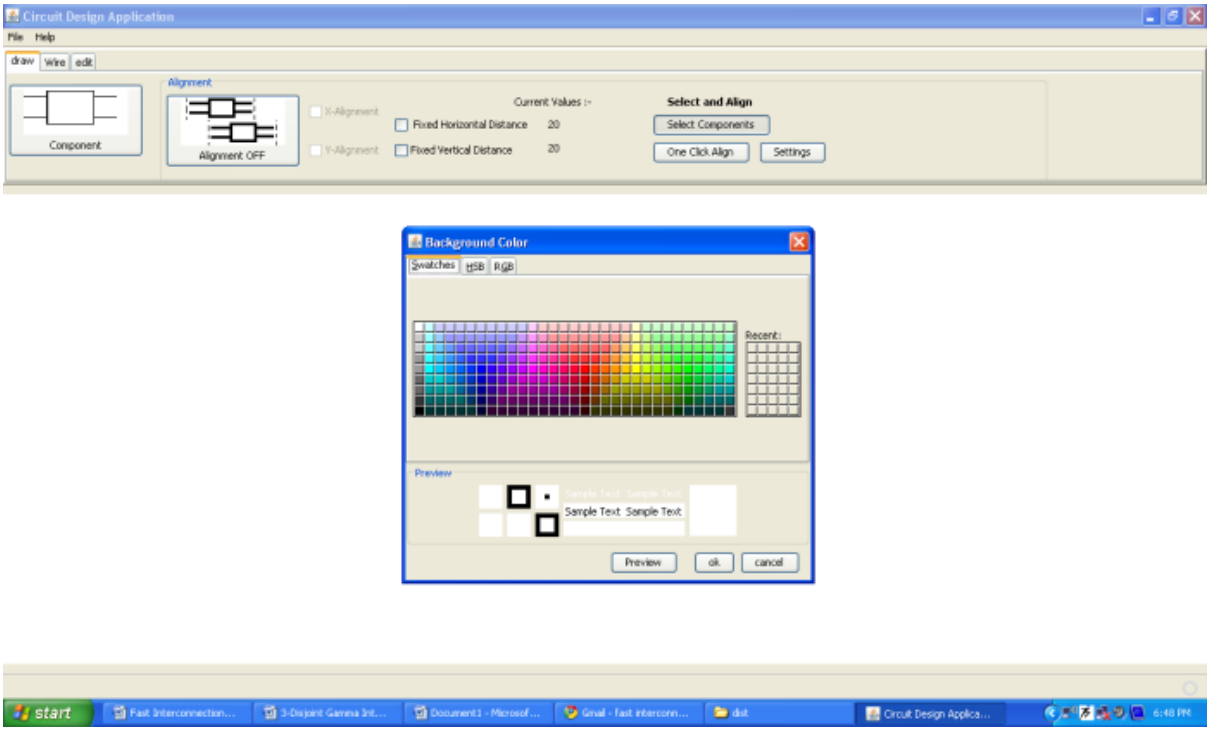
**GUI design tool**

Formerly known as *project Matisse*, the GUI design-tool enables developers to prototype and design Swing GUIs by dragging and positioning GUI components.
The GUI builder has built-in support for JSR 295 (Beans Binding technology), but the support for JSR 296 (Swing Application Framework) was removed in 7.

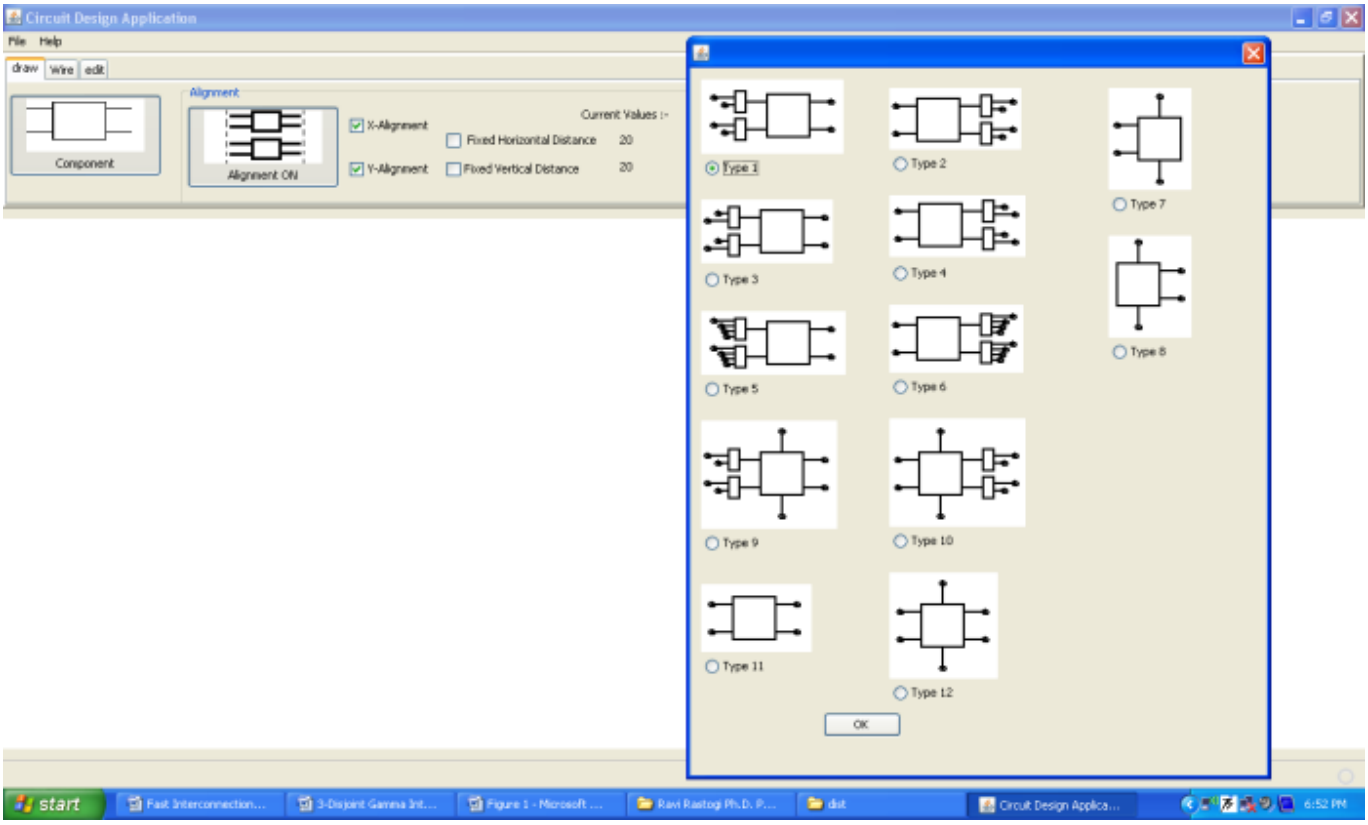
# 7. Screen Shots



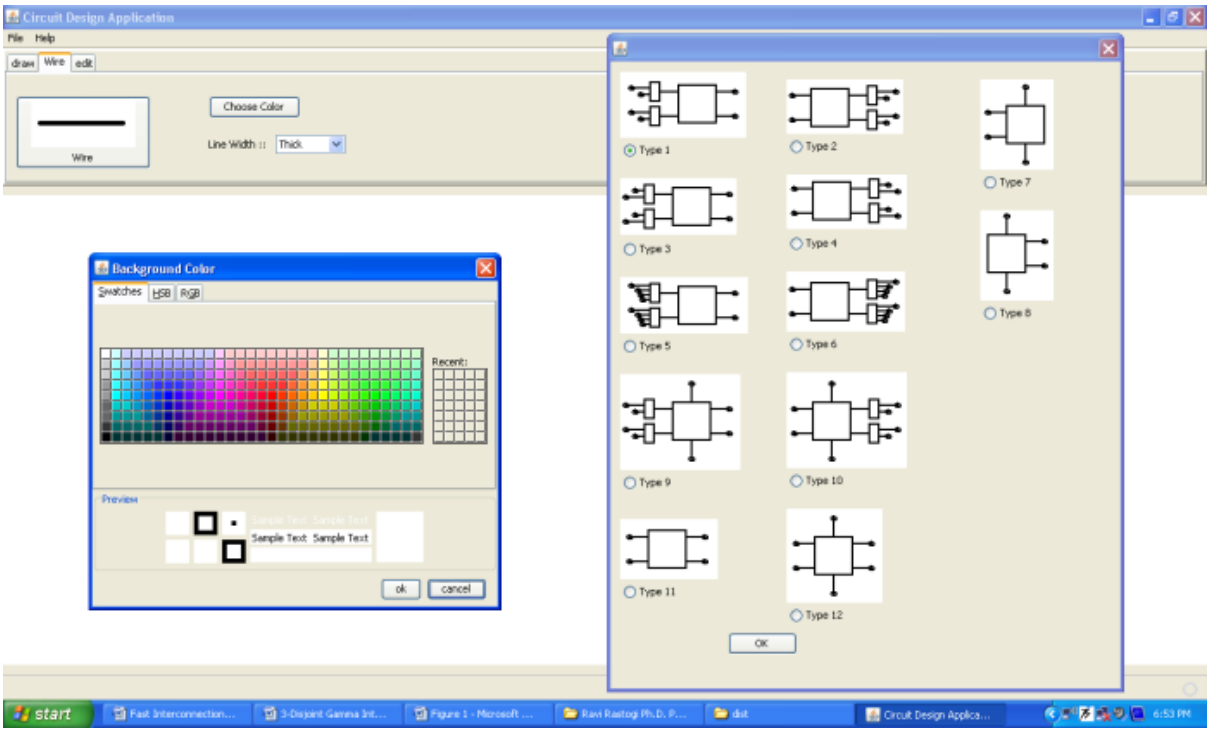

**The front window of the case tool.**



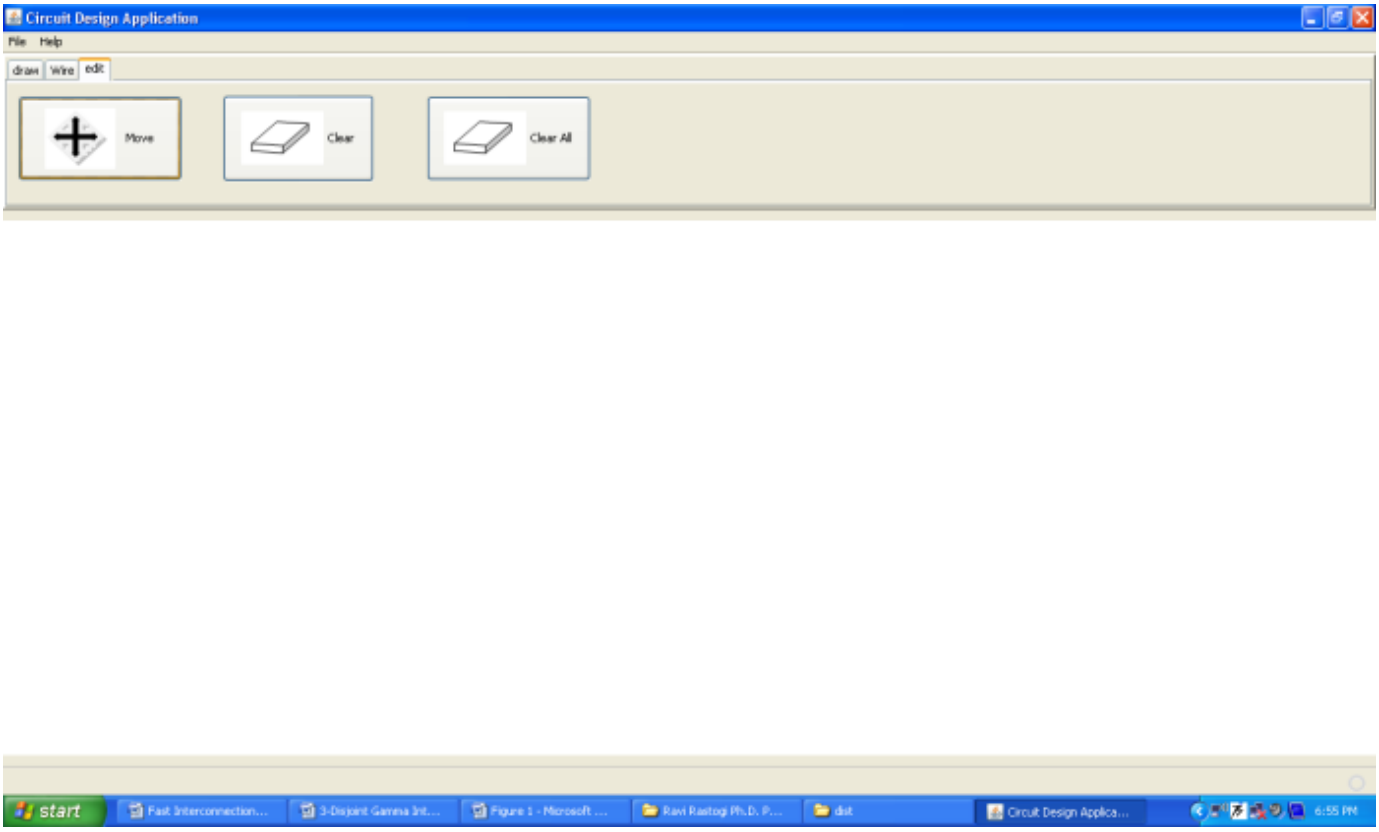

**The front window of the software, where you can change the background color.**



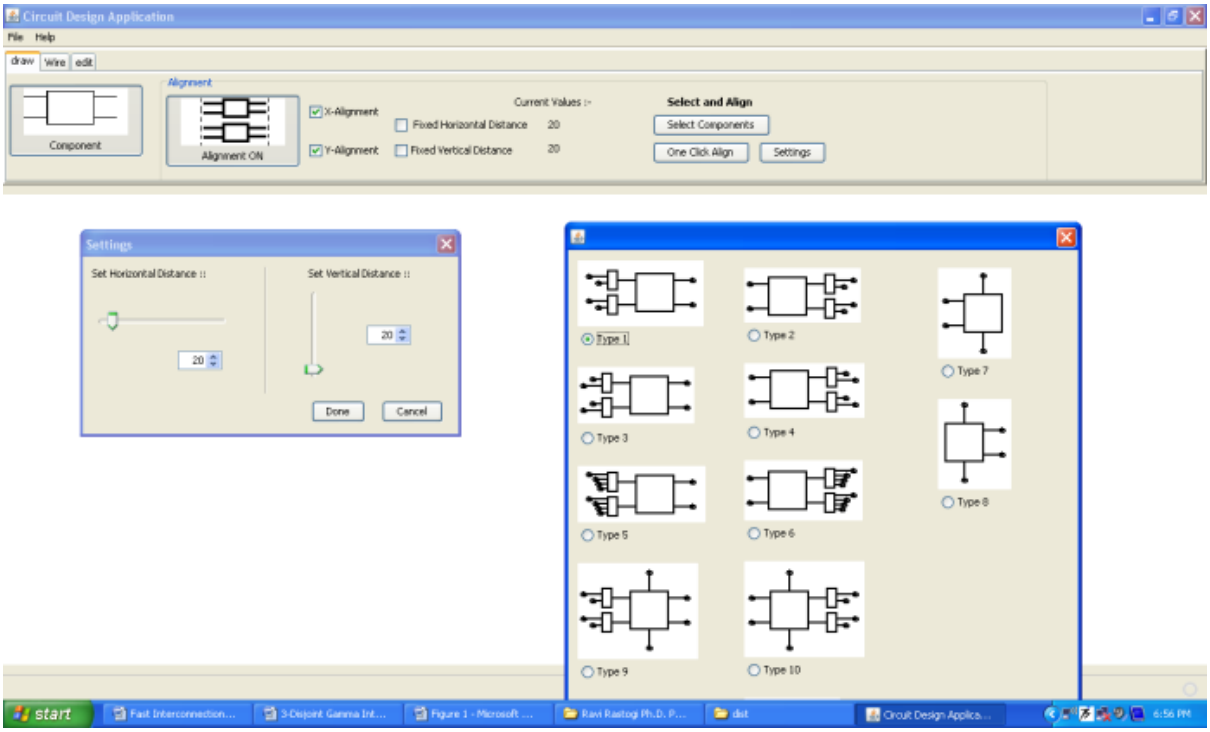

**The front window case tool with the various components.**



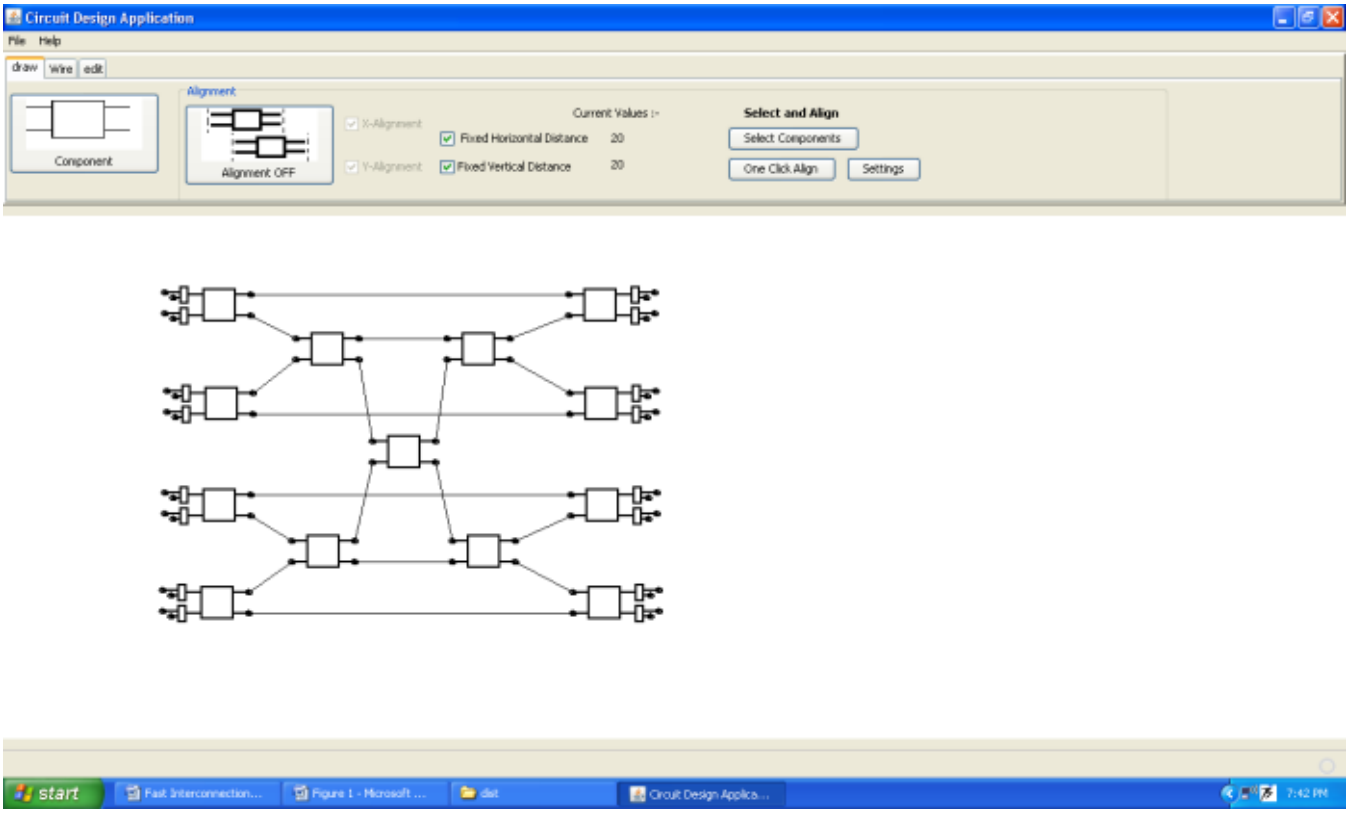

**The front window with different widths of the wire and with different background color**.




**the tool with the various edit components.**




**The front window of the case tool with various components where the size of the application window can be fixed in terms of horizontal and vertical distance.**




**A Interconnection Network has been designed using the Simulator**

# 8. Code

```
package abc;

import java.awt.BasicStroke;
import java.awt.Color;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.Point;
import java.awt.Stroke;
import java.io.Serializable;

public class cmp1 implements Serializable{
private Point p[]=new Point[22];
int ht=30,w=30,type;
Point centre;
Point[] node=new Point[12];
int no_node=4;
int no_pt;
Color fill=Color.WHITE;

void setColor(Color fill)
{
    this.fill=fill;
}



int top_bot(int x)
```

```
{
    int a=0;
    if(type==7)
    {   if(x==2)
              a=1;
        if(x==3)
              a=2;
    }

    if(type==8)
        {   if(x==2)
              a=1;
        if(x==3)
              a=2;
    }
    if(type==9)
        {   if(x==6)
              a=1;
        if(x==7)
              a=2;
    }
    if(type==10)

    {   if(x==6)
              a=1;
        if(x==7)
              a=2;
    }

    if(type==12)
        {   if(x==2)
              a=1;
        if(x==3)
              a=2;
    }
    return a;
}

void cmp1()
{
    for(int i=0;i<p.length;i++)
        p[i]=new Point(0,0);
}

void set(int x1,int y1)
{
    if(type >=1 && type <= 6)
    {
        p[0] = new Point(x1-w, y1-ht/3);
        p[1] = new Point(x1-w, y1+ht/3);
        p[2] = new Point(x1-w/2, y1-ht/2);
        p[3] = new Point(x1-w/2, y1-ht/3);
        p[4] = new Point(x1-w/2, y1+ht/3);
        p[5] = new Point(x1-w/2, y1+ht/2);
        p[6] = new Point(x1+w/2, y1-ht/2);
        p[7] = new Point(x1+w/2, y1-ht/3);
```

```
p[8] = new Point(x1+w/2, y1+ht/3);
p[9] = new Point(x1+w/2, y1+ht/2);
p[10] = new Point(x1+w, y1-ht/3);
p[11] = new Point(x1+w, y1+ht/3);

if(type == 1)
{
    p[12]=new Point(x1-w-w/4,y1-ht/3-ht/4);
    p[13]=new Point(x1-w-w/4,y1+ht/3-ht/4);
    p[14]=new Point(x1-w-w/4-w/2,y1-ht/3-ht/4+ht/6);
    p[15]=new Point(x1-w-w/4-w/4,y1-ht/3-ht/4+ht/3);
    p[16]=new Point(x1-w-w/4-w/2,y1+ht/3-ht/4+ht/6);
    p[17]=new Point(x1-w-w/4-w/4,y1+ht/3-ht/4+ht/3);
no_node=6;
node[0]=p[14];
node[1]=p[15];
node[2]=p[16];
node[3]=p[17];
node[4]=p[10];
node[5]=p[11];
}

if(type == 2)
{
    p[12] = new Point(x1+w, y1-ht/3-ht/4);
    p[13] = new Point(x1+w, y1+ht/3-ht/4);
    p[14] = new Point(x1+w+w/4+w/2, y1-ht/3-ht/4+ht/6);
    p[15] = new Point(x1+w+w/4+w/4,y1-ht/3-ht/4+ht/3);
    p[16] = new Point(x1+w+w/4+w/2,y1+ht/3-ht/4+ht/6);
    p[17] = new Point(x1+w+w/4+w/4,y1+ht/3-ht/4+ht/3);
    no_node=6;
node[0]=p[14];
node[1]=p[15];
node[2]=p[16];
node[3]=p[17];
node[4]=p[0];
node[5]=p[1];
}

if(type == 3)
{
    p[12]=new Point(x1-w-w/4,y1-ht/3-ht/4);
    p[13]=new Point(x1-w-w/4,y1+ht/3-ht/4);
    p[14]=new Point(x1-w-w/4-w/4, y1-ht/3-ht/4+ht/6);
    p[15]=new Point(x1-w-w/4-w/2, y1-ht/3-ht/4+ht/3);
    p[16]=new Point(x1-w-w/4-w/4, y1+ht/3-ht/4+ht/6);
    p[17]=new Point(x1-w-w/4-w/2, y1+ht/3-ht/4+ht/3);
     no_node=6;
node[0]=p[14];
node[1]=p[15];
node[2]=p[16];
node[3]=p[17];
node[4]=p[10];
node[5]=p[11];
}
```

```
if(type == 4)
{
    p[12] = new Point(x1+w, y1-ht/3-ht/4);
    p[13] = new Point(x1+w, y1+ht/3-ht/4);
    p[14] = new Point(x1+w+w/4+w/4, y1-ht/3-ht/4+ht/6);
    p[15] = new Point(x1+w+w/4+w/2,y1-ht/3-ht/4+ht/3);
    p[16] = new Point(x1+w+w/4+w/4,y1+ht/3-ht/4+ht/6);
    p[17] = new Point(x1+w+w/4+w/2,y1+ht/3-ht/4+ht/3);
     no_node=6;
node[0]=p[14];
node[1]=p[15];
node[2]=p[16];
node[3]=p[17];
node[4]=p[0];
node[5]=p[1];
}

if(type == 5) //to be modified
{
    p[12]=new Point(x1-w-w/4,y1-ht/3-ht/4);
    p[13]=new Point(x1-w-w/4,y1+ht/3-ht/4);
    p[14]=new Point(x1-w-w/4-w/2,y1-ht/3-ht/4+ht/20);
    p[15]=new Point(x1-w-w/4-w/3,y1-ht/3-ht/4+ht/6+1);
    p[16]=new Point(x1-w-w/4-w/4,y1-ht/3-ht/4+ht/5+4);
    p[17]=new Point(x1-w-w/4-w/5,y1-ht/3-ht/4+ht/4+7);

    p[18]=new Point(x1-w-w/4-w/2,y1+ht/3-ht/4+ht/20);
    p[19]=new Point(x1-w-w/4-w/3,y1+ht/3-ht/4+ht/6+1);
    p[20]=new Point(x1-w-w/4-w/4,y1+ht/3-ht/4+ht/5+4);
    p[21]=new Point(x1-w-w/4-w/5,y1+ht/3-ht/4+ht/4+7);
     no_node=10;
node[0]=p[14];
node[1]=p[15];
node[2]=p[16];
node[3]=p[17];
node[4]=p[10];
node[5]=p[11];
node[6]=p[18];
node[7]=p[19];
node[8]=p[20];
node[9]=p[21];
}

if(type == 6)
{
    p[12]=new Point(x1+w,y1-ht/3-ht/4);
    p[13]=new Point(x1+w,y1+ht/3-ht/4);
    p[14]=new Point(x1+w+w/4+w/2,y1-ht/3-ht/4+ht/20);
    p[15]=new Point(x1+w+w/4+w/3,y1-ht/3-ht/4+ht/6+1);
    p[16]=new Point(x1+w+w/4+w/4,y1-ht/3-ht/4+ht/5+4);
    p[17]=new Point(x1+w+w/4+w/5,y1-ht/3-ht/4+ht/4+7);

    p[18]=new Point(x1+w+w/4+w/2,y1+ht/3-ht/4+ht/20);
    p[19]=new Point(x1+w+w/4+w/3,y1+ht/3-ht/4+ht/6+1);
    p[20]=new Point(x1+w+w/4+w/4,y1+ht/3-ht/4+ht/5+4);
    p[21]=new Point(x1+w+w/4+w/5,y1+ht/3-ht/4+ht/4+7);
```

```
        no_node=10;
    node[0]=p[14];
    node[1]=p[15];
    node[2]=p[16];
    node[3]=p[17];
    node[4]=p[0];
    node[5]=p[1];
    node[6]=p[18];
    node[7]=p[19];
    node[8]=p[20];
    node[9]=p[21];
    }
}

if(type >= 7 && type <= 8)
{
    p[0] = new Point(x1-w, y1-ht/3);
    p[1] = new Point(x1-w, y1+ht/3);
    p[2] = new Point(x1-w/2, y1-ht/2);
    p[3] = new Point(x1-w/2, y1-ht/3);
    p[4] = new Point(x1-w/2, y1+ht/3);
    p[5] = new Point(x1-w/2, y1+ht/2);
    p[6] = new Point(x1+w/2, y1-ht/2);
    p[7] = new Point(x1+w/2, y1-ht/3);
    p[8] = new Point(x1+w/2, y1+ht/3);
    p[9] = new Point(x1+w/2, y1+ht/2);
    p[10] = new Point(x1+w, y1-ht/3);
    p[11] = new Point(x1+w, y1+ht/3);
   p[12] = new Point(x1, y1+ht);
    p[13] = new Point(x1, y1-ht);
    no_node=4;
    if(type==7)
    {
    node[0]=p[0];
    node[1]=p[1];
    node[2]=p[12];
    node[3]=p[13];
    }
    if(type==8)
    {
    node[0]=p[10];
    node[1]=p[11];
    node[2]=p[12];
    node[3]=p[13];
    }

}

if(type >= 9 && type <= 10)
{
    p[0] = new Point(x1-w, y1-ht/3);
    p[1] = new Point(x1-w, y1+ht/3);
    p[2] = new Point(x1-w/2, y1-ht/2);
    p[3] = new Point(x1-w/2, y1-ht/3);
    p[4] = new Point(x1-w/2, y1+ht/3);
    p[5] = new Point(x1-w/2, y1+ht/2);
```

```
    p[6] = new Point(x1+w/2, y1-ht/2);
    p[7] = new Point(x1+w/2, y1-ht/3);
    p[8] = new Point(x1+w/2, y1+ht/3);
    p[9] = new Point(x1+w/2, y1+ht/2);
    p[10] = new Point(x1+w, y1-ht/3);
    p[11] = new Point(x1+w, y1+ht/3);
    p[18] = new Point(x1, y1+ht);
    p[19] = new Point(x1, y1-ht);

    if(type == 9)
    {
        p[12]=new Point(x1-w-w/4,y1-ht/3-ht/4);
        p[13]=new Point(x1-w-w/4,y1+ht/3-ht/4);
        p[14]=new Point(x1-w-w/4-w/2,y1-ht/3-ht/4+ht/6);
        p[15]=new Point(x1-w-w/4-w/4,y1-ht/3-ht/4+ht/3);
        p[16]=new Point(x1-w-w/4-w/2,y1+ht/3-ht/4+ht/6);
        p[17]=new Point(x1-w-w/4-w/4,y1+ht/3-ht/4+ht/3);
        no_node=8;
    node[0]=p[14];
    node[1]=p[15];
    node[2]=p[16];
    node[3]=p[17];
    node[4]=p[10];
    node[5]=p[11];
    node[6]=p[18];
    node[7]=p[19];
    }

    if(type == 10)
    {
        p[12]=new Point(x1+w,y1-ht/3-ht/4);
        p[13]=new Point(x1+w,y1+ht/3-ht/4);
        p[14]=new Point(x1+w+w/4+w/2,y1-ht/3-ht/4+ht/6);
        p[15]=new Point(x1+w+w/4+w/4,y1-ht/3-ht/4+ht/3);
        p[16]=new Point(x1+w+w/4+w/2,y1+ht/3-ht/4+ht/6);
        p[17]=new Point(x1+w+w/4+w/4,y1+ht/3-ht/4+ht/3);
        no_node=8;
    node[0]=p[14];
    node[1]=p[15];
    node[2]=p[16];
    node[3]=p[17];
    node[4]=p[0];
    node[5]=p[1];
    node[6]=p[18];
    node[7]=p[19];
    }

}
 if(type == 11)
{
    p[0] = new Point(x1-w, y1-ht/3);
    p[1] = new Point(x1-w, y1+ht/3);
    p[2] = new Point(x1-w/2, y1-ht/2);
    p[3] = new Point(x1-w/2, y1-ht/3);
    p[4] = new Point(x1-w/2, y1+ht/3);
    p[5] = new Point(x1-w/2, y1+ht/2);
```

```
        p[6] = new Point(x1+w/2, y1-ht/2);
        p[7] = new Point(x1+w/2, y1-ht/3);
        p[8] = new Point(x1+w/2, y1+ht/3);
        p[9] = new Point(x1+w/2, y1+ht/2);
        p[10] = new Point(x1+w, y1-ht/3);
        p[11] = new Point(x1+w, y1+ht/3);
        no_node=4;
        node[0]=p[0];
        node[1]=p[1];
        node[2]=p[10];
        node[3]=p[11];
    }

    if(type == 12)
    {
        p[0] = new Point(x1-w, y1-ht/3);
        p[1] = new Point(x1-w, y1+ht/3);
        p[2] = new Point(x1-w/2, y1-ht/2);
        p[3] = new Point(x1-w/2, y1-ht/3);
        p[4] = new Point(x1-w/2, y1+ht/3);
        p[5] = new Point(x1-w/2, y1+ht/2);
        p[6] = new Point(x1+w/2, y1-ht/2);
        p[7] = new Point(x1+w/2, y1-ht/3);
        p[8] = new Point(x1+w/2, y1+ht/3);
        p[9] = new Point(x1+w/2, y1+ht/2);
        p[10] = new Point(x1+w, y1-ht/3);
        p[11] = new Point(x1+w, y1+ht/3);
        p[12] = new Point(x1, y1+ht);
        p[13] = new Point(x1, y1-ht);
        no_node=6;
        node[0]=p[0];
        node[1]=p[1];
        node[2]=p[12];
        node[3]=p[13];
        node[4]=p[10];
        node[5]=p[11];
    }

    centre = new Point(x1, y1);
}


void set(int x1,int y1,int t)
{
    type=t;
    if(type >=1 && type <= 6)
    {
        p[0] = new Point(x1-w, y1-ht/3);
        p[1] = new Point(x1-w, y1+ht/3);
        p[2] = new Point(x1-w/2, y1-ht/2);
        p[3] = new Point(x1-w/2, y1-ht/3);
        p[4] = new Point(x1-w/2, y1+ht/3);
        p[5] = new Point(x1-w/2, y1+ht/2);
        p[6] = new Point(x1+w/2, y1-ht/2);
        p[7] = new Point(x1+w/2, y1-ht/3);
        p[8] = new Point(x1+w/2, y1+ht/3);
```

```
p[9] = new Point(x1+w/2, y1+ht/2);
p[10] = new Point(x1+w, y1-ht/3);
p[11] = new Point(x1+w, y1+ht/3);

if(type == 1)
{
    p[12]=new Point(x1-w-w/4,y1-ht/3-ht/4);
    p[13]=new Point(x1-w-w/4,y1+ht/3-ht/4);
    p[14]=new Point(x1-w-w/4-w/2,y1-ht/3-ht/4+ht/6);
    p[15]=new Point(x1-w-w/4-w/4,y1-ht/3-ht/4+ht/3);
    p[16]=new Point(x1-w-w/4-w/2,y1+ht/3-ht/4+ht/6);
    p[17]=new Point(x1-w-w/4-w/4,y1+ht/3-ht/4+ht/3);
no_node=6;
node[0]=p[14];
node[1]=p[15];
node[2]=p[16];
node[3]=p[17];
node[4]=p[10];
node[5]=p[11];
no_pt=18;
}

if(type == 2)
{
    p[12] = new Point(x1+w, y1-ht/3-ht/4);
    p[13] = new Point(x1+w, y1+ht/3-ht/4);
    p[14] = new Point(x1+w+w/4+w/2, y1-ht/3-ht/4+ht/6);
    p[15] = new Point(x1+w+w/4+w/4,y1-ht/3-ht/4+ht/3);
    p[16] = new Point(x1+w+w/4+w/2,y1+ht/3-ht/4+ht/6);
    p[17] = new Point(x1+w+w/4+w/4,y1+ht/3-ht/4+ht/3);
    no_node=6;
node[0]=p[14];
node[1]=p[15];
node[2]=p[16];
node[3]=p[17];
node[4]=p[0];
node[5]=p[1];
no_pt=18;
}

if(type == 3)
{
    p[12]=new Point(x1-w-w/4,y1-ht/3-ht/4);
    p[13]=new Point(x1-w-w/4,y1+ht/3-ht/4);
    p[14]=new Point(x1-w-w/4-w/4, y1-ht/3-ht/4+ht/6);
    p[15]=new Point(x1-w-w/4-w/2, y1-ht/3-ht/4+ht/3);
    p[16]=new Point(x1-w-w/4-w/4, y1+ht/3-ht/4+ht/6);
    p[17]=new Point(x1-w-w/4-w/2, y1+ht/3-ht/4+ht/3);
     no_node=6;
node[0]=p[14];
node[1]=p[15];
node[2]=p[16];
node[3]=p[17];
node[4]=p[10];
node[5]=p[11];
no_pt=18;
```

```
}

if(type == 4)
{
    p[12] = new Point(x1+w, y1-ht/3-ht/4);
    p[13] = new Point(x1+w, y1+ht/3-ht/4);
    p[14] = new Point(x1+w+w/4+w/4, y1-ht/3-ht/4+ht/6);
    p[15] = new Point(x1+w+w/4+w/2,y1-ht/3-ht/4+ht/3);
    p[16] = new Point(x1+w+w/4+w/4,y1+ht/3-ht/4+ht/6);
    p[17] = new Point(x1+w+w/4+w/2,y1+ht/3-ht/4+ht/3);
     no_node=6;
node[0]=p[14];
node[1]=p[15];
node[2]=p[16];
node[3]=p[17];
node[4]=p[0];
node[5]=p[1];
no_pt=18;
}

if(type == 5) //to be modified
{
    p[12]=new Point(x1-w-w/4,y1-ht/3-ht/4);
    p[13]=new Point(x1-w-w/4,y1+ht/3-ht/4);
    p[14]=new Point(x1-w-w/4-w/2,y1-ht/3-ht/4+ht/20);
    p[15]=new Point(x1-w-w/4-w/3,y1-ht/3-ht/4+ht/6+1);
    p[16]=new Point(x1-w-w/4-w/4,y1-ht/3-ht/4+ht/5+4);
    p[17]=new Point(x1-w-w/4-w/5,y1-ht/3-ht/4+ht/4+7);

    p[18]=new Point(x1-w-w/4-w/2,y1+ht/3-ht/4+ht/20);
    p[19]=new Point(x1-w-w/4-w/3,y1+ht/3-ht/4+ht/6+1);
    p[20]=new Point(x1-w-w/4-w/4,y1+ht/3-ht/4+ht/5+4);
    p[21]=new Point(x1-w-w/4-w/5,y1+ht/3-ht/4+ht/4+7);
     no_node=10;
node[0]=p[14];
node[1]=p[15];
node[2]=p[16];
node[3]=p[17];
node[4]=p[10];
node[5]=p[11];
node[6]=p[18];
node[7]=p[19];
node[8]=p[20];
node[9]=p[21];
no_pt=22;
}

if(type == 6)
{
    p[12]=new Point(x1+w,y1-ht/3-ht/4);
    p[13]=new Point(x1+w,y1+ht/3-ht/4);
    p[14]=new Point(x1+w+w/4+w/2,y1-ht/3-ht/4+ht/20);
    p[15]=new Point(x1+w+w/4+w/3,y1-ht/3-ht/4+ht/6+1);
    p[16]=new Point(x1+w+w/4+w/4,y1-ht/3-ht/4+ht/5+4);
    p[17]=new Point(x1+w+w/4+w/5,y1-ht/3-ht/4+ht/4+7);
```

```
        p[18]=new Point(x1+w+w/4+w/2,y1+ht/3-ht/4+ht/20);
        p[19]=new Point(x1+w+w/4+w/3,y1+ht/3-ht/4+ht/6+1);
        p[20]=new Point(x1+w+w/4+w/4,y1+ht/3-ht/4+ht/5+4);
        p[21]=new Point(x1+w+w/4+w/5,y1+ht/3-ht/4+ht/4+7);
        no_node=10;
    node[0]=p[14];
    node[1]=p[15];
    node[2]=p[16];
    node[3]=p[17];
    node[4]=p[0];
    node[5]=p[1];
    node[6]=p[18];
    node[7]=p[19];
    node[8]=p[20];
    node[9]=p[21];
    no_pt=22;
    }
}

if(type >= 7 && type <= 8)
{
    p[0] = new Point(x1-w, y1-ht/3);
    p[1] = new Point(x1-w, y1+ht/3);
    p[2] = new Point(x1-w/2, y1-ht/2);
    p[3] = new Point(x1-w/2, y1-ht/3);
    p[4] = new Point(x1-w/2, y1+ht/3);
    p[5] = new Point(x1-w/2, y1+ht/2);
    p[6] = new Point(x1+w/2, y1-ht/2);
    p[7] = new Point(x1+w/2, y1-ht/3);
    p[8] = new Point(x1+w/2, y1+ht/3);
    p[9] = new Point(x1+w/2, y1+ht/2);
    p[10] = new Point(x1+w, y1-ht/3);
    p[11] = new Point(x1+w, y1+ht/3);
    p[12] = new Point(x1, y1+ht);
    p[13] = new Point(x1, y1-ht);
    no_node=4;
    if(type==7)
    {
    node[0]=p[0];
    node[1]=p[1];
    node[2]=p[12];
    node[3]=p[13];
    }
    if(type==8)
    {
    node[0]=p[10];
    node[1]=p[11];
    node[2]=p[12];
    node[3]=p[13];
    }
    no_pt=14;

}

if(type >= 9 && type <= 10)
{
```

```
    p[0] = new Point(x1-w, y1-ht/3);
    p[1] = new Point(x1-w, y1+ht/3);
    p[2] = new Point(x1-w/2, y1-ht/2);
    p[3] = new Point(x1-w/2, y1-ht/3);
    p[4] = new Point(x1-w/2, y1+ht/3);
    p[5] = new Point(x1-w/2, y1+ht/2);
    p[6] = new Point(x1+w/2, y1-ht/2);
    p[7] = new Point(x1+w/2, y1-ht/3);
    p[8] = new Point(x1+w/2, y1+ht/3);
    p[9] = new Point(x1+w/2, y1+ht/2);
    p[10] = new Point(x1+w, y1-ht/3);
    p[11] = new Point(x1+w, y1+ht/3);
    p[18] = new Point(x1, y1+ht);
    p[19] = new Point(x1, y1-ht);

    if(type == 9)
    {
        p[12]=new Point(x1-w-w/4,y1-ht/3-ht/4);
        p[13]=new Point(x1-w-w/4,y1+ht/3-ht/4);
        p[14]=new Point(x1-w-w/4-w/2,y1-ht/3-ht/4+ht/6);
        p[15]=new Point(x1-w-w/4-w/4,y1-ht/3-ht/4+ht/3);
        p[16]=new Point(x1-w-w/4-w/2,y1+ht/3-ht/4+ht/6);
        p[17]=new Point(x1-w-w/4-w/4,y1+ht/3-ht/4+ht/3);
        no_node=8;
    node[0]=p[14];
    node[1]=p[15];
    node[2]=p[16];
    node[3]=p[17];
    node[4]=p[10];
    node[5]=p[11];
    node[6]=p[18];
    node[7]=p[19];
    }

    if(type == 10)
    {
        p[12]=new Point(x1+w,y1-ht/3-ht/4);
        p[13]=new Point(x1+w,y1+ht/3-ht/4);
        p[14]=new Point(x1+w+w/4+w/2,y1-ht/3-ht/4+ht/6);
        p[15]=new Point(x1+w+w/4+w/4,y1-ht/3-ht/4+ht/3);
        p[16]=new Point(x1+w+w/4+w/2,y1+ht/3-ht/4+ht/6);
        p[17]=new Point(x1+w+w/4+w/4,y1+ht/3-ht/4+ht/3);
        no_node=8;
    node[0]=p[14];
    node[1]=p[15];
    node[2]=p[16];
    node[3]=p[17];
    node[4]=p[0];
    node[5]=p[1];
    node[6]=p[18];
    node[7]=p[19];
    }
    no_pt=20;

}
```

```
    if(type == 11)
    {
        p[0] = new Point(x1-w, y1-ht/3);
        p[1] = new Point(x1-w, y1+ht/3);
        p[2] = new Point(x1-w/2, y1-ht/2);
        p[3] = new Point(x1-w/2, y1-ht/3);
        p[4] = new Point(x1-w/2, y1+ht/3);
        p[5] = new Point(x1-w/2, y1+ht/2);
        p[6] = new Point(x1+w/2, y1-ht/2);
        p[7] = new Point(x1+w/2, y1-ht/3);
        p[8] = new Point(x1+w/2, y1+ht/3);
        p[9] = new Point(x1+w/2, y1+ht/2);
        p[10] = new Point(x1+w, y1-ht/3);
        p[11] = new Point(x1+w, y1+ht/3);
        no_node=4;
        node[0]=p[0];
        node[1]=p[1];
        node[2]=p[10];
        node[3]=p[11];
        no_pt=12;
    }

    if(type == 12)
    {
        p[0] = new Point(x1-w, y1-ht/3);
        p[1] = new Point(x1-w, y1+ht/3);
        p[2] = new Point(x1-w/2, y1-ht/2);
        p[3] = new Point(x1-w/2, y1-ht/3);
        p[4] = new Point(x1-w/2, y1+ht/3);
        p[5] = new Point(x1-w/2, y1+ht/2);
        p[6] = new Point(x1+w/2, y1-ht/2);
        p[7] = new Point(x1+w/2, y1-ht/3);
        p[8] = new Point(x1+w/2, y1+ht/3);
        p[9] = new Point(x1+w/2, y1+ht/2);
        p[10] = new Point(x1+w, y1-ht/3);
        p[11] = new Point(x1+w, y1+ht/3);
        p[12] = new Point(x1, y1+ht);
        p[13] = new Point(x1, y1-ht);
        no_node=6;
        node[0]=p[0];
        node[1]=p[1];
        node[2]=p[12];
        node[3]=p[13];
        node[4]=p[10];
        node[5]=p[11];
        no_pt=14;
    }

    centre = new Point(x1, y1);
}

int no_node()
{
    return no_node;
}
```

```
void draw(Graphics g1)
{
    Graphics2D g=(Graphics2D) g1;
    Stroke d= new BasicStroke(2);
    g.setStroke(d);
    if(type >= 1 && type <= 6)
    {
    g.drawRect(p[2].x,p[2].y, w, ht);
    g.setColor(fill);
    g.fillRect(p[2].x,p[2].y, w, ht);
    g.setColor(Color.BLACK);
    g.drawLine(p[0].x,p[0].y,p[3].x,p[3].y);
    g.drawLine(p[1].x,p[1].y,p[4].x,p[4].y);
    g.drawLine(p[7].x,p[7].y,p[10].x,p[10].y);
    g.drawLine(p[8].x,p[8].y,p[11].x,p[11].y);

    if(type==1)
    {

        g.drawRect(p[12].x, p[12].y, w/4, ht/2);
        g.drawRect(p[13].x, p[13].y, w/4, ht/2);
        g.setColor(fill);
        g.fillRect(p[12].x, p[12].y, w/4, ht/2);
        g.fillRect(p[13].x, p[13].y, w/4, ht/2);
        g.setColor(Color.BLACK);
        g.drawLine(p[14].x,p[14].y,p[12].x, p[14].y);
        g.drawLine(p[15].x,p[15].y,p[12].x, p[15].y);
        g.drawLine(p[16].x,p[16].y,p[13].x, p[16].y);
        g.drawLine(p[17].x,p[17].y,p[13].x, p[17].y);

    }
    if(type==2)
    {
        g.drawRect(p[12].x, p[12].y, w/4, ht/2);
        g.drawRect(p[13].x, p[13].y, w/4, ht/2);
         g.setColor(fill);
       g.fillRect(p[12].x, p[12].y, w/4, ht/2);
        g.fillRect(p[13].x, p[13].y, w/4, ht/2);
        g.setColor(Color.BLACK);
        g.drawLine(p[14].x,p[14].y,p[12].x+w/4, p[14].y);
        g.drawLine(p[15].x,p[15].y,p[12].x+w/4, p[15].y);
        g.drawLine(p[16].x,p[16].y,p[13].x+w/4, p[16].y);
        g.drawLine(p[17].x,p[17].y,p[13].x+w/4, p[17].y);

    }

    if(type == 3)
    {

        g.drawRect(p[12].x, p[12].y, w/4, ht/2);
        g.drawRect(p[13].x, p[13].y, w/4, ht/2);
         g.setColor(fill);
        g.fillRect(p[12].x, p[12].y, w/4, ht/2);
        g.fillRect(p[13].x, p[13].y, w/4, ht/2);
        g.setColor(Color.BLACK);
        g.drawLine(p[14].x,p[14].y,p[12].x, p[14].y);
```

```
    g.drawLine(p[15].x,p[15].y,p[12].x, p[15].y);
    g.drawLine(p[16].x,p[16].y,p[13].x, p[16].y);
    g.drawLine(p[17].x,p[17].y,p[13].x, p[17].y);
    }

if(type == 4)
{

    g.drawRect(p[12].x, p[12].y, w/4, ht/2);
    g.drawRect(p[13].x, p[13].y, w/4, ht/2);
     g.setColor(fill);
    g.fillRect(p[12].x, p[12].y, w/4, ht/2);
    g.fillRect(p[13].x, p[13].y, w/4, ht/2);
    g.setColor(Color.BLACK);
    g.drawLine(p[14].x,p[14].y,p[12].x+w/4, p[14].y);
    g.drawLine(p[15].x,p[15].y,p[12].x+w/4, p[15].y);
    g.drawLine(p[16].x,p[16].y,p[13].x+w/4, p[16].y);
    g.drawLine(p[17].x,p[17].y,p[13].x+w/4, p[17].y);
    }

if(type == 5)
{

    g.drawRect (p[12].x, p[12].y, w/4, ht/2);
    g.drawRect (p[13].x, p[13].y, w/4, ht/2);
     g.setColor(fill);
    g.fillRect(p[12].x, p[12].y, w/4, ht/2);
    g.fillRect(p[13].x, p[13].y, w/4, ht/2);
    g.setColor(Color.BLACK);
    g.drawLine(p[14].x,p[14].y,p[12].x, p[14].y);
    g.drawLine(p[15].x,p[15].y,p[12].x, p[15].y);
    g.drawLine(p[16].x,p[16].y,p[13].x, p[16].y);
    g.drawLine(p[17].x,p[17].y,p[13].x, p[17].y);
    g.drawLine(p[18].x,p[18].y,p[12].x, p[18].y);
    g.drawLine(p[19].x,p[19].y,p[12].x, p[19].y);
    g.drawLine(p[20].x,p[20].y,p[13].x, p[20].y);
    g.drawLine(p[21].x,p[21].y,p[13].x, p[21].y);
   }

if(type == 6)
{

    g.drawRect(p[12].x, p[12].y, w/4, ht/2);
    g.drawRect(p[13].x, p[13].y, w/4, ht/2);
     g.setColor(fill);
     g.fillRect(p[12].x, p[12].y, w/4, ht/2);
    g.fillRect(p[13].x, p[13].y, w/4, ht/2);
    g.setColor(Color.BLACK);
    g.drawLine(p[14].x,p[14].y,p[12].x+w/4, p[14].y);
    g.drawLine(p[15].x,p[15].y,p[12].x+w/4, p[15].y);
    g.drawLine(p[16].x,p[16].y,p[13].x+w/4, p[16].y);
    g.drawLine(p[17].x,p[17].y,p[13].x+w/4, p[17].y);
    g.drawLine(p[18].x,p[18].y,p[12].x+w/4, p[18].y);
    g.drawLine(p[19].x,p[19].y,p[12].x+w/4, p[19].y);
    g.drawLine(p[20].x,p[20].y,p[13].x+w/4, p[20].y);
    g.drawLine(p[21].x,p[21].y,p[13].x+w/4, p[21].y);
```

```
    }
}
if(type >= 7 && type <= 8)
{
    g.drawRect(p[2].x,p[2].y, w, ht);
     g.setColor(fill);
    g.fillRect(p[2].x,p[2].y, w, ht);
    g.setColor(Color.BLACK);
    g.drawLine(p[12].x, p[12].y, centre.x, p[5].y);
    g.drawLine(p[13].x, p[13].y, centre.x, p[2].y);

    if(type == 7)
    {
        g.drawLine(p[0].x,p[0].y,p[3].x,p[3].y);
        g.drawLine(p[1].x,p[1].y,p[4].x,p[4].y);
    }

    if(type == 8)
    {
        g.drawLine(p[7].x,p[7].y,p[10].x,p[10].y);
        g.drawLine(p[8].x,p[8].y,p[11].x,p[11].y);
    }
}

if(type >= 9 && type <= 10)
{
    g.drawRect(p[2].x,p[2].y, w, ht);
    g.drawRect(p[12].x, p[12].y, w/4, ht/2);
    g.drawRect(p[13].x, p[13].y, w/4, ht/2);
     g.setColor(fill);
     g.fillRect(p[2].x,p[2].y, w, ht);
    g.fillRect(p[12].x, p[12].y, w/4, ht/2);
    g.fillRect(p[13].x, p[13].y, w/4, ht/2);
    g.setColor(Color.BLACK);
    g.drawLine(p[0].x,p[0].y,p[3].x,p[3].y);
    g.drawLine(p[1].x,p[1].y,p[4].x,p[4].y);
    g.drawLine(p[7].x,p[7].y,p[10].x,p[10].y);
    g.drawLine(p[8].x,p[8].y,p[11].x,p[11].y);
    g.drawLine(p[18].x, p[18].y, centre.x, p[5].y);
    g.drawLine(p[19].x, p[19].y, centre.x, p[2].y);
        if(type == 9)
        {

            g.drawLine(p[14].x,p[14].y,p[12].x, p[14].y);
            g.drawLine(p[15].x,p[15].y,p[12].x, p[15].y);
            g.drawLine(p[16].x,p[16].y,p[13].x, p[16].y);
            g.drawLine(p[17].x,p[17].y,p[13].x, p[17].y);
            }

    if(type == 10)
    {
            g.drawLine(p[14].x,p[14].y,p[12].x+w/4, p[14].y);
            g.drawLine(p[15].x,p[15].y,p[12].x+w/4, p[15].y);
            g.drawLine(p[16].x,p[16].y,p[13].x+w/4, p[16].y);
            g.drawLine(p[17].x,p[17].y,p[13].x+w/4, p[17].y);
     }
```

```
    }

    if(type == 11)
    {
        g.drawRect(p[2].x,p[2].y, w, ht);
        g.setColor(fill);
        g.fillRect(p[2].x,p[2].y, w, ht);
        g.setColor(Color.BLACK);
    g.drawLine(p[0].x,p[0].y,p[3].x,p[3].y);
    g.drawLine(p[1].x,p[1].y,p[4].x,p[4].y);
    g.drawLine(p[7].x,p[7].y,p[10].x,p[10].y);
    g.drawLine(p[8].x,p[8].y,p[11].x,p[11].y);
    }

    if(type == 12)
    {
    g.drawRect(p[2].x,p[2].y, w, ht);
    g.setColor(fill);
    g.fillRect(p[2].x,p[2].y, w, ht);
    g.setColor(Color.BLACK);
    g.drawLine(p[0].x,p[0].y,p[3].x,p[3].y);
    g.drawLine(p[1].x,p[1].y,p[4].x,p[4].y);
    g.drawLine(p[7].x,p[7].y,p[10].x,p[10].y);
    g.drawLine(p[8].x,p[8].y,p[11].x,p[11].y);
    g.drawLine(p[12].x, p[12].y, centre.x, p[5].y);
    g.drawLine(p[13].x, p[13].y, centre.x, p[2].y);
    }
    for(int i=0;i<no_node;i++)
    {
        g.setColor(Color.BLACK);
        g.drawOval(node[i].x-3,node[i].y-2,6,4);
    }

}

void del(Graphics g1,Color c)
{
    Graphics2D g=(Graphics2D) g1;
    Stroke d= new BasicStroke(2);
    g.setStroke(d);
    g.setColor(c);
    if(type >= 1 && type <= 6)
    {
    g.drawRect(p[2].x,p[2].y, w, ht);
    g.fillRect(p[2].x,p[2].y, w, ht);
    g.drawLine(p[0].x,p[0].y,p[3].x,p[3].y);
    g.drawLine(p[1].x,p[1].y,p[4].x,p[4].y);
    g.drawLine(p[7].x,p[7].y,p[10].x,p[10].y);
    g.drawLine(p[8].x,p[8].y,p[11].x,p[11].y);

    if(type==1)
    {

        g.drawRect(p[12].x, p[12].y, w/4, ht/2);
        g.drawRect(p[13].x, p[13].y, w/4, ht/2);
        g.fillRect(p[12].x, p[12].y, w/4, ht/2);
```

```
    g.fillRect(p[13].x, p[13].y, w/4, ht/2);
    g.drawLine(p[14].x,p[14].y,p[12].x, p[14].y);
    g.drawLine(p[15].x,p[15].y,p[12].x, p[15].y);
    g.drawLine(p[16].x,p[16].y,p[13].x, p[16].y);
    g.drawLine(p[17].x,p[17].y,p[13].x, p[17].y);

}
if(type==2)
{
    g.drawRect(p[12].x, p[12].y, w/4, ht/2);
    g.drawRect(p[13].x, p[13].y, w/4, ht/2);
   g.fillRect(p[12].x, p[12].y, w/4, ht/2);
    g.fillRect(p[13].x, p[13].y, w/4, ht/2);
    g.drawLine(p[14].x,p[14].y,p[12].x+w/4, p[14].y);
    g.drawLine(p[15].x,p[15].y,p[12].x+w/4, p[15].y);
    g.drawLine(p[16].x,p[16].y,p[13].x+w/4, p[16].y);
    g.drawLine(p[17].x,p[17].y,p[13].x+w/4, p[17].y);

}

if(type == 3)
{

    g.drawRect(p[12].x, p[12].y, w/4, ht/2);
    g.drawRect(p[13].x, p[13].y, w/4, ht/2);
    g.fillRect(p[12].x, p[12].y, w/4, ht/2);
    g.fillRect(p[13].x, p[13].y, w/4, ht/2);
    g.drawLine(p[14].x,p[14].y,p[12].x, p[14].y);
    g.drawLine(p[15].x,p[15].y,p[12].x, p[15].y);
    g.drawLine(p[16].x,p[16].y,p[13].x, p[16].y);
    g.drawLine(p[17].x,p[17].y,p[13].x, p[17].y);
    }

if(type == 4)
{

    g.drawRect(p[12].x, p[12].y, w/4, ht/2);
    g.drawRect(p[13].x, p[13].y, w/4, ht/2);
    g.fillRect(p[12].x, p[12].y, w/4, ht/2);
    g.fillRect(p[13].x, p[13].y, w/4, ht/2);
    g.drawLine(p[14].x,p[14].y,p[12].x+w/4, p[14].y);
    g.drawLine(p[15].x,p[15].y,p[12].x+w/4, p[15].y);
    g.drawLine(p[16].x,p[16].y,p[13].x+w/4, p[16].y);
    g.drawLine(p[17].x,p[17].y,p[13].x+w/4, p[17].y);
    }

if(type == 5)
{

    g.drawRect(p[12].x, p[12].y, w/4, ht/2);
    g.drawRect(p[13].x, p[13].y, w/4, ht/2);
    g.fillRect(p[12].x, p[12].y, w/4, ht/2);
    g.fillRect(p[13].x, p[13].y, w/4, ht/2);
    g.drawLine(p[14].x,p[14].y,p[12].x, p[14].y);
    g.drawLine(p[15].x,p[15].y,p[12].x, p[15].y);
    g.drawLine(p[16].x,p[16].y,p[13].x, p[16].y);
```

```
    g.drawLine(p[17].x,p[17].y,p[13].x, p[17].y);
    g.drawLine(p[18].x,p[18].y,p[12].x, p[18].y);
    g.drawLine(p[19].x,p[19].y,p[12].x, p[19].y);
    g.drawLine(p[20].x,p[20].y,p[13].x, p[20].y);
    g.drawLine(p[21].x,p[21].y,p[13].x, p[21].y);
   }

if(type == 6)
{

    g.drawRect(p[12].x, p[12].y, w/4, ht/2);
    g.drawRect(p[13].x, p[13].y, w/4, ht/2);
     g.fillRect(p[12].x, p[12].y, w/4, ht/2);
    g.fillRect(p[13].x, p[13].y, w/4, ht/2);
    g.drawLine(p[14].x,p[14].y,p[12].x+w/4, p[14].y);
    g.drawLine(p[15].x,p[15].y,p[12].x+w/4, p[15].y);
    g.drawLine(p[16].x,p[16].y,p[13].x+w/4, p[16].y);
    g.drawLine(p[17].x,p[17].y,p[13].x+w/4, p[17].y);
    g.drawLine(p[18].x,p[18].y,p[12].x+w/4, p[18].y);
    g.drawLine(p[19].x,p[19].y,p[12].x+w/4, p[19].y);
    g.drawLine(p[20].x,p[20].y,p[13].x+w/4, p[20].y);
    g.drawLine(p[21].x,p[21].y,p[13].x+w/4, p[21].y);
    }
}
if(type >= 7 && type <= 8)
{
    g.drawRect(p[2].x,p[2].y, w, ht);
    g.fillRect(p[2].x,p[2].y, w, ht);
    g.drawLine(p[12].x, p[12].y, centre.x, p[5].y);
    g.drawLine(p[13].x, p[13].y, centre.x, p[2].y);

    if(type == 7)
    {
        g.drawLine(p[0].x,p[0].y,p[3].x,p[3].y);
        g.drawLine(p[1].x,p[1].y,p[4].x,p[4].y);
    }

    if(type == 8)
    {
        g.drawLine(p[7].x,p[7].y,p[10].x,p[10].y);
        g.drawLine(p[8].x,p[8].y,p[11].x,p[11].y);
    }
}

if(type >= 9 && type <= 10)
{
    g.drawRect(p[2].x,p[2].y, w, ht);
    g.drawRect(p[12].x, p[12].y, w/4, ht/2);
    g.drawRect(p[13].x, p[13].y, w/4, ht/2);
     g.fillRect(p[2].x,p[2].y, w, ht);
    g.fillRect(p[12].x, p[12].y, w/4, ht/2);
    g.fillRect(p[13].x, p[13].y, w/4, ht/2);
    g.drawLine(p[0].x,p[0].y,p[3].x,p[3].y);
    g.drawLine(p[1].x,p[1].y,p[4].x,p[4].y);
    g.drawLine(p[7].x,p[7].y,p[10].x,p[10].y);
    g.drawLine(p[8].x,p[8].y,p[11].x,p[11].y);
```

```
        g.drawLine(p[18].x, p[18].y, centre.x, p[5].y);
        g.drawLine(p[19].x, p[19].y, centre.x, p[2].y);
            if(type == 9)
            {

                g.drawLine(p[14].x,p[14].y,p[12].x, p[14].y);
                g.drawLine(p[15].x,p[15].y,p[12].x, p[15].y);
                g.drawLine(p[16].x,p[16].y,p[13].x, p[16].y);
                g.drawLine(p[17].x,p[17].y,p[13].x, p[17].y);
                }

        if(type == 10)
        {
                g.drawLine(p[14].x,p[14].y,p[12].x+w/4, p[14].y);
                g.drawLine(p[15].x,p[15].y,p[12].x+w/4, p[15].y);
                g.drawLine(p[16].x,p[16].y,p[13].x+w/4, p[16].y);
                g.drawLine(p[17].x,p[17].y,p[13].x+w/4, p[17].y);
         }
    }

    if(type == 11)
    {
        g.drawRect(p[2].x,p[2].y, w, ht);
        g.fillRect(p[2].x,p[2].y, w, ht);
    g.drawLine(p[0].x,p[0].y,p[3].x,p[3].y);
    g.drawLine(p[1].x,p[1].y,p[4].x,p[4].y);
    g.drawLine(p[7].x,p[7].y,p[10].x,p[10].y);
    g.drawLine(p[8].x,p[8].y,p[11].x,p[11].y);
    }

    if(type == 12)
    {
    g.drawRect(p[2].x,p[2].y, w, ht);
    g.fillRect(p[2].x,p[2].y, w, ht);
    g.drawLine(p[0].x,p[0].y,p[3].x,p[3].y);
    g.drawLine(p[1].x,p[1].y,p[4].x,p[4].y);
    g.drawLine(p[7].x,p[7].y,p[10].x,p[10].y);
    g.drawLine(p[8].x,p[8].y,p[11].x,p[11].y);
    g.drawLine(p[12].x, p[12].y, centre.x, p[5].y);
    g.drawLine(p[13].x, p[13].y, centre.x, p[2].y);
    }
    for(int i=0;i<no_node;i++)
    {
        g.drawOval(node[i].x-3,node[i].y-2,6,4);
    }


}

Point[] getPoints()
{
    return node;
}
Point getPoint(int i)
{
    return node[i];
```

```
}



int getX()
{
    int min=9999;
    for(int i=0;i<no_pt;i++)
        if(p[i].x<min)
        {
            min=p[i].x;
        }
    return (min);
}
int getY()
{
    int min=9999;
    for(int i=0;i<no_pt;i++)
        if(p[i].y<min)
        {
            min=p[i].y;
        }
    return (min);
}
void setH(int h)
{
    ht=h;
}
void setW(int w)
{
    this.w=w;
}
int getH()
{
    int min=9999;
    for(int i=0;i<no_pt;i++)
        if(p[i].y<min)
        {
            min=p[i].y;
        }
    int max=-1;
    for(int i=0;i<no_pt;i++)
        if(p[i].y>max)
        {
            max=p[i].y;
        }
    return (max-min);
}
int getW()
{
    int min=9999;
    for(int i=0;i<no_pt;i++)
        if(p[i].x<min)
        {
            min=p[i].x;
        }
```

```
    int max=-1;
    for(int i=0;i<no_pt;i++)
        if(p[i].x>max)
        {
            max=p[i].x;
        }
    return (max-min)/2;
}
}
```

# 9. References


- Research paper on fast interconnection.(Author Ravi Rastogi and Dr. Nitin)
- http://www.researchgate.net/publication/220419528_Fast_Interconnections_A_Case_Tool_for_Developing_Fault-tolerant_Multi-stage_Interconnection_Networks
- http://en.wikipedia.org/wiki/Network_topology
- http://research.ijcaonline.org/volume43/number21/pxc3878691.pdf(Author-Shubhangi D Chawade, Mahendra A Gaikwad, Rajendra M Patrikar)
- http://www.cs.mcgill.ca/~cs535/lect_notes/Lecture13-InterconNetworks.pdf
- http://www.csc.ncsu.edu/faculty/efg/506/f07/docs/lecture_notes/lec24.pdf
- http://pnewman.com/papers/thesis/chapter4.pdf
- http://research.ijcaonline.org/volume43/number21/pxc3878691.pdf