ADDIS ABABA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

FACULTY OF TECHNOLOGY

A survey on VLSI routing-algorithms and implementation

BY

Fasil W/Gebriel Belay

September, 2006

ADDIS ABABA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

FACULTY OF TECHNOLOGY

A survey on VLSI routing-algorithms and implementation

A thesis submitted to the School of Graduate Studies of Addis Ababa University in partial fulfillment for the Degree of Master of Science in Computer Engineering

By

Fasil W /Gebriel Belay

Advisor

Professor Santtanam. A

and

Dr. GebreAmanuel Gessese

Addis Ababa, Ethiopia

September, 2006

ADDIS ABABA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

FACULTY OF TECHNOLOGY

A survey on VLSI routing-algorithms and implementation

BY

Fasil W/Gebriel Belay

Approval by Board of Examiners

Dr. Getachew Biru	_____
Chairman, Department of Electrical and Computer Engineering	Signature
Professor Santtanam. A	_____
Advisor	Signature
_____	_____
External Examiner	Signature
_____	_____
Internal Examiner	Signature

Acknowledgments

My first and foremost gratitude goes to my advisors Dr. GebreAmanuel Gessess and Professor Santhanam .A for their tireless efforts in giving an exhaustive guidance through the thesis development. My special appreciation goes to Dr. GebreAmanuel Gessess, for his voluntary advisory regardless of the geographical distance between us. As the subject matter of the thesis is different from what I have studied in my stay in the postgraduate studies, professor Santtanam has assisted me in understanding important microelectronics concepts related to my thesis work. Therefore I would like to offer my thankfulness once again to Professor Santtanam .A.

I am also very much grateful to my colleagues for their continued assistance and encouragement which has been a great contribution in bringing the thesis work to come to its final stage. Especially I would like to express my thankfulness to W/t Etagegn, my colleague, for editing important images in the thesis manuscript and for her encouragement when I found the going through the process tough.

Fasil w/gebriel

Addis Ababa University

September, 2006

Table of contents

Table of contents ..... **i**

Acknowledgments ..... **iii**

Abstract **1**

Thesis overview **2**

Chapter one **3**

Introduction 3

1.1 Very Large Scale Integration (VLSI) technology **3**

1.2 VLSI Design Flow **6**

1.3 VLSI design Automation **7**

Chapter two **8**

Thesis background 8

2.1 VLSI physical design **8**

2.1.1 Partitioning 10

2.1.2 Placement 10

2.1.3 Routing 11

2.1.4 Compaction 12

2.2 Physical Design Styles **12**

2.2.1 Full Custom 12

2.2.2 Standard Cell 14

2.2.3 Gate array 15

2.2.4 Field Programmable Gate Array (FPGA) 16

2.3 VLSI Routing **17**

2.3.1 Defining the problem 17

2.3.2 Routing Models 20

Chapter three **21**

Switchbox Routing Problems in depth 21

3.1 Definition **21**

3.2 Single Row Routing **22**

3.3 Channel Routing **24**

3.3.1 Horizontal Constraints 26

3.3.2 Vertical Constraints 27
3.3.3 Classification of channel routing algorithms 28
3.3.4 Channel routing algorithms based on the strategy they use. 29
3.3.5 Channel Routing algorithms based on the routing models used 33
3.4 Gamma Routing 35
3.5 General Switchbox Routing 35
Chapter four 38
Switchbox Routing In Manhattan Model 38
4.1 The Square-Shaped Switchbox 38
4.2 The Switchbox of Arbitrary Shape 43
Chapter five 44
Maze routing algorithms 44
5.1 Lee's Algorithm 44
5.2 Soukup's Algorithm 47
Chapter six 50
Implementation of Routing Algorithms 50
6.1 Channel routing algorithm 51
6.1.1 Left edge channel routing algorithm 51
6.1.2 Methodology used in the implementation of single row routing and channel routing algorithms 54
6.1.3 Left edge algorithm applied to single row and channel routing problems 58
6.1.4 Implementation results of benchmark problems 62
6.2 Maze routing algorithm 64
6.2.1 Methodology used in the implementation of Maze routing algorithm (Lee's maze routing algorithm) 65
6.2.2 Single layer maze routing 68
6.2.3 Multilayer maze routing 70
6.3 A Two way Solution to channel routing problems 73
6.3.1 Vertical constraint cycle in left edge channel routing algorithm. 74
6.3.2 Channel routing problem with vertical constraint cycle 75
6.3.3 Two way solution in detail 77
6.3.4 Another benchmark for the two way solution 81
6.3.5 Comparison of results of routing algorithms 86
References 90

Abstract

VLSI (Very Large Scale Integration) physical design automation has been an area of great interest for the researchers in integrated circuit design. In earlier times physical design was done manually. The manual generation of layout designs has proved to be backbreaking, time consuming and error prone. In this thesis a rigorous study of VLSI switchbox routing algorithms and their implementation is done to evaluate the performance of these algorithms and to propose an algorithm which provides a better solution for some of the benchmark problems. A java implementation provides a graphical display of the result of execution of an algorithm under investigation and it also provides the number of tracks and vias a given problem requires to complete the routing.

Thesis overview

This section presents the organization of the thesis. In chapter one I have considered three important concepts as introduction to my thesis. The introduction begins with the new trends in Very Large Scale Integration (VLSI) technology, followed by a discussion on VLSI design flow which presents a series of steps an engineered VLSI design and production process should pass through in order to manufacture the desired chip. The last concept I have discussed in chapter one is VLSI design automation where I have mentioned the significance of VLSI physical design automation tools in facilitating the most difficult and tedious layout generation processes. Chapter two has been organized as the background for the rest of the thesis. I have selected three relevant issues in this chapter. The first issue is a detailed discussion on the physical design of VLSI systems. Physical design styles have been considered as an important scheme to handle the complex nature of the physical design phase. I have specially used the standard cell design style for channel routing problems. The last issue in chapter two is VLSI routing where I gave an introductory idea to VLSI routing, classification of VLSI routing in to general routing and detailed routing is also discussed in this chapter. In chapter three different classes of switchbox routers are discussed. In this chapter, switchbox routers are categorized as single row routers, channel routers and general switchbox routers. Special emphasis is also given in the same chapter for channel routers and general switchbox routers. In chapter four specific switchbox routers like square shaped switchbox routers and arbitrary shaped switchbox routers are discussed. Chapter five discusses about some maze routing algorithms in relation to detailed VLSI routing. A maze routing algorithm is used in this thesis as a clean up router after the main router completes its task. In this respect I have obtained optimal solutions to channel routing problems by using the two way solution strategy (main router followed by maze router). Chapter six discusses the implementation aspects of this thesis. Chapter seven contains the conclusion and recommendations for future work

Chapter one

Introduction

1.1 Very Large Scale Integration (VLSI) technology

The birth of electronics dates back 100 years and is marked by the invention of the triode in 1907, as indicated in **[1].** Since then the electronics industry has transformed the life style of nations through progressive innovations made by different scientists around the world. The industry has achieved quite an unusual accomplishment in the last two decades, mainly due to advanced technology in integrated circuit (**IC**) design and manufacturing. Applications of VLSI systems span from house hold electronics appliances to sophisticated telecommunication, and control systems. The field of microelectronics is reaching its peak by the high performance requirements imposed by applications that make use of VLSI systems. Figure- 1.1, taken from Mlynek **[2],** gives an overview of the important trends in information technologies. According to Mlynek **[2]** *"The current leading-edge technologies (such as low bit-rate video and cellular communications) already provide the end-users a certain amount of processing power and portability*". With the rapid development of VLSI systems design this trend is expected to continue in the years to come. Information services of our day are characterized by a very high need for processing power and bandwidth (scarce resource) in order to support real-time audio/video applications. An ideal characteristic of the current day information processing systems is their personalized nature, which imposes additional requirements on them. These devices are expected to have better intelligence to deal with individual demands, and at the same time be small enough to allow a high degree of mobility as in the case of cellular systems **[2] [4]**.

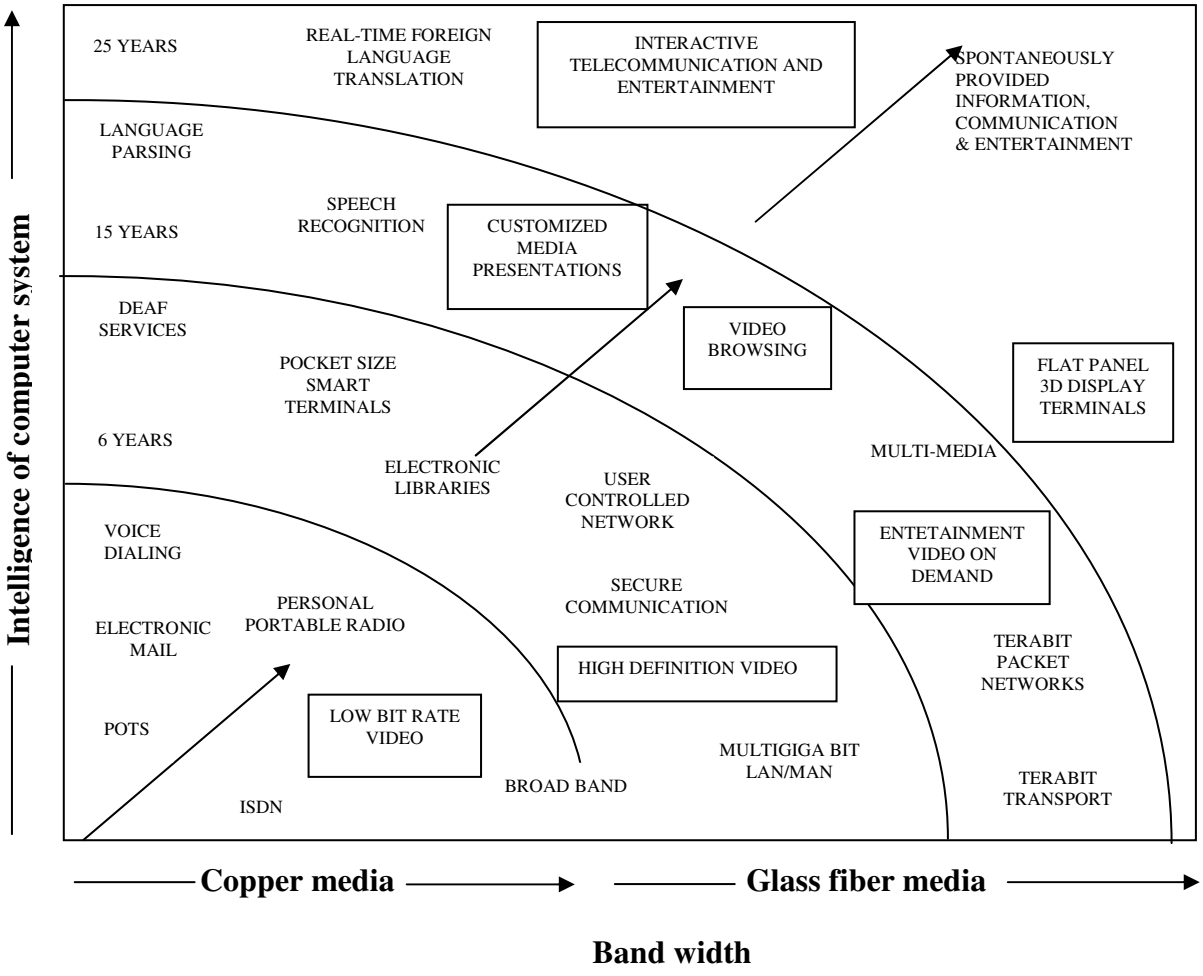


Figure-1.1 prominent trends in information service technologies. **Source Bellcore**

Current day technology is imposing stringent requirements on devices required for various data processing and communication purposes to support complex functions within very small space area. According to Mlynek **[2]**, *"The level of integration as measured by the number of logic gates in a monolithic chip has been steadily rising for almost three decades, mainly due to the rapid progress in integration technology and interconnects technology"*. In the same writing **[2]**, a logic block (gate) has been defined to consist of 10 to 100 transistors, depending on their function. According to a report in 2004 Intel, one of the largest microprocessor producing companies introduced Pentium 4 processor with processing speed of 3.4 GHz and transistor count of 55 million. **[4]**

Category (Number of logic blocks per chip)	**Time**	**Size**
Unit logic (one gate)	1960	1
Multi-function	1962	2 - 4
Complex function	1964	5 - 20
Medium Scale Integration (MSI)	1967	20 - 200
Large Scale Integration (LSI)	1972	200 - 2000
Very Large Scale Integration (VLSI)	1978	2000-20000
Ultra Large Scale Integration (ULSI)	1989	20000 - ?

Table-1.1: *Evolution of logic complexity in integrated circuits.* **[2]**

The most important inference that can be made from table-1.1 is that there is an exponential increase in the logic complexity per chip in each era. The massive integration of a large number of functions on a single chip usually provides the following advantages as indicated in **[2].**

- Less area or volume and therefore, ***high level of miniaturization.***
- Low power ***consumption***-this is becoming an important area of research because increased power consumption threatens Moore's law prediction.
- Higher ***reliability***, basically due to improved on-chip interconnects.
- Higher ***speed***, due to high level of ***proximity*** among interconnected devices.
- Remarkable ***cost reduction.***

Current day progress in sophisticated device manufacturing technology and especially the steady reduction of miniaturization size supports high degree of integration level. Generally, for the same chip size the transistor count of logic chips is significantly fewer than those of memory chips. In logic chips much of the area is taken by the complex interconnects. Memory chips however, are highly regular reducing chip area required for interconnection thus more number of transistors can be integrated in these chips.

1.2 VLSI Design Flow

As any engineered system, design flow for integrated circuits starts with a given set of requirements. A starting design is developed and checked against the requirements. When requirements are not met, the design has to be improved iteratively until the desired requirements are met. If achieving the desired requirement becomes difficult or becomes costly, then revision of requirements should be done to come up with an optimal design. **[2]**

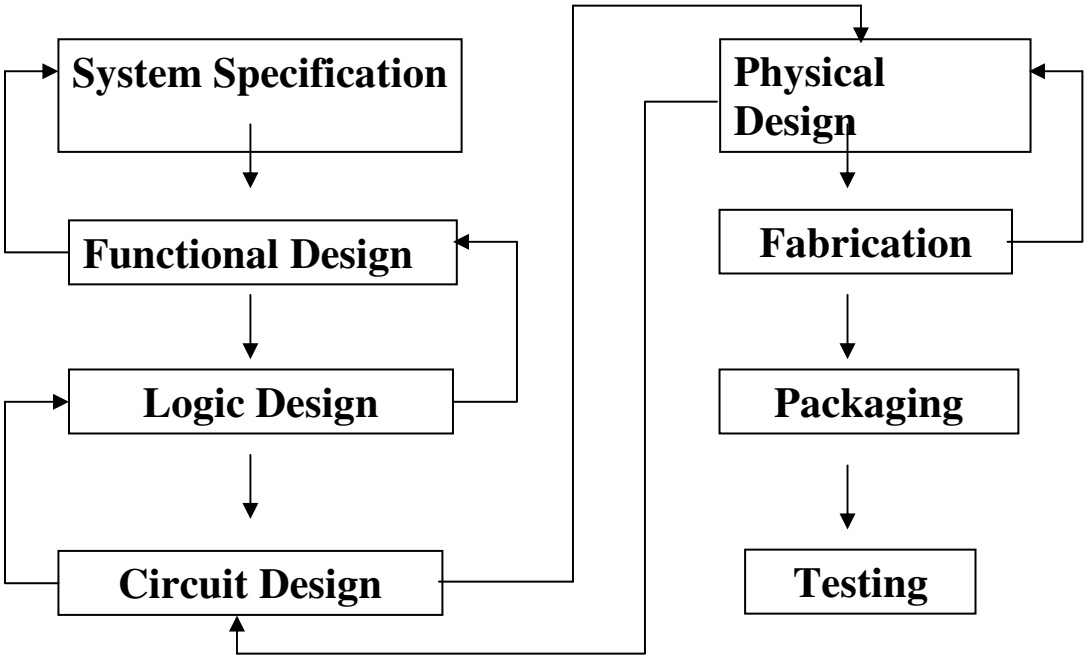


Figure-1.2: VLSI design flow.

Figure-1.2 shows VLSI design flow, where feed back mechanisms are supplied in the flow in order to make sure that in each and every stage specific requirements are well met. For example after functional design stage is completed, the result has to be checked against all the requirements in the system specification. Basically the feed back paths help to enforce verification mechanisms along the paths of the design flow. One very important scheme in such engineering designs is the necessity of inserting verification mechanisms at the early phases of the design. This helps to localize any kind of design

flaws at the early stages of the design before they are propagated to the latter stages where identification and maintenance of such flaws becomes too costly and time consuming.

1.3 VLSI design Automation

Naveed Sherwani **[3]** describes the significant achievement in VLSI design as *"VLSI design has brought the power of the large sized computers like mainframe computers to the Laptop"*. As one can easily realize such a tremendous growth is only possible by the development of sophisticated design tools and software. Sherwani argues, "*To deal with the complexity of the millions of components and to achieve a reasonable response time, VLSI tools must not only be computationally fast but also perform close to optimal"*.

The research and development of VLSI design automation tools are determinant to the future advances in VLSI systems. The central concern of VLSI design automation is basically the research and development of algorithms and data structures related to the VLSI design processes. The general objective of the automation is to produce an optimized arrangement of devices in a chip (e.g. optimization against the amount of area required to place and route the devices) and to ensure an efficient interconnection among these devices.

Naveed Sherwani describes **[3]** *"Since space on a wafer is a very expensive real state, algorithms must use the space very efficiently to lower costs and improve yield"*. In addition, the arrangement of devices plays a key role in determining the performance of a chip. *"Algorithms for physical design must ensure that the layout generated abides by all the rules required by the fabrication process"*.

Chapter two

Thesis background

2.1 VLSI physical design

VLSI technology has well influenced every one living in the 21st century. Almost all digital equipments around us consist of VLSI chips. Hence an optimized design of VLSI devices is becoming an important issue for VLSI manufacturers. One of the important steps in manufacturing VLSI chips is its physical design. "*The input to the physical design is a logical representation of the system under design. The out put of this step is the lay out of the physical package that optimally or near-optimally realizes the logical representation"*. Figure-2.1 shows a typical physical design process of VLSI chips diagram from **[8]**.

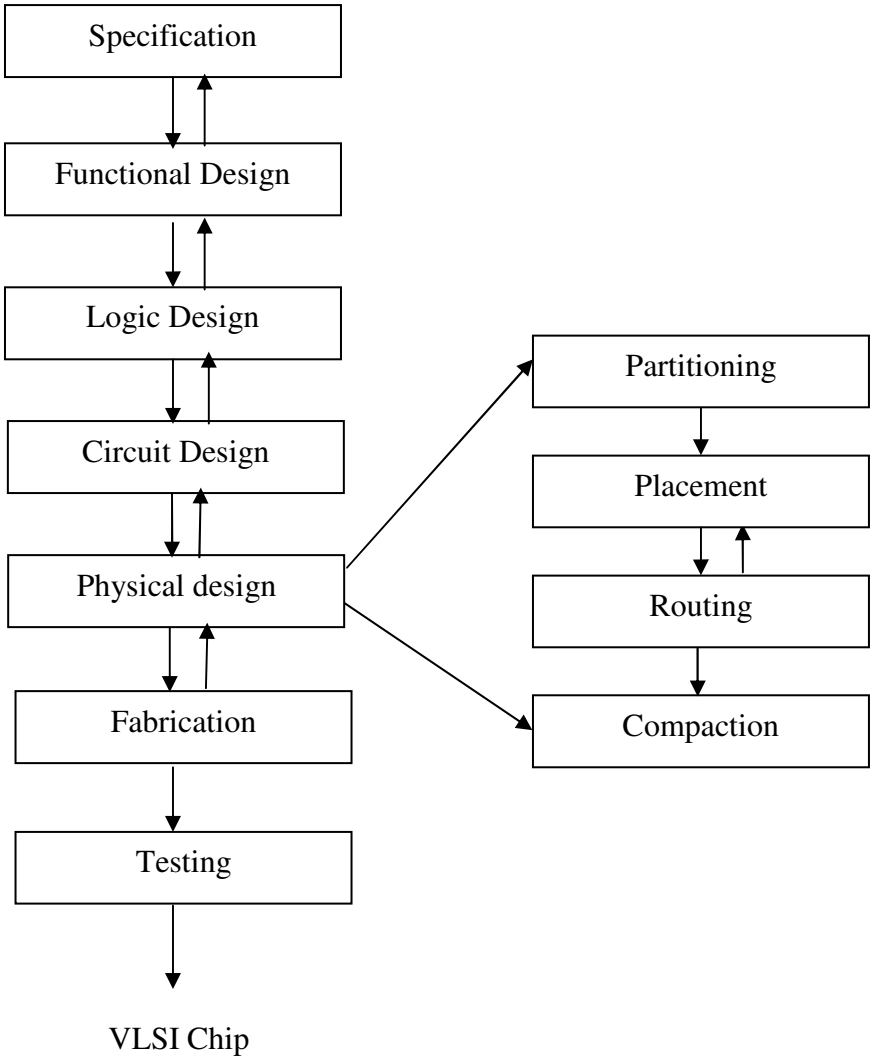


Figure-2.1 The design process of VLSI chips

It is not difficult to understand from figure 2.1 that physical design of VLSI chips involves an iterative or feedback technique to verify whether each goal within each stage of the design process is well met or not.In order to master complexity of the physical design in the design process of VLSI chips, it is a good idea to divide the problem into sub-problems. Physical design problem can be divided into four sub problems as follows: ***Partitioning, placement, routing and compaction.***

2.1.1 Partitioning

One of the most important things to note in dealing with VLSI physical design automation is that the input size to be handled at one time is extremely large. Hence the famous scheme of divide and conquer approach should be employed in order to minimize the input to a size that could be suitably handled with computer programs. The first step in VLSI physical design, partitioning, ensures the division of circuits into smaller parts. The circuit is often divided into portions that are implemented separately. The goal is to partition the circuit such that the sizes of the parts are within prescribed ranges and the complexity of the interconnection between these parts is minimized. In a complex VLSI chip the process of partitioning is hierarchical. At the top level the chip can be divided into relatively larger blocks and these blocks could also be divided recursively in the remaining hierarchies **[8].**

2.1.2 Placement

Next to partitioning phase comes placement stage where cells of the circuit are assigned to their geometrical locations on the chip *("a cell may be a single transistor, an adder or sub circuit, etc"*). According to Sherwani **[3]**, *"The goal in placement is to minimize the total lay out area of the chip that allows completion of interconnections between cells while meeting the performance constraints"*. The quality of the placement phase is determined not only by the end of the placement phase itself but also by the end of the next phase, routing. This is so because some placement techniques may lead to design placements that are very difficult or impossible to route. In that case another iteration of the placement algorithm may be necessary. To limit the number of iterations of the placement algorithm, an estimate of the required routing space is used during the placement phase. The effectiveness of routing results is highly dependent on the effectiveness of the placement algorithms.

2.1.3 Routing

The third phase in the VLSI physical design is the routing phase. Routing can be defined in simple terms as the process of connecting terminals of nets subject to a set of routing constraints. This phase contains the main focus of my thesis. According to Sherwani **[3],** VLSI routing is divided into two:

1. Global routing
2. Detailed routing

1. ***Global routing*** is concerned with finding approximate interconnection paths in the different routing regions provided.

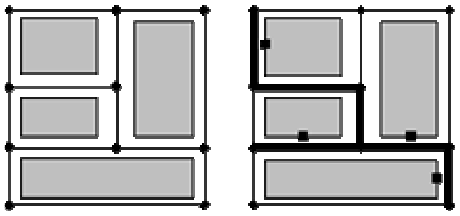


Figure-2.2 A routing graph ***(a)*** *and a global route for a four terminal net* ***(b)***

2. ***Detailed routing*** is concerned with wiring of interconnections in to exact positions inside a routing region. In the detailed routing phase the exact routes for the wires have to be determined.

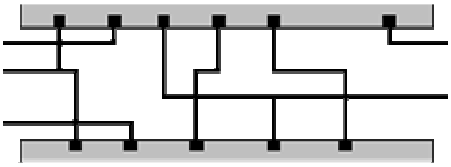


Figure-2.3 The detailed routing inside a channel

The overall goal of the routing phase is to complete all routing, subject to certain quality constraints like:

- Minimizing the lengths of the interconnections
- Avoiding overlapping of interconnection wires

2.1.4 Compaction

Compaction is the last step in the physical layout design. The overall objective of compaction phase is to minimize (compress) the layout in all direction to reduce the total area without violation of fabrication constraints **[8]**.

Similar to all engineered systems an optimized design of each phase in the physical design of VLSI systems determines the efficiency of the subsequent phases.

2.2 Physical Design Styles

According to Sherwani **[3]**, *"market requirements demand quick time-to-market and high yield, as a result, restricted models and design styles are used in order to reduce the complexity of physical design"*. Design styles can be broadly classified as either full custom or semi custom. In a full custom layout, different blocks of a circuit can be placed at any location on a silicon wafer. Sherwani describes, *"Selection of a layout style depends on many factors including the type of chip, cost and time to market. Full custom layout is a preferred style for mass produced chips, since the time required to produce a highly optimized layout can be justified. On the other hand, to design an Application Specific Integrated Circuit (ASIC), a semi-custom layout is usually preferred"*.

2.2.1 Full Custom

The full custom design allows the flexibility for functional blocks to have different sizes. One very distinctive characteristics of the full custom design style is that blocks can be placed at any location on the chip irregularly as shown in figure 2.4. Due to the absence of constraint, full custom layout style results in a very compact design. However,

computer automation of full custom designs is more difficult than other restricted models **[6]**. For this reason this design style is only appropriate whenever the final design must have minimum area and design time is not considered. *"Some phases of physical design of a full-custom chip may be done manually to optimize the layout"*.

One can identify two important points while investigating full custom design style:

1. In placing the various blocks of the chip, there should be an optimized allocation of chip area to each block being placed.
2. In doing the first task however, there must be enough space left between the blocks so that it is always possible to complete routing within the space allocated for routing.

Full custom design is time consuming; thus the method is inappropriate especially for very large circuits, unless chip size is of great importance. **[3]**

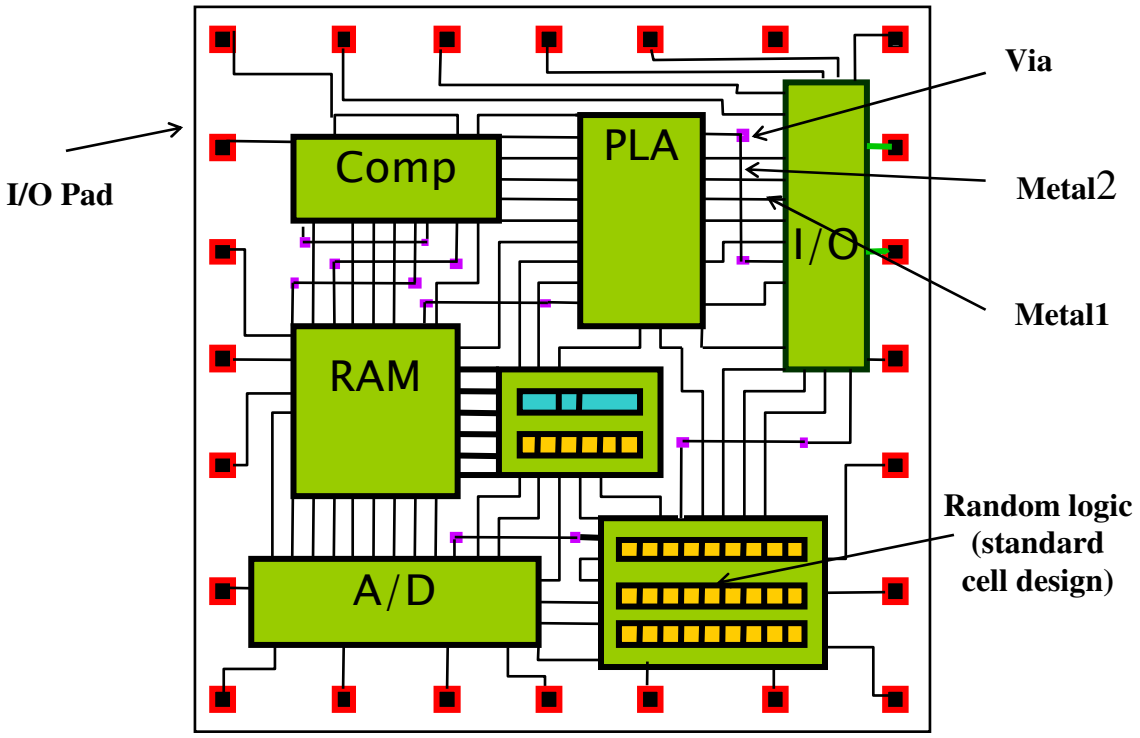


Figure-2.4 full custom design

2.2.2 Standard Cell

The full custom design style is the most difficult design style to automate. Standard cell layout style, owing to its regular nature, provides a relatively easy layout as compared to full custom layout. The design process in the standard cell design is relatively simpler than full custom design style **[7]**. Important steps in the standard cell design style are outlined below **[3]**.

1. According to some predefined cell, a given circuit is divided into several smaller blocks.
2. Analysis, test and specification of each predefined cell are done in order to conform to the appropriate functionality and electrical characteristics.
3. A collection of cells which satisfy the conditions in 1 and 2 forms a cell library.

According to Sherwani **[3]** *"a cell library consists of 500-1200 cells"*. As shown in figure 2.5 blocks are placed in rows and the space between the two rows forms a channel for interconnection. The usual scenario for routing is to use the channel for the interconnection of terminals in the same row or in adjacent rows. Another scenario for routing is when terminals to be connected are on two non-adjacent rows. In this scenario there must be empty blocks or gaps in the cells that can be used for routing in the intermediate rows. Such empty blocks are called feed-throughs.

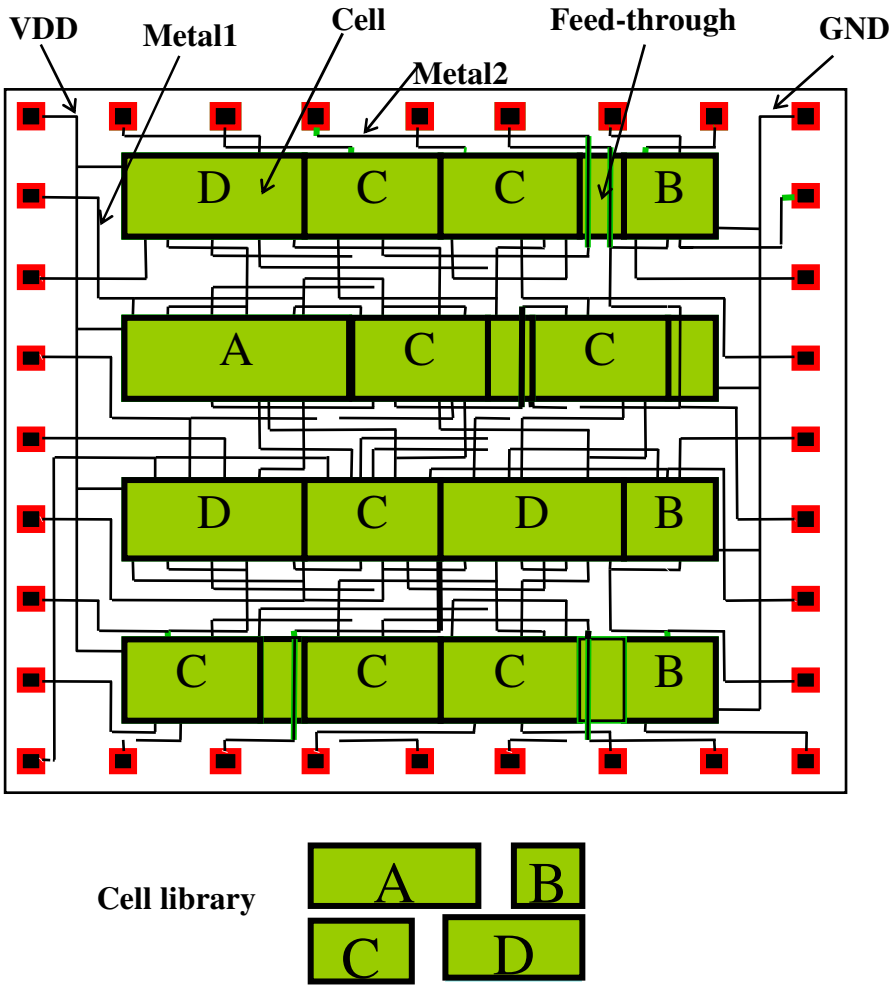


Figure-2.5 Standard cell design

2.2.3 Gate array

Gate array design (uncommitted logic array) style is created in order to simplify the standard cell design. The gate array design is used in the manufacturing of application specific integrated circuits (ASICS). *"Unlike standard cell design, all cells in gate array design are identical",* Sherwani explains the advantage of gate arrays, *"the steps involved for creating any prefabricated gate array wafer are the same and only the last few steps in the fabrication process actually depend on the application for which the design will be used"*. The gate array design is one of the most restricted design styles, which is very simple for automation purposes as shown in figure-2.6.

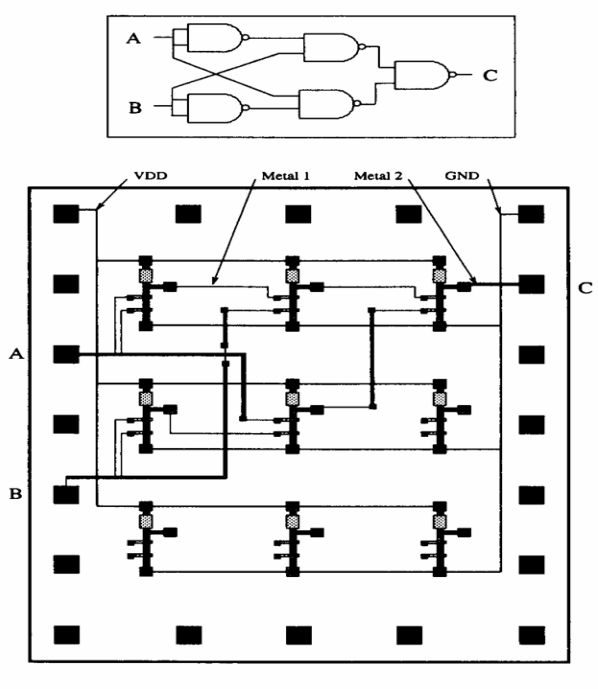


Figure-2.6 Gate array design **[3]**

2.2.4 Field Programmable Gate Array (FPGA)

Field programmable gate array (FPGA) is a design style used to manufacture semiconductor device containing programmable logic components and programmable interconnects. FPGA (field programmable gate array) chips are available these days in order to implement desired functionality on field by custom hardware programming **[2]**. *"This design style provides a means for fast prototyping and also for cost-effective chip design, especially for low-volume applications"*. Typical field programmable gate array (FPGA) architecture consists of an array of configurable logic blocks (CLBs), input output pads, routing channels and programmable interconnects.

"The programming of interconnects is implemented by programming of RAM cells whose output terminals are connected to the gates of MOS pass transistors". A general architecture of FPGA is shown in Figure-2.7.

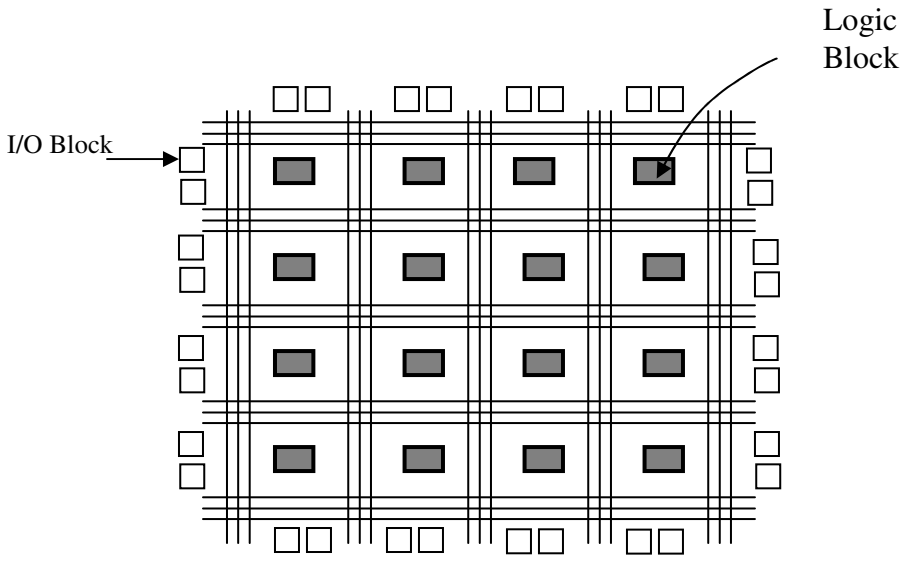


Figure-2.7 General Architecture of FPGA **[2]**

2.3 VLSI Routing

2.3.1 Defining the problem

Definition of VLSI routing as described by David Szsezler in his PhD dissertation **[5]** is as follows, "*A set of circuit elements is given; each circuit element has a few terminals (or pins). Furthermore, a description of the circuit to be designed is also given, which is nothing else but a list of pair-wise disjoint subsets of terminals, called nets."*

The ultimate purpose of routing is creating an interconnection among the terminals of same nets by wires such that the circuit elements and the interconnecting wires are embedded into the plane. Besides achieving a complete routing some additional fabrication requirements should also be met. The most important of these requirements is that a minimum distance has to be maintained between any two distinct wires. The easiest and most common way to achieve this is that the wires must conform to the edges of a given rectangular grid.

The range of cost functions that need to be minimized in various routing problems include:

1. Area, that is, the number of grid vertices on a layer. This basically affects the amount of miniaturization possible in manufacturing VLSI chips.
2. Total wire length to be used for interconnection.
3. Number of vias used to make interlayer connections.

In this thesis, these are the major cost functions used to compare the different algorithms in order to evaluate their performance.

Routing takes the third step in the overall physical design process of VLSI systems. Preceding physical design phases have a significant effect in the subsequent phases. VLSI Placement is dependent on VLSI partitioning step and VLSI routing is dependent on VLSI placement phase. In the placement phase, the exact locations of circuit blocks and pins are determined as discussed in section 2.1. Required interconnections are specified using a net list. Spaces not occupied by the blocks can be viewed as a collection of routing regions. Nets are always routed within the routing regions. In addition contact between wires of different nets must be avoided.

Figure 2.8 presents a diagrammatic description of the two-phase routing (general routing and detailed routing) flow explaining the detailed activities involved in each phase.

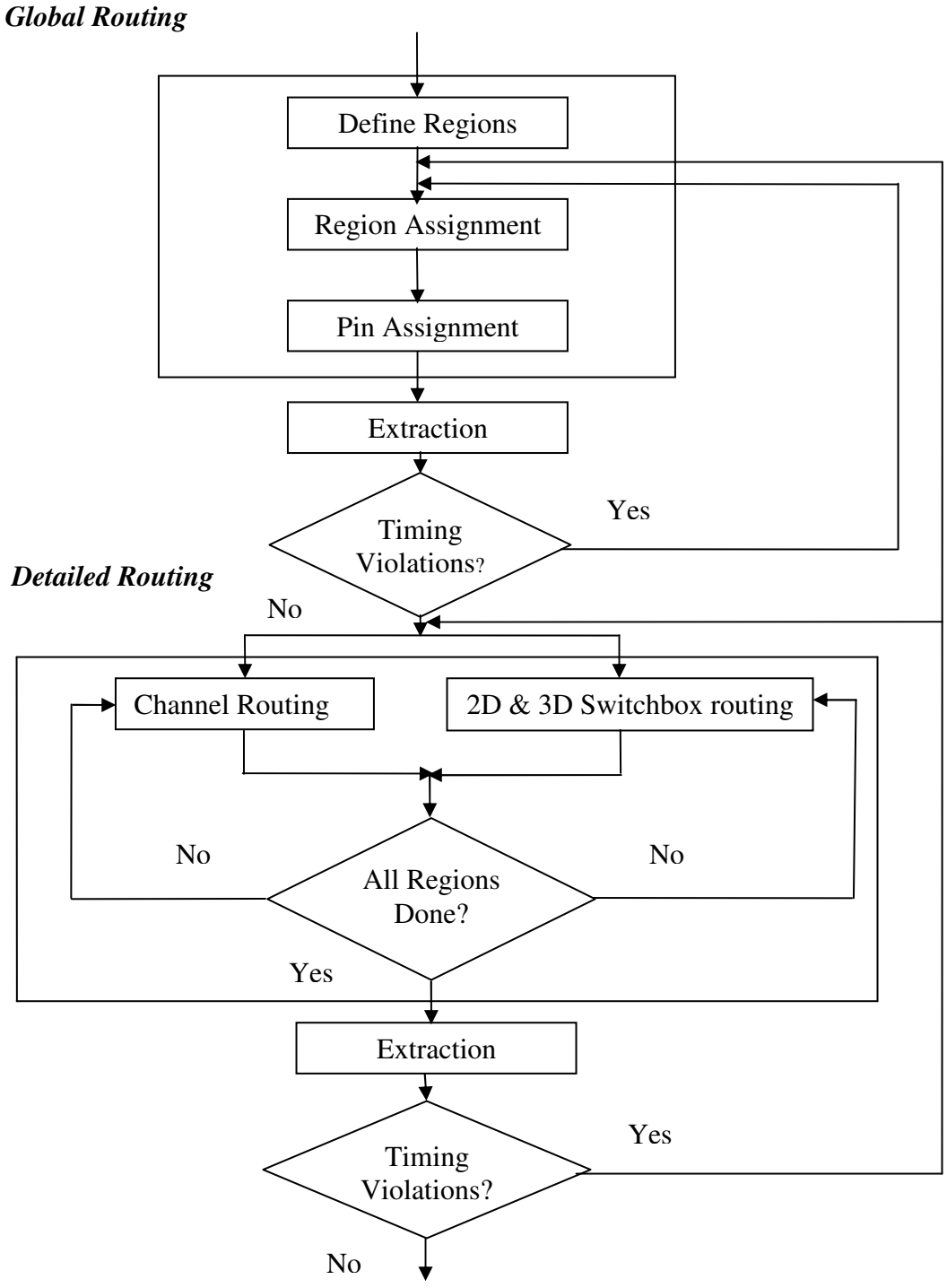


Figure-2.8 The two phase routing flow **[3]**

2.3.2 Routing Models

2.3.2.1 Grid based model

Generally two routing models are available: grid-based and gridless model. In grid-based model, routing wires are restricted to follow paths along the grid lines. Other components of routing such as terminals, vias are also required to conform to the edges of the routing grid **[11]**.

2.3.2.2 Gridless model

Any approach that doesn't follow the grid-based approach is categorized as a gridless model. This approach allows arbitrary location of terminals, nets and vias. The griddles approach is relatively difficult to handle using computer programs **[11]**.

Chapter three

Switchbox Routing Problems in depth

3.1 Definition

Andras Recski et al **[13]** defines a switchbox *as "a rectangular grid* ***G*** *consisting of horizontal tracks (numbered from* ***0*** *to* ***w+1****) and vertical columns (numbered from* ***0*** *to* ***h+1****), where* ***w*** *is the width and* ***h*** *is the height of the switchbox. Whenever* ***h= w*** *the switchbox is called square shaped switchbox".*

Based on the boundary of the switchbox on which the different terminals of a net are located; they are called *Northern*, *Southern*, *Eastern* or *Western terminals of a net*. The corners of the switchbox are not considered as terminals; therefore they are not used for routing.

In the paper by Andras Recski et al **[13]** a net has been defined as:

"A net is a collection of terminals. A switchbox routing problem is a set $N = \{N_1 \ldots N_n\}$ *terminals of pair-wise disjoint nets."*

According to the above definition solutions of a routing problem consists of nets whose edges don't contact with one another, i.e only terminals of the same net should be connected with each other and the solution has to make sure that all nets are completely disjoint. In the literatures surveyed two types of definitions exist regarding switchbox routing problems. In some literatures **[3], [11], [16]** switchbox routing problem only refers to a routing where terminals of nets are placed on all four sides. In other literatures **[13]**, **[5]** it is a more general routing problem consisting of many different subclasses.

This thesis adopts the second definition given by Andras Recski et al. He divides the general switchbox routing problem into five subclasses as follows.

1. Single Row Routing problems, in these types of problems terminals of the nets are located on one straight line; terminals can take any one of the four boundaries.
2. Channel Routing problems, in these types of problems terminals of nets take any two parallel boundaries of the four boundaries, i.e. either the northern and southern boundaries (called horizontal channels) or the western and the eastern boundaries (called vertical channels).
3. Γ-(Gamma) Routing problems, in these types of problems terminals of the nets can be located in any two adjacent boundaries. The name gamma refers only to one variety of the problems, for example we can have an L-shape routing problem in this category and also others.
4. C–Routing problems, in these types of problems terminals of nets are located on any three boundaries. For example, northern, western and southern boundaries make the C-shape routing.
5. General Switchbox problems, this category refers to a switchbox routing problems where terminals of nets are located in all the four boundaries of a box.

3.2 Single Row Routing

According to David Szeszeler's definition **[5]** single row routing is defined as *"a special case of the switchbox routing problem in which all the terminals of each net are located, say, on northern or southern or eastern or western boundaries. Given a set of two-terminal or multi-terminal nets defined on a set of evenly spaced terminals on a line, the single row routing problem is to realize the interconnection of the nets by means of non-crossing paths"*. Gallai's linear time algorithm constitutes the first classic algorithm in the topic of VLSI routing **[14]** that solves the single row routing problem with optimal width in the 2-layer Manhattan model. In Manhattan model routing layers contain either horizontal or vertical wires not both. According to David Szeszeler's definition **[5],** *"for every vertical line e that cuts the grid into two its congestion is defined as c (e): it is the*

number of nets that are divided into two by e (that is, the number of nets that have terminals both left to e and right to e)". Figure 3.1 presents a single row routing problem from **[5].** The congestion of line e in the same figure is c (e) = 3. The density of the problem is defined as the maximum of the congestion lines that could be drawn along the routing region. The density of the routing provides the lower bound required for the minimum width required for routing as indicated in the following theorem.

T. Gallai **[14]** has proved a theorem, which states **"***the minimum width of a solution of a single row routing problem in the 2-layer Manhattan model is equal to the density of the problem. Moreover, such a minimum width routing can be found in linear time"*.

A single row routing problem is shown in Figure-3.1. The interval graph of Figure-3.2 corresponds to the routing problem of Figure-3.1. This graph is colored using three colors. The solution of the routing problem obtained from this coloring is shown in Figure-3.1. Solid dots in the figure show terminals to be connected and empty dots show vias used for interlayer connection. **[5]**

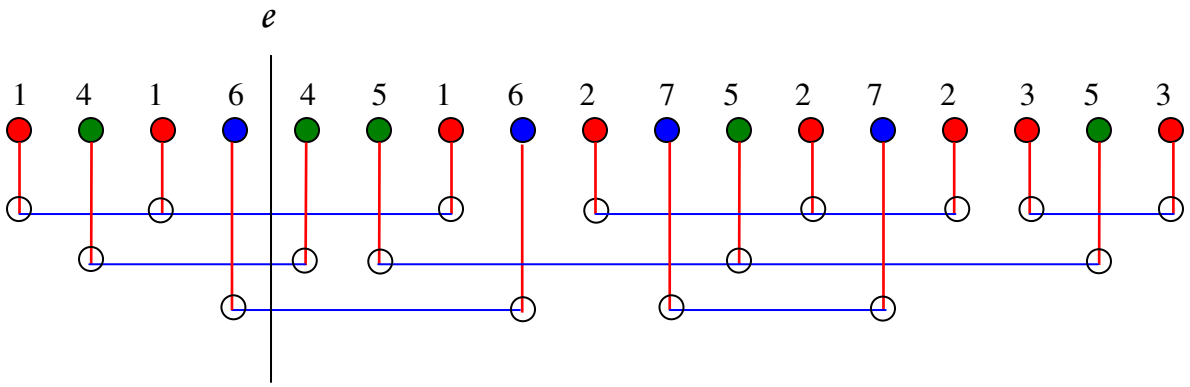


Figure-3.1 single row routing

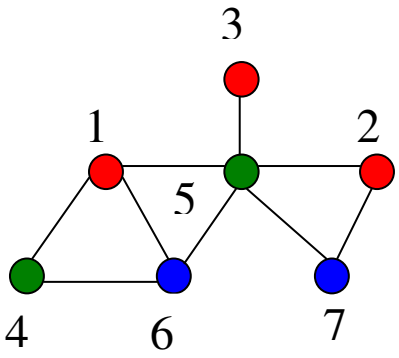


Figure-3.2 interval graph of Figure-3.1

3.3 Channel Routing

According to David Szeszeler's definition **[5]** channel routing is defined as *"a special case of the switchbox routing problem in which all the terminals of each net are situated on two opposite boundaries of the grid (say, they are all northern or southern). Hence a channel is a routing region bounded by two parallel rows of terminals"*. Each terminal is assigned a number or a symbol, which represents the net to which the terminal belongs. The horizontal dimension (sometimes measured in terms of number of columns used in the routing region) of a channel is called the channel length and the vertical dimension (sometimes measured in terms of number of tracks used in the routing region) of the channel is called the channel height. Important terminologies related to channel routing from **[3], [11]** have been include. *"Horizontal segment of net is called a trunk and vertical segments that connect the trunk to the terminals are called branches. The horizontal line along which the trunk lies is called a track. A dogleg is a vertical segment that is used to connect two trunks of a net on two different tracks"*, Figure-3.3 shows channel routing terminologies.

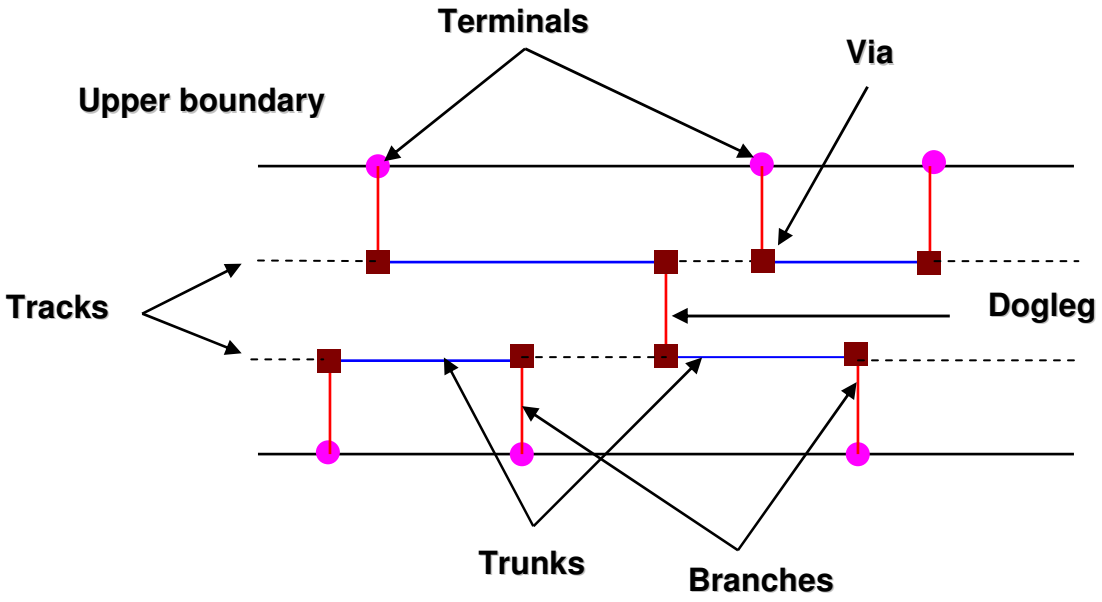


Figure-3.3 Channel routing terminologies

The main objective of a channel routing problem is to minimize the channel height **[3], [5], [14]**. Additional objectives include minimizing the total number of vias used in multilayer routing and minimizing the length of any particular net **[3]**.

Generally Grid-based routing involves assignment of horizontal segments of nets to tracks, as shown in figure 3.3. The horizontal segments of nets are located on the tracks of the routing region. Vertical segments are used to connect horizontal segments of the same net if they occupy different tracks and to connect the terminals of nets to the horizontal segments. There are two key constraints, which must be satisfied while assigning the horizontal and vertical segments **[3], [11]**.

3.3.1 Horizontal Constraints

"There is a horizontal constraint between two nets if the trunks of these two nets overlap each other when placed on the same track" **[11]**. A horizontal constraint graph is defined to represent the horizontal constraint in a channel routing problem as shown in figure 3.4.

Horizontal constraint graph (HCG) – each vertex in the HCG corresponds to a net; there is an edge between two vertices if there is a horizontal constraint between corresponding nets.

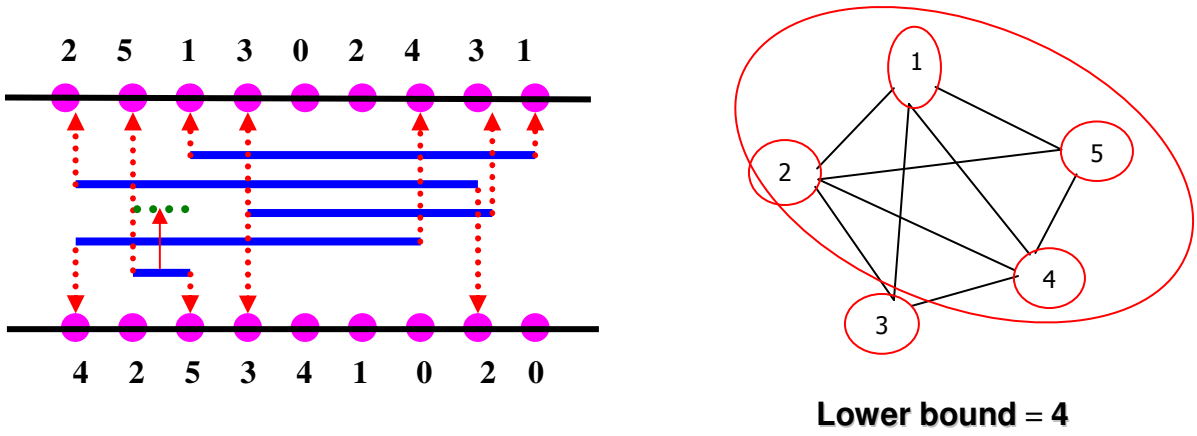


Figure-3.4 horizontal constraint illustration

"In graph theory, a clique in an undirected graph G, is a set of vertices V such that for every two vertices in V, there exists an edge connecting the two". This is equivalent to saying that the sub graph induced by V is a complete graph. The size of a clique is the number of vertices it contains and the maximum clique in the HCG is a lower bound for channel height.

Unfortunately finding cliques within a graph can be a hard problem. Finding the largest clique in any graph for example, is NP-complete. NP-complete problems are problems for which polynomial time algorithm can't be found.

3.3.2 Vertical Constraints

Vertical constraints have been defined in **[3], [11]** as follows. *"A net N_i, in a grid based model, has a vertical constraint with net Nj if there exists a column such that the top terminal of the column belongs to Ni and the bottom terminal belongs to N_j and $i \neq j$"*. In case of the griddles model, the definition of vertical constraint is similar except that the overlap is between the actual vertical segments rather than terminals in a column **[8]**.

***"Vertical Constraint Graph (VCG)**-Given a channel routing problem, a vertical constraint graph (VCG) is a directed graph Gv = (V,Ev), where,*

$Ev = \{(v_i, v_j) \mid N_i$ has vertical constraint with $N_j\}$" **[11].**

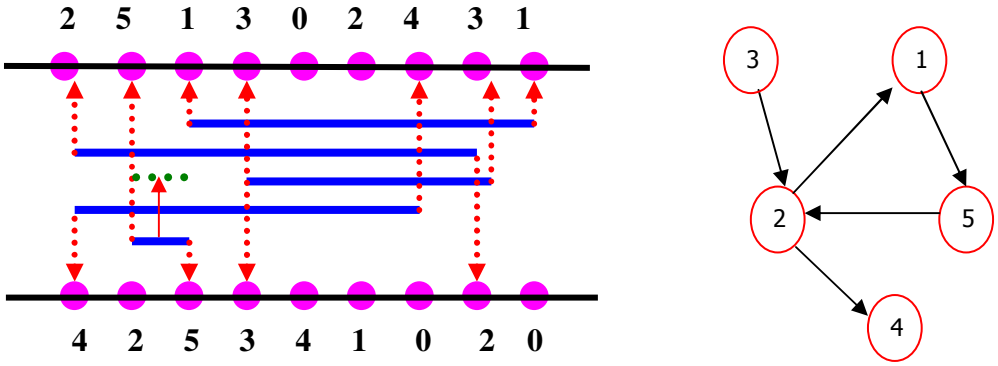
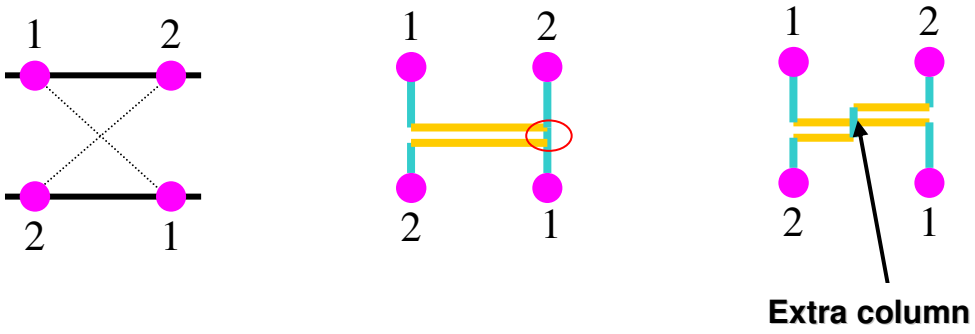


Figure-3.5 vertical constraint illustration



Note: one way of solving vertical constraint problem is by inserting additional track and column as shown in Figure-3.6.

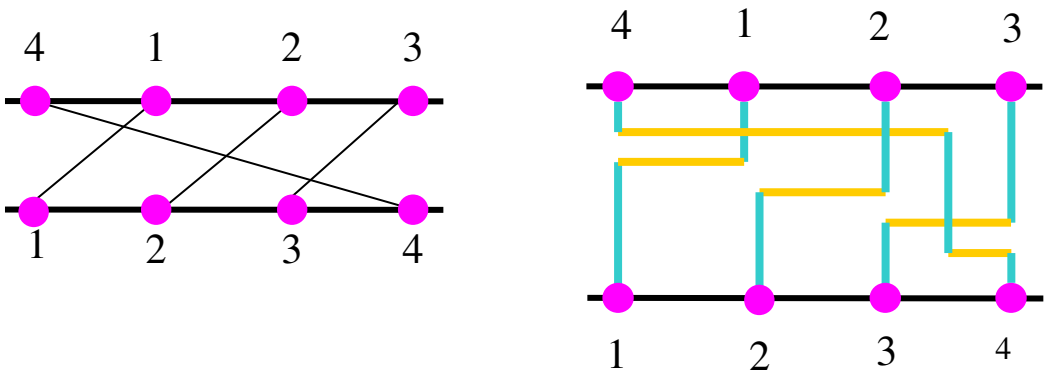


Figure-3.6 solution to vertical constraint

Note: One cyclic constraint → require one track and a column to solve the constraint

3.3.3 Classification of channel routing algorithms

There are many ways of classifying the detailed routing algorithms in general and the channel routing algorithms in particular. The algorithms could be classified on the basis of the routing models used. Some routing algorithms use grid-based models while other algorithms use a gridless model. Some algorithms use a reserved layer model while other algorithms use a model without reserved layers. The griddles model is more flexible as all the wires in the design need not have the same widths. In the reserved layer model the routing layers are constrained to contain either horizontal or vertical segments of a wire for multi-terminal layer routing. Another possible classification scheme could be to classify the algorithms based on the strategy they use. In this respect one can have greedy channel router, left edge channel router, hierarchical channel router, etc. Channel routing algorithms could be classified based on the number of layers they involve for routing. In this respect we have single layer routing, two layer routing, three layer routing etc…

3.3.4 Channel routing algorithms based on the strategy they use.

3.3.4.1 Left Edge Algorithm

The left edge algorithm (LEA) was originally proposed by Hashimoto and Stevens, **[15]**. The basic left edge algorithm uses a reserved layer model where routing layers are restricted to contain either horizontal wires or vertical wires (this model is referred to as Manhattan routing model). The basic left edge algorithm is applicable to channel routing problems, which do not require doglegs and are free from vertical constraints. A description of the basic left edge algorithm taken from **[3]** has been presented in figure 3.7 and the detailed illustration of the algorithm has been included in chapter 6.

```
Algorithm LEFT-EDGE (N,I)
Begin
        FORM-INTERVAL (N, I);
        FORM-HCG (I, HCG);
        d=DENSITY (HCG);
        Let T= {T1T2… Td} denote the set of routing
            tracks from top to bottom;
        SORT-INTERVAL (I);
        For i=1 to n do
             For j=1 to d do
                  If  DOES-NOT-OVERLAP(Ii,Tj) then
                      assign interval Ii to Tj;
        For i=1 to n do
        (*connect the vertical segments of net Ni to its*)
        (*horizontal segment *)
        VERTICAL–SEGMENT (left (Ii), left (Ni));
        VERTICAL-SEGMENT (right (Ii), right (Ni));
End.
```

Figure-3.7 Left edge algorithm.

Description of functions and variables in the left edge algorithm of figure 3.7

- INTERVAL (I_i) - is the horizontal distance spanned by each net.
- FORM-INTERVAL (N, I) - forms interval set I= {I_1, I_2,…, I_n} from a set of two terminal nets N.
- DENSITY- calculates the maximum clique size in the horizontal constraint graph (HCG).
- The maximum clique size is a lower bound for the number of track necessary on a given channel routing problem and it is obtained from HCG.
- SORT-INTERVAL-sorts the intervals formed by FORM-INTERVAL in the ascending order of their x-coordinates on their left edge.
- DOES-NOT-OVERLAP- return true if there is no overlap false otherwise. This function is called every time one assigns a trunk to a net to check the availability of a horizontal track on the routing region.
- VERTICAL–SEGMENT- joins vertical segment of a net with its horizontal segment both on the left and right side of the interval of each net.

Figure-3.8 shows a routing produced by LEA, **[3]**. Chapter six describes java implementation of the left edge algorithm. Naveed Sherwani has proved the following theorem.

"*Given a two-layer channel routing problem with no vertical constraints, LEA produces a routing solution with minimum number of tracks".*

The time complexity of the algorithm in figure-3.7 is O (nlogn).

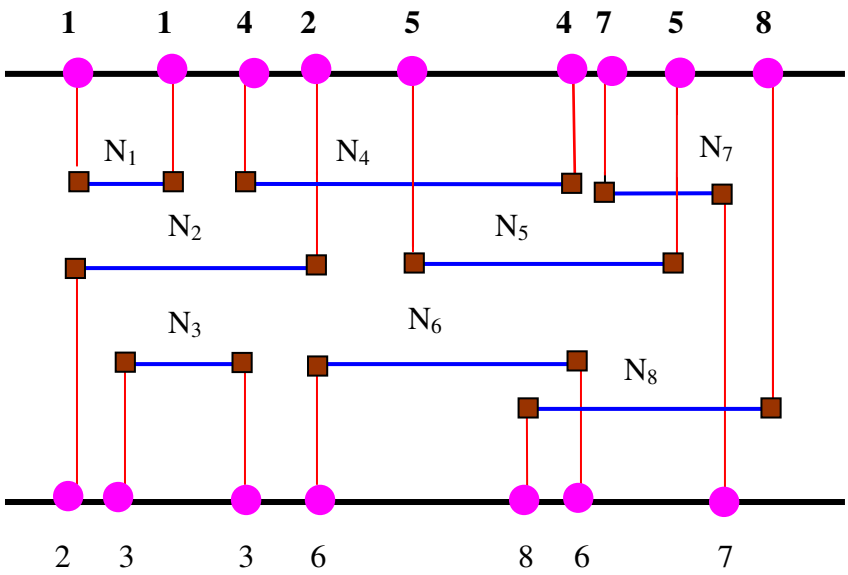


Figure-3.8 Left edge channel routing

The requirement of the basic left edge algorithm that prohibits the situation of two nets sharing the same common end point is very restrictive. Because of the stated restrictions LEA on its own is not practical for most channel routing problems. However LEA can be used as an initial router for routing channels with vertical constraints. One can simply employ LEA knowing that it cannot solve channel routing problems with vertical constraint and deal with the incomplete routing result using other algorithms (maze routing algorithms or rip-up reroute algorithms).

3.3.4.2 Dogleg router

LEA produces a routing with more number of tracks than necessary as it places the entire net on a single track. Figure-3.9 (a) and (b) taken from **[3]**, show a simple channel routing problem solved using LEA and Dogleg router respectively. One can easily observe from figure 3.9 that the dogleg router result in a routing solution with one less number of track than the left edge router.

The reduction in the number of tracks in the dogleg router is due to the possibility of assigning horizontal segment of a net in more than one track; however in the left edge algorithm horizontal segment of a net can only be assigned to a single track. The insertion of doglegs may not necessarily reduce channel density (the maximum clique size in HCG) as a wrongly placed dogleg can lead to an increase in channel density. **[3]**

"Finding the smallest number and locations of doglegs to minimize the channel density is NP complete". **[31]**

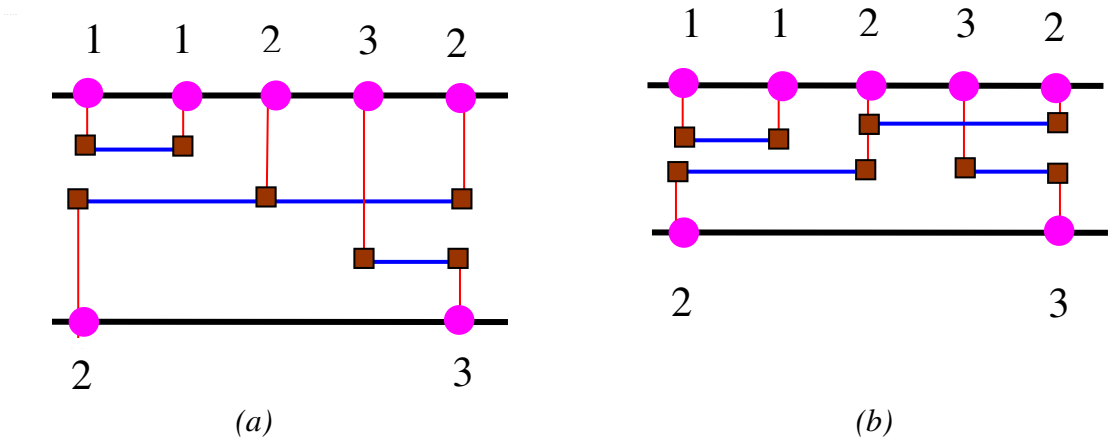


Figure-3.9 (a) LEA router (b) Dogleg router

3.3.4.3 Symbolic channel router: YACR2

The limitation of not allowing vertical constraint cycles makes LEA incapable of solving all channel routing problems. Vertical constraint violation is a localized problem and can be resolved by any one of the two methods as described in **[3].**

1. Local rip-up and reroute.
2. Localized maze routing.

The basic idea of YACR2 is to select nets in an order and assign nets to tracks in such a way that vertical constraint cycle is kept as minimum as possible for each column c_i. In order to minimize vertical constraint cycle, the algorithm does the following in sequence:

1. Assigns tracks to nets belonging to the maximum density column,
2. Uses LEA to assign tracks to nets that are to the right of the maximum density column and
3. Uses LEA to assign tracks to the nets that are to the left of the maximum density column.

After assigning nets to tracks, specialized maze routing techniques, are used to resolve the violations. If a vertical constraint violation cannot be resolved using maze routing technique, additional tracks are used to complete the routing.

3.3.5 Channel Routing algorithms based on the routing models used

3.3.5.1 Two layers Manhattan model channel routing

Manhattan model channel routing is a constrained channel routing where routing layers contain either horizontal or vertical segments. As discussed in single row routing finding the optimum width solution in the 2-layer Manhattan model is always possible in linear time **[5]**. However, in channel routing an optimal linear time solution similar to that of the single row routing may not be possible as described in the following theorem.

David Szeszeler **[5]** has proved the following theorem, *"It is NP-complete to decide whether a channel routing problem is solvable in the 2-layer Manhattan model with width at most w (where w is part of the input)"*

One can try to solve a channel routing problem by assuming the problem as consisting of nets whose terminals are arranged on one row, either on the southern or northern region. The simplest approach to such problem is to use the single row routing algorithm

(Gallai's algorithm) twice. The result of application of Gallia's algorithm is at most twice the optimum width; therefore Gallai's algorithm can't be applied for channel routing problems.

3.3.5.2 Two layers dogleg free Manhattan model

"If each net in the solution of a channel routing problem in the 2-layer Manhattan model contains a single horizontal wire segment only, then the solution is called dogleg free" **[5]**. The importance of such a routing lies in the fact it minimizes the number of necessary vias.

The minimum width solution of the channel routing problem of Figure-3.10 requires 3 tracks for the Manhattan model and 4 tracks for the dogleg free model as described in Figure-3.10 (a) and figure-3.10 (b) respectively.

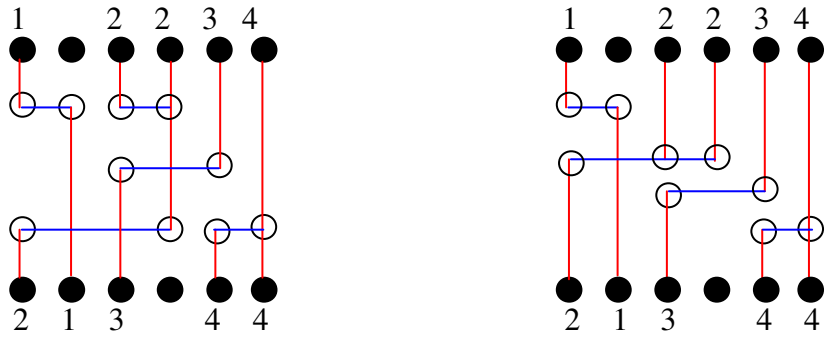


Figure-3.10 channel routing in two layer dogleg free Manhattan model

3.3.5.3 Two layers, unconstrained model

The unconstrained model has no routing restriction on the layers as in the case of Manhattan model. *"In the unconstrained model every channel routing problem is solvable in polynomial time on 2 layers with a sufficiently large width even without extending the length of the channel",* **[36].**

3.4 Gamma Routing

Andras Recski et al **[13]** define the gamma routing problem as *"The special case of the switchbox routing problem in which all the terminals of each net are situated on two adjacent boundaries of the grid (say, they are all northern or western) is called gamma routing. The objective in a gamma routing problem is again to minimize the number of layers needed (since both the length n and the width w are fixed)"*. The complexity of the gamma routing problem is not known, however it is generally assumed that the problem is significantly easier than the general switchbox routing or even the channel routing. A trivial observation shows that if each net has at least one terminal both on the northern and on the western boundaries then the problem instance can be solved on two layers in the Manhattan model, **[5].**

3.5 General Switchbox Routing

This section is only meant to give a bird's eye view of the general switchbox problem. In general switchbox routing, terminals are placed on all four boundaries of the grid. As has already been explained the usage of the term switchbox may refer to a four-sided routing box or to one of the subclasses of the general switchbox problem. In contrast to single row routing and channel routing, both the length and the width of the switchbox routing problem are fixed. David Szsezler states that the objective of switchbox routing is *"to minimize the number of layers needed (or to decide solvability on a given number of layers)"*. For channel routing problem we had a theorem by David Szsezler which states deciding solvability of the problem in the 2-layer Manhattan model is NP-complete. Since channel routing problem is a subset of switchbox routing problem deciding solvability of a four sided switchbox problem in the 2-layer Manhattan model is also NP-complete, **[5].**

3.6 Performance parameters in channel routing

In order to compare the routing algorithms it is always important to lay out an appropriate performance evaluation measurements. Below performance parameters against which the algorithms are compared have been discussed.

3.6.1 Wire length

In the design and manufacturing of VLSI chips one has to always think of working against minimum space constraint. In this regard as the length of interconnecting wire in the chip increases the area required to place the wires increases ultimately increasing the chip size. Another disadvantage of lengthy interconnection wires in the design of chips is the effect of undesirable heat generated inside the chip which possibly leads to a complete destruction of the chip itself. The more the length of the interconnection wire the more will be the total resistance of the wire correspondingly the power dissipated by the wire increases directly with increase in its resistance. Reduction of crosstalk between interconnections is becoming an important consideration in today's VLSI design. Crosstalk between two interconnections is proportional to their coupling capacitance **[19]**. The coupling capacitance is proportional to the coupling length of the two interconnections (the total length of their overlapping segments) and inversely proportional to their separation. Reduction of interconnecting wires means reduction of the total length of wire required for interconnection in the chip. Hence minimizing the overall wire length in the chip is useful for the indicated reasons.

3.6.2 Number of vias

Vias are necessary to make interlayer connection among terminals of the same nets. Many routers use a simple reserved layer model and produce routing solution with a large number of vias. Such models are easy for automation but create big problems during manufacturing. The reasons for minimizing the number of vias in a layout have been outlined.

1. Generally excessive use of vias is considered as an undesirable condition because the probability of manufacturing the chips without defects decreases as the number of vias increases.
2. The resistance associated with each via affects the performance of the chip.
3. Usually the size of vias is greater than those of the interconnecting wires thus the more vias used inside the chip the more routing space would be required which ultimately increases the chip size.

For the above concrete reasons, via minimizing algorithms are by far preferred as compared to other algorithms which neglect the via minimization constraint, **[3].**

3.6.3 Number of tracks

Channel routing problems are basically problems of finding minimum width solution that is nothing but finding minimum number of tracks to complete the routing, **[3] [5]**. Because the majority of switchbox routing algorithms are grid based algorithms and we measure the width of the routing in terms of number of tracks and the distance between two tracks is fixed based on many factors such as the minimum distance needed to avoid crosstalk effects, optimization of the number of tracks necessary for the routing is very important.

All algorithms are not equally good for all channel routing problems. Therefore many benchmark channel routing examples have been proposed. The most famous benchmark is the Deutsche difficult example. In chapter six comparisons of different algorithms in solving the Deutsche difficult channel routing problem has been presented.

Chapter four

Switchbox Routing In Manhattan Model

This section presents the description of switchbox routing problems where some problems from literature are diagrammatical analyzed in order to show the number of layers required in their routing solutions.

4.1 The Square-Shaped Switchbox

A square shaped switchbox is the one for which the width and the height of the routing are equal. As indicated by Hambrusch **[41]** the necessary number of layers to solve such a problem is at least 4 in the worst case, even in the unconstrained model. Furthermore, it is verified by Andras Recski **[42]** that the number of layers needed is at least 6 in the Manhattan model for the 2×2 problem of Figure-4.1 taken from **[5]**.

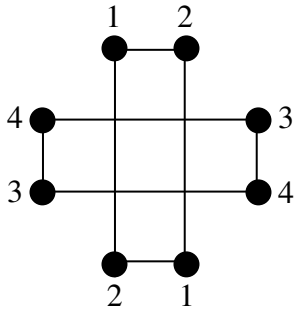


Figure- 4.1 a 2×2 switchbox problem

David Szeszler argues, "*Every square-shaped switchbox can be solved on 6 layers in the Manhattan model",* **[5].** In the section below two switchbox routing problems are investigated to verify what David Szeszler asserts in the above theorem.

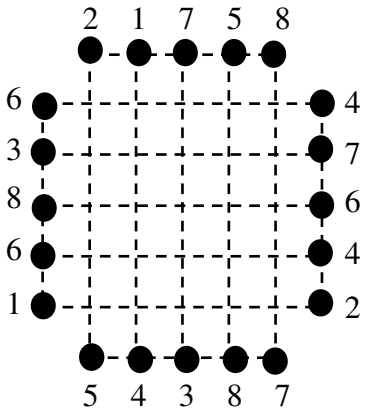


Figure- 4.2 switchbox routing problem

Figure 4.2 is a switchbox routing problem taken from **[5].** Here the solution is presented in an easy way to make visualization of the problem simple.

Nets are categorized as follows

1. Net 1 is NW net (one terminal on north boundary and another on west boundary)
2. Net 2 is NE net (one terminal on north boundary and another on east boundary)
3. Net 3 is WS net (one terminal on west boundary and another on south boundary)
4. Net 4 is SE net (one terminal on south boundary and another on east boundary)
5. Net 5 is NS net (one terminal on north boundary and another on south boundary)
6. Net 6 is EW net (two terminals on west boundary and one terminal on east boundary)
7. Net 7 is NSE net (one terminal on north, another on south boundary and last terminal on east boundary)
8. Net 8 is NSW net (one terminal on north, another on south boundary and last terminal on west boundary). By assigning nets to the different layers according to the location of terminals as analyzed above, the solution in figure 4.3 is obtained.

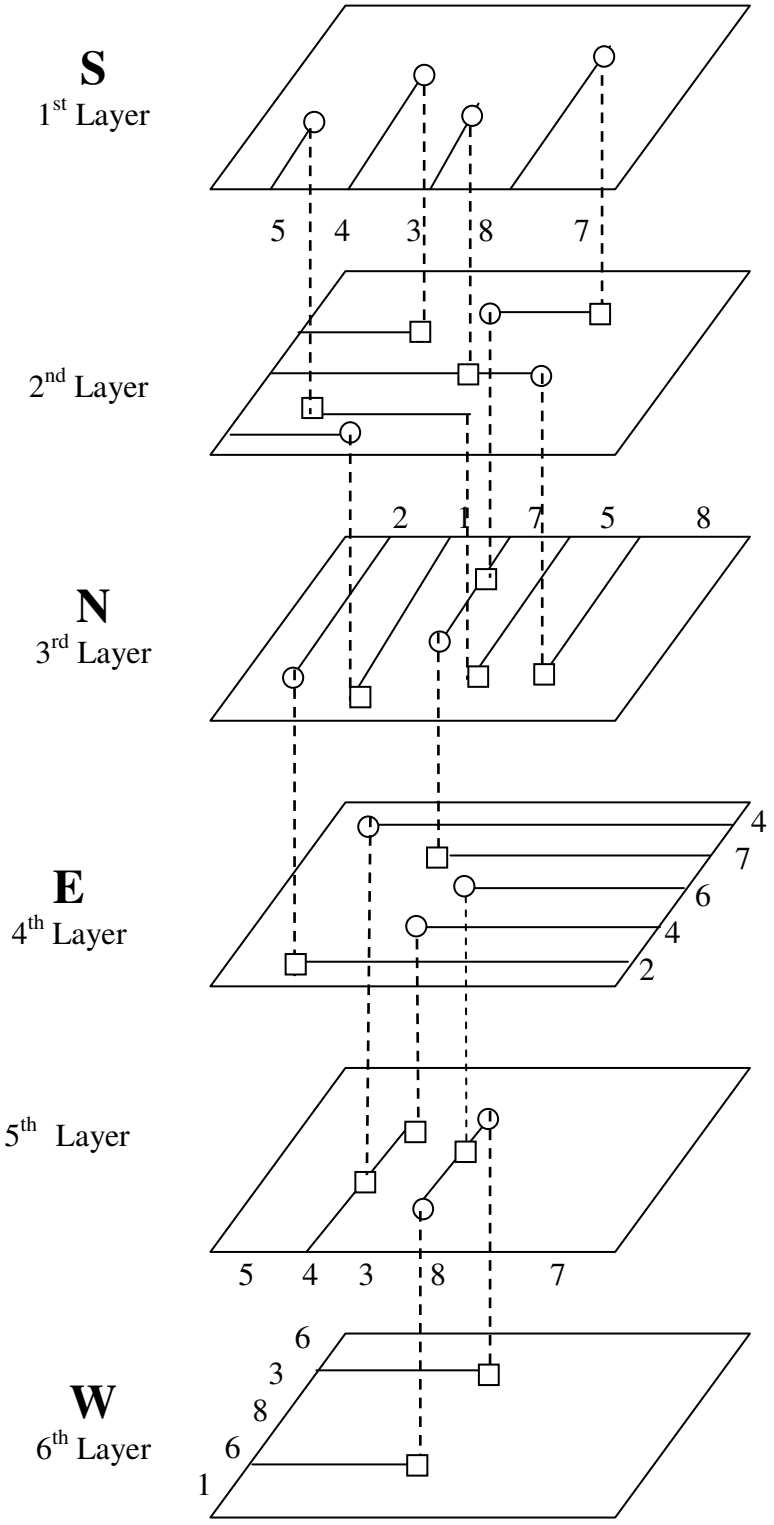


Figure- 4.3 solution to figure-4.2

In the multilayer Manhattan model consecutive layers must contain wire segments of different directions as defined in page 33. Thus layers with horizontal east-west and with vertical north-south wire segments alternate as shown in figure 4.3. The routing solution in figure 4.4 shows a complete solution in six layers.

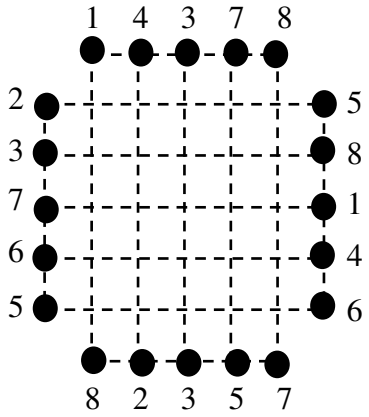


Figure 4.4 switchbox routing problem

Figure 4.4 is another switchbox routing problem its solution is presented below to verify what David Szeszler asserts in the theorem above.

Nets are categorized as follows

1. Net 1 is NE net (one terminal on north boundary and another on east boundary)
2. Net 2 is WS net (one terminal on west boundary and another on south boundary)
3. Net 3 is WS net (one terminal on north boundary , one on west boundary and another on south boundary)
4. Net 4 is NE net (one terminal on north boundary and another on east boundary)
5. Net 5 is WSE net (one terminal on west boundary , one on south boundary and another on east boundary)
6. Net 6 is EW net (two terminals on west boundary and one terminal on east boundary)
7. Net 7 is NWS net (one terminal on north, one on west boundary and another on south boundary)

8. Net 8 is NSW net (one terminal on north boundary, one on east boundary and another on west boundary). By assigning nets to the different layers according to the location of terminals as analyzed above, the solution in figure 4.5 is obtained.

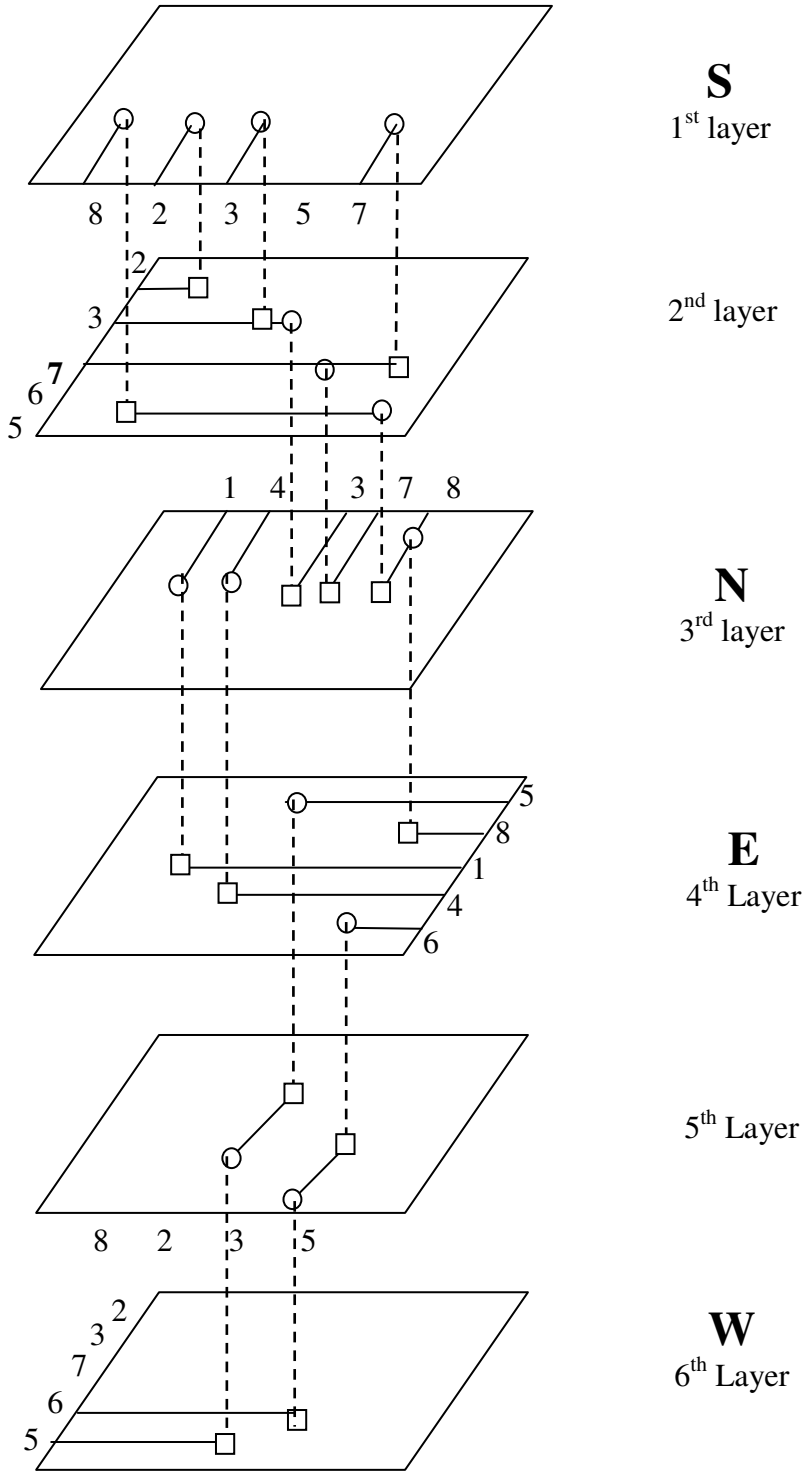


Figure 4.5 switchbox problem

The above two solutions confirm the assertion of David Szeszler's theorem which states, "*Every square-shaped switchbox can be solved on 6 layers in the Manhattan model".*

4.1.1 Draw backs of Manhattan model switchbox routing

- Generally the Manhattan model leaves some tracks and columns unused
- The number of layers in the Manhattan model is relatively high as compared to an unconstrained switchbox routing model.

4.2 The Switchbox of Arbitrary Shape

David Szeszler has shown a new result in his PhD dissertation **[5]** about the minimum number of layers needed for a switchbox routing problem in the Manhattan model.

"*Assume that a switchbox routing problem instance with length n and width w is given such that n ≥ w. Denote by d the maximum congestion of all vertical lines that cut the grid into two. Then the minimum number of layers needed for a solution in the Manhattan model is between* $2\left(\frac{d}{w}\right)-1$ *and* $2\left(\frac{d}{w}\right)+4$. *A solution on* $2\left(\frac{d}{w}\right)+4$ *layers can be found in linear time"*. The following description has been included to make the above theorem clear. *"If e is a vertical line with congestion d then d wires must intersect e. At most w wires can intersect e on each layer reserved for horizontal wire segments. Thus the number of such layers is at least* $\left(\frac{d}{w}\right)$ *which proves that* $2\left(\frac{d}{w}\right)-1$ *is a lower bound on the total number of layers"*. In the same literature **[5]** a summarized description of linear time solution for any type of switchbox routing problem has been presented.

"*Every switchbox can be solved in linear time on 2m + 4 layers in the Manhattan model".*

Chapter five

Maze routing algorithms

In this chapter some important maze routing algorithms are discussed. The discussion of maze algorithms in this chapter are all considered in relation to detailed VLSI routing because these algorithms are also widely applicable in global routing.

Generally maze routing algorithms are used to find a path between a pair of points, called the source and the target respectively, in a planar rectangular grid graph. Standard cell and gate array design styles are pretty convenient to model the layout as a grid. In a maze routing region two types of areas are available. An already occupied routing region is represented as blocked area. The remaining routing region is represented as unblocked area. The aim of maze routing algorithm is to find a minimum path (if possible) between the source and the target vertex using only unblocked routing regions. The first phase of finding the path is called exploration phase where several paths start at the source in all directions and are expanded until one of them finds the target. The second phase is called the retrace phase where vertices are retraced back to the source in order to identify the actual routing path, **[3]**.

5.1 Lee's Algorithm

A routing algorithm for a two terminal net is introduced by Lee **[37]**. This algorithm is most widely used for finding a path between any two vertices on a planar rectangular grid in either standard or gate array design style. *"The key to popularity of Lee's maze router is its simplicity and its guarantee of finding the optimal solution if one exists"*.

The search in Lee's algorithm can be understood in the same way as wave propagation. Wave propagates from its source in all direction to the neighboring vicinity. All unblocked neighboring vertices of the source are labeled '1'. All, unblocked vertices adjacent to vertices with a label '1' are marked with a label '2', all unblocked vertices adjacent to vertices with a label '2' are marked with a label '3' and so on. The process is repeated until the target vertex is found or no other unblocked vertices are left. Illustration of Lees maze routing algorithm is shown in figure 5.1.

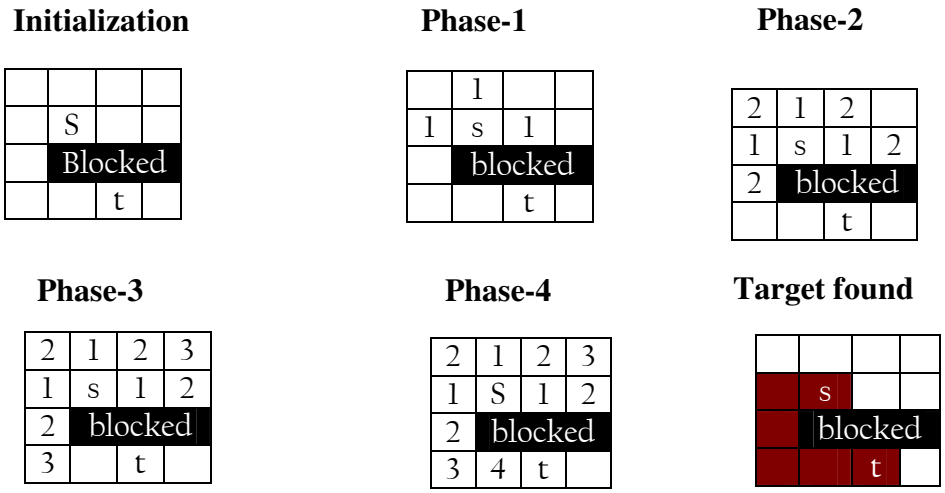


Figure 5.1 Lee algorithm illustrations

The major drawback of Lee's algorithm is that it requires a large amount of memory and its performance diminishes as the size of the grid increases. *."The time and space complexity of Lee's algorithm for a rectangular grid of dimension hxw is O (hxw)"*. The complete listing of Lee's algorithm is given in figure 5.2 from **[3]**.

Algorithm LEE-ROUTER(B,s,t,P)

Input: B,s,t

Output:P

Begin

plist=s;

nlist=Ø;

Temp=1;

```
    Path_exists= FALSE;
    while plist do
        for each vertex vi in plist do
            for each vertex vj neighbouring vi do
            if B[vj] = UNBLOCKED then
                L[vj]=temp;
                INSERT(vj. nlsit);
                If vj=t then
                    path_exists=TRUE;
                    exit while;
            temp=temp+1;
            plist=nlist;
            nlist= Ø;
    If path_exists= TRUE then RETRACE (L. P);
    Else path does not exist;
end.
```

Figure 5.2 Lee router algorithm

Description of functions and variables in figure-5.2

- S – The source or starting terminal used as input to the algorithm.
- T – The target or final terminal to be connected with the source terminal used as input to algorithm.
- B – Input array
- P – Provides the shortest path between the source terminal and the target terminal as an out put of the Lee's algorithm.
- Plist – propagation linked list used to keep track of the vertices on the wavefront.
- nlilst – neighbor linked list used to keep track neighboring vertices.
- L[v] – is an array used to contain the distance from the source to the vertex v.
- RETRACE (L. P); – uses L[v] to retrace the vertices to form a path P.
- INSERT (vj. Nlsit) – inserts the current vertex in to nlist linked list.

Single layer and multiple layer implementation results of Lee's maze routing algorithm have been presented in chapter six.

5.2 Soukup's Algorithm

Lee's algorithm searches in a manner of wave propagation symmetrically in all directions which takes quite a lot of time. In order to overcome this limitation, an iterative algorithm that explores in the direction of the target (not symmetrically in all directions as in the case of Lee's algorithm) until the target or an obstacle is reached, has been proposed by Soukup. Figure 5.3 presents the description of Soukup algorithm as found in **[3]**.

```
Algorithm LEE-ROUTER (B,s,t,P)
        Input: B,s,t
        Output:P
Begin
        plist=s;
        Nlist=Ø;
        Temp=1;
        Path_exists= FALSE;
        while plist ≠ Ø  do
                for each vertex vi in plist do
                        for each vertex vj neighbouring vi do
                            if  vj=t  then
                                 L[vj]=temp;
                                  Path_exists=TRUE;
                                   Exit while;
                            if B[vj] = UNBLOCKED then

                              (*if the direction of the search if toward the target, the
                               search continues in this direction*)
                                  If DIR(Vi,Vj)= TO-TARGET
```

```
                then L[vj]=temp;
                     temp=temp+1;
                     INSERT(vj. plsit);
                     While B[NGHBR-IN –DIR(Vi,Vj)]=
                          UNBLOCKED do
                          Vj=NGHBR -IN –DIR(Vi,Vj);
                           L[vj]=temp;
                           Temp=temp+1;
                           INSERT(vj. plsit);
                else
                   L[vj]=temp;
                   Temp=temp+1;
                   INSERT(vj. nlsit);
          plist=nlist;
          nlist= Ø;
     If  path_exists= TRUE then RETRACE (L. P);
     else path does not exist;
end.
```

Figure 5.3 SOUKUP routing algorithm

Description of functions and variables of the algorithm in figure-5.3

The variables used in Soukup algorithm is similar with those of Lee's algorithm, here description of variables not mentioned in the Lee's algorithm are included.

- L[v] – shows the order in which the vertex v is visited during the exploration phase of the algorithm.
- DIR (Vi,Vj) – returns the direction from Vi to Vj.
- NGHBR–IN –DIR(Vi,Vj) – returns the neighbor of Vj which is in the direction from Vi to Vj.

- TO-TARGET – is a variable which indicates wether the direction of the search is towards the target. Described inside the two asterisks indicated in the middle algorithm of figure 5.3

Chapter six

Implementation of Routing Algorithms

As stated in the proposal of my thesis, this thesis basically comprises two aspects in VLSI routing. The first one is the literature survey on switchbox routing algorithms the second one is implementation of selected algorithms. This section deals with implementation aspects of selected switchbox routing algorithms. Specifically, object oriented design and implementation of the following algorithms are presented:

1. Single row routing algorithm,
2. Left edge channel routing algorithm and
3. Lee's maze routing algorithms (single layer, multi layer).

Objectives of the implementation include:

To characterize VLSI routing algorithms in terms of important VLSI routing parameters such as number of vias (interconnection across the layers) it requires in completing the given routing, number of tracks it uses etc. Evaluation of the implemented algorithms involves telling the pros and cons of the algorithm in comparison with an acceptable optimal solution of the problem.

Before going to the implementation of the algorithms it is necessary to recap on the theoretical descriptions of the algorithms.

6.1 Channel routing algorithm

The detailed description of the various channel routing algorithms have been discussed in chapter three section 3.3 of this thesis. Routing is the task of finding a set of connections that will wire together the terminals of different modules on a VLSI chip. In these problems, connections may be placed on one of a small number of layers. Wires on the same layer that cross each other form a connection. Wires on different layers that cross each other do not connect unless an explicit interconnection called via is constructed there to connect the wires on the different layers.

Channel routing is a constrained form of routing in which all connections are made in a rectangular region called a channel (a region bounded by two rows containing the terminals to be interconnected). Terminals to be connected in the routing are located in the top or bottom rows of the channel. The connections to be made in a channel routing problem are specified by a netlist. Each net in the netlist describes a set of terminals that must be connected together. Connections are made by wire segments on two different layers which are electrically insulated from each other unless a via is used. Inside the channel, horizontal wire segments on one layer are placed along lines called tracks. Vertical wire segments on another layer connect terminals to the horizontal wire segments, with vias at the intersection.

6.1.1 Left edge channel routing algorithm

In this section a high level description of the left edge channel routing algorithm has been outlined. A discussion of vertical constraints, in relation to channel routing problems, will be described first. This will be followed by a description of two flavors of the left edge algorithm. The first approach uses an unconstrained left edge algorithm i.e. one that does not look at vertical constraints. The second flavor takes vertical constraint into consideration when assigning tracks. The left edge algorithm can not solve channel routing problems with cyclic vertical constraints.

Unconstrained left Edge algorithm (without considering vertical constraint) **[11]**

- Basic concept – put as many nets as possible on a track
    1) Sort all nets by the coordinate of the leftmost end points in an increasing order
    2) Put the first net of the sorted list on the left side of the lowest available track and remove it from the list
    3) Select next non-overlapping net, put it on the right side and remove it from the list; repeat this process until this track is full
    4) Repeat previous two steps until all nets have been assigned to a track.

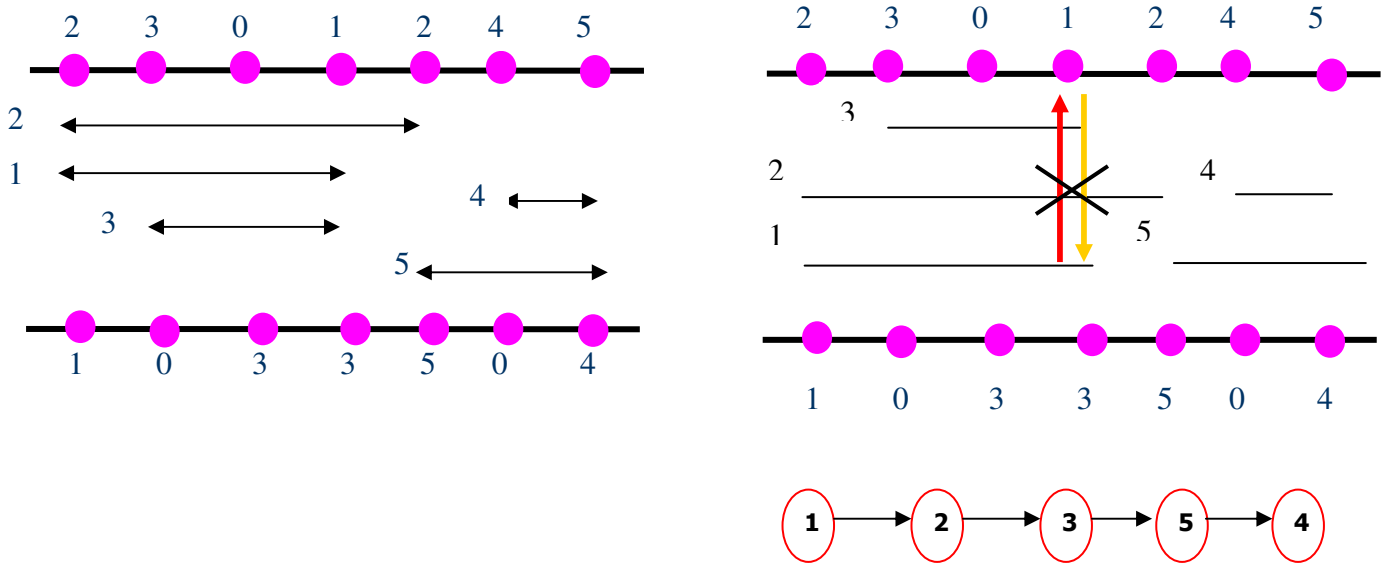


Figure 6.1 Unconstrained left edge channel algorithm illustration

The list on the bottom right side of figure 6.1 shows a sorted net list of the unconstrained left edge algorithm. Assignment of horizontal tracks begins with the first net of the list, ones the net is assigned a track it is removed from the list and the next non-overlapping net from the list is assigned a track at the right side of the previously assigned net. In this algorithm the assignment of track doesn't take in to consideration the case of vertical constraints. As shown in the figure, net 1 and net 3 are sharing the same column. One can conclude that this algorithm can not solve a channel routing problem with vertical constraint.

Constrained left Edge algorithm (considering vertical constraint) **[11]**

- Enhancement over the first flavor (cannot solve cyclic VCG)

    1. Sort all nets by their leftmost end points in an increasing order
    2. Scan the net list and select the first net that has no descendent in VCG, put it on the lowest available track, and remove it from the list and VCG
    3. Select next non-overlapping net that has no descendent in VCG, put it on the right side, and remove it from the list and VCG
    4. Repeat previous step until this track is full
    5. Repeat the previous three steps until all nets are assigned to a track

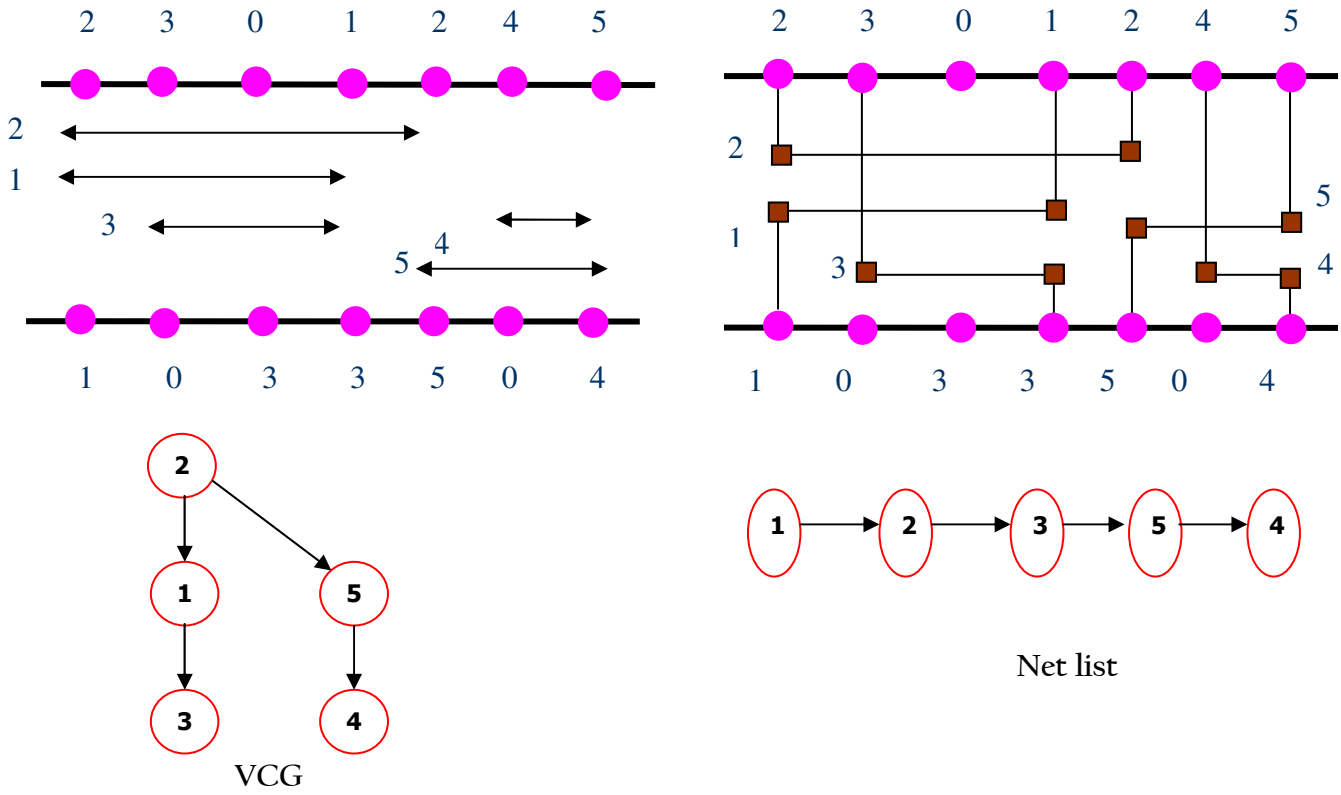


Figure 6.2 Constrained left edge channel algorithm illustration

Constrained left edge algorithm is an improvement over the unconstrained left edge algorithm where vertical constraints are successfully handled. In order to deal with vertical constraint problem one needs to maintain a vertical constraint graph (VCG). Allocation of track begins with a net at the bottom of the VCG. Nets 3 and 4 are assigned to the bottom track for they are found at the bottom of the VCG and they are not

overlapping nets. Once a net is assigned a track it is removed from the VCG and the process continues in this manner until all the nets are assigned a track.

Figure-6.3 presents a class diagram description of the left edge channel routing algorithm. The class diagram is included to show the relation among the different classes in the design of the algorithm.

6.1.2 Methodology used in the implementation of single row routing and channel routing algorithms

The single row routing problem is a subclass of channel routing problem where its terminals are located on one side of the routing region.
On the investigation of these problems the following entities are identified for each type of problems.

Terminals

In VLSI routing problems the basic entity one usually identifies is a pin or a terminal to be connected with other terminals. A pin or a terminal is a basic unit for interconnection. A pin could be an input or an output terminal of the logic block. In a VLSI physical design, especially in the routing phase one encounters quite a large number of pins to be interconnected in a certain routing region. While making the placement of the logic blocks in the partitioned area the position of pins is determined automatically. Hence the problem in the routing is to find an interconnection path among the pins without changing their placement.
In the single row routing problem terminals take only one of the four sides of the switchbox problem, for example either the northern or the southern boundary.

Nets

Another entity identified in the single row and channel routing problems is a Net. A Net basically consists of terminals of logic blocks which need to be connected with each other. In VLSI routing considering a certain routing region there are many nets which

should be routed independently, i.e without short circuit between two different nets. Besides the short circuit constraint cross is another constraint that needs to be considered in routing different nets. In both single row and channel routing problems implemented in this thesis, Manhattan model routing is adopted which dictates that no two consecutive routing layers should consist wires running in the same direction. This basically decreases the capacitive coupling that might arise as a result of parallel wires in two consecutive routing layers. As a result of adopting Manhattan model in the implementation of the two problems it is possible to minimize cross talk among wires of different nets.

Net List

Another entity identified in the single row and channel routing problem is a Net List. A Net list is basically list of different nets that are used as an input in the routing problem. A Net list is constructed using the Terminal and Net entities defined above. A Net list has been defined using a dynamic data structure which changes the size of the container based on the number of nets to be stored in the data structure.

Canvas

One very important entity identified in the analysis and implementation of the two routing problems is a Canvas entity of a Canvas object. Canvas means a drawing area drawing space or better defined a drawing object. In the single row routing problem it is defined as SingleRowRouterCanvas and in the channel routing problem it is defined as ChannelRouterCanvas. One of the most important contributions of this thesis is to provide a graphical visualization of VLSI routing results. In order to provide a graphical display of routing results one has to identify an object through which routing results could be drawn. One of the reasons of preference for java programming in this thesis is, java is very rich for its graphical visualization tools and Canvas object has such a capability.

User interface

The user interface is the last entity identified in the analysis of the two problems. The user interface entity is necessary in order to simplify interaction of the routing engine with the user. The user interface is designed using a boarder layout method. In this method the various components are arranged in to five regions: Northern, southern, eastern, western and central regions. In case of single row routing terminals to be connected are located on the northern region in the user interface. In the case of channel routing terminals are located on northern and southern boundaries. The central boundary is used for the Canvas object for drawing the routing wires. Buttons to initiate routing and remove routing are located on the southern boundary. There also labels for displaying the count of tracks and vias used in routing.

The above five entities identified in the analysis of the two problems are converted in to java classes. The relationship among these classes is described in the class diagram in figure 6.3. In the class diagram description classes are depicted using rectangle. The first row in the rectangle consists the class name the second row consists attributes of a class and the last row consists the methods of the class. There are different types of relationships among these classes, for example the ChannelRouterUserInterface has an inheritance relationship with the java Frame class. ChannelRouterUserInterface class has an association relationship with ChannelRouterCanvas. Upward directed arrow indicates inheritance relationship and down ward arrow indicates association relationship.

Class diagram for left edge channel routing algorithm

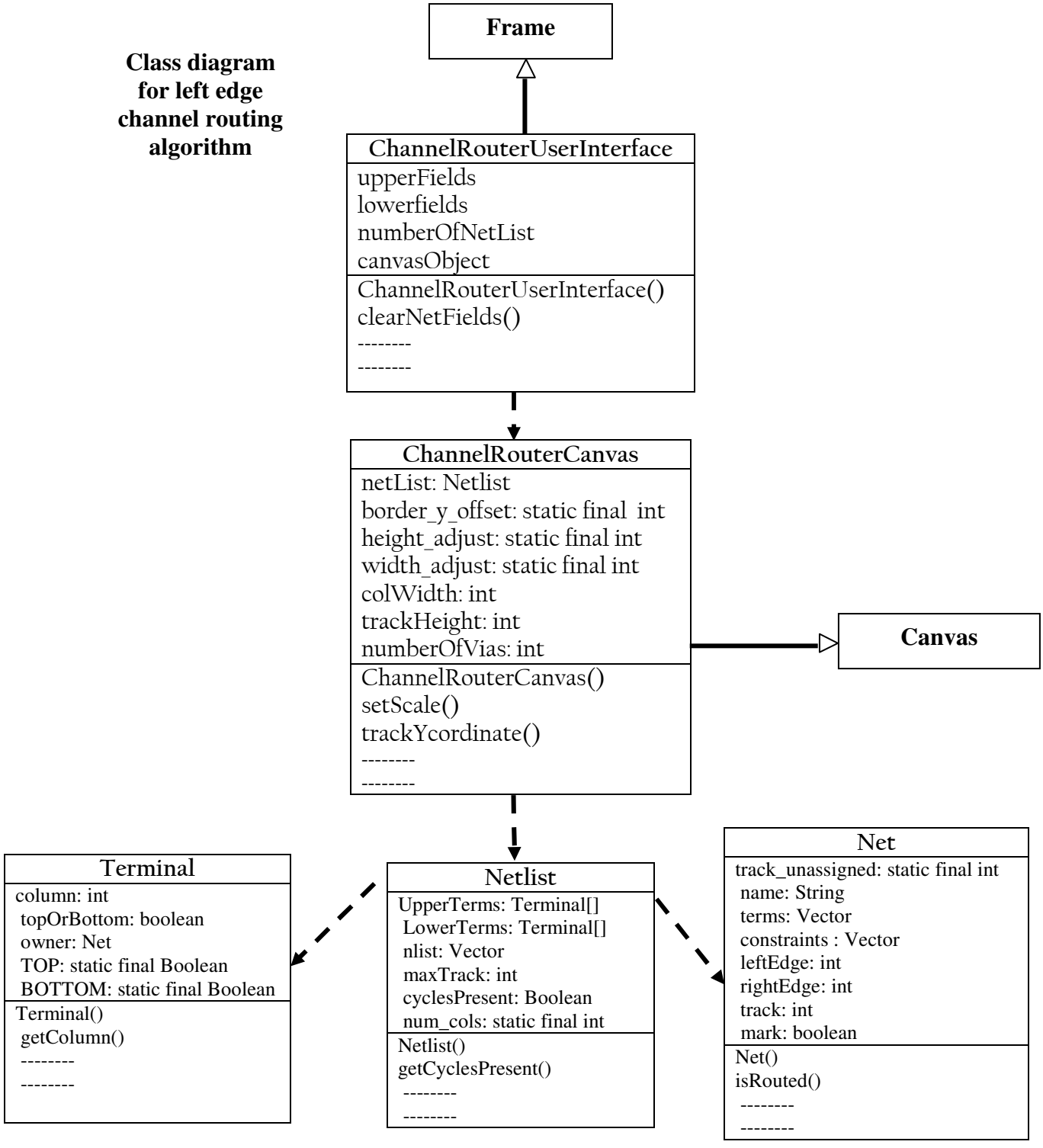


Figure 6.3 Left edge channel router class diagram

6.1.3 Left edge algorithm applied to single row and channel routing problems

Screen captured java implementation results of different algorithms and their description are shown below. The solutions to single row routing problem and to the channel routing problems are done using Manhattan's two layer model. The red vertical lines are all placed in one layer and the blue horizontal lines are all placed in another layer. Therefore the numbers of layers used are two in all cases. A routing in more than two layers in maze routing implementations has been included at the end of this chapter.

Single row routing problem

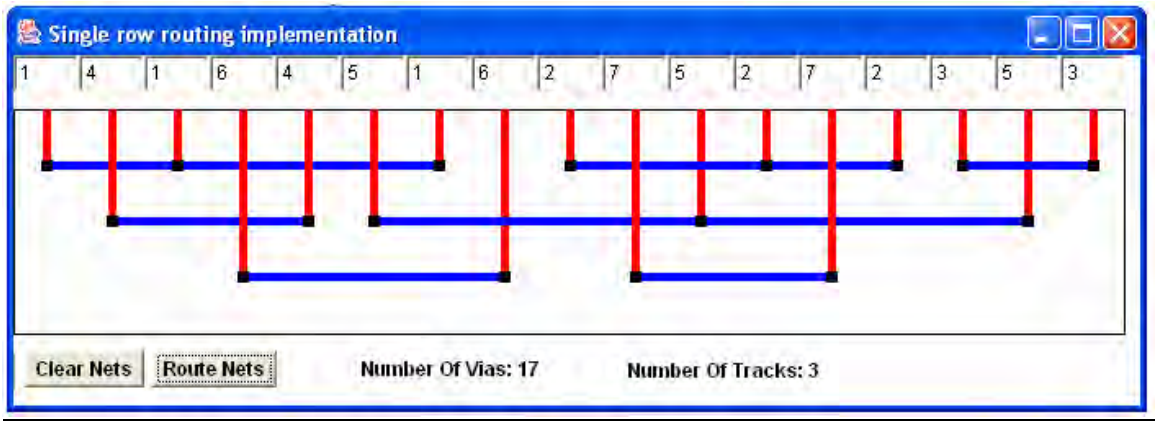
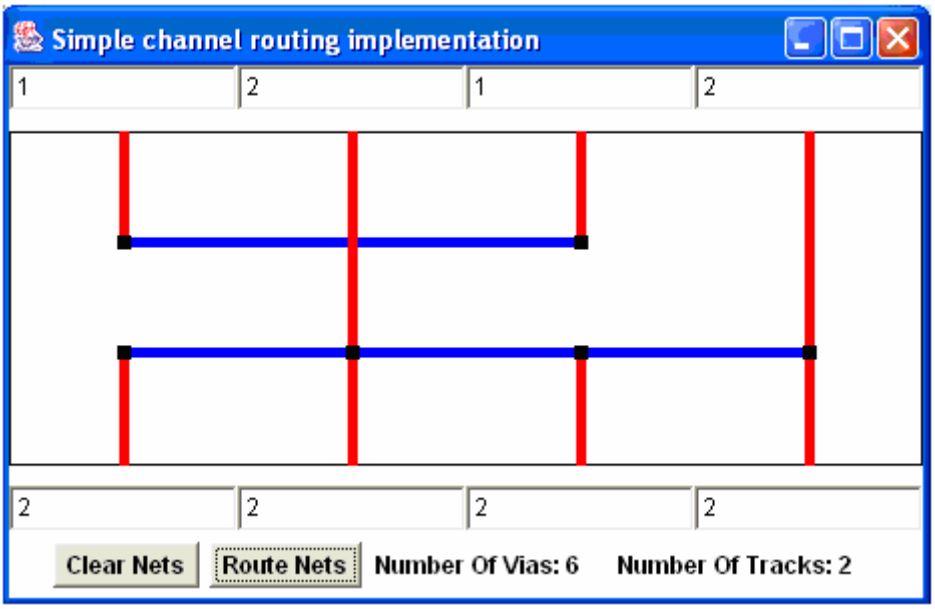


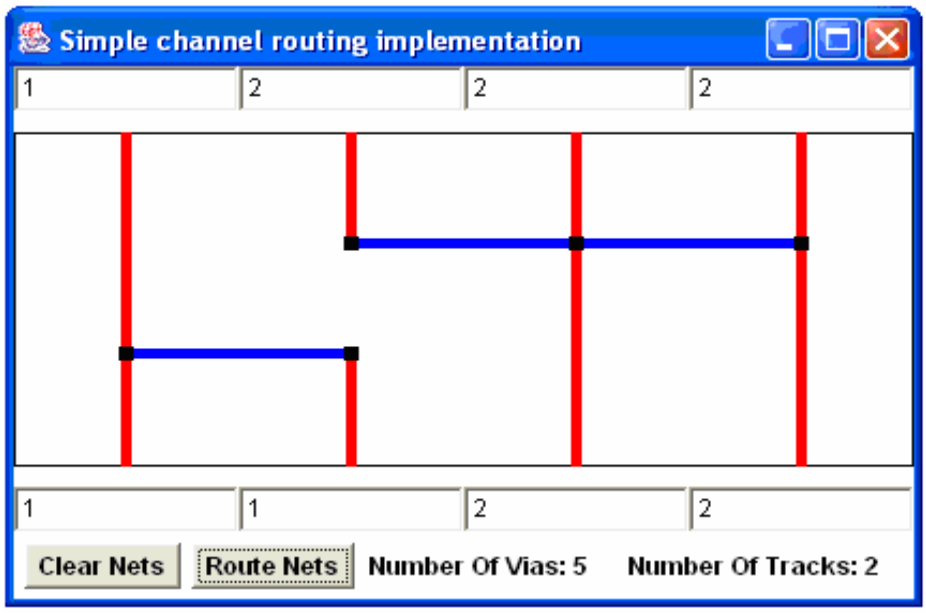
Figure 6.4 single row routing implementation out put

Figure 6.4 is implementation result of the single row routing algorithm discussed in section 3.2. The congestion at a given point, for a single row routing problem, is defined as the number of nets bisected by the vertical line drawn at that point as discussed earlier. The maximum congestion of all such lines defines the density of the problem. The density of the single row routing problem indicated in figure 3.1 matches the implementation result of figure 6.4.

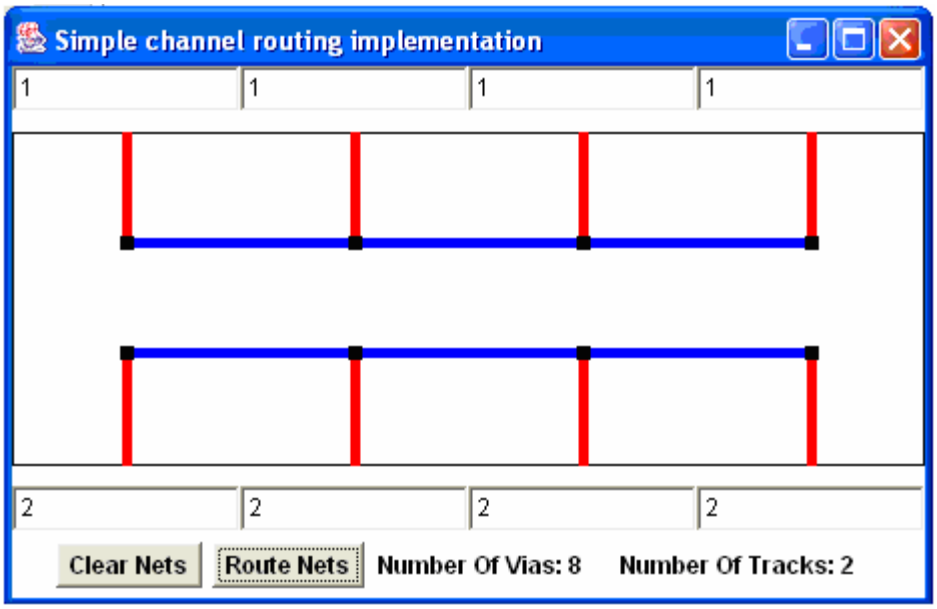
Simple channel routing implementation results



(a)



(b)

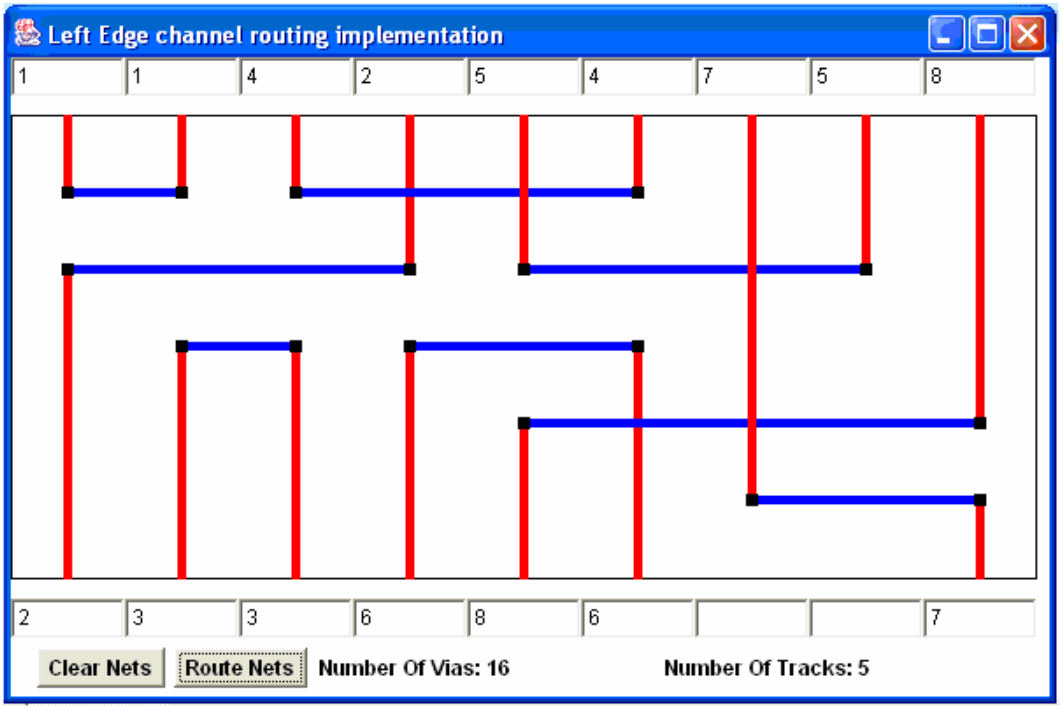


(c)

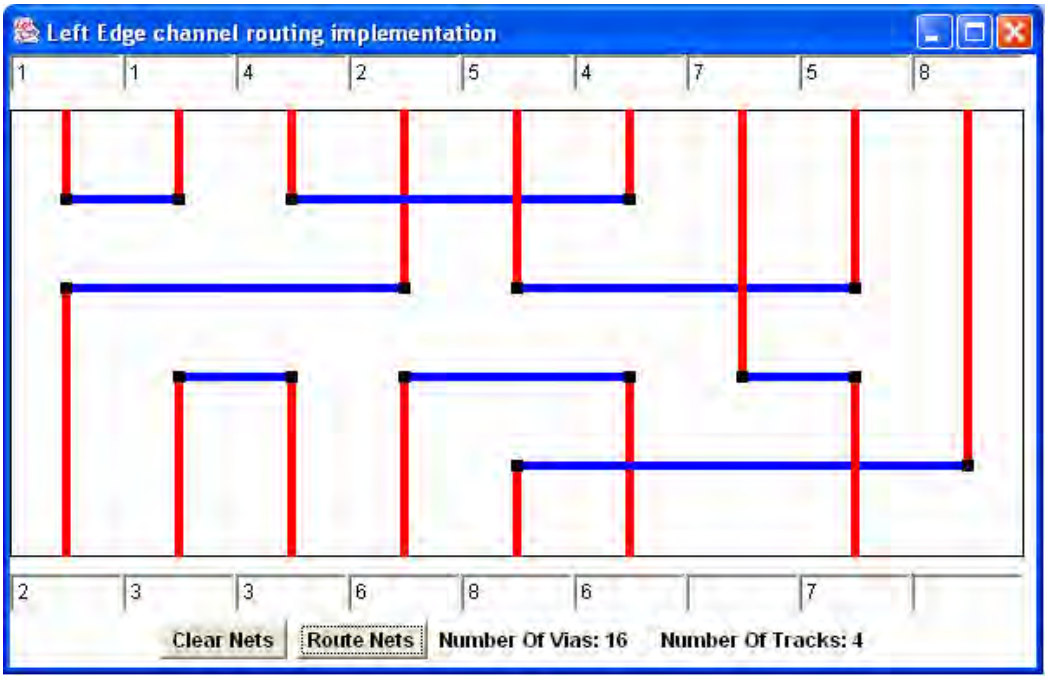
Figure 6.5 (a), (b), (c) simple channel routing implementation out puts

Figure 6.5 shows three simple channel routing problems that that were routed using the java implementation. As can be seen in the three figures terminals of same net are connected together, i.e. terminals which have the same label belong to the same net. The above channel routing problems are simple problems because the length of the channel is small i.e. 4 in this case. Routing metrics at the bottom of the figures are as follows. Figure 6.5 (a) required 6 vias, figure 6.5 (b) required 5 vias and figure 6.5 (c) required 8 vias. In all the figures the number of tracks required is 2.

Implementation result of channel routing problem in figure 3.8



(a)



(b)

Figure 6.6 (a), (b) channel routing implementation out puts of figure 3.8

The implementation results in figure 6.6 show the successful work of the routing engine. The channel routing problem in figure 3.8 is indicated to use only four tracks. The two outputs of figure 6.6 are the same in terms of number of vias (both use 16 vias), but figure 6.6 (a) used one less number of tracks than figure 6.6 (b). The change in the number of tracks used is due to the slight change in the position of a terminal belonging to net 7 at the bottom left of the figures. Therefore the minimum number of tracks necessary to complete the given routing problem is four. Optimum routing of the nets is not only dependant on the effectiveness of the routing algorithms (both global and detailed) used but also on the effectiveness of placement algorithms used before the routing phase has began. The difference in the routing result of the same problem is basically due to the change of position (placement) of terminal 7.

6.1.4 Implementation results of benchmark problems

This section discusses one additional channel routing problem from (Vladimir N. Davidenko, Victor M. Kureichik,Victor V. Miagkikh) **[40].** Figure 6.9 describes a benchmark channel routing problem solved using genetic algorithm. The following results were obtained by the (Vladimir N. Davidenko, Victor M. Kureichik,Victor V. Miagkikh) of paper **[40].**

Results: Channel width (number of tracks used) = 12 **[40]**

Vias used = 57

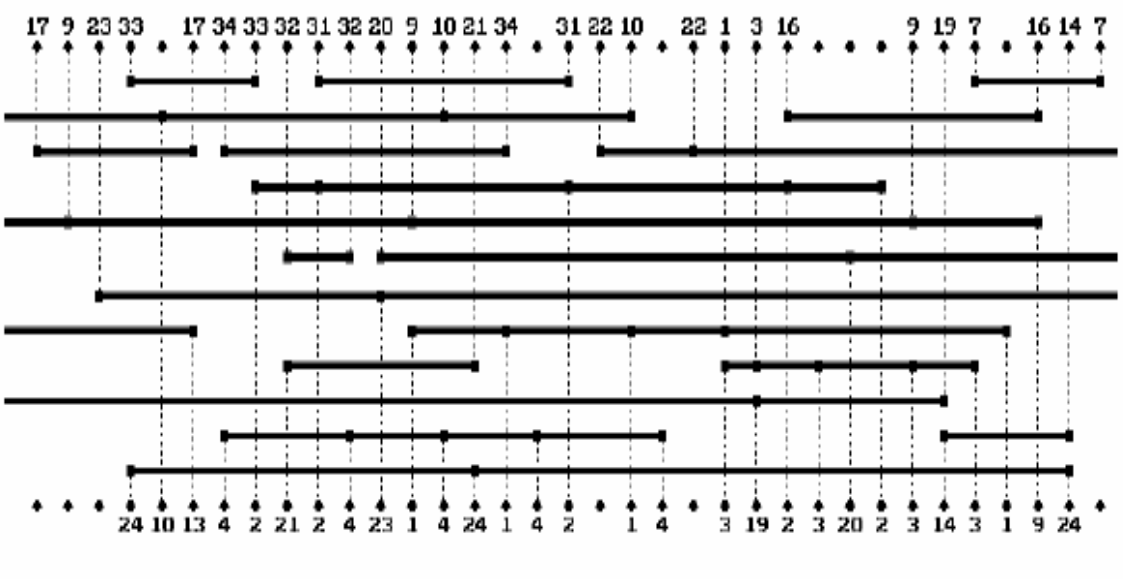


Figure 6.7 genetic algorithm result of channel routing problem

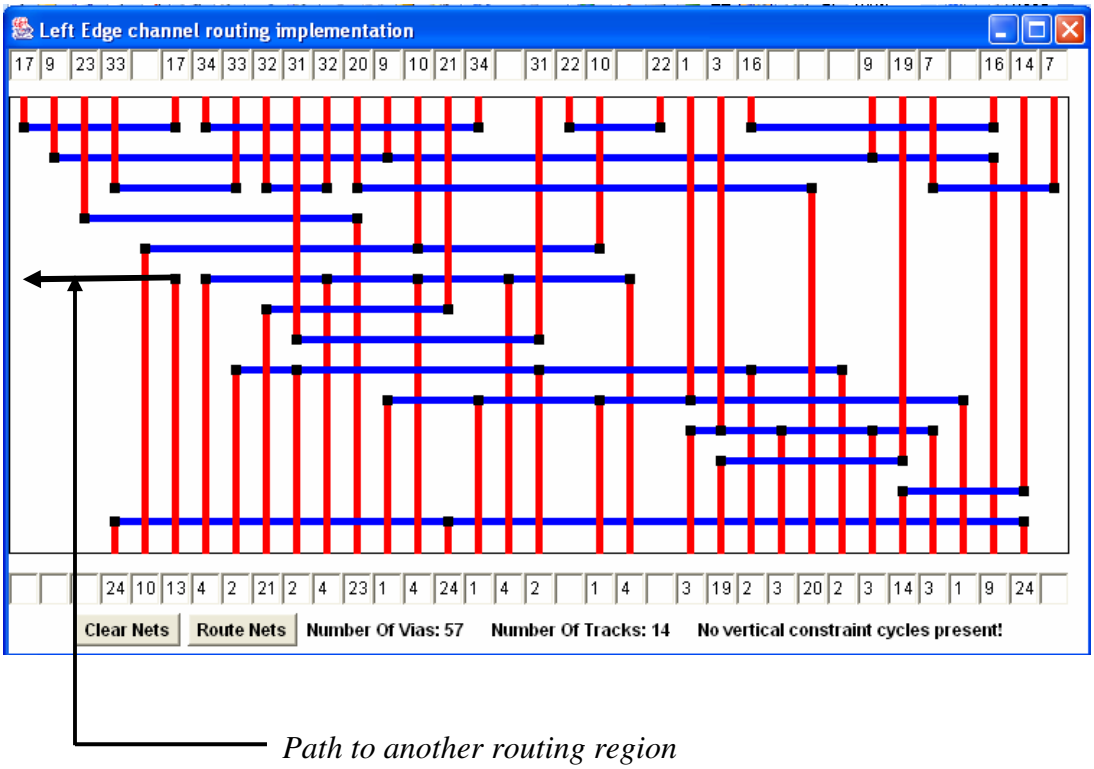


Figure 6.8 left edge implementation of channel routing problem in figure 6.7

Implementation of a benchmark problem in **[37]** when routed using my engine gave the following results.

Results: Channel width (number of tracks used) = 14
Vias used = 57

The Java implementation required the same number of vias as the genetic algorithm used in **[40].** However the number tracks used in the java implementation is two more than those used in the genetic algorithm. As can be seen from figure 6.8 a single terminal wire – terminal 13 is left alone. This is because net 13 is having only one terminal in the routing region. Generally, routing in VLSI systems is done using a divide and conquer algorithm i.e. the total routing region is divided into smaller regions for the detailed router. Therefore we may not find all terminals of the net to be placed in the same routing region.

6.2 Maze routing algorithm

The detailed description of maze routing algorithms has been included in chapter five. In the same chapter two maze routing algorithms: Lee's maze routing algorithm and Soukup's maze routing algorithm were given special emphasis. To appreciate the functioning of maze routing algorithm implementation of Lee's maze routing algorithm has been done. Another goal of implementing Lee's maze routing algorithm is to see how the algorithm can be used in conjunction with the left edge algorithm to complete the routing of any channel routing problems involving vertical constraint cycles. Selection of Lee's maze router over Soukup's maze router for implementation is based on the simplicity of Lee's maze router.

In the next two sections screen captured images of single layer maze routing and multiple layer maze routing implementation results are presented.

6.2.1 Methodology used in the implementation of Maze routing algorithm (Lee's maze routing algorithm)

Maze routing algorithms are basically area routing algorithms, i.e, given two terminals in the routing area the maze router finds a shortest path between the terminals if such a path exists. While investigating Lee's maze routing algorithm, I have identified the following entities which are ultimately converted in to java classes.

Grid

Obviously the routing is done over a grid. The routing area is divided into horizontal and vertical grids hence we need a Grid entity.
The grid is basically constructed as a three dimensional array where the two dimensions describe the width and the height of the routing area and the third dimension holds the number of layers the maze router spans.
Corresponding to the width and height of the routing region pixel width and pixel height should be determined.
The total number of columns and row necessary for constructing the routing grid is also determined.

Grid Point

After constructing the routing grid we identify another entity called Grid Point which deals with everything related with each grid point. Each grid point is described with three variables: position x, position y, position z. for single layer routing the third variable is not necessary.
In the exploration and trace back phase of the maze routing algorithm we are basically concerned with tacking of the neighboring grid points. In the expansion phase we keep track of each grid point with its distance from the starting terminal which is later used to determine the minimum path between the two terminals to be connected. In the expansion and retrace phases painting of grid points involved in the two phases are painted.

The painting of grid points involves different colors showing the different phase of the routing and for this the grid Point entity uses another entity called color sequencer as described below.

Color Sequencer

This is a very simple entity which is used to determine the various colors used for painting of grid points in the previous entity. The color sequencer is designed in such a way it is possible to use different sequences of colors for exploration, retrace and actual routing of the minimum path as determined by the maze routing algorithm.

Maze application

This is another entity identified in the maze routing. It is basically used for construction of the user interface. All the layout design for the of the user interface including the button used for the interaction are incorporated in this entity.
This entity also contains all the necessary java elements used for the animation of the maze routing algorithm.

As in the single row and channel routing algorithms, after identifying the necessary entities, they are all converted in to java classes. The relationship among these classes is depicted using the class diagram in figure 6.9. How classes are depicted and all the relations among the various classes is defined in the implementation methodology section of single row routing and channel routing problem.

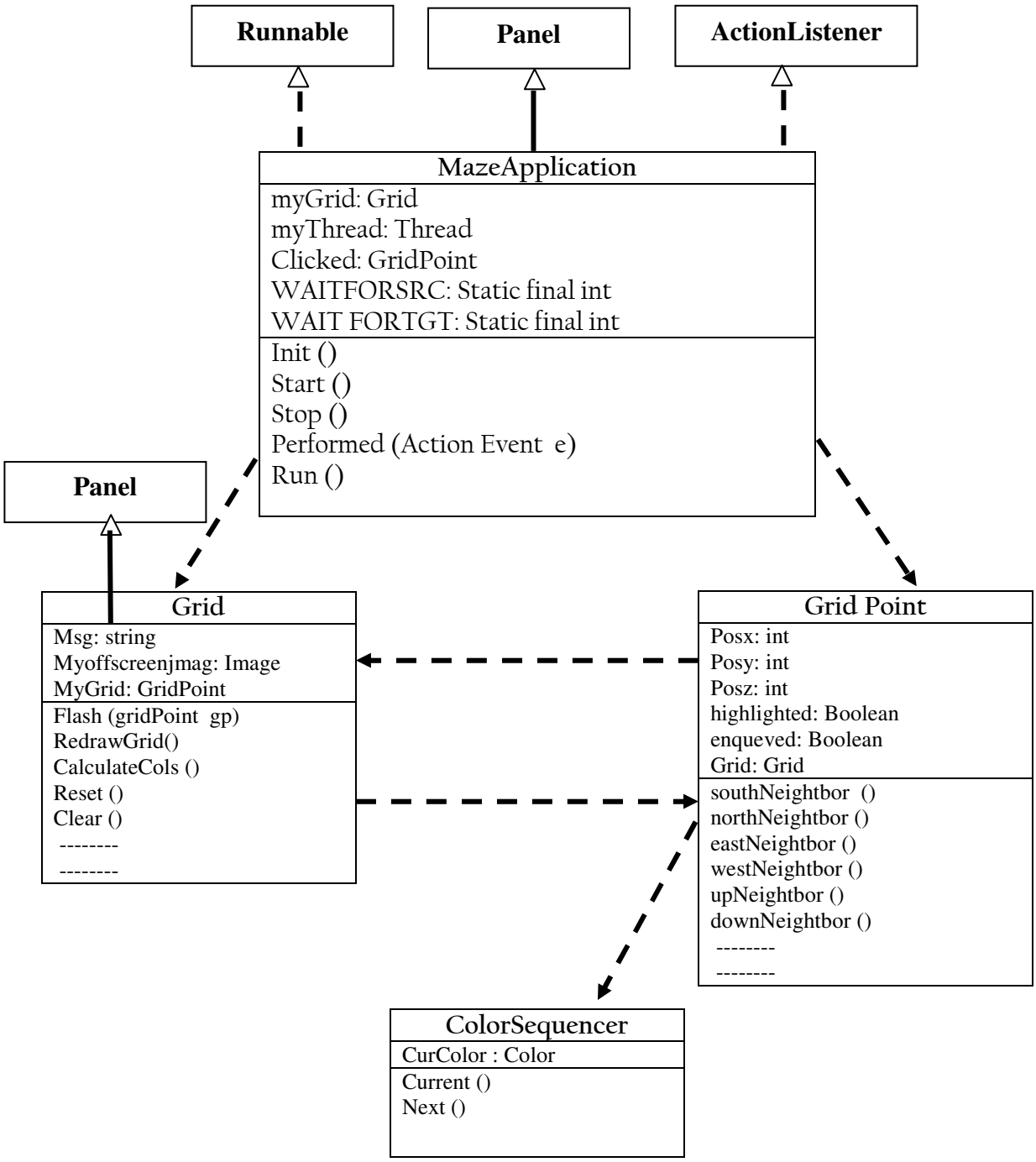
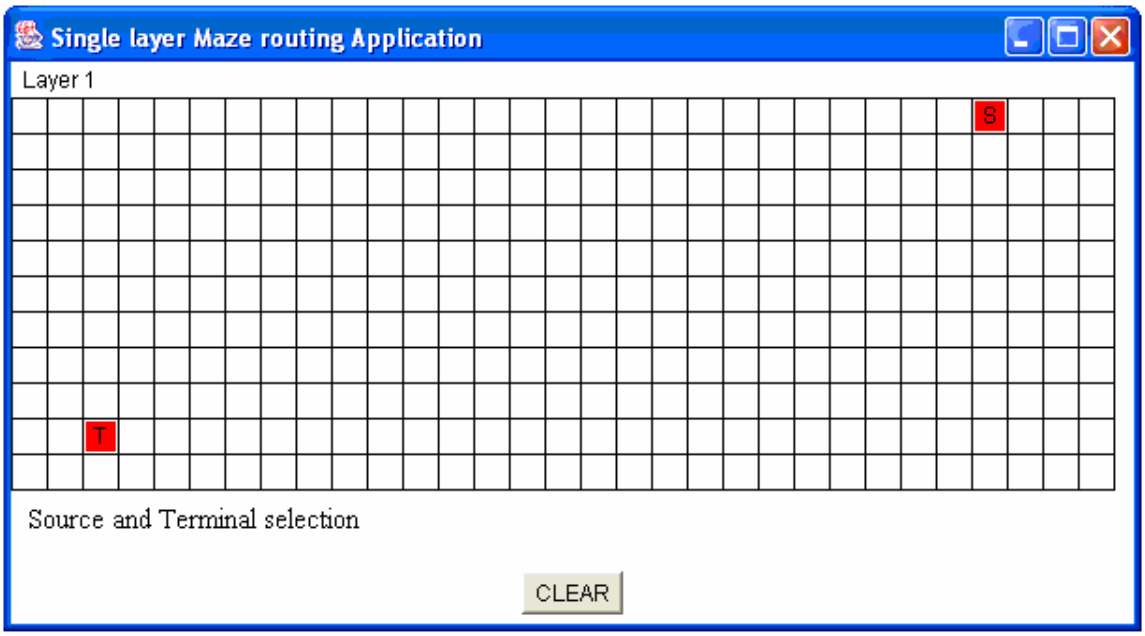
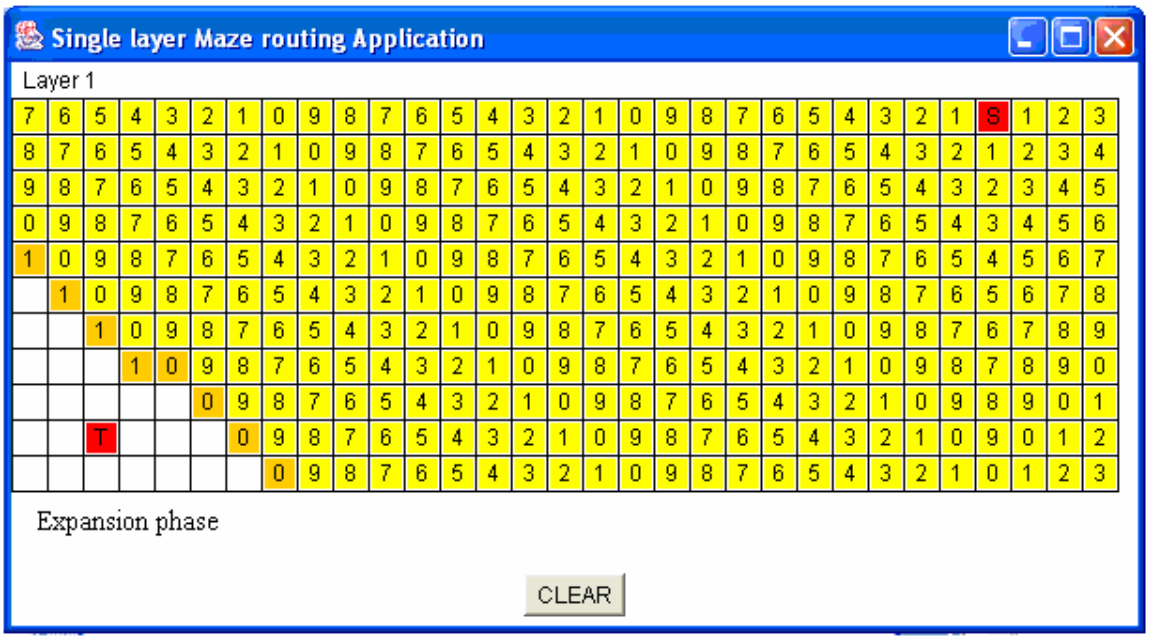


Figure 6.9 Lee's maze router class diagram

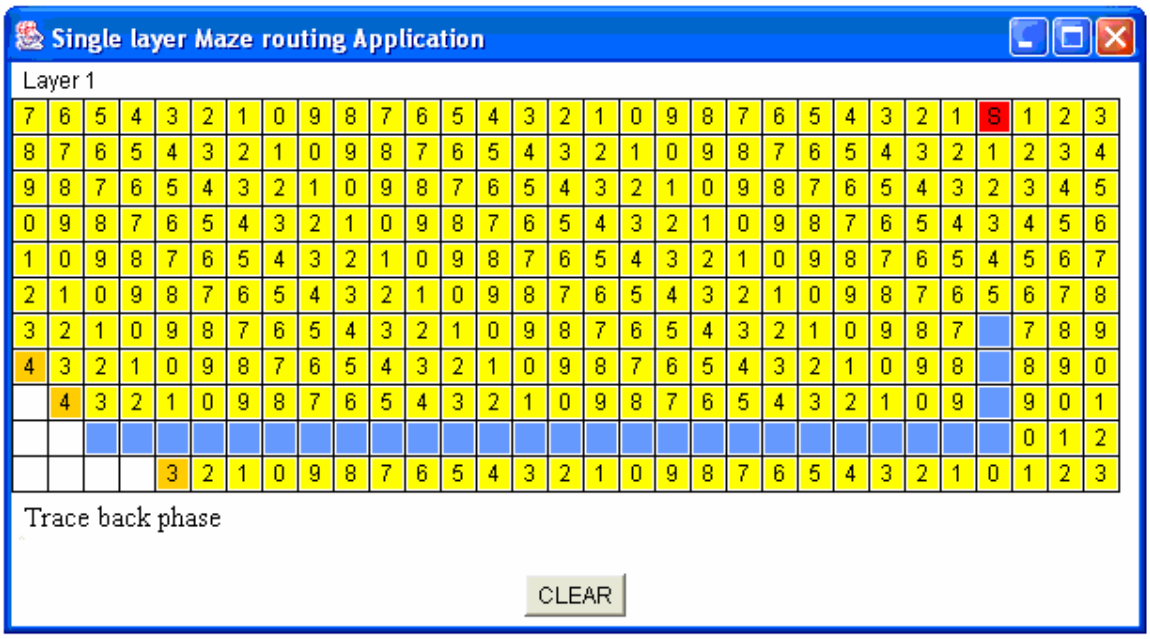
6.2.2 Single layer maze routing



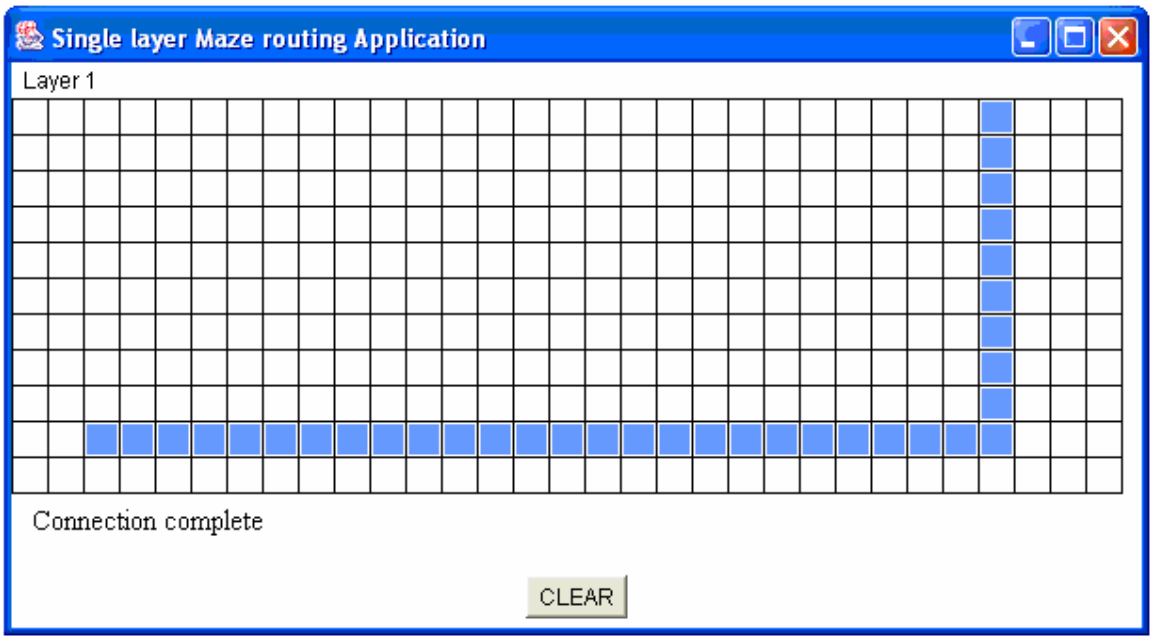
(a)



(b)



(c)

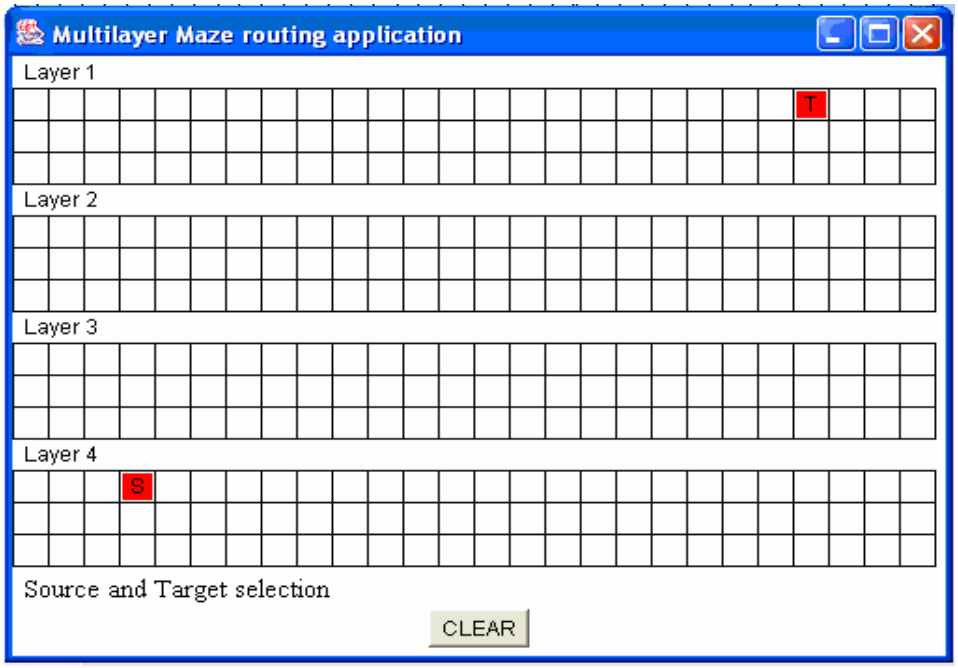


(d)

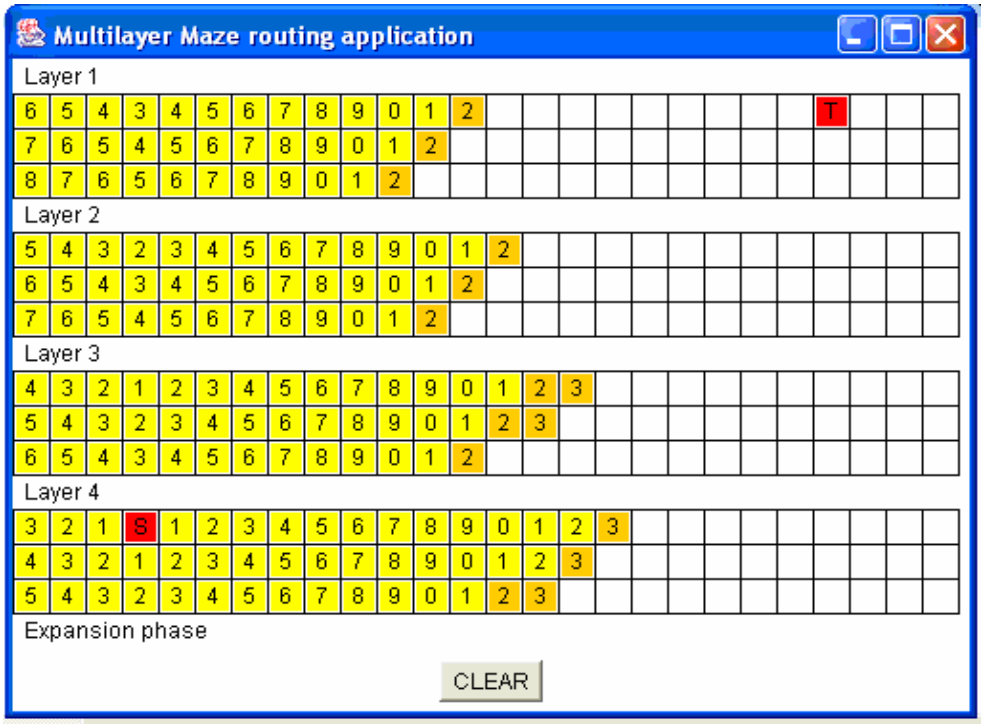
Figure 6.10 (a), (b), (c), (d) single layer maze routing implementation outputs

In figure 6.9 the routing is basically a two terminal routing, i.e the router is capable of routing only two terminals of the same net at a time. Figure (a) shows selection of two terminals of the same net. The first terminal selected is considered as a source terminal the second terminal is considered as a target terminal as described in the figure. Figure (b) shows the expansion phase of the algorithm, it basically employs a breadth first search mechanism labeling each grid point with its distance from the source terminal. Figure (c) shows the trace back phase after the expansion phase is completed. The expansion phase makes sure that each node from the target to the source has been labeled with its distance from the source hence the tarceback phase follows the shortest distance from the target to the source to find the shortest path. Figure (d) shows the completed interconnection between the source and the target.

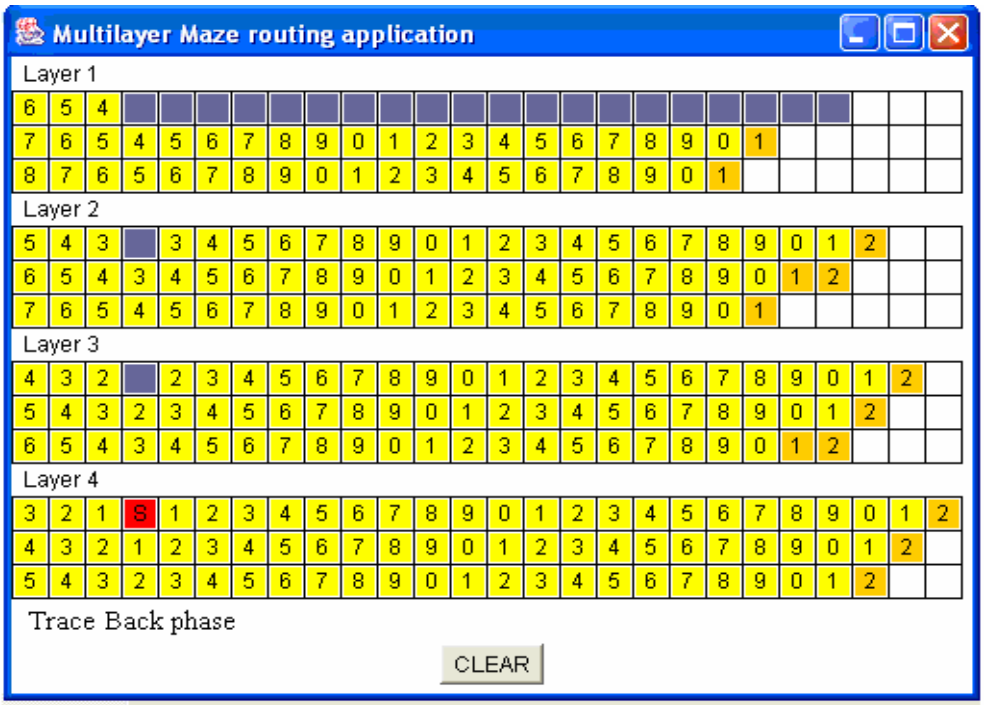
6.2.3 Multilayer maze routing



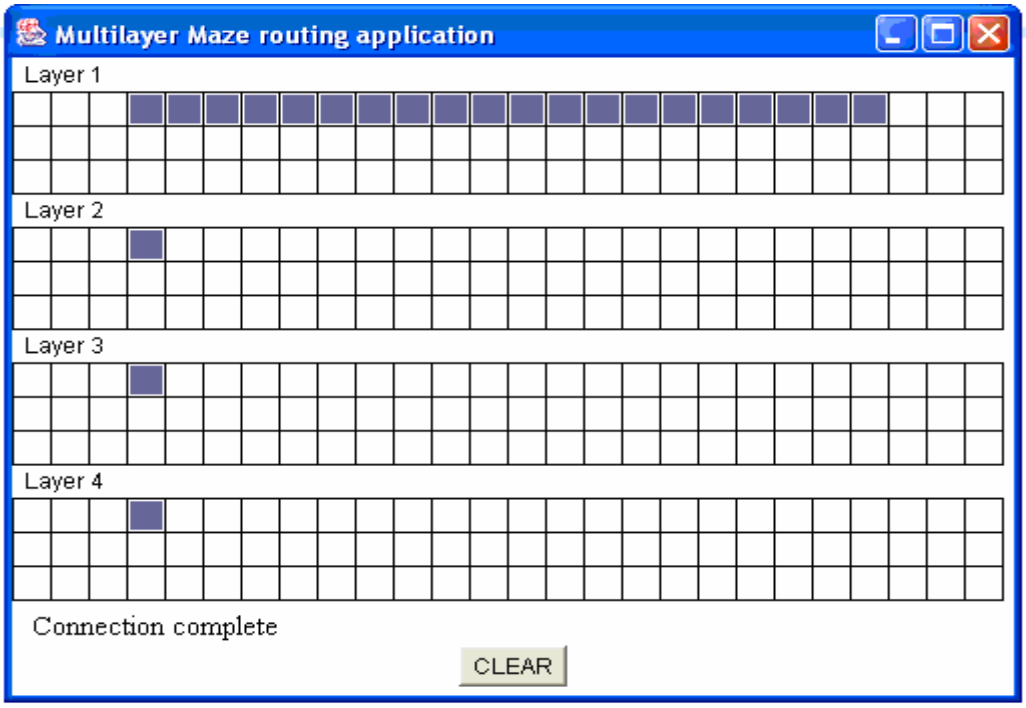
(a)


Multilayer Maze routing application
Layer 1
Layer 2
Layer 3
Layer 4
Expansion phase
CLEAR

(b)


Multilayer Maze routing application
Layer 1
Layer 2
Layer 3
Layer 4
Trace Back phase
CLEAR

(c)



(d)

Figure 6.11 (a), (b), (c), (d) multilayer maze routing implementation outputs

The multilayer implementation is basically the same with single layer implementation except here the number of layers is more than one; i.e. the source and target terminals can be located in different layers. Expansion and trace-back phases have been indicated in the figures.

As described at the beginning of this section one objective of studying and implementing Lee's maze routing is to resolve overlapped interconnection results from left edge channel routing algorithm. Generally speaking, the maze router creates an interconnection between two terminals of the same net with minimum path as long one such path exists. The detailed description of the two way solution to all channel routing problems has been described in the next section.

6.3 A Two way Solution to channel routing problems

One of the important implications of my thesis has been described in this section. In the literature survey part the pros and cons of different routing algorithms are discussed especially left edge channel routing algorithm. The inherent limitation of left edge channel routing algorithm is its inability to resolve vertical constraint cycles. However the algorithm has good characteristics of completing a routing problem with minimum number of tracks if the problem is free from vertical constraint cycles as proved by Navid Sherwani.

> **"***Given a two-layer channel routing problem with no vertical constraints, LEA produces a routing solution with minimum number of tracks*". **[3]**

In order to exploit the stated advantage the overlapped wires due to vertical constraint cycles and any left out terminals should be dealt with to complete the routing. Two possible solutions have been investigated: rip-up rerouting and localized maze routing.

The selection of Lee's maze routing algorithm among the different maze routing algorithms is due to its simplicity. The two way algorithm has been presented by discussing practical results exposing the limitation of left edge channel routing algorithm. The famous Deutsche channel routing problem has been routed using the first router (LEA) where incomplete and overlapped interconnections were detected. Maze router which deals with difficulties faced by the first router has also been discussed in this section.

6.3.1 Vertical constraint cycle in left edge channel routing algorithm.

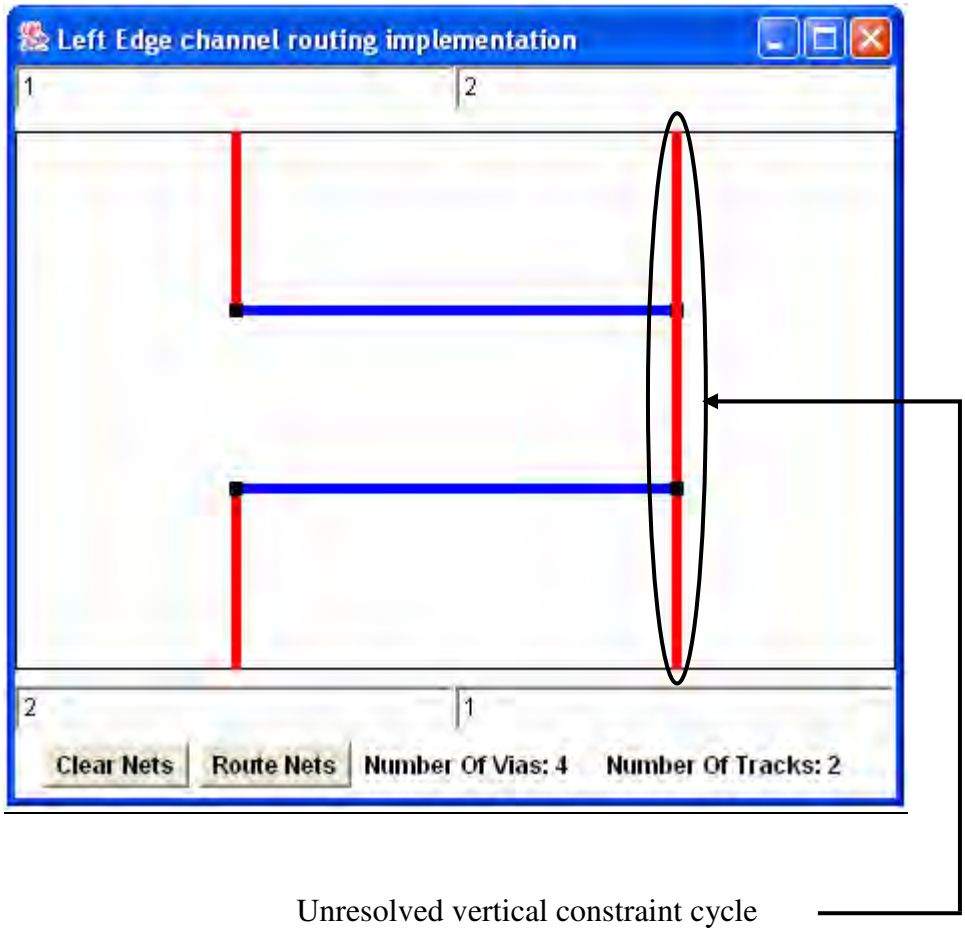
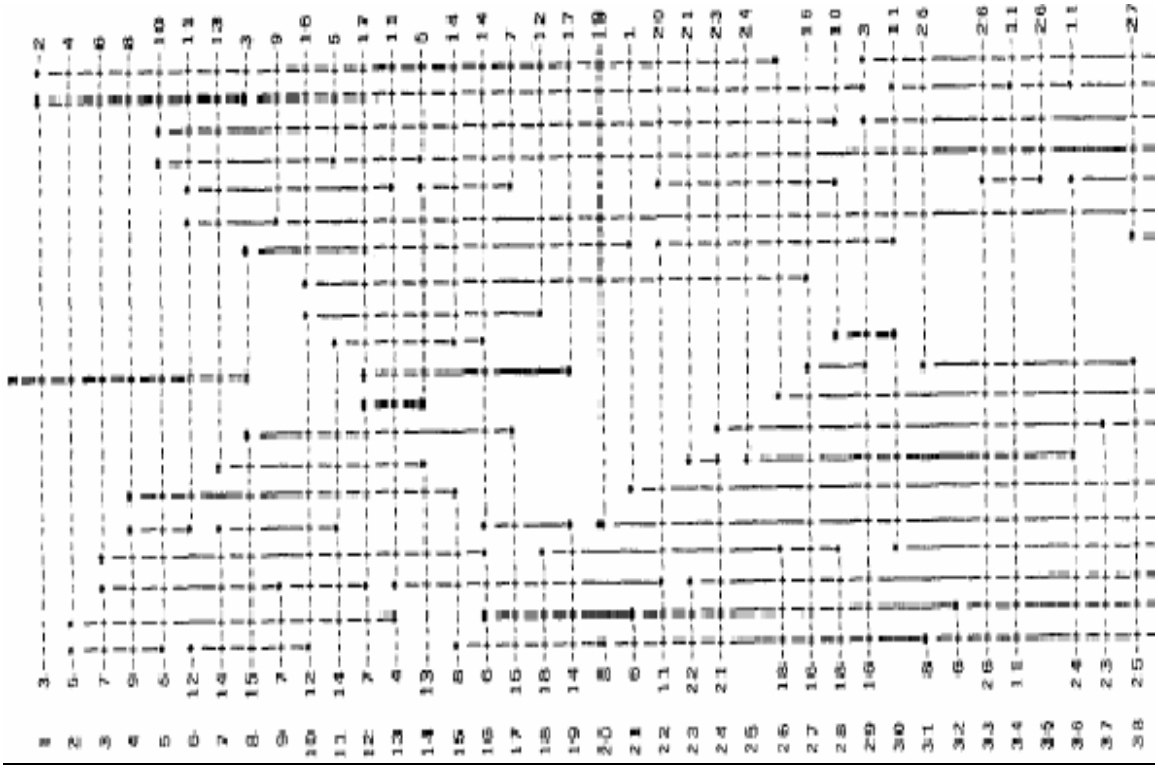


Figure 6.12 limitation of left edge algorithm

Figure 3.6 shows the phenomenon with vertical constraint cycle. Figure 6.11 is an implementation result to show the limitation of left edge algorithm in solving vertical constraint cycles. Vertical constraint cycles can be avoided by moving terminals. This may require additional columns and tracks.

6.3.2 Channel routing problem with vertical constraint cycle

In this section I have considered a channel routing problem with vertical constraint cycle. Figure 6.12 (a) shows a copy of segment of a problem from **[36].** Inadvertently the last pin at the right top is changed from 27 to 7 in the implementation I have used. This made the problem to have vertical constraint cycles; I have used the same problem to evaluate how the left edge algorithm deals with vertical constraint cycles.



(a)

Figure 6.13 (b) shows the routing result of the same problem solved using Left Edge algorithm.

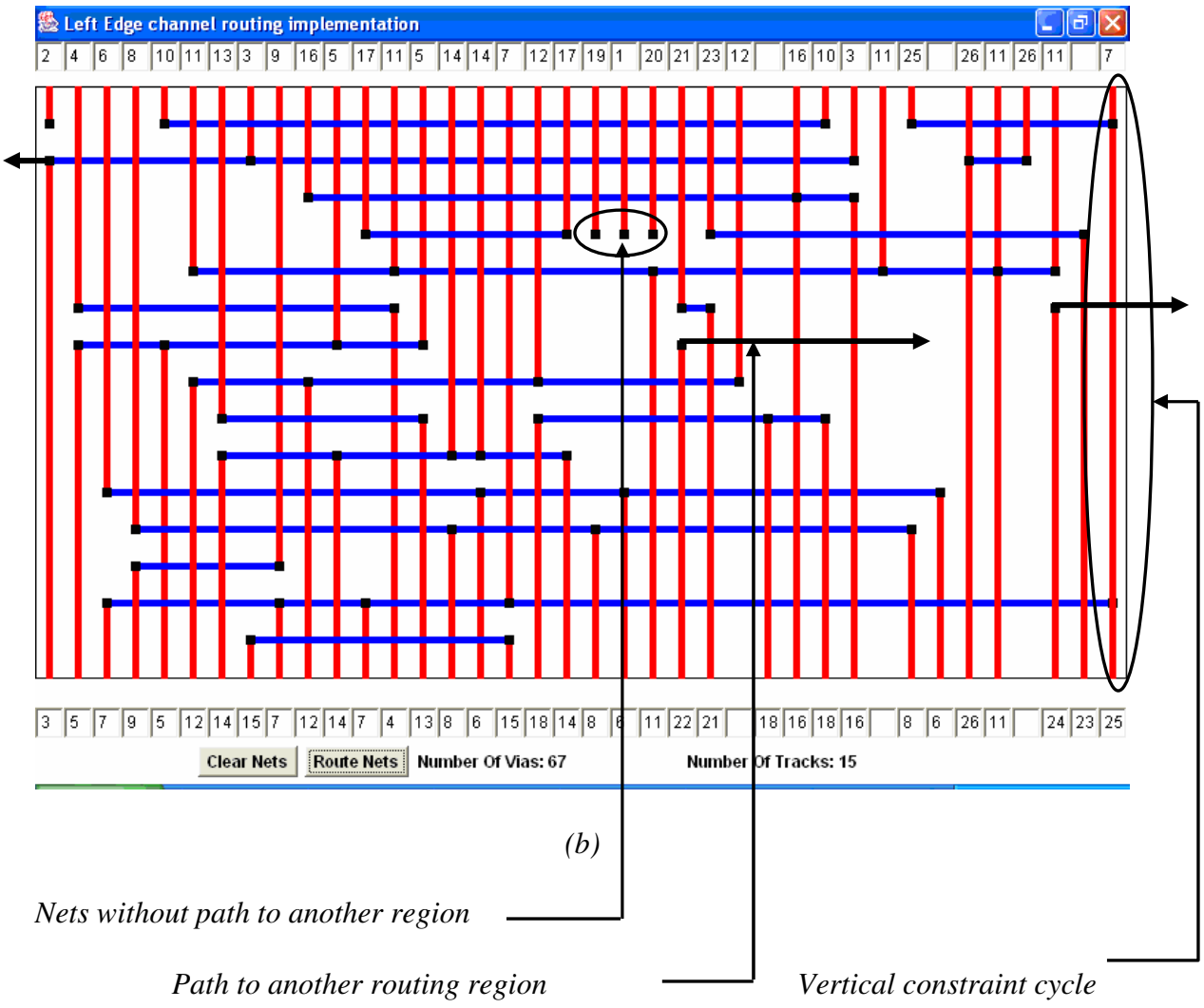


Figure 6.14(a), (b) channel routing problem with vertical constraint cycle

Results: Channel width (number of tracks used) = 15 **[36]**

Vias used = 67

As can be seen in figure 6.12 left edge algorithm ended up overlapping the interconnection wires of two nets involved in vertical constraint cycle. The above result is obtained for the incomplete solution to the problem and shouldn't be used for comparison with other algorithms. The complete solution to the problem is given in the two way solution to channel routing problems discussed at the end of this section.

It is clearly seen from the figure that some of the nets of the problem have been left out like nets **19,1,20.** In order to complete the routing additional tracks, vias and columns are required. In order to create pathways for nets **19, 1** and **20** at most two additional tracks are necessary one extending to the left the other extending to the right. It is also possible to let the interconnecting path from terminal **1** go down the routing region until it finds an unblocked track to extend its path to a next routing region; in this case we have one track reduction. In order to eliminate the vertical constraint cycle at the left of the figure one additional track and one additional column are required, however this is not always a requirement, sometimes adding only a column without the need to add additional track would solve the stated problem.

6.3.3 Two way solution in detail

In this section a two way solution strategy procedure is discussed in detail. The first strategy is to start the routing with left edge channel router and identify any incomplete nets and overlapped nets, then to launch the second phase routing using a maze router which basically identifies and routes any incomplete nets. Additional tracks and columns are added as required.

Note

The incomplete routing result of the problem in figure 6.12 is caused by the following two points.

1. Not all terminals of a net are located in the same routing region. Therefore, there should always be a path way (unblocked track) which leads from one terminal of the net to another terminal of the same net located in a different routing region. Terminals **19,1,20** from figure 6.12 are left with out a path to another routing region.
2. Overlapping nets due to vertical constraint cycles. Terminals of nets **25** and **7** are overlapping.

Solution to problem 1

For every terminal which is left alone in the current routing region there should be a path way either to the right or to the left of the routing region.

- If there is an empty track segment leading to another routing region no additional track would be required, however if track segments to the left and right of the terminal are already occupied then one additional track needs to be inserted.
- If two or more number of terminals belonging to different nets occur on the same track (like terminal 19, 1 and 20) then at most half as many number of tracks as the number of left out terminals need to be inserted. One track would suffice to create a path way for two terminals to another routing region, one leading to the right and the other to the left.

Solution to problem 2

The number of constraint cycles need to be identified first and should be cleaned.
According to the illustration in figure 3.6 to alleviate one vertical constraint cycle at most one additional track and one additional column is necessary. Hence for each and every vertical constraint we encountered one column and one track need to be inserted.

The two way solution strategy to channel routing problem is not completely automated. The procedure to complete the routing is presented below.

1. Launch the first router i.e. left edge channel router assuming all problems are free from vertical constraint cycles. (Automated)
2. If there are no vertical constraint cycles and all terminals leading to another routing region are provided with an unblocked path, the first engine results in a complete solution to the problem else,
3. Apply the above two solutions in order to remove vertical constraint cycles and to give path way for left out terminals leading to another routing region so that the second router (maze router) completes the remaining interconnection. (manual)
4. Based on the routing result of the first router, indicate blocked tracks and column on the grids of the maze router. (manual)
5. Launch the maze router by selecting the source and destination terminals. (Automated).

By following the above procedure for the two way algorithm, it is possible to complete channel routing problems with vertical constraint cycles and with "lonely" terminals in a given routing region.

Figure 6.13 describes the result of localized maze router. Each column and track of the routing area in the localized maze router is assumed to be proportional with the column and track of the routing area in left edge algorithm result described in figure 6.12.

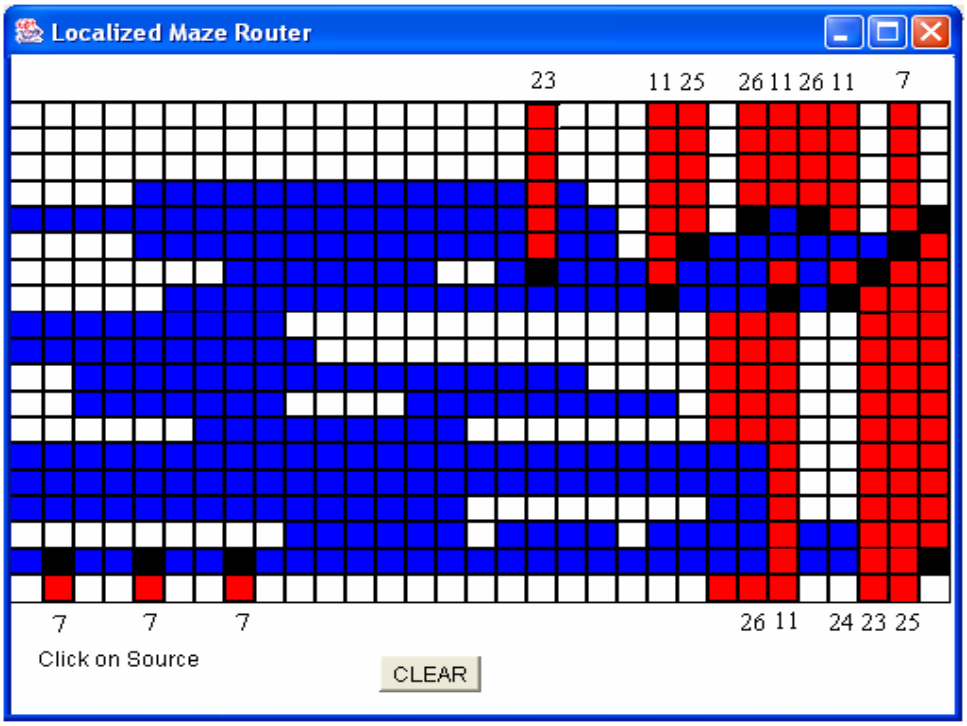


Figure 6.14 representing blocked paths on the maze

Note:

- As described in the procedure of the two way algorithm all blocks (used tracks and columns in the first router) around the path of the terminals to be connected should be represented on the maze routing area before the maze router commences its routing.
- Figure 6.13 is a supper-imposed result of two layers one for horizontal wires (the blue ones) the other for the vertical wires (red ones).
- The black wires (vias) connect wires of two layers.
- Terminal identification numbers are inserted manually.

By combining figure 6.12 (b) and figure 6.13 manually, the complete routing solution of a problem in figure 6.12 (a) is shown in figure 6.14.

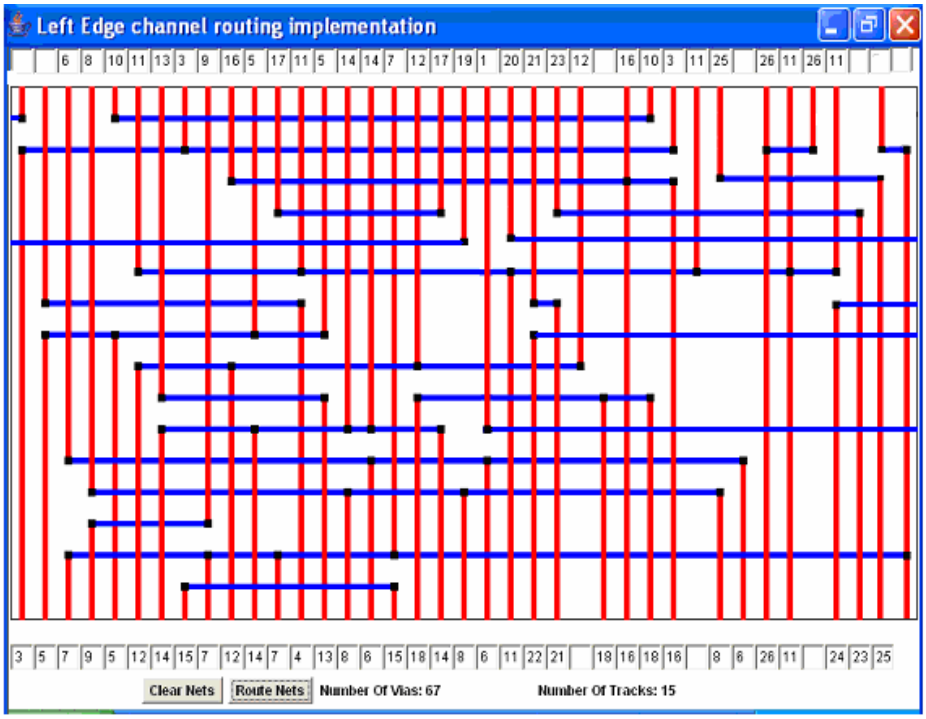
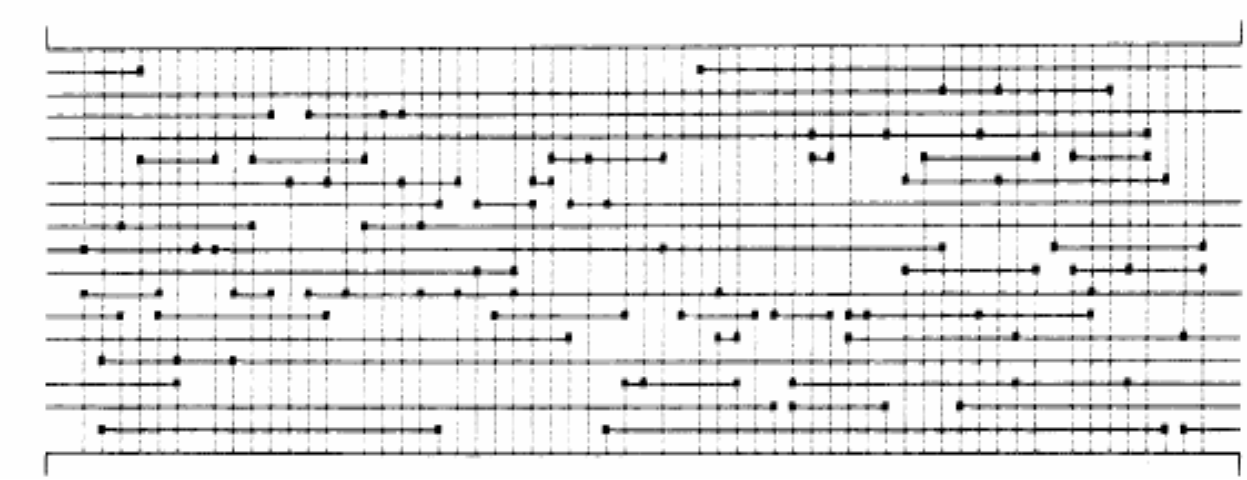


Figure 6.15 a complete solution to problem in figure 6.12(a)

6.3.4 Another benchmark for the two way solution

In this section I have considered another benchmark channel routing problem to test the proposed two way channel routing algorithm. The benchmark problem is of length 61. figure 6.15 is a copy of the problem solved using an efficient channel router algorithm by Takeshi Yoshimura and Ernest S. Kuh, Fellow, IEEE **[43].**



`*Figure 6.16 channel routing benchmark problem (example 3b)*

They obtained the following result

Results: Channel width (number of tracks used) = 17
Vias used = 107

According to the two way solution strategy the channel router is the first router to start the routing solution and it will be followed by the localized maze router as outlined in the previous sections of this thesis. Figure 6.16 show routing result of the problem using left edge router only.

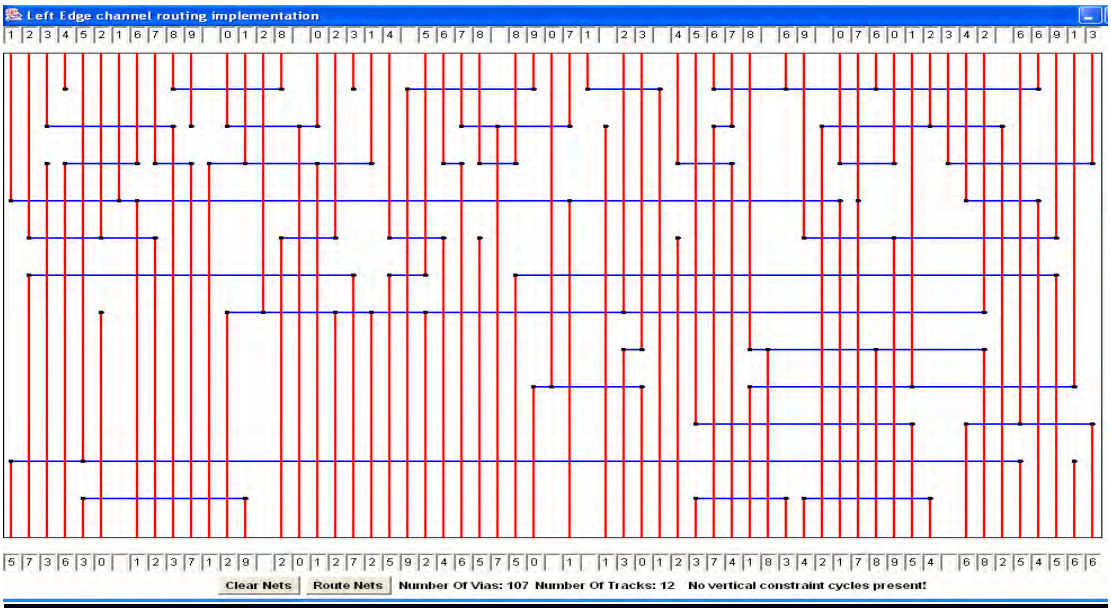


Figure 6.17 result of LEA algorithm to problem in figure 6.15

As can be seen in the routing result of left edge algorithm, the solution is not complete nets like 4, 9, 13, 47, 38, 40, 35, 41, 42, 47, 46 are left out with out having an exit from the given routing region. All these nets consist of a single terminal in the considered routing region; the left edge algorithm always makes sure that terminals within the considered routing regions are connected to each other regardless of constraint violations. The problem with the left edge algorithm is it doesn't create an exit path for single terminal nets in the considered routing region. In figure 6.17 I have shown how the localized maze router completes the left out terminals by the left edge algorithm.

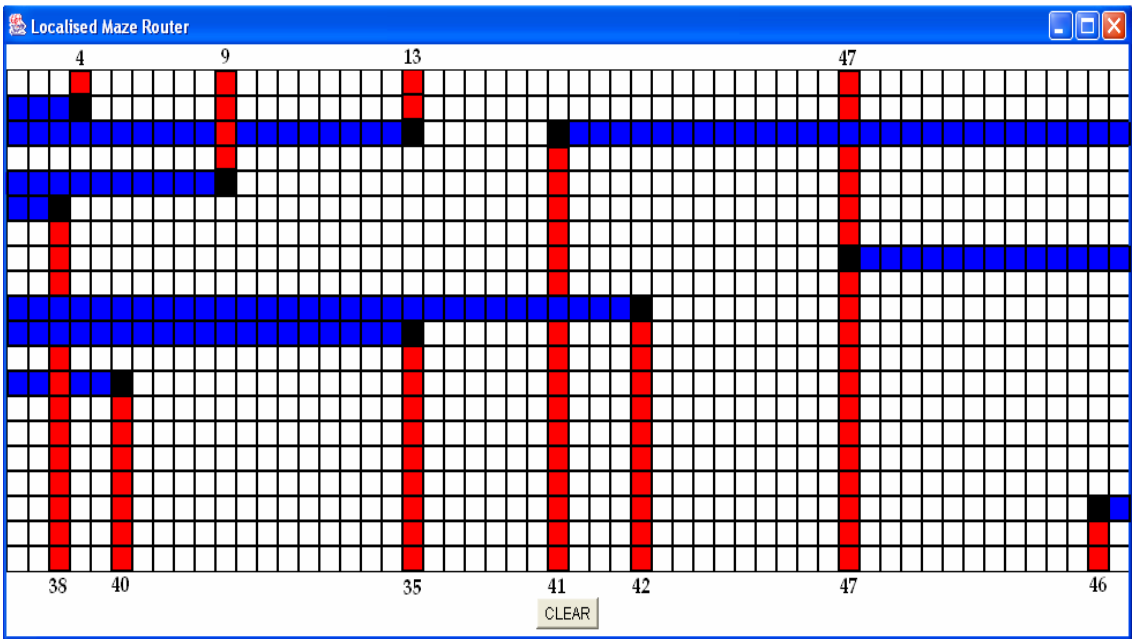


Figure 6.18 result of localized maze router to left out terminals in figure 6.16

As can be seen in the routing result of the localized maze router, all terminals which were without an exit to another routing region are supplied with the necessary path. In creating the shortest path between terminals and an exit point additional tracks are used. The left edge algorithm used 12 tracks to create the initial routing and the localized maze router used 5 additional tracks to complete the routing. The superimposed of the results of the two routers is shown in figure 6.18.

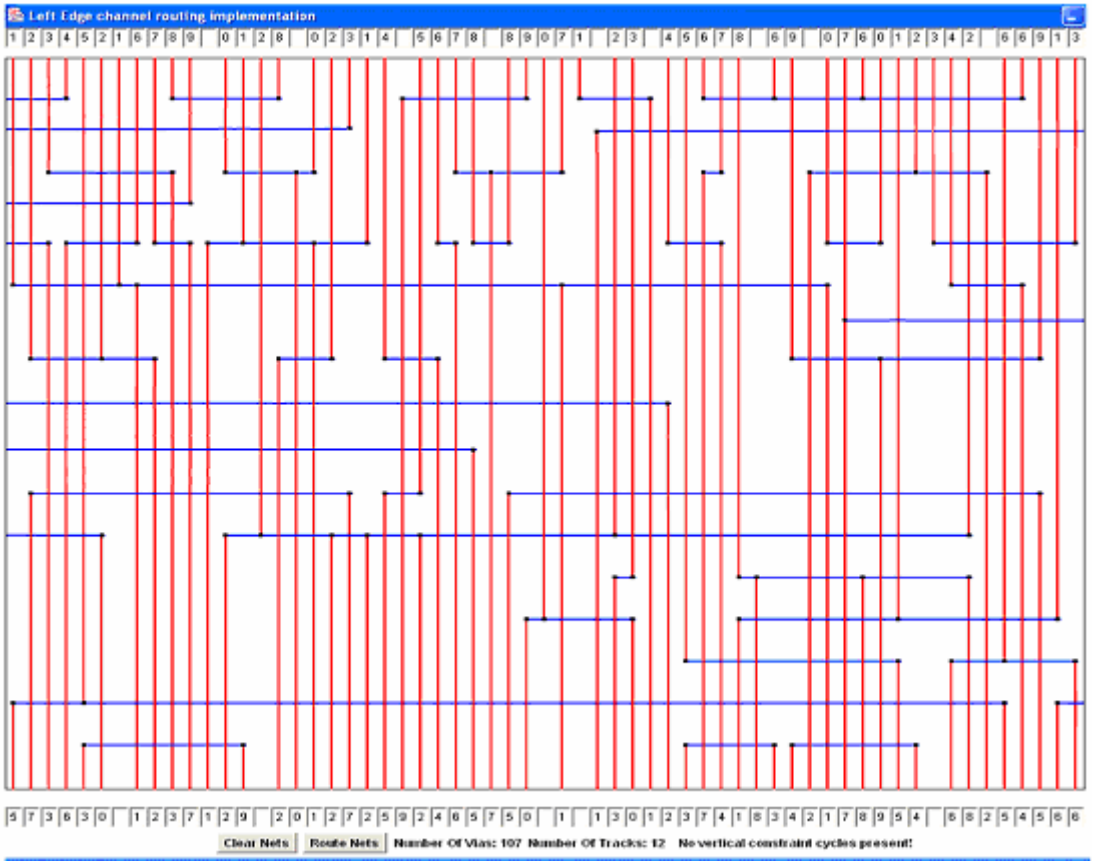


Figure 6.19 complete solution to routing problem in figure 6.15

In this section I have presented comparison results of the two way algorithm and other algorithms over the same benchmark problem. Routing results of benchmark problem, referred as example 3.b in the literatures with channel density of 17, is described in table 6.1.

Router	**Tracks**	**Vias**
Two way algorithm based router	17	107
Efficient channel router **[43]**	17	107
Genetic algorithm based router **[40]**	17	-
Permeation router **[44]**	18	-
Topological channel router **[45]**	38	94

Table 6.1 comparison of routing results

6.3.5 Comparison of results of routing algorithms

As can be seen in the routing results of table 6.1 the number of tracks used in the two way solution is equal to the channel density, which is a lower bound to the number of tracks required to complete the routing. Permeation router result over the same routing problem is one more than required by the two way solution. On the other hand topological router doesn't seem to give optimal results in the count of track but provides significant reduction in the number of vais required for interlayer connection. I have also tried the two way routing algorithm for other benchmark problems and I have obtained near optimal results which can be used for practical channel routing problems. According to **[46]** channel routing algorithms with one or two number of tracks beyond channel density can be used for practical problems.

Chapter seven
Conclusion and Recommendation

In this chapter conclusive remarks on the major achievement of the thesis is presented followed by recommendation on the further work. One of the subclasses of switchbox routing problem I have investigated is a single row routing problem. In the single row routing problem Gallai asserts, *"The minimum width routing solution of the problem is the density of the problem and such a solution can be found in liner time"*. **In my work I have constructed an interval graph corresponding to the horizontal segments of nets of the problem and verified that an optimal coloring of the graph which determines the minimum width solution of the problem can be obtained in linear time.** I have also implemented single row routing problem based on Gallai's assertion.

Another subclass of switchbox routing problem I have considered in my work is the channel routing problem. Among the various channel routing algorithms I have worked on the left edge channel routing algorithm. In my work I have exposed the inherent limitations of the left edge algorithm. **The basic left edge algorithm can't solve vertical constraints and constraint cycles and can't be used to route multi terminal nets.**

I have also considered a four sided switchbox routing problem in my work. David Szeszeler in his PhD dissertation asserts, *"Every square shaped switchbox routing can be solved using six layer in Manhattan model"*. **I have investigated two square shaped switchbox routing problems and verified that the minimum layer requirement for square shaped Manhattan model switchbox routing problem is six.**

Another important investigation in my thesis work is on maze routing algorithms, Maze routing algorithms are analyzed and implemented in this thesis in order to see whether they can complimentarily be used with switchbox routing algorithms. **Through the implementation I have arrived to a conclusion that specialized maze routing algorithm can be used in conjunction with switchbox routing algorithms especially in the clean up phase where overlapped interconnections can be avoided.**

One of the most important contributions of my work is a proposal for the two way solution strategy to channel routing problems. The two way solution exploits the strong sides of the two algorithms: left edge channel router algorithm and Lee's maze router algorithm. The strong side of left edge algorithm is that **it results in a minimum track solution to channel routing problems without vertical constraint cycles.** The strong side of Lee's maze routing algorithm is that **it always guarantees to find the minimum path between two terminals of a net.** The weak side of left edge algorithm is its incapability to solve vertical constraint cycles. **However vertical constraint cycles are inherently localized problems and can be solved by another router efficient in connecting terminals within limited locality.** This is exactly the place to employ Lee's maze router which is efficient for such small area routing. **I have also made a test of the two way solution using benchmark problems and obtained optimal and near optimal results. For benchmark problem example 3.b, the two way algorithm achieves the same number of tracks as the density of the channel routing problem. In comparison to some already existing algorithms like permeation router and topological router the two way solution provides less number of tracks.** With respect to the count of via, the inherent nature of left edge algorithm which doesn't allow doglegs helps in minimizing the necessary vias in the two way routing solution.

Implementations of the different switchbox algorithms have been an interesting undertaking for me. One immediate extension of my work would be to integrate the two routers (left edge channel router and localized maze router) into one application in order to avoid the manual intervention. The first router can pass results of its routing to the second router through a file. The second router will know the areas already used (blocked

areas) by the first router and makes sure that only unblocked areas are available for it's routing. The other reasonable extension would be implementation of a four sided switchbox. To begin with, it would be easy to visualize the four sided switchbox as consisting of two channel routing problems, i.e. vertical and horizontal channels.

References

[1] Behind the Front Panel: The Design & Development of 1920's Radios (Paperback) by David Rutland, September 1994

[2] Design of VLSI systems: VLSI e-book, by Dr. Mylenk and Dr. Yusuf Leblebici, 1998

[3] NAVEED SHERWANI, Intel Corporation: *Algorithms for VLSI physical design Automation,* VLSI text book, Third Edition, 1999

[4] SHAAHIN HESSABI: advanced VLSI design, overview of VLSI design, Department of Computer Eng., Sharif University of Technology

[5] DAVID SZESZLER: Combinatorial Algorithms in VLSI Routing PhD Dissertation 2005.

[6] DAEBUM LEE EECS: Aspects of Full-Custom VLSI Microprocessor Design and Implementation: Department, University of California, Berkeley, Technical Report No. UCB/CSD-89-504, 1989

[7] B. KICK, U. BAUR: Standard-cell-based design methodology for high-performance support chips: IBM journal of research and development. 1997

[8] JENS LIENIG TANNER RESEARCH, Inc*: Physical design of VLSI circuits and application of Genetic algorithms, 1997.*

[9] SHAWKI AREIBI AND ZHEN YANG: Congestion Driven Placement for VLSI Standard Cell Design:School of Engineering, University of Guelph, Ontario, Canada.

[10] Computer-Aided Design of VLSI Circuits, by Newton, A.R, IEEE, Proceedings, vol. 69, Oct. 1981, p. 1189-1199.

[11] Detailed routing concepts: routing considerations and routing models, lecture slides.

[12] KIA BAZARGAN*, VLSI Design Automation I: Routing, Lecture slides:* University of Minnesota*, 2004.*

[13] ANDRAS RECSKI, GABOR SALAMON and DAVID SZESZLER: *Improving size-bounds for subcases of square shaped switchbox routing, 2003.*

[14] Gallai T., His unpublished results were announced in A. Hajnal Annales Univ. Sci. Budapest. (1958) 1, 115-123.

[15] A.HASHIMOTO AND J.STEVENS. Wire routing by optimization channel assignment within large apertures. Proceedings of the 8th Design and automation workshop, pages 155-163, 1971

[16] H. BOLLINGER. A mature system for pc layout. Proceedings of first International Printed Circuit Conference, 1979

[17] PATRICK H. MADDEN: VLSI physical design Automation lecture slides at Binghamton

[18] KURT KEUTZER, RICHARD NEWTON, and JASON CONG: Routing in Integrated Circuits CS, UCLA Prof. EECS, UC Berkeley, 1997.

[19] JENS LIENIG TANNER RESEARCH, INC: Channel and Switchbox routing with minimized cross-talk a parallel genetic approach

[20] VLSI Datapath Choices: Cell-Based Versus Full-Custom Andrew Liang Ping Chang Concurrent VLSI Architecture Group, Artificial Intelligence Laboratory Massachusetts Institute of Technology and Computer Systems Laboratory Stanford University,1998.

[21] Physical Design for Nanometer ICs Yao-Wen Chang Graduate Institute of Electronics Engineering Department of Electrical Engineering National Taiwan University Spring, 2005.

[22] CHANNEL HEIGHT ESTIMATION IN VLSI DESIGN, *Lun Li, Theodore W. Manikas,2002.*

[23] ON OPTIMAL INTERCONNECTIONS FOR VLSI, Andrew B. Kahng University of California Los Angeles, California, USA and Gabriel Robins University of Virginia Charlottesville, Virginia, USA

[24] A Force-Directed Maze Router, Fan Mo, Abdallah Tabbara and Robert K. Brayton Department of EECS, University of California at Berkeley.

[25] CAD Prelim Study Sheet, Summer 2002, Version 1.0, Doug Densmore January 23, 2003

[26] GLOBAL ROUTING FOR VLSI STANDARD CELLS**,** *Hao Sun and Shawki Areibi* School of Engineering,University of Guelph, CANADA N1G 2W1.

[27] Introduction to Computer Design, Lecture 14-Routing
Eli Bozorgzadeh, Computer Science Department-UCI, winter 2005

[28] Rabaey, Chandrakasan, Nikolic "Digital Integrated Circuits: A Design Perspective". Prentice Hall/Pearson, 2003.

[29] THE CHIPMAP™: VISUALIZING LARGE VLSI PHYSICAL DESIGN DATASETS, PhD dissertation, December 2002

[30] Requirements for Models of Achievable Routing**.** Andrew B. Kahng, Stefanus Mantik, Dirk Stroobandt

[31] T.G Szeymanski. Dogleg channel routing is NP-complete. IEEE Transactions on Computer Aided Design, CAD-4:31-41, January 1985.

[32] D. N. Deutsch. Compacted channel routing. Proceedings of IEEE International conference on Computer Aided Design-85:223-225, 1985.

[33] J Reed, A. Sangiovanni-Vincentelli. Convergence and finite time behavior of simulated annealing. Proceedings of the 24th Conference on Decision and Control, 1985.

[34] Middendorf, M., Manhattan channel routing is NP-complete under truly restricted settings, Chicago J. Theoret. Comput. Sci., Article 6, 1996

[35] Deutsch, D, A dogleg channel router, Proc. 13th Design Automation Conference. (1976), 425-433.

[36] Marek-Sadowska, M. and E. Kuh, General channel-routing algorithm, Proc. IEE (GB) (1983) 130, 83-88.

[37] C.Y. Lee. An algorithm for path connections and its applications. IRE Transactions on Electronic Computers, 1961.

[38] S.B Akers. A modification of Lee's path connection algorithm. IEEE Transactions on Computers, February 1967.

[39] A "GREEDY" CHANNEL ROUTER ' by Rouald L. Rivest and Charles M. Fiduccia

MIT Laboratory for Computer Science, Cambridge, Mass. 02139, and GE Research and Development Center, Schenectady, New York 12301, March 1981

[40] Genetic Algorithm For Restrictive Channel Routing Problem
Vladimir N. Davidenko, Victor M. Kureichik, Victor V. Miagkikh

[41] S. E. Hambrusch: Channel routing in overlap models, IEEE Trans. Computer-Aided Design of Integrated Circ. Syst. CAD-4 (1985), 23-30.

[42] Golda, B., B. Laczay, Z. _A. Mann, Cs. Megyeri and A. Recski, Implementation of VLSI Routing Algorithms, Proc. IEEE 6th Internat. Conf. on Intelligent Engineering Systems, Croatia, 2002, 383-388.

[43] Efficient algorithms for channel routing by Takeshi Yoshimura and Ernest S. Kuh, IEEE transactions on computer-aided design of integrated circuits and systems January 1982.

[44] A permeation router by YOICHI SHIRAISHI, and JUN'YA SAKEMI, IEEE transactions on computer-aided design, May 1987

[45] Topological routing using geometric information by shinichiro haruyama, D.F,Wong, and Don Fussell

[46] Generating more compactable channel routing solutions, by Jingsheng Cong and D.F. Wong, department of computer Science University of Illinois