A Multiple-objective ILP based Global Routing Approach for VLSI ASIC Design

by

Zhen Yang

A thesis

presented to the University of Waterloo

in fulfilment of the

thesis requirement for the degree of

Doctor of Philosophy

in

Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2008

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

A VLSI chip can today contain hundreds of millions transistors and is expected to contain more than 1 billion transistors in the next decade. In order to handle this rapid growth in integration technology, the design procedure is therefore divided into a sequence of design steps. Circuit layout is the design step in which a physical realization of a circuit is obtained from its functional description. Global routing is one of the key subproblems of the circuit layout which involves finding an approximate path for the wires connecting the elements of the circuit without violating resource constraints. The global routing problem is NP-hard, therefore, heuristics capable of producing high quality routes with little computational effort are required as we move into the Deep Sub-Micron (DSM) regime.

In this thesis, different approaches for global routing problem are first reviewed. The advantages and disadvantages of these approaches are also summarized. According to this literature review, several mathematical programming based global routing models are fully investigated. Quality of solution obtained by these models are then compared with traditional Maze routing technique. The experimental results show that the proposed model can optimize several global routing objectives simultaneously and effectively. Also, it is easy to incorporate new objectives into the proposed global routing model. To speedup the computation time of the proposed ILP based global router, several hierarchical methods are combined with the flat ILP based global routing approach. The experimental results indicate that the bottom-up global routing method can reduce the computation time effectively with a slight increase of maximum routing density.

In addition to wire area, routability, and vias, performance and low power are also important goals in global routing, especially in deep submicron designs. Previous efforts that focused on power optimization for global routing are hindered by excessively long run times or the routing of a subset of the nets. Accordingly, a power efficient multi-pin global routing technique (PIRT) is proposed in this thesis. This integer linear programming based techniques strives to find a power efficient global routing solution. The results indicate that an average power savings as high as 32% for the 130-nm technology can be achieved with no impact on the maximum chip frequency.

Acknowledgements

I would like to take this opportunity to express my sincere appreciation and thanks to my supervisor professor Anthony Vannelli and professor Shawki Areibi for their guidance and assistance. Many thanks to professor Mohab Anis, professor Geatherinc H. Gebotys, professor Samir Elhedhi and professor Richard Shi for reviewing this thesis. Their invaluable comments made this work a much better one. The work in this thesis has been founded by National Resources of Canada. I would like to thank this institution for supporting me during my studies and making this work possible.

I want to especially thank my husband Wenxin Wang, my sister, my parents and my parents in-law for their continuous encouragement and support.

And finally, many thanks to all my friends and well-wishers who exhorted me to work dedicatedly towards the fulfilment of the objectives of this research.

To

my family

whose love and encouragement helped accomplish this thesis.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The last few decades brought explosive growth in the electronics industry due to the rapid advances in integration technologies and the different benefits of large-scale system design. As a result, System-on-Chip (SoC) designs have become one of the main drivers of the semiconductor technology in recent years. By employing third party intellectual property (IP) cores, designers today can improve design productivity and cut development costs and time. However, as more and more complex functions are integrated into a small package, State-of-the-art VLSI chips, such as the INTEL *Pentium IV* or *Itanium II*, tend to contain hundreds of millions transistors [Kang03]. Designing such a multi-million transistor chip and ensuring that it operates correctly when the first silicon returns is a daunting task that is virtually impossible without the help of Computer Aided Design (CAD) tools [Raba03b].

The phrase associated with the task of automatically designing a circuit using CAD tools is called Design Automation (DA). The ultimate goal of the DA research

field is to fully automate the tasks of designing, verifying, and testing a circuit. Unfortunately, there is still a long way from achieving this goal. No CAD design flow is currently capable of handling the enormous and often contradicting design goals required by the modern VLSI circuits. For such a complicated problem, the feasible approach is to use a divide-and-conquer strategy in which the whole design task is broken down into several sub-tasks that are more manageable for a single design tool.

The VLSI design cycle starts with a formal specification of a VLSI chip that follows a series of steps, and eventually produces a packaged chip. A typical VLSI design process is illustrated in Figure 1.1. The design process of a VLSI circuit

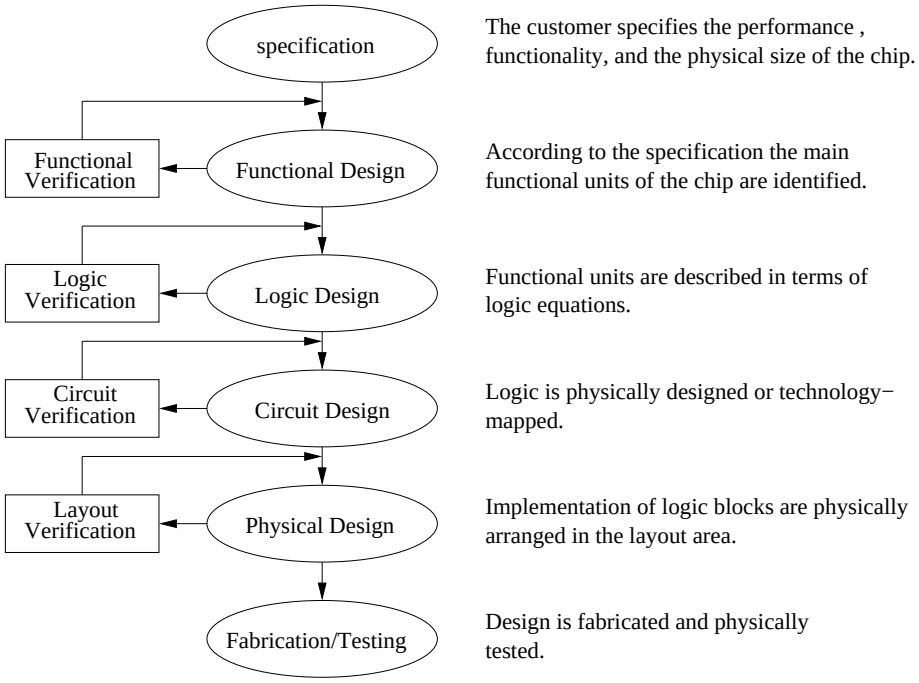


Figure 1.1: VLSI Design Flow

begins with a formal specification of the circuit. The factors to be considered in

this process include: performance, functionality, and physical dimensions. The end results are specifications for the size, speed, power, and functionality of the VLSI circuit. The basic architecture of the circuit is also specified.

Following the specification step, the main functional units of the circuit are determined. In the functional design step the interconnect requirements between the units, area, power, and other parameters of each unit are also identified and estimated [Sher99]. These functional units could either be implemented using Application Specific Integrated Circuits (ASIC) or Field Programmable Gate Arrays (FPGA) design styles. The description of this design step is a high-level description and usually expressed as Register Transfer Logic (RTL).

In the logical design stage, the functional units are described in terms of primitive logic operations (NAND, NOT, etc.). This description could be expressed in a Hardware Description Language (HDL), such as VHDL and Verilog, which can be used in simulation and verification [Sher99].

Following the logical design, a technology-dependent description of the circuit is created. At this design level, the whole circuit is implemented as transistors. In some implementation topologies, logic equations are broken down and mapped to available physical circuit blocks in the circuit topology (called *technology mapping*), or pre-designed logic circuit implementations (e.g., a standard-cell library) [Thom00].

The physical design step involves converting the circuit representation of each component into a geometric representation (also called a layout). Connections between different components are also expressed as geometric patterns. The end result of physical design is a placed and routed design, from which the photolithography

masks can be derived for chip fabrication [Sher99]. Since the physical design problem is an NP-hard problem it is usually broken down into several sub-problems, referred to as partitioning, placement and routing. This thesis is mainly concerned with the global routing stage.

Finally, the wafer is manufactured and diced in a fabrication facility. Each chip is then packaged and tested to ensure that it meets all design specifications and that it functions properly.

Note that in Figure 1.1 the *verification* following each design step plays a very important role in the entire design cycle. The failure to properly verify a design in its early phases typically causes significant and expensive re-design at later stages, which ultimately tend to increase the time-to-market [Kang03]. In addition, although the design procedure has been described as a sequence of steps, the individual design steps are not mutually exclusive. Each step influences subsequent steps, and the results of any step may be used as feedback information to revise an earlier step. Several iterations of the design procedure may be required to obtain a satisfactory design for any given Integrated Circuit.

1.1 Research Motivations

In this section, we will highlight the main motivations driving the PhD research.

1.1.1 Interconnect in DSM Design

In deep sub-micron (DSM) regimes, the shrinking geometries and increasing average wirelength due to the increasing design complexities make interconnect capacitance

C_{wire} become dominant over gate capacitance C_{gate}. It has been reported that the interconnection delay consumes a major portion of the clock cycle today and is projected to continue to do so in the future [Asso97], as illustrated in Figure 1.2. Apart from the circuit delay, wire capacitance also dominates the switched

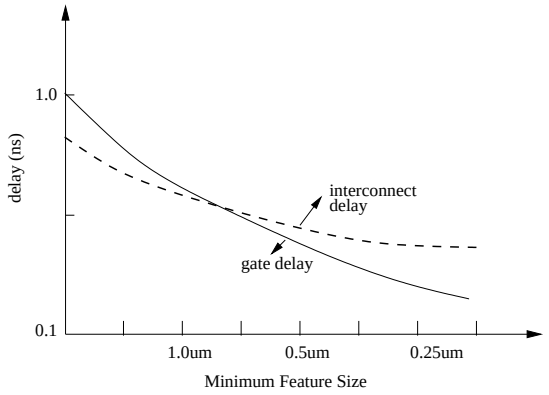


Figure 1.2: Interconnect and Gate Delay

capacitance power, as shown in equation 1.1.

$$P_{switching} = \alpha_{0\to1} \cdot V_{dd}^2 \cdot f_{clk} \cdot (C_{gate} + C_{wire}) \tag{1.1}$$

Since wire load has an important impact on circuit delay and power, global routing which determines the approximate path for each net plays critical role in today's performance-driven low-power VLSI design.

Yet, another important implication of decreasing devices and wire geometries is that the number of wires used to connect the components in the circuits has grown exponentially. As a result, a global routing heuristic that produces excellent results for small size problem may take days or weeks to obtain a good result. Obviously, a computationally expensive technique is often useless to the modern just-in-time

fabrication mentality.

1.1.2 Hierarchical ILP based Global Routing

Since the computation time of a global routing algorithm must be appropriate for today's large circuits, an approach that operates in a reasonable amount of time, while still achieving good solutions is desirable. When solving the global routing problem, sequential approaches, such as Maze router [Kast], tend to route one net at time. Once a net has been routed, it may block other nets which are yet to be routed, and in the worst case, it may lead to unroutable solutions. Therefore, these methods are highly dependent on the order in which the nets are routed. Also, when a sequential approach does find a solution, it is not known whether or not this solution is optimal, or how far it is from the optimal solution. Modeling the global routing problem as a mathematical optimization problem gives a global view of the routing problem and eliminates the dependency on the ordering of the nets. In addition, the evaluation of optimality of solutions becomes possible. However, global routing approaches that operate on flat large circuits tend to have a scaling problem. Accordingly, a hierarchical global routing approach becomes essential to shorten the design period. The hierarchical global routing problem has been studied in the literature [Burs83b, Luk87, Heis91, MS84, Hu85, Gegu00, Cong01a, L02] and the general approaches have been developed: the top-down method, the bottom-up method, and the multilevel method. The hierarchical nature of the multilevel algorithm makes it very scalable to large designs. Moreover, a multilevel hierarchical algorithm is more flexible than other hierarchical approaches. In the coarsening passes, more routing resources information can be obtained, which provides a good

guide to the coarse level path searching. During the uncoarsening process, the fine level router has the flexibility to refine the coarse level result based on more detailed information about local resource and congestion. All these features make the multilevel method attractive to achieve good solutions efficiently. Hence, one motivation of this research is to identify the effectiveness of the multilevel hierarchical techniques on solutions obtained by the proposed flat ILP based global routing technique.

1.1.3 Multi-objective Global Routing

With the maturity of DSM technology, more and more complex circuits and functions with tens to hundreds of millions transistors can be integrated into a single chip. For these circuits, area and routability become the fundamental goals of global routing. The shrinking of geometries also makes the interconnect capacitance a dominant factor in determining circuit speed and switching power dissipation. At the global routing stage, which determines how signal nets will finally be connected, it is necessary to consider the wire delays and power so that the overall chip performance can be maximized accurately. Meanwhile, to satisfy the increasing needs of interconnect, over-the-cell (OTC) multi-layer (more than two layers) routing is becoming more popular. As a result, a large number of vias are introduced to connect the wire segments on different layers. However, it is desirable to reduce the number of vias because they not only increase the interconnect area, but also increase the manufacturing cost and cause several problems in reliability and performance. Clearly, in the DSM regime, not only is global routing required to optimize the fundamental objectives (e.g. area and routability), but it also needs

to address other important design factors (e.g. timing, via minimization, power dissipation, crosstalk). Therefore, another motivation of this research is to develop an effective multi-objective modeling method to solve the global routing problem that has several conflicting goals.

1.1.4 Power-Efficient ILP based Global Routing

Interconnect wires are slowly dominating the performance of deep submicron chips. In fact, the number of interconnects and buffers used to achieve timing closure are becoming one of the major challenges facing designers in the sub-90nm ICs. This is attributed to the continuous increase of the number of logic blocks facilitated by the continuous shrinking of device dimensions. In order to mitigate the impact of these interconnects, designers have shifted their focus to interconnect centric designs pioneered by Cong et. al. [Cong96], where the wire is the center of the chip design flow.

Traditionally, delay and routing were the focus of most optimization efforts of the interconnect problem. However, the power consumption of the interconnects is becoming a crucial factor in determining the overall chip performance [Bane02]. In order to tackle the interconnect bottleneck, researchers have developed several sub-problems that deal with the various aspects of the interconnects. These problems start from the simple problem of buffer insertion determining the number and the positions of buffers to minimize delay and increase in complexity up to the multi-objective maze routing, where a combined routing and power optimization under relaxed delay constraints is introduced. The drawbacks of these power-aware routing methods are their inability or inefficiency to find the routing tree for multi-pin

nets. These nets approximately represent 30-40% of the interconnects on a modern chip [Cong01b]. Due to such drawbacks, another motivation of this research is to develop an approach that is capable of timing optimization, buffer insertion and power reduction simultaneously with routability consideration.

1.1.5 Overview of Research Approaches

The overall research approaches used to tackle the circuit layout problem is separated into two stages (i.e. wirelength-driven or congestion-driven placement and multi-objective flat/hierarchical global routing) as illustrated in Figure 1.3. The re-

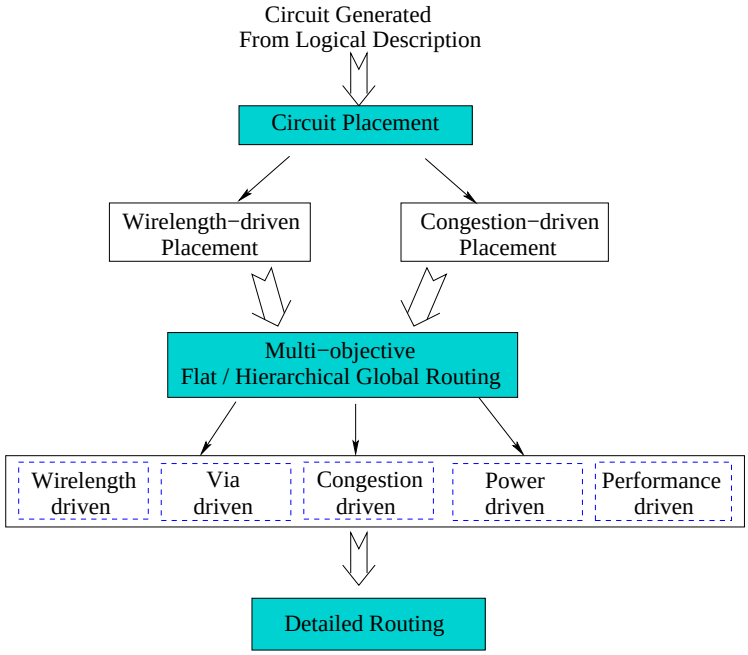


Figure 1.3: Overall Research Approaches

search first attempts to investigate several mathematical models and explore their effectiveness (introduced in Chapter 4). Following that, a hybrid global routing

framework that combines a pure IP model with a traditional sequential router is proposed. Quality of solution obtained by this routing framework is then compared with state-of-the-art routers. In addition, several hierarchical global routing methods are combined with the flat ILP based global router to reduce the computation time. Finally, power optimization problem is considered in the global routing stage and the performance of power-efficient ILP based global routing model is evaluated.

1.2 Contributions

The main contributions of the thesis can be summarized as:

- Investigating novel flexible ILP based global routing models (WVEM & WVZM) to optimize several design objectives simultaneously and efficiently.
- Proposing a novel pure IP based model to route 98% of nets without generating any overflow and combining it efficiently with a sequential router to provide better solutions with fast runtime.
- The hybrid model proposed is almost 8x faster than the multi-commodity flow approach [Albe01] and achieves better quality in terms of wirelength.
- Integrating a flat ILP based global routing approach with a top-down and bottom-up hierarchical methods which differs from traditional methods that seek to use Maze router within the flow [L02].
- Proposing a power-efficient ILP based global routing method to perform the timing optimization, buffer insertion and power reduction simultaneously with routability consideration.

1.3 Thesis Organization

Chapter 2 provides essential background on the physical design automation problem, the global routing sub-problem and the multi-objective optimization techniques. In addition, all benchmarks used to evaluate the performance of the proposed global routing results are introduced. The different layout styles that affect physical design are also described. Chapter 3 reviews state-of-the-art global routing approaches and different buffer insertion techniques. In Chapter 4 a flat ILP based global routing method with different formulations is introduced. A hybrid global routing framework that combines a pure IP model with a traditional sequential router is also presented. Chapter 5 introduces two hierarchical global routing methods and gives the experimental results. Chapter 6 presents a power-efficient multi-pin ILP based global routing method. Finally, Chapter 7 provides conclusions and a summary of the future work.

Chapter 2

Background

In the combinatorial sense, physical design automation is a constrained optimization problem [Sher99]. We are given a circuit (usually a module-wire connection-list called a *netlist*) which is a description of switching elements and their connecting wires. We seek an assignment of X and Y coordinates of the circuit components (in the plane or in one of a few planar layers) that satisfies the requirements of the fabrication technology (sufficient spacing between wires, restricted number of wiring layers, and so on) and that minimizes certain cost criteria. Practically, all aspects of the physical design problem as a whole are intractable; that is, they are NP-hard [Hach89]. Consequently, we have to resort to heuristic methods to solve this complex problem. One of these methods is to break up the problem into subproblems (partitioning, placement and routing), which are then solved one after the other. Another technique that is used to simplify the complexity of physical design automation is to narrow the search to localized regions of the search space through circuit clustering.

This chapter gives a detailed background on physical design automation in general and the global routing in particular.

2.1 Physical Design

Physical Design of VLSI circuits is a process of determining the location of devices and connecting them inside the boundary of a VLSI chip. It is one of many interrelated complex tasks in VLSI circuit design. Not surprisingly, this complex task is handled by dividing the original task into more tractable sub-tasks such that a physical design can be realized in reasonable amount of time. These sub-tasks may be performed in a slightly different order, iterated or omitted depending on the layout style used, the desired time, the desired chip size, and so on. The different stages of physical design cycle are shown in Figure 2.1.

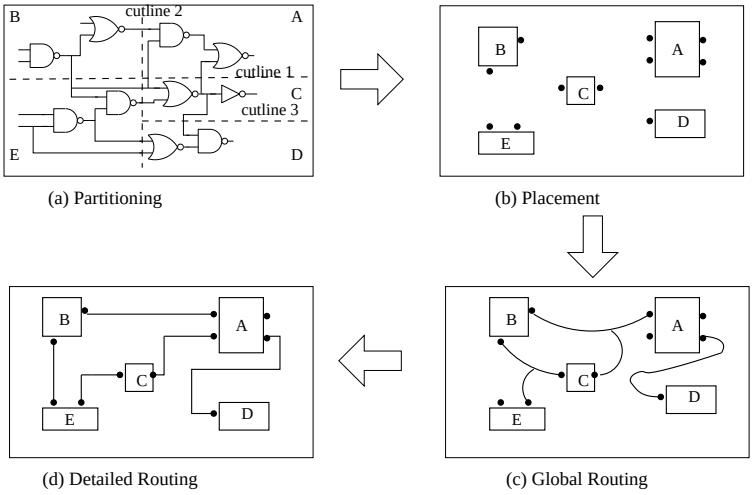


Figure 2.1: Physical Design Cycle

2.1.1 Circuit Partitioning

A chip may contain millions of transistors. Layout of the entire circuit cannot be handled due to the limitation of memory space as well as computation power available. Therefore, it is normally partitioned by grouping the components into blocks/subcircuits. The actual partitioning process considers many factors such as, the size of the blocks, number of blocks, and number of interconnections between the blocks. The output of partitioning is a set of blocks and the interconnections required between the blocks. Figure 2.1(a) shows that the input circuit is partitioned into five blocks (i.e. a,b,c,d and e). In large circuits, the partitioning process is hierarchical and at the topmost level a chip may have 5 to 25 blocks. Each block is then partitioned recursively into smaller blocks [Sher99].

Partitioning has been an active area of research for at least a quarter of a century and many algorithmic techniques for other sub-tasks of physical design, such as placement are originated in application to partitioning. For a recent survey on the partitioning problem, see [Kuca04].

2.1.2 Circuit Floorplanning and Placement

Following the partitioning stage, the next step in the physical design cycle is to determine the shapes and locations of the partitioned subcircuits (i.e. blocks). The process of determining block shapes and positions with area minimization objective and aspect ratio requirement is referred to as Floorplanning. Floorplanning is a critical step, since it sets up the ground work for a good layout. However, it is computationally hard due to the unfixed shape of blocks. A common strategy for

blocks floorplanning is to determine, in the first phase, the relative location of the blocks to each other based on connection-cost criteria. In the second step, block sizing is performed with the goal of minimizing the overall chip area and the location of each block is finalized.

As a restricted version of the floorplanning problem, circuit placement attempts to assign locations to fixed shape blocks on a layout surface such that the wirelength and area are minimized. Figure 2.1(b) shows that five blocks have been placed. Note that some space between the blocks is intentionally left empty to allow interconnections between blocks. It has been shown that placement is an NP-hard problem[Chan99]. When a large number of components are involved an optimal solution can not be obtained by using the exhaustive search method in reasonable amount of time. Therefore, heuristic algorithms are often used to obtain good solutions that satisfy certain computation time.

The quality of the placement will not be evident until the routing phase has been completed. A good routing and circuit performance will heavily depend on the outcome of the placement tool. For example, traditional placement algorithms mainly focus on minimizing total estimated wirelength to obtain better routability and smaller layout area [AD85, Sun93, Klei91]. However, a placement with less total wirelength but highly congested regions often leads to routing detours around the region, in turn resulting in a larger routed wirelength [Yang01b]. Congested areas can also downgrade the performance of global routers, and in the worst case, create an unroutable placement in the fix-die regime [Cald00]. Although the congestion problem is often addressed in routing stage, the optimization performance is constrained because very little can be done to improve the routing and the overall

circuit performance if the positions of cells are fixed. Hence, considering routability in the placement stage is becoming an important objective for VLSI circuit placement.

2.1.3 Global and Detailed Routing

Following the placement stage, interconnections between components are physically assigned to allowable routing regions. Due to the complexity of this problem, traditional routing is separated into two phases. The first phase is called Global Routing and generates a "rough" route for each net. In fact it assigns a list of routing regions to each net without specifying the actual geometric layout of wires, as shown in Figure 2.1(c). The second phase, refereed to as Detailed Routing, attempts to find the actual geometric layout of each net within assigned routing regions. Unlike Global Routing, which considers the entire layout, a detailed router considers just one region at a time [Sher99]. A detailed routing corresponding to the global routing is shown in Figure 2.1(d).

Physical design is iterative in nature and many steps are repeated several times to obtain a better layout. For example, an unroutable layout might need to be re-placed or re-partitioned several times such that routing can be completed successfully. Clearly, earlier steps have more influence on the overall solution quality. In this sense, partitioning and placement tend to play an important role in determining the area and chip performance, as compared to routing.

2.2 Layout Styles

Physical design is an extremely complex process and even after breaking the entire process into several conceptually easier steps, it has been shown that each step is computationally hard. However, market requirements demand a quick time-to-market and high yield [Raba03b]. As a result, restricted models and design styles are used in order to reduce the complexity of physical design. An overview of the different layout styles are shown in Figure 2.2. The design styles can be broadly classified as either Full-Custom or Semi-Custom. In a Full-Custom layout, the entire circuit is designed by hand. On the other hand, in Semi-Custom layout, some parts of a circuit are predesigned and placed on some specific location on the chip. The popular Semi-Custom layout styles include Standard-cells, Mixed size, and Gate Arrays.

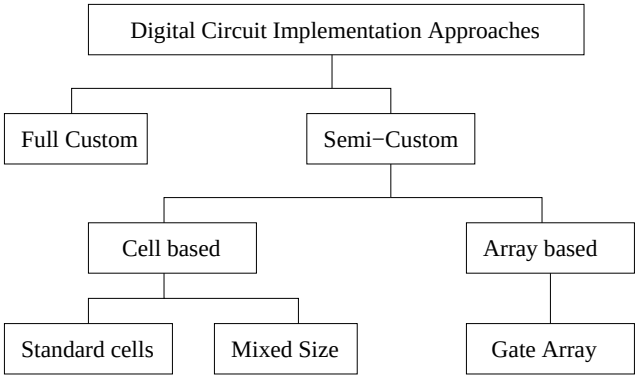


Figure 2.2: Different Layout Styles for Digital Integrated Circuits

2.2.1 Gate Array Layout

Gate array layout is a term given to a set of topologies, such as sea-of-gates, maskable gate array and a number of other gate array topologies. Gate array layout style are highly structured topologies, generally consisting of a grid array of prefabricated generalized logic blocks, as shown in Figure 2.3.

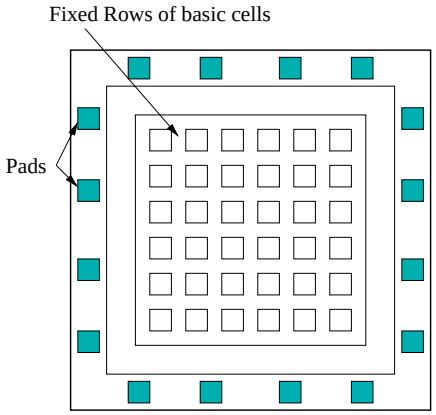


Figure 2.3: Gate Array Layout

All the blocks have identical size and are separated by vertical and horizontal spaces called vertical and horizontal channels. A special case of the gate array is the *Field Programmable Gate Array*, or FPGA. The feature that makes FPGA stand out among gate array topologies is that, instead of effecting a design with a photo-mask, all wires and interconnections are manufactured on the chip, and programmable fuse are fabricated into the interconnections. The desired design can be implemented by programming the interconnections between the wires and gates. The low cost of implementation and short time needed to physically realize a given design provide enormous advantages over traditional approaches which have to completed in foundries. However, FPGA is not very space-efficient, because

all the wires and interconnections are purposely generic to allow a variety of uses. Further more, the total cost increases largely for the high-density FPGA products compared to the traditional full-custom or standard-cell based design.

2.2.2 Full-Custom Layout

When performance or area is of primary importance, handcrafting the circuit topology and physical design seems to be the only option. The chip topology for this style of design is called full-custom layout. An example of full-custom design is shown in Figure 2.4. Full-custom design has the greatest flexibility and results in

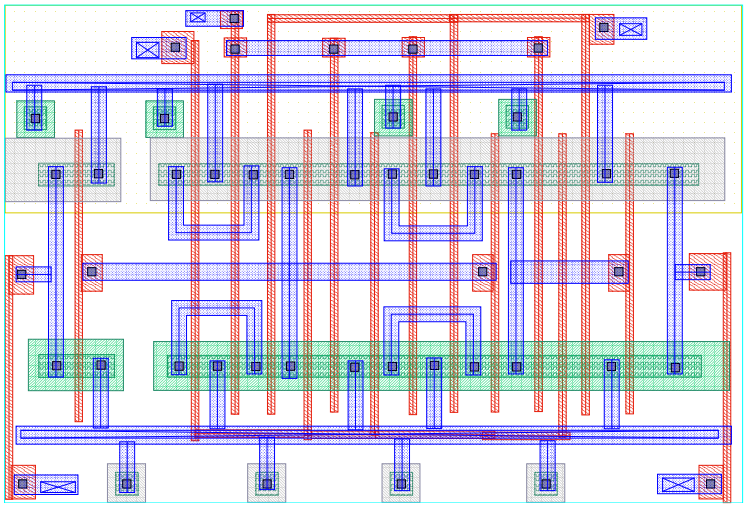


Figure 2.4: Full Custom Layout

the smallest chip area. However, it is also the most complicated, and therefore most time-consuming layout style. Because of the prohibitive design cost involved, this layout style is inappropriate for very large circuits unless performance is of utmost importance. Usually, Full-custom is used for microprocessor layouts [Sher95].

2.2.3 Standard Cell Layout

Standard-cell layout style (shown in Figure 2.5) is a topology between full-custom based and gate array based layout styles. Initially, a circuit is partitioned into

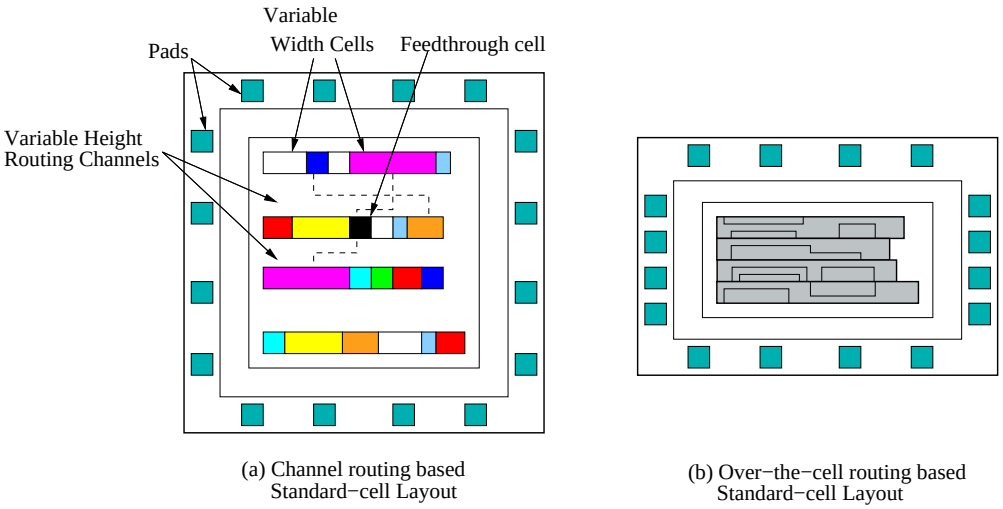


Figure 2.5: Standard Cell Layout

several smaller blocks each of which is equivalent to some predefined sub-circuit (cell). The functionality and the electrical characteristics of each pre-defined cell are tested, analyzed, and specified. A collection of these cells is called a cell library. Terminals on cells may be located either on the boundary or distributed throughout the cell area. All standard cells in the library are restricted to having the same height, but their width can be chosen by the standard-cell library designer to accommodate the area of the functional block design. Once a circuit is mapped the cells are laid out in rows within the chip boundaries. In the traditional single- or two-metal-layer technology, all connections were restricted to the area between standard cell rows, called routing channel. If two cells to be interconnected lie

in the same row or in adjacent rows, then the channel between the rows is used for interconnection. However, if two cells to be connected lie in two non-adjacent rows, then their interconnection wire passes through an empty space (also called Feedthrough) as shown in Figure 2.5(a). With the advent of the N-metal-layer process ($N > 3$), it is possible to route all nets in the unused metal layers over the cell. In Standard-cell designs, the problem of Over-The-Cell routing refers to routing a subset of nets over the cell regions to minimize the overall layout area. The objective is to eliminate channel areas by maximizing the routing in the Over-The-Cell areas [Sher95]. An example of Over-The-Cell routing based Standard-cell layout style is illustrated in Figure 2.5(b).

Fixed-die versus Variable-die

Standard-cell based placement and routing can be performed in two ways: variable-die or fixed-die. For traditional 2-metal-layer process, the variable-die placement and routing is appropriate. In this methodology, the inter-row spacings (that is the space between two adjacent rows), row lengths and sometimes even the number of rows may not be fixed before the placement, but rather are determined during placement and routing. Congestion is relieved by increasing the inter-row spacings. The fixed-die methodology is becoming popular due to the presence of N-metal-layer technologies ($N > 3$), mixed-size layout style, fixed power and clock distribution networks, etc. It is typically applied to design blocks rather than whole chips. For such a regime, the design block geometry, area, inter-row space and available cell sites in cell rows are fixed before placement and routing. Therefore, congestion analysis and hot-spot removal becomes paramount [Cald00].

The standard-cell design style provides a compromise between good design time and production size because it uses pre-designed standard cell library. It is also well-suited for automated design because the topology has a great deal of structure. However, the variable-width aspect causes complications in automation, and the final result must be fully fabricated. Current State-of-art processors, such as the *Pentium IV* make full use of standard cell within their design.

2.2.4 Mixed Size Layout

From above discussion, it is obviously that different styles have advantages, but they suffer from some drawbacks as well. To produce a circuit with high performance, good reusability and lower cost, a combined design style, called "Mixed Size Layout" is used. Normally in the mixed layout style, a design is laid-out as blocks of other layout styles, where each block is matched to the layout style which best represents it. For example, a logic array might be laid-out as a gate-array, while a memory array most likely be laid-out by hand for speed and space efficiency, as illustrated in Figure 2.6. In Mixed layout style, the irregular sizes of general blocks introduce complexity to the placement problem. But the number of modules involved is usually much less than standard cells. Mixed layout style can be a very powerful design style, and is the style used in industry for very large and very high production chip design, such as personal computer microprocessors.

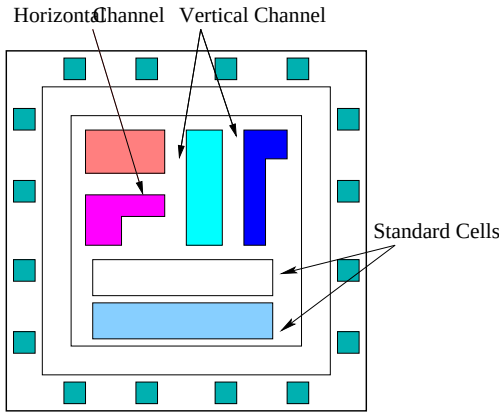


Figure 2.6: Mixed Size Layout

2.3 Global Routing

Following the placement stage, the exact location of cells are determined. The goal of global routing is to decide the connection pattern for each net and satisfy different objectives. The input to the global routing problem consists of a netlist that indicates the interconnections between terminals and placement information including the terminal positions and the location of wiring channels in between them. The typical objective of global routing involves wirelength and congestion minimization. In order to fit more functionality into a given chip area and reduce the capacitive delays associated with longer nets and speed up the operation of the chip, we need to optimize the chip area usage. In order to improve the routability and therefore, make the routing stage more manageable, we need to minimize the congestion areas in the global routing stage. For high performance design, the total wirelength may not be a major concern. Instead, the interconnect delay of some critical nets is minimized to maximize the chip's performance.

2.3.1 Problem Overview

The global routing problem is typically presented as a graph problem, where the routing regions and the module connections are modeled using a grid graph [Leng90]. Initially, a given chip is partitioned into a set of rectangular regions, called global bins. The cells are placed into these bins and each cell is assumed to be placed in the center of the global bin, as shown in Figure 2.7 (a). It is easy to see that the global bins and edges can be transformed by a grid graph (Figure 2.7 (b)). The vertices of the graph represent possible positions of net terminals, and the horizontal and vertical edges (called grid edge) that lie between two adjacent vertices represent channels along or regions through which wiring can be routed. A net is an unordered set of points on the grid graph. A route (or tree) of a net is a set of grid edges used to connect all the terminals of the net. As there is finite routing resources, each grid edge has a capacity. With such a graph representation, one can solve the graph version of the global routing problem instead of the original problem.

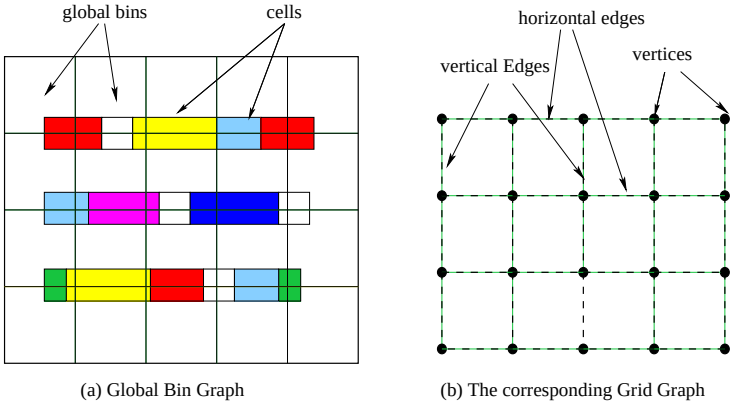


Figure 2.7: Grid Graph for Standard-cell Layout Style

2.3.2 Route Construction

In the practical circuit, the terminals in a net are connected by the horizontal and vertical wires. These routes can be defined by Rectilinear Spanning Trees (RSTs). The number of the RSTs grows exponentially with the number of terminals in a net. Therefore, only the RSTs that have some special feature, such as minimum length or limited number of vias are considered as the acceptable routes.

In [Hana66], it is proved that any Steiner Minimum Tree can be built on a grid obtained by drawing horizontal and vertical lines that pass through the vertices of the graph. This grid is usually referred to as Hanan grid. Figure 2.8 illustrates an example of 4-terminal net and its Hanan grid. The problem of finding Rectilin-

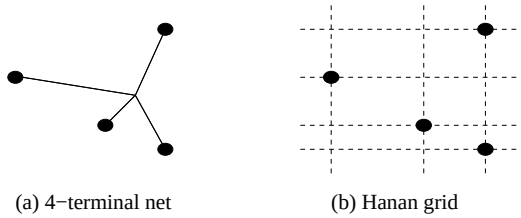


Figure 2.8: Four-terminal net and its Hanan grid

ear Steiner Minimum Trees (RSMTs) is NP-hard [Gare77]. Most of the heuristic algorithms take the minimum RSTs as an approximation for the RSMTs due to the special relationship between these two types of trees. In the case of a Steiner tree, a route can branch at any Steiner point so that the total length of the tree is minimum, while in the case of minimum spanning tree, branching is allowed only at the terminal point locations. It has been proved by Hwang [Hwan76] that the ratio of the cost of minimum Spanning tree (MST) to that of an optimal rectilinear Steiner tree is no greater than $\frac{3}{2}$. A minimum spanning tree for a given net can

be found in polynomial time [Tarj83]. Therefore, to produce a RSMT, a minimum spanning tree is first built for each net. Then, a near RSMT can be obtained by linearizing each edge of the minimum spanning tree. This linearizion is done by changing each non-rectilinear edge of the minimum spanning tree to either an L shape or a Z shape tree.

2.3.3 Global Routing Cost Functions

Every global routing method depends on the evaluation metric employed to measure the goodness of the technique. There are two traditional primary objectives in the automated global routing problem: minimizing chip area and achieving routable designs. For the standard-cell layout style, since the total chip area is approximately equal to the area of the modules plus the area occupied by the interconnect, minimizing the wire-length is approximately equivalent to minimizing the chip area [Shah91]. Congestion minimization is another important objective of global routing. Each connection may have multiple possible paths, and by selecting an appropriate set, the routing demand in any given area can be reduced such that all the nets can be routed without exceeding the track capacity of each routing channel. As the routing layers have been increased to more than two, the number of vias or vias plays a vital role in the routability and timing of the nets. Minimizing the number of vias tends to reduce the number of metal contacts in the routing, therefore, improving the performance of the circuit.

Total Wire Length

Almost all automatic routing tools use Manhattan geometry; that is, only horizontal and vertical lines are used to connect any two points. (i.e. two layers are used such that horizontal lines are allowed in one layer and vertical lines in the other). Usually, a Rectilinear Steiner tree or spanning tree based wirelength is used in the global routing. The shortest route for connecting a set of pins together is a *Steiner Tree* [Shah91] (Figure 2.9a). For a Steiner tree, a wire can branch at any Steiner point along its length so that the total route is minimum, while for a minimal Spanning tree (Figure 2.9b) connections allow branching only at the pin locations. Hence, the pins are connected in the form of the minimal spanning tree of a graph.

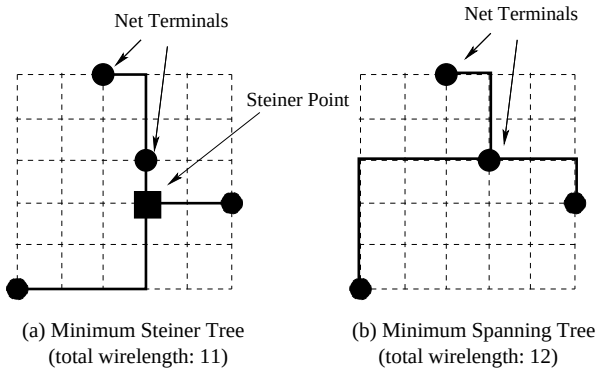


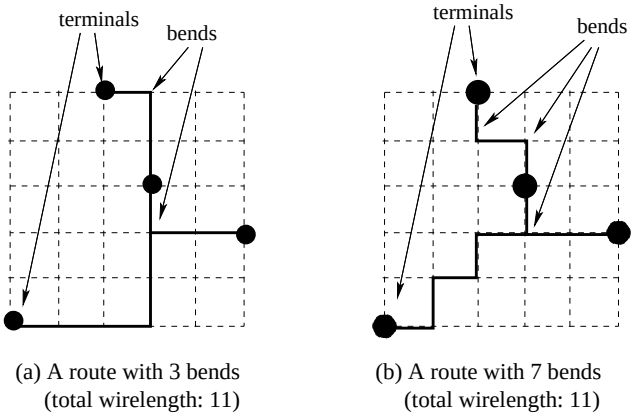
Figure 2.9: Total Wire Length

The Minimum Rectilinear Steiner Tree (MRST) problem is one of the fundamental problems in VLSI computer aided design. A lot of work has been devoted to designing good exact and heuristic algorithms. The MRST problem is NP-hard [Gare77], therefore any exact algorithm is expected to have an exponential worst case running time. However, the *GeoSteiner* algorithm presented by Warme et al.

in [Warm97, Warm01] has made a spectacular progress on the exact methods for MRST problem. On the other hand, many heuristic algorithms have been developed to produce near minimum Steiner trees fast. It has been proved by Hwang [Hwan76] that the ratio of the cost of minimum Spanning tree to that of an optimal rectilinear Steiner tree is no greater than $\frac{3}{2}$. Based on Hwang's result, Ho et al. generate a Steiner tree by improving a minimum spanning tree topology [Ho90]. In [Kahn92], a Batched Iterated 1-Steiner heuristic by Kahng and Robins produces the best quality solution among MRST heuristics. An improved implementation is presented in [Grif94] with computation time of $O(n^3)$. A more efficient approach was later proposed by Borah et al. [Bora94]. The performance of their algorithm is similar to the iterated 1-Steiner algorithm [Kahn92], while the worst case running time is $\Theta(n^2)$. The most recently effort in this field is a new heuristic by Zhou [Zhou04], which is based on the edge-substitution approach of Borah et al. [Bora94] and Zhou et al.'s spanning graph. The experimental results show that the performance of [Zhou04] is close to that of the 1-Steiner algorithm and the worst case running time is $O(nlogn)$.

Total Number of Vias

To minimize the number of vias in a layout, admissible routes in the global routing are limited to those that have lower number of vias. The total number of vias of a tree is equal to the sum of the number of vias between any two terminals, which equals to the route direction change from horizontal to vertical or vice versa. Figure 2.10(a) and (b) shows an example on how to calculate the total number of vias of a route.



(a) A route with 3 bends (total wirelength: 11)

(b) A route with 7 bends (total wirelength: 11)

Figure 2.10: Total Number of Bends

Congestion

Congestion in a layout indicates that there are too many nets routed in a local area, which leads to routing detours around the region, in turn resulting in a larger routed wirelength. In the worst case, the detailed router may not be able to find a feasible routing solution. The congestion problem has been widely addressed in several global routing algorithms [Hads03, Agni03, Yan04, Hu02, Jing04]. One of the most important issues for interconnect management is the ability to accurately and efficiently predict the routability of a given design. There are many interconnect estimation techniques, such as *Rent's rule based methods* [Yang01b], *Probabilistic based methods* [Lou01, Chen04], and *Simple bounding box based method* [Chen94], and so on.

To examine the effectiveness of the congestion minimization, different measures are used to evaluate the results. The most commonly used congestion metrics are congestion map, the maximum routing density of all the grid edges, and the total overflow [Lou01, Hads03, Chen04, Gegu00, Yan04, Hu02, Jing04]. A congestion map

visually plots the congestion in the design by assigning different colors to different congestion costs. A lighter color usually means the higher congestion cost. For a given routing graph $G = (V, E)$, the *routing demand* d_e of edge e, is the number of wires crossing e, the *routing supply* s_e is the number of wires that are allowed to cross edge e. The difference between routing demand and supply is formally described as: $overflow_e = \begin{cases} d_e - s_e & \text{if } d_e > s_e; \\ 0 & \text{if } d_e \leq s_e. \end{cases}$ The total overflow of a layout is defined as the summation of the overflow for all global edges. The amount of total overflow reflects the amount of total shortage of routing resources in the layout. The maximum routing density Max_{rd} can be described as: $Max_{rd} = max_{e \in E}\{d_e\}$. Figure 2.11 shows a grid graph of a simple circuit. The circuit has 5 modules and 4 nets. If the routing supply of all the edges is set to 1, then the total overflow is 3 and the maximum routing density is 3.

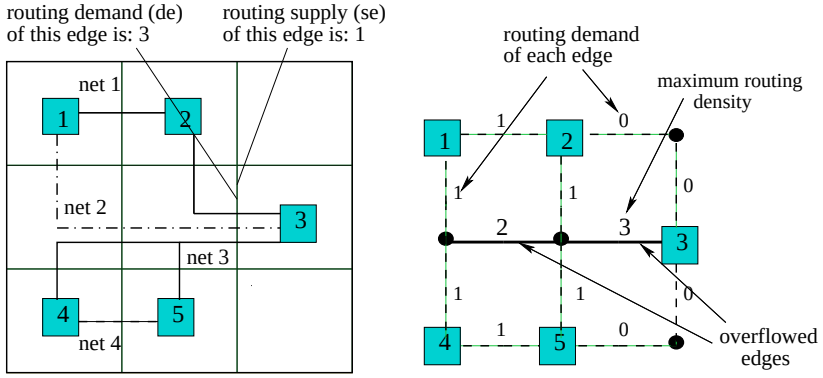


Figure 2.11: Example of Congestion Measure

Switching Power

As interconnect capacitance dominates gate capacitance in DSM technologies, minimizing power dissipation due to the interconnects becomes another important objective of routing. Switched capacitance power (a type of dynamic power) is the dominant factor of power dissipation in most CMOS digital circuits [Deng94]. It depends on the gate transition activity factor $\alpha_{0\rightarrow1}$, the supply voltage V_{dd}^2, the clock frequency f_{clk} and the load capacitance C_{load} (i.e. $P_{switch} = C_{load}\alpha_{0\rightarrow1}f_{clk}V_{dd}^2$). Traditionally, the load capacitance consists of two components: the gate capacitance C_{gate} and the wire (i.e. interconnect) capacitance C_{wire}. However, as the geometrics continuously scaling down, the coupling capacitance which occurs between adjacent wires on the same metal routing layer make up an ever growing amount of the total capacitance [Zube05]. Since the produce of switching frequency and square of the supply voltage of the gate is invariant after logic synthesis, we can use a constant C_i to represent it as $C_i = \alpha_{0\rightarrow1}f_{clk}V_{dd}^2$. Accordingly, the switched capacitance power of net k can be expressed as [Zube05]:

$$P_{net_k} = \sum_i C_i l_i (\frac{1}{d_{li}} + \frac{1}{d_{ri}}) \tag{2.1}$$

where l_i is the length of the ith wire segment of net k, and d_{li} and d_{ri} are the left and right distances to the next metal objects, respectively.

2.4 Multi-objective based Optimization

With the continuous and rapid increase in complexity of VLSI designs and fabrication technologies, global routing problem becomes more difficult to solve. Different and conflicting design objectives need to be considered in the global routing stage. Thus, modeling the global routing problem with several conflicting objectives, as well as maintaining the simplicity and efficiency of the models becomes an important issue for today's complex and large VLSI circuit design.

2.4.1 General multi-objective Optimization Problem

Multi-objective optimization problem can be defined as the problem of finding a vector of decision variable which satisfies constraints and optimizes a vector function whose elements represent the objective functions. The term "optimize" means finding such a solution which would give the values of all the objective functions acceptable to the designer [Osyc85]. A general multi-objective optimization problem can be expressed as follows [Ande99]:

$$\begin{aligned} \text{Minimize } & F(X) = (f_1(X), f_2(X)...., f_k(X))^T \\ s.t. \quad & X \in S \\ & X = (x_1, x_2, ..., x_n)^T \end{aligned} \tag{2.2}$$

where $f_1(X), f_2(X)$,...,$f_k(X)$ are the k objective functions, $(x_1, x_2$,...,$x_n)$ are the n optimization parameters, and S is the solution space. Figure 2.12 provides a visualization of the two-objective optimization problem. f_1^*, f_2^* represent the minima of the respective objective function. The ideal solution for these two-objective prob-

lem is defined as $F^* = (f_1^*, f_2^*)^T$. Usually, the vector X^* is called Pareto optimal (or efficient point) if there exists no feasible vector X that scores at least as well in all objective functions and strictly better in one. The Pareto optimum almost always gives a set of solutions, known as the Pareto optimal frontier P [Coel00] (as shown in Figure 2.12). The Pareto optimal frontier forms the boundary of the region defined by objective values for feasible solutions. When the behavior of the

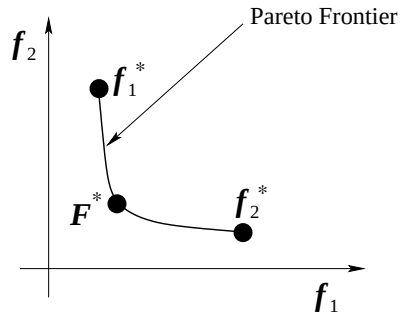


Figure 2.12: Optimization Problem with Two Objectives

different objective functions is more or less known and the preference information can be obtained prior to the optimization, the most common way of conducting multi-objective optimization is by aggregating the different objectives into one single figure of merit. The popular aggregating approaches include: Weighted Sum Approach, Lexicographic Optimization and Goal Programming. A weighted sum approach consists of adding all the objective functions using different weighting coefficients for each one. The typical formulation of this method is as follows:

$$\begin{aligned} \text{Min} \ & \sum_{i=1}^{K} \beta_i f_i(x) c_i \\ s.t \quad & X \in S \\ & \sum_{i}^{k} \beta_i = 1 \end{aligned} \tag{2.3}$$

where β_i are the weighting coefficients representing the relative importance of the objectives. As the objective functions are generally of different magnitudes, so c_i is used as a constant multipliers that will scale the objectives properly. This method can be applied to generate strongly Pareto optimal solution. But it is difficult to decide the appropriate weights because there is no clear relation between the weights and the obtained multi-objective solution [Ande99]. As an alternative scheme, lexicographic method performs multi-objective optimization by considering objectives one at a time. The designer ranks the objectives in order of importance. The most important objective is first optimized, then the second most important is optimized subject to a requirement that the first objective achieve its optimal value, and so on. Lexicographic approaches place very great emphasis on the highest-priority objective, with all later steps of other objectives optimization limited to alternative optima of the highest-priority objective. In some cases, when alternative optima of the highest-priority objective are rare, this method is actually optimizing the priority objective while ignoring all the others [Rard98]. In stead of minimizing or maximizing the objective functions, goal programming approaches solve the multi-objective optimization problem by minimizing the deviations from the targets or goals to the objectives. In the beginning, designers have to specify the goals that they wish to achieve for each objective. These values are then incorporated into the problem as additional constraints. Finally, the objective function attempts to find a solution that has the smallest deviation from the Utopian solution. It may be a very efficient approach if the desired goals are known. However, it still involves finding appropriate weights or priorities for the objectives [Coel00].

2.5 Test Circuits

Table 2.1 shows the general information of benchmarks used to measure the performance of the global routing algorithms in this research. The circuits used are the MCNC'91 benchmarks [MCNC91]. This test set consists of seven circuits ranging in size from 125 cells to over 25,000 cells The circuits have been grouped into three categories according to size: small, medium and large, indicated by horizontal lines in Table 2.1. Column "Short Nets" lists the number of 2-terminal and 3-terminal nets, while column "Long Nets" lists the number of remaining nets. The column "H/V Cap" lists the horizontal and vertical edge capacity and the last column shows the total number of columns and rows of the grid graph. To further illustrate the performance of the proposed ILP based global routing algorithms, the ISPD98/IBM benchmarks [ISPD98] and ISPD2007 benchmarks [ISPD07] are also used. The statistic information of ISPD98/IBM benchmarks and ISPD2007 benchmarks is listed in table 2.2. It is important to notice that 2-terminal and 3-terminal nets constitute the majority of the nets in all the test benchmarks.

2.5.1 Software Packages Used

The proposed techniques are implemented in 'C++' programming language on a 900MHz Sun Blade 2000 workstation with 1GB memory. Originally, GeoSteiner3.1 package [Warm01] is used to construct the steiner minimal trees. A much faster and accurate rectilinear steiner minimal tree algorithm [Chu04] is adopted later. The comparison of these two packages is shown in Appendix B. In addition, a commercial solver iLog Cplex10.0 is used to solve the Linear Programming and

Integer Linear Programming problems.

Bench marks	Tot Cells	Tot Nets	Short Nets	% Short Nets	Long Nets	% Long Nets	H/V Cap	Grid size
Fract	125	147	114	77.55%	33	22.45%	4/4	8x8
Prim1	752	876	698	79.68%	178	20.32%	6/6	19x19
Struct	1888	1920	1888	98.33%	32	1.67%	5/5	21x21
Ind1	2271	2478	2002	80.79%	476	19.21%	10/10	15x15
Prim2	3014	3136	2307	73.57%	829	26.43%	10/10	26x26
Ind2	12142	13419	11436	85.22%	1983	14.78%	11/11	72x72
avq.large	25114	25384	23050	90.81%	2334	9.19%	9/9	86x86

Table 2.1: MCNC Tested Circuit Statistics

Bench Marks	Tot Net	Routed Nets	Short Nets	%Short Nets	Long Nets	%Long Nets	H/V Cap	Grid Size	Chip Size(um)
ibm01	11507	9128	6299	69%	2829	31%	14/12	64x64	655x655
ibm02	19291	14270	8708	61%	5562	39%	34/22	64x80	754x750
ibm03	21621	15289	11316	74%	3973	26%	30/20	64x80	-
ibm04	26163	19738	14337	72%	5401	28%	23/20	64x96	-
ibm05	27777	21135	12351	58%	8784	42%	63/42	64x128	-
ibm06	33354	25797	18129	70%	7668	30%	33/20	64x128	-
ibm07	44394	34378	24104	70%	10274	30%	36/21	64x192	1142x1139
ibm08	47944	35195	24548	70%	10611	30%	32/21	64x192	1224x1224
ibm09	50393	39592	27957	71%	11635	29%	28/14	64x256	1231x1209
ibm10	64227	49491	30663	62%	18828	38%	40/27	64x256	1579x1577
ispd01	219794	176715	122973	70%	53742	30%	70/70	324x324	500x500
ispd02	260159	207972	160466	77%	47506	23%	80/80	424x424	500x500
ispd03	466295	323887	249037	77%	74850	23%	92/92	474x479	498x502
ispd04	515304	357104	287215	80%	69889	20%	92/92	474x479	498x502

Table 2.2: ISPD98 IBM Benchmark & ISPD2007 Benchmark Statistics

2.6 Summary

The VLSI design process is broken down into several smaller and tractable sub-tasks to manage the high complexity of the task. One of these sub-tasks is physical design,

which is still incredibly complex. As a result, this complexity is handled by dividing the physical design task into more tractable sub-tasks and circuit placement is one of such sub-tasks. Physical design automation highly depends on the layout style used. Different layout styles, such as full-custom, gate-array, and standard-cell layouts can achieve different trade offs among speed, cost, fabrication time, and degree of design automation.

The layout topology used to demonstrate the research work in this proposal is the standard-cell layout style. The most commonly used metric that measures the quality of a global routing solution is the total wirelength for all the nets, the total number of vias (i.e. bends), the maximum routing density of all the grid edges, and the total overflow. In addition, the multi-objective optimization problem and several aggregating approaches are also introduced in this chapter.

In the following chapter, different global routing techniques are reviewed and their advantages and drawbacks are also analyzed.

Chapter 3

Literature Review

Since all versions of the global routing problem are NP-hard [Sher99], a variety of heuristic algorithms have been developed for it, as shown in Figure 3.1. Basically, these algorithms can be classified into three classes: *Sequential global routing*, *Concurrent global routing* and *Meta heuristic based methods*.

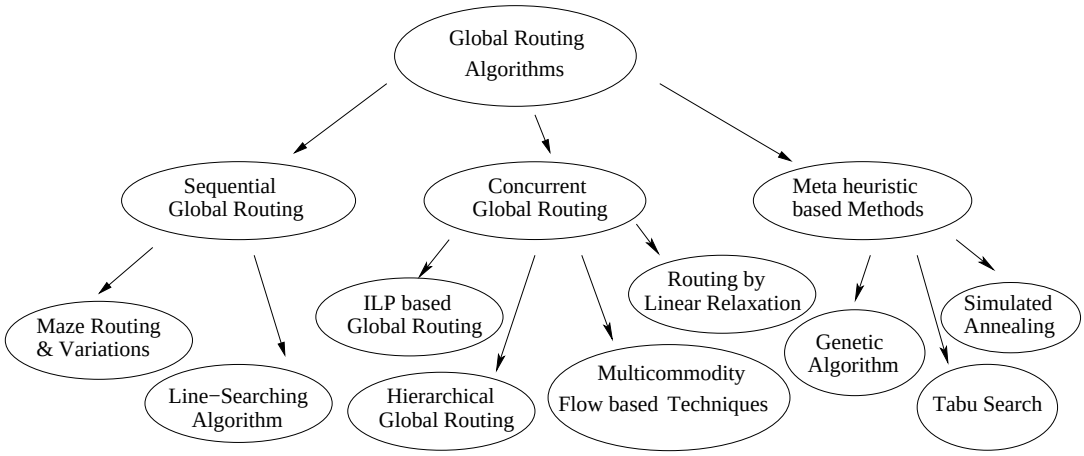


Figure 3.1: Different Approaches for Global Routing Problem

3.1 Sequential Global Routing

The most common approach to global routing is sequential routing. In such an approach, nets are first ordered according to their importance, then based on the ordering nets are routed sequentially. Sequential Routing can be grouped into two types: *Maze routing algorithms (with variations)* and *Line-Probe Algorithms*.

In 1961, a maze routing algorithm for finding a shortest path between two points on a grid graph was first introduced by Lee [Lee61]. The searching in Lee's algorithm is conducted by using the breath-first search technique (BFS). In order to find the shortest path between a source point "S" and a target "T", the maze global router first constructs several initial paths at "s" and then expands them until one path reaches "T" (as seen in Figure 3.2). The reason of the popularity of Lee's maze

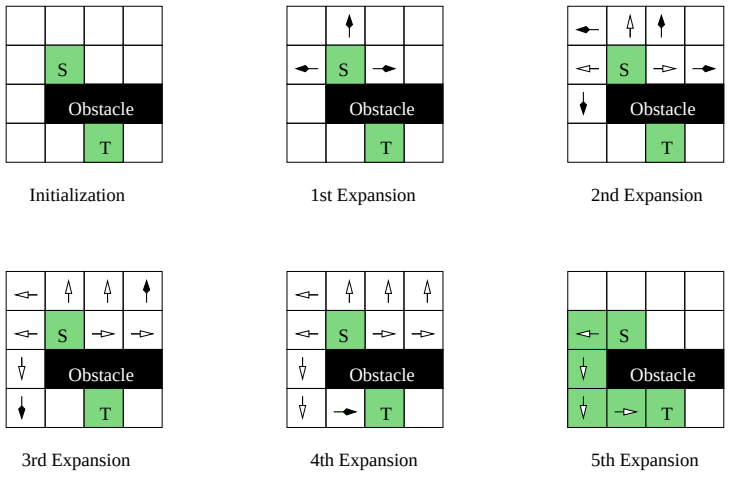


Figure 3.2: Maze Routing Algorithm

router is its simplicity and its guarantee of finding an optimal solution if one exists. However, the spacing and computation time requirements of Lee's maze router is also large. For an $N \times N$ grid plane, the time and space complexity is $O(N^2)$

[Sait95]. To reduce the problem of storage, a coding scheme for labeling of grid cells have been proposed in [Aker67]. Several authors discuss goal-directed search techniques in [Souk78] and [Hadl75] to speed up maze routing algorithms. The major drawback of Lee's maze routing algorithm and many of their variations is that the information data must be kept for each vertex. Thus, a very large memory space is needed for solving a large grid graph. To further reduce the space requirement, line-probe algorithms have been introduced by Hightower [High69] and Mikami [Mika68]. In line-probe algorithms, the routing is performed by a bidirectional searching. Straight search lines are extended from both source point "S" and target point "T" in all four directions, as shown in Figure 3.3. If a search line extended from source meets a line extended from target, then a path is constructed. When these trial lines hit some obstacles, new directions are extended for some base points on the trial lines. The time and space complexities of these algorithms is $O(L)$,

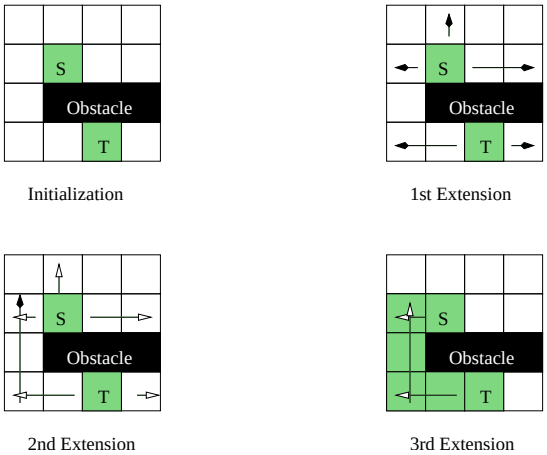


Figure 3.3: Line Probe Routing Algorithm

where L is the number of segments produced by these algorithms. Clearly, Line-

Searching algorithms are faster than maze routing algorithms and require much less memory. However, they do not guarantee finding the shortest path between two points.

The maze routing and line-probe algorithms are not designed to route multi-terminal nets. In [Sche86], the multi-terminal nets are decomposed into several two-terminal nets and the resulting two-terminal nets are routed by a maze routing or line-probe algorithms. For any sequential approach, once a net has been routed it may block other nets which are yet to be routed. As a result, these approaches highly depend on the order in which the nets are processed. However, it is hard to decide which net ordering is better than others [Abel72]. In practice, a technique called "rip-up and reroute" is often used after the sequential routing to remove blockages when further routing of nets is not possible [Aosh83, Ting83, Nair87, Lee91]. In [Hads03], an amplified congestion estimate is used to influence a rip-up and reroute approach. During each rip-up and reroute iteration, the congestion information is obtained by a feed-back loop to guide the maze router to avoid the congested areas in the subsequent reroute process. The congestion estimation consists of two parts: the static congestion given by a probabilistic algorithm and the dynamic congestion obtained from the previous iterations of rip-up and reroute process. The congestion estimation is amplified in the early rip-up iterations to strongly influence the routes in the congested areas and scaled down in the later iterations to decrease the impact on the total wirelength. Compared to the global routing work of [Kast01], the proposed approach produces shorter wirelength and lower overflow with less computation time.

A minimum-length Steiner tree is not appropriate when it traverses through

congested regions since it deteriorates the routability and may lead to unroutable layout. On the other hand, a Steiner tree that only tries to avoid crowded regions, regardless of its length, is not suitable due to its excessive length. To overcome the drawbacks of dealing with length or density alone, a weighted Steiner tree concept is applied to sequential global routing by Chiang et al. [Chia94]. The weight of a Steiner tree p is calculated as: $W(p) = \sum_i^t l_i w_i$ where l_i denote the length of tree p in region R_i and w_i is the weight of region R_i. The weight of a region depends on its area, the number of nets going through it, and the number of terminals it contains. By finding the minimum-weight Steiner tree for each net, a global routing that minimizes congestion and net length simultaneously is achieved. Experimental results show that the proposed algorithm gives lower maximum channel density and shorter wirelength than [Chia90]. Kastner et al. [Kast00] developed a pattern routing method that allows nets to be predictably routed. The authors show that using pattern routing, only a constant number of edges are searched for overflow information and therefore, the global routing process speeds up. The predictability of the pattern routing also makes the quick and accurate congestion and wirelength estimation becomes possible in the higher design level. The results in [Kast00] show that up to 80% of the small nets can be pattern routed with little degradation in the quality of the routing solution. While Manhattan routing architectures have dominated circuit design, there is growing interest in non-Manhattan architecture. In [Agni03], a congestion-driven global routing method that takes both Manhattan and non-Manhattan architectures into consideration is studied. The lower total wirelength, number of vias and congestion are achieved by optimizing the tree topologies and routing layer assignments. The authors assume that each

routing layer has a default routing cost at the beginning of the routing. A combined heuristic is then performed to build trees for all signal nets and the resource usage is examined. Based on the updated routing layer cost, a layer balancing approach is invoked to smoothly adjust interconnect topologies such that they can match the available routing resources better. To make the accurate interconnect information available during the placement process, Pan et al. [Pan06] presented a fast and high-quality global router. Different from traditional approaches, in [Pan06] a congestion-driven Steiner tree topology generation technique and an edge shifting technique are proposed to determine the good Steiner tree topologies and Steiner point positions. Based on these congestion-driven Steiner tree topologies, pattern routing and maze routing are applied to obtain the high quality global routing solutions. The experimental results show that the promising runtime makes it possible to incorporate global routing directly into the placement process without much runtime penalty.

The quality of a sequential global router largely depends on the ordering of nets. Usually, nets are ordered according to their criticality, half-perimeter wirelength and number of terminals [Sher99]. Once a net is routed, it occupies the routing resources and therefore, it can block the nets that will be routed later. Methods such as rip-up and reroute algorithms have been developed to further improve the quality of the final solution. Due to the sequential nature of these techniques, they fail to give adequate results. Besides, the sequential heuristic techniques can not provide a certain answer as to whether or not a feasible solution exists. In other words, if they fail to find a feasible solution, it is no clear whether this is attributable of the non-existence of a feasible solution or because of shortcomings

of the heuristic. Moreover, when a heuristic does find a feasible solution, it is not known whether or not this solution is optimal, or how far it is from optimality.

3.2 Concurrent Global Routing

3.2.1 Integer Programming based Global Routing

To avoid the net ordering problem and make the solution more predicable, Integer Programming based global routing algorithms have been developed to route all the nets simultaneously. In the Integer Programming based approach, the general global routing problem is formulated as a 0/1 Integer Programming problem. Given a set of Steiner trees for each net and a routing graph, the objective of the Integer Programming technique is to select a Steiner tree for each net from its set of Steiner trees without violating the channel capacities while minimizing the total wirelength. This approach tends to result in a more global solution and no initial ordering of nets is required. A general formulation of the ILP based global routing is as follows:

$$\text{Minimize} \sum_{j=1}^{t} b_j x_j \tag{3.1}$$

Subject to

$$\sum_{x_j \in N_k} x_j = 1, \quad k \in \{1, ..., n\}$$

$$\sum_{j=1}^{t} a_{ij} x_j \leq c_j, \quad i \in \{1, ..., p\}$$

$$x_j \in \{0, 1\} \quad j \in \{1, ..., t\}$$

where constant b_j is the cost of connecting a net using jth tree. There are two types of constraints in this formulation. The selection of one and only one routing for each net is forced by the first set of constraints and the edge capacity requirements of each edge is represented by the second set of constraints.

The time to solve the Integer Programming problem increases exponentially with the number of Steiner trees generated in the formulation of the program. Therefore, techniques to solve the ILP based global routing problem efficiently becomes an important issue.

3.2.1.1 An ILP based Global Routing Method

In [Behj02], a research work that focuses on enhancing the solution of the global routing problem using an Integer Linear Programming (ILP) based approach is presented. The flowchart in Figure 3.4 shows the main steps of the approach proposed in [Behj02]. The first important step in the ILP based global routing is producing a set of admissible routes for each net. In the practical circuit, the terminals in a net are connected by the horizontal and vertical wires. Therefore, only the Rectilinear Spanning Trees or Rectilinear Steiner Trees are considered in the tree construction process. These trees become the unknown variables of the ILP problem. Obviously, the number of trees for each net should not be too large, because the size of the ILP problem is proportional to the number of these trees. On the other hand, a number of trees should build for each net so that the feasibility of the problem is guaranteed. To remedy this problem, an additional tree generation step is proposed in [Behj02] to reduce the number of trees created for each net, while ensuring that the constructed trees will likely result in a promising (feasible and optimal) solu-

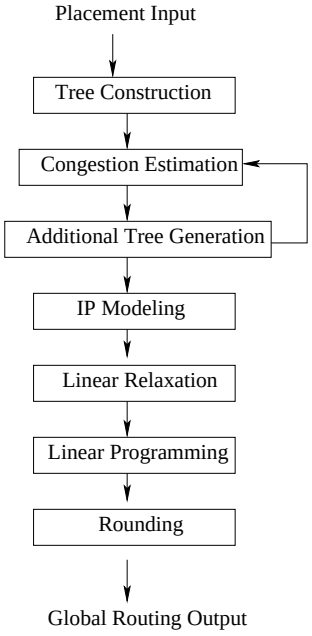


Figure 3.4: Flow Chart of the Global Routing Methodology

tion. Initially, the potentially congested areas in the routing graph are predicted by a congestion estimation technique [Behj02] in the congestion estimation stage. This priori congestion information is then used in the additional tree generation stage to eliminate the congested areas by adding trees to the nets passing through these areas iteratively. The congestion of the circuit is re-estimated after each additional tree generation step. Following the additional tree generation step, the global routing problem is formulated as an ILP problem. The different ILP formulations of the global routing problem is introduced in section 4.3. Because of the large size of today's VLSI problems, solving the ILP problem using traditional method, e.g., branch and bound, can be impractical. Thus, the ILP is typically relaxed and solved as a Linear Programming (LP) problem followed by rounding heuristics to obtain an integer solution. The relaxation is performed by replacing the integer

constraints imposed on the variables with linear boundary constraints.

Rectilinear Spanning Tree Construction

The number of trees produced by Hanan grid method for nets with two or three terminals is very limited, but these nets constitute a high percentage of the nets. Therefore, the set of produced trees could be an infeasible set because of the maximum channel capacity. On the other hand, Hanan method generates too many trees for nets with four or more terminals. Hence, the number of variables in the ILP model can become very large. To deal with these issues, in [Behj02] the nets are classified into two types: short nets[1] and long nets [2]. For short nets, a limited number of minimum or near minimum length spanning trees are built based on the Hanan grid. Following that, the congestion is estimated and the new spanning trees are added to the 2-terminal nets that pass through the congested areas to eliminate the congestion and therefore increase the feasibility of the produced tree set. For 2-terminal nets, a maximum of two minimum length trees are generated. To construct the minimum or near minimum length trees for 3-terminal nets, these nets are first transformed to two 2-terminal nets. Each one of these 2-terminal nets is then routed separately. Finally, the generated routes are combined to obtain the spanning trees for the original 3-terminal net.

[1]Nets with 2 or 3 terminals

[2]Nets with 3 more terminals

Additional Tree Construction

When formulating the global routing problem as an ILP, it is very important to make sure that the global routing formulation will result in a feasible solution, i.e., all the constraints must be satisfied. One main set of constraints in the global routing is the channel capacity constraints. If the number of wires passing through a channel exceeds the maximum channel capacity, the routing becomes infeasible. To increase the feasibility of the final solution, the exceeding channel usage should be avoided. In circuit, an area with exceeding channel usage can also be defined as a congested area. In [Behj02], a congestion estimation technique is proposed to predict potentially congested areas. This priori information about the congestion is then used to eliminate the congested areas (i.e., reduce the exceeding channel usage) in the circuit by adding trees to the nets passing through these areas in four iterations. Initially, trees that are shorter and have the least number of vias and do not pass through **any** congested edges are added. Then, congestion estimation is recalculated. If there are still congested edges, additional trees, which have either longer length or larger number of vias are constructed. This procedure is continued until either no congested edges existed or an iteration limit is reached. Figure 3.5 shows an example of additional trees construction in four iterations. Following that, the global routing problem is then formulated as an Integer Linear Programming (ILP) problem and relaxed to the Linear Programming (LP) problem. The fractional solutions of the LP problem is then transformed to integer solution using a rounding technique.

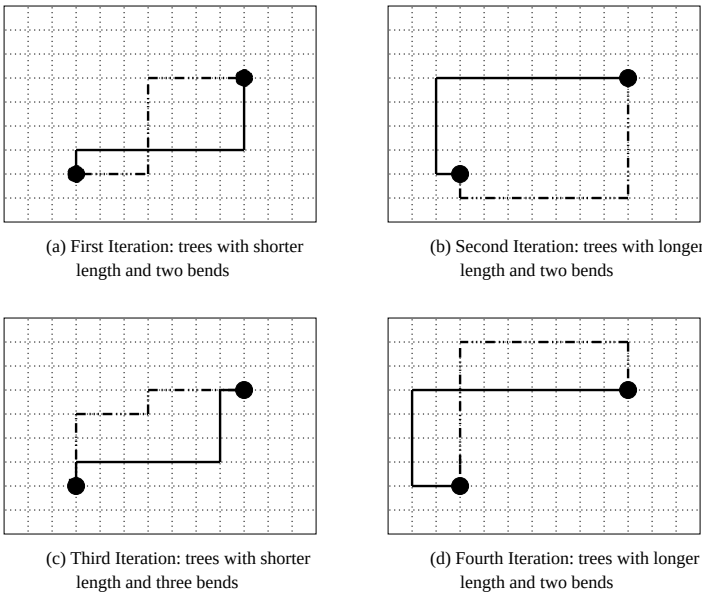


Figure 3.5: Additional Tree Construction

3.2.2 Hierarchical Global Routing

One alternative approach to solve the ILP based global routing problem is to build the multilevel hierarchical representations of the routing region to reduce the complexity of the overall global routing problem.

3.2.2.1 Top-down Hierarchical Global Routing

The Top-down hierarchical routing algorithms attempt to use a hierarchy on the routing graph to decompose the large routing problem into small, manageable pieces. The hierarchy can be obtained based on a uniform quadrisection partitioning or based on a cut tree resulted from the preceding floorplanning phase. Figure 3.6 shows an example of top-down hierarchical decomposition. The depth of hierarchy is called a *level*. Initially, on level 1, the layout area is divided into 2 x 2

blocks, and subsequently, on each level l, the whole chip area will be divided into 2^l x 2^l blocks. At each level of the hierarchy, the nets in each 2 x 2 blocks are routed simultaneously by solving the corresponding linear programming problem. This

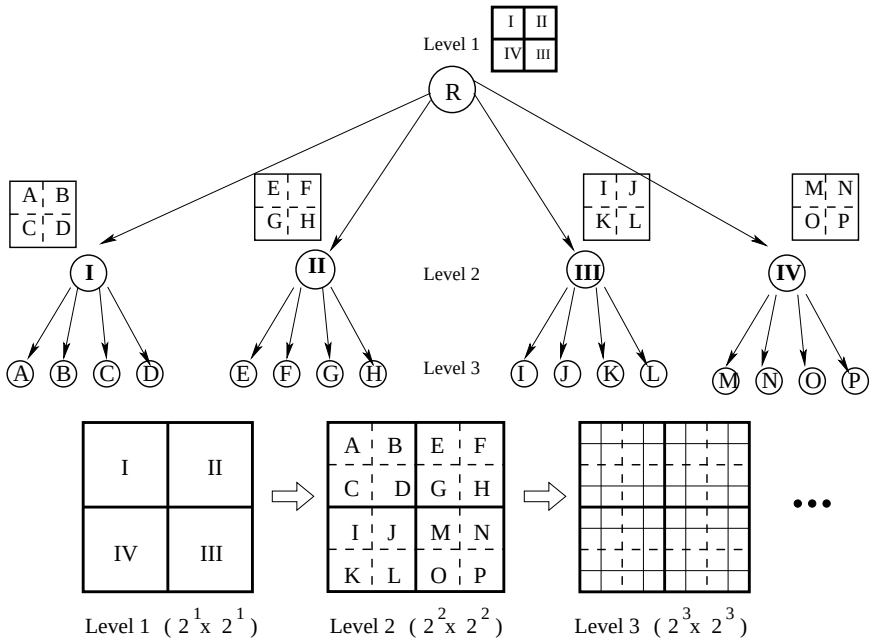


Figure 3.6: Hierarchical Decomposition

method was first introduced by Burstein and Pelavin [Burs83b] for the gate-array layout style. However, it can not be applied to a custom chip design environment because of the assumption of the uniform wiring substrate. In [Luk87], Luk et al. proposed a top-down hierarchical global routing method for custom circuit design based on a slicing floorplan. Usually, the subproblems of hierarchical method corresponding to the routing problems at each interior node of the cut tree can be solved exactly by integer programming techniques. But, this may take a large amount of computation time due to the large number of subproblems (i.e. integer programs). In order to speed up the computation, Heistermann et al. [Heis91]

propose a greedy preprocessing method that can solve these small Integer Programming problems efficiently. Burstein and Hong [Burs83a] combined hierarchical floorplanning and global routing together in the context of gate array layout and was later extended for general floorplans by Dai et al. [Dai87] and Lengauer et al. [Leng93]. In [Gegu00], a method based on hierarchical routing [Burs83b] and multi-layer routing model for high performance standard cell layout is presented. The routing area is divided into sub-regions recursively at each hierarchical level and the routes of nets are determined by solving a Linear Programming problem considering wire-sizing and buffer-insertion under timing constraints.

Through the hierarchical top-down methods, the scaling problem on large designs is overcame by building multilevel hierarchical representations of the routing regions from the coarsest level to the finest level. However, it is very difficult to make appropriate decisions at the coarser level due to the lack of detailed routing information. Therefore, if an unwise decision is made at the coarser level, a large computation effort would be needed to revise it at the finer level.

3.2.2.2 Bottom-up Merging Global Routing

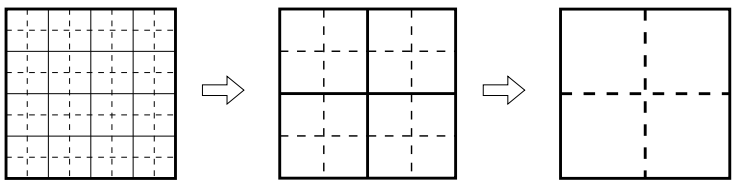


Figure 3.7: Bottom Up Hierarchical Routing

To overcome the limitation of uniform wiring substrate of Burstein and Pelavin's

method [Burs83b], a bottom-up hierarchical global routing method is described by Marek-Sadowska in [MS84]. Initially, the whole routing region is partitioned into a set of 2 x 2 supper-cell bins. In Figure 3.7(a), each bin is separated by the solid lines from other bins. The super-cells within a bin are depicted by the dashed lines. At the lowest hierarchical level, the routes are restrained within each bin individually (i.e. no wires are allowed to cross the solid border). The routing is performed in each 2 x 2 bin by a maze router. Next, every other solid border is open to form a new set of 2 x 2 bins as shown in Figure 3.7(b). This process is repeated until the top is reached, where there is only one 2 x 2 bin. In contrast to the uniform quadrisections in [MS84], the work in [Hu85] performs recursive bisections according to the net distributions.

3.2.2.3 Multilevel Global Routing

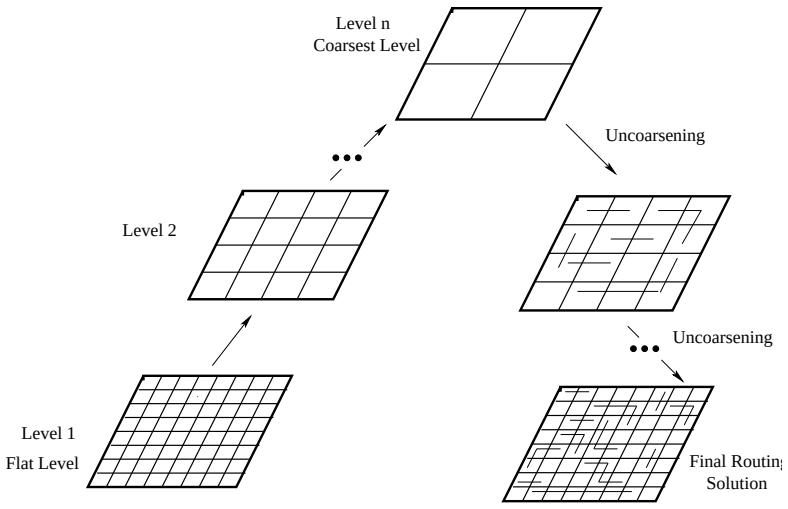


Figure 3.8: Multilevel Routing Flow

Inspired by the successful application of multilevel methods in circuit parti-

tioning and placement [Kary97, Cong00, Chan00], the different multilevel routing approaches were proposed by Cong et al.[Cong01a] and Lin et al. [L02]. In [Cong01a], the router starts by recursively coarsening the global bins, and an estimation of routing resource is computed at each level, as illustrated in Figure 3.8. At the coarsest level, a multicommodity flow algorithm is used to obtain an initial global routing solution. During the uncoarsening stage, a maze-searching algorithm is used to further improve the routing solution level by level. The final global routing solution is then fed into a gridless detailed router to find the exact connection for each net. The experimental results shown in [Cong01a] indicate that multilevel scheme provides better routing solutions in terms of the completion rate than classical hierarchical routing flow with much less computation time. The completion rate is further improved largely in [Cong02]. The major difference between [L02] and [Cong01a] is that: (i) during the coarsening stage, the work in [L02] performs the global routing, detailed routing and routing resource estimation at each level, while in [Cong01a] only routing resource estimation is performed. (ii) At the coarsest level, no initial routing solution is produced in [L02], while an initial solution is generated by a multicommodity flow algorithm in [Cong01a]. (iii) In addition to routability, timing objective is also incorporated in the routing process of [L02]. In [Ho04], the work of [L02] is extended to considering the antenna effect in the multilevel routing.

The hierarchical nature of the multilevel algorithm makes it very scalable to large designs. Moreover, a multilevel algorithm is more flexible than a top-down or bottom-up hierarchical approach. In the coarsening passes, more routing resources information can be obtained, which provides a good guide to the coarse level path

searching. During the uncoarsening process, the fine level router has the flexibility to refine the coarse level result based on more detailed information about local resource and congestion. All these features make the multilevel method achieve good solutions efficiently.

3.2.3 Routing by Linear Relaxation

One disadvantage of hierarchical approaches is that a large number of Integer Programming problems have to be solved during the course of global routing [Heis91]. To reduce the computation time of hierarchical global routing, the original Integer Programming problem is relaxed and solved as a Linear Programming problem The fractional solution is then transformed into an integer solution using several heuristic techniques.

In [Hu85], Hu and Shing directly solve the linear relaxation version of the Integer Programming based global routing problem. They use column generation techniques [Dant63] to generate potential Steiner trees and select the Steiner tree with the highest fractional value as the final route of each net. Karp et al. [Karp87] introduce a global routing algorithm based on linear programming relaxation and randomized rounding, which is a technique discussed in detail by Raghavan and Thompson [Ng87, Ragh91]. In [Vann91], an adaptation of Karmarkar's Interior Point Method [Karm84] is used to solve the linear relaxation formulation presented in [Hu85]. The experimental results indicate that the Interior Point Method runs much faster than an equivalent simplex implementation. In [Behj02], an Integer Programming based global router that combines wirelength, congestion, and the number of vias optimization is proposed. In the first stage, a set of routes is

produced for each net based on minimizing the wirelength and an approximation estimation of congestion. The global routing problem is then formulated as an Integer Linear Programming problem with the objective of minimizing the wirelength. Congestion and number of vias are also incorporated as penalty functions in the objective function.

3.2.4 Multicommodity Flow based Methods

The global routing problem can be formulated as a multicommodity flow problem. A multicommodity flow operates on a network that is a graph $G = (V, E)$, where $V = \{v_1, v_2, ..., v_n\}$ is a set of n vertices and $E = \{e_1, e_2, ...e_m\}$ is a set of m edges. Usually, an undirected graph formulation is used for multi-terminal net global routing. In global routing, each net N_i is treated as a commodity and the demand of commodity N_i always equals to one. Each edge has a flow capacity u_e. The routing can be expressed in terms of either edges or trees. In the edge-based expression, a variable $f_i(e)$ represents the amount of flow passing through edge $e \in E$. The tree-based expression assumes that there is a set of possible routing trees $T_i = \{T_{i,1}, T_{i,2}, ...\}$ for each net N_i. The binary variable $x_{i,j}$ is set to one if $T_{i,j} \in T_i$ is selected for net N_i, otherwise, $x_{i,j}$ is zero. Typically, there are two types of objective functions for the multicommodity flow problem. One formulation tries to minimize the total cost of transportation, and it is called the min-cost multicommodity flow problem. Another formulation attempts to minimize the maximum edge (or routing) density (as defined in Chapter 2 Section 2.3.3) and it is called the concurrent multicommodity flow problem. Generally, there are two set of constraints that must be satisfied in a multicommodity flow problem. The

first set is the *demand constraint*, which requires that the amount of flow shipped for each commodity should be equal to its demand, and the second set is the *edge capacity constraint*, which states that the total amount of flow $f(e)$ passing through each $e \in E$ should not exceed its capacity $u(e)$. A concurrent multicommodity flow based global routing formulation is as follows:

$$\text{Minimize } \lambda \tag{3.2}$$

Subject to

$$\sum_{T_{i,j} \in T_i} x_{i,j} = 1, \quad \forall N_i \in N$$
$$\sum_{e \in E_{i,j}} x_{ij} \leq \lambda u(e), \quad \forall e \in E$$
$$x_j = \{0, 1\} \quad \forall N_i \in N, \forall T_{i,j} \in T_i$$

where λ is variable to scale the capacity of edge e to $\lambda u(e)$. Due to computational complexity issues related to integer programming and the large size of the global routing problem, the zero-one integer multicommodity flow formulation is often relaxed to the fractional formulation, whose solution is transformed to an integer solution after a rounding procedure. Since linear programming solvers are slow for the sizes of problems encountered in global routing, research on the multicommodity flow problem has mostly focused on heuristics and combinatorial approximation algorithms.

In [Shra87], Shragowitz and Keel first presented a work on global routing using multicommodity flow model. This approach is based on a directed network graph and is restricted to two-pin nets. A min-cost multicommodity flow formulation is used in this approach to model the global routing problem. Since the original multi-

commodity flow problem is NP-complete [Sarr96], Shragowitz and Keel developed a polynomial time approximation algorithm to solve it. The authors investigated the feasibility of the convergence and the convergence rate, but no statements about the optimality are made with the analysis. Initially, by removing the edge capacity constraints, the problem can be decomposed into k subproblems, where k is the number of commodity (Nets). Each of these subproblems is similar to the shortest path problem and effective shortest path techniques can be used to solve the problem efficiently [Shra87]. Next, it iteratively reduces the violations on the edge capacity constraints while maintaining a minimum feasible cost.

Shahrokhi et al. [Shah90] proposes an approximation algorithm that can solve the 2-terminal multicommodity fractional flow problem in a relatively short period of time. When directly applied to global routing problem, this algorithm requires 2-terminal nets and fractional flow assignments. Carden et al. [Card91] extend Shkhrohki et al.'s algorithm to handle multi-terminal nets by decomposing a multiterminal net into several two-terminal nets. Furthermore, the fractional solutions are transformed to the integer solutions by a randomized rounding technique. In contrast with Shragowitz and Keel's approach [Shra87], their method is directed towards a concurrent multicommodity flow formulation (as shown in equation 3.2).

The work of [Algr01] is an application of Grag and Konemann's multicommodity fractional flow approximation algorithm [Garg98] (which is simpler and faster than the algorithm presented in [Shah90]) for global routing problem. Both approaches presented in [Shah90] and [Algr01] are based on the linear programming duality theory to solve the global routing problem. First, a multicommodity flow model is formulated (same as equation 3.2) and relaxed to the linear programming prob-

lem (called LP) by omitting the integer constraint. The dual linear programming problem (called DLP) is then formulated as follows:

$$\text{Maximize} \sum_{N_i \in N} \theta_i \tag{3.3}$$

Subject to

$$\sum_{e \in E} u(e)l(e) = 1,$$
$$\sum_{e \in T_{i,j}} l(e) \geq \theta_i, \quad \forall N_i \in N, \forall T_{i,j} \in T_i$$
$$l(e) \geq 0 \quad \forall e \in E$$

The variable $l(e)$ is the dual variable corresponding to each edge e and the variable θ_i represents the throughput of the flow from net N_i. According to the linear programming duality, a feasible solution to DLP provides a lower bound for the optimal solution of LP, and the LP solution reaches its optimum when it equals the DLP solution. Based on this property, different approximation algorithms are developed by Sharrokhi et al. and Albrecht in which the LP and DLP solutions are pushed closer after each iteration. Both of these algorithms begin by constructing a minimum weight Steiner tree for each net to obtain an initial solution without considering the *edge capacity constraint* in 3.2. In subsequent steps, the algorithms iteratively recomputes the Steiner trees to decrease the maximum edge density. The final gap between the LP and DLP solutions provides an upper bound on how far the LP solution away from the optimal solution. Besides congestion minimization, the total weighted wirelength optimization is also considered in [Algr01] by incorporating the target total weighted wirelength into the formulation as an additional constraint. Superficially, the algorithms proposed in [Shra87, Shah90, Algr01] looks

similar to the rip-up and reroute method. In [Algr01], the proposed algorithm is compared with the rip-up and reroute method. Through the results comparison, the author argue that (i) The proposed algorithm can solve the global routing problem for difficult large benchmarks where the rip-up and reroute method fails. (ii) The rip-up and reroute method increase the length of the Steiner tree for some nets substantially due to the limited view and limited number of choices when decreasing the congestion of an edge. Multicommodity flow methods have also been combined with timing-driven global routing to produce the high performance solution [Huan93, Wang96]. (introduced in section 3.4). In [Cho06], a global router (BoxRouter) that is based on the idea of box expansion and progressive ILP routing is proposed. Initially, a PreRouting is performed to decide the congestion areas. The most congested area is selected as the starting point of BoxRouting and expanded progressively. Within each expanded area, wires are routed by an efficient ILP based router and an adaptive maze router. After the expanded area covers the whole circuit, a PostRouting is then performed to remove the unnecessary detour and overflow created by area expansion. The experimental results indicates that the BoxRouter outperforms the state-of-the-art global routers in terms of wirelength, routability and runtime.

3.3 Meta Heuristic based Methods

In addition to the previous mentioned techniques, different meta heuristic methods have been applied to the global routing problem. In [Vecc83], the iterative technique: Simulated Annealing is applied to global routing, where only two-pin nets

are considered and the number of vias for each net is no more than two. Initially, the nets are assigned to the horizontal routing channels one at a time such that the overall channel density is minimized. A simulated annealing search technique is then used to refine the global routing solution produced by the previous sequential algorithm. The cost function used in simulated annealing search is equal to the sum of the total channel densities. Simulated Annealing is also applied to placement and global routing in the well-known TimberWolf package [Sech85]. In this package, a set of candidate routing trees is created for each net and one of the trees is randomly chosen as the initial solution. Each move implies a switch from one tree to another for a net. The cost to be minimized is the total wiring overflow over the entire routing graph. This algorithm yields high quality solution, however, it requires large computation time and highly depends on the determination of critical cooling schedule [Aart85, Lund86].

Other meta heuristic methods applied to global routing include simulated evolution [Chen89], genetic algorithm [Esbe94] and tabu search [Yous99].

3.4 Performance-driven Global Routing

As the VLSI fabrication technology reaches sub-micron device dimension and gigahertz frequency, interconnection delay has become the dominant factor in determining circuit speed [Bako90]. Traditional global routing methods that produce good results in terms of wirelength and congestion minimization can not achieve performance optimization in the design of high-speed ICs and MCMs. Hence, performance-driven global routing has received attention in recent years. A lot of

approaches that take interconnect delay into account through the routing stage have been reported in the literature [Wang96, Huan93, Hong97, Cong97, Zhu98, Hu02].

In [Wang96], an initial routing solution is obtained by a timing-driven Steiner tree algorithm [KDB94]. The interconnect wire distribution of the initial solution is then improved by using an iterative rip-up and rerouted approach. During each rip-up and rerouted process, all the nets that pass through the most congested edges are ripped-up. The rerouting problem of the ripped-up nets is formulated as a multicommodity flow problem and solved by a hierarchical grid routing algorithm. The uniformed wire distribution is achieved by maximizing the global flow uniformity and minimizing the maximum flow density. This algorithm is applied to the multi-layer building block layouts and can improve flow uniformity effectively. Similar to [Wang96], in [Huan93, Hong97], a set of trees that satisfy the critical-path-based timing constraint are generated for each net. An iterative algorithm is then performed to detour the path from the congested edges as well as maintain the timing requirement. The routing problem is then formulated as a multicommodity flow problem with the maximum edge density minimization objective. A primal dual approach [Card91] is used to find the fractional solution and the final integer routing result is obtained by an integerization process. Experimental results shows that the proposed approach can yield smaller delay and similar maximum channel density compare to TimberWolf 5.6 under both the standard cell and gate array layout styles. The paper presented in [Cong97] incorporates the timing issue with an iterative deletion technique [Cong92] for standard-cell based global routing. Initially, a set of routes is generated for each net and optimized based on the global delay objectives. Feedthrough cells are then assigned to each row with the

consideration of resources and requirements across all rows and nets. Finally, the route is selected for each net using the iterative deletion technique to minimize channel density and congestion across the circuit. In addition to the interconnect topology optimization, variable-width wire sizing optimization is also supported by the global router developed in [Cong97].

The authors in [Hu02] take advantage of the topology flexibilities to reduce the congestion while ensuring that timing constraints are satisfied. The algorithm first route each net to meet the timing constraints without considering the congestion. During this step, each routing tree is connected by a set of "soft edges" (A soft edge is an edge connecting two nodes without specifying the precise route between these two nodes.) so that the topology flexibilities can be exploited in the latter phases. Following that, the routing region is recursively bisected into subregions and the soft edges is assigned to the boundaries on the bisector line without exceeding any boundary supply and violating any delay constraint. At the end, a timing-constrained rip-up and reroute technique is used to reduce the congestion. The congestion reduction is evaluated by the total overflow and the maximum demand density $D_{max} = max_{e \in E}\{d(e)/s(e)\}$, where $d(e)$ is the routing demand of edge e and $s(e)$ is the routing supply of edge e.

In [Jing04], the timing and congestion objectives are incorporated into a unified objective through a shadow price mechanism. The global routing problem is initially formulated as a multicommodity flow primal problem and then converted into a dual formulation. The dual formulation is used to calculated the shadow congestion cost and delay cost. Based on these shadow cost, the primal operation reroutes nets to minimize the congestion and delay. The primal dual process is performed

iteratively until the two objective converge or the number of iterations exceeds the limit. A heuristic method is also proposed to find the integer flows effectively. The experimental results show the proposed algorithm optimize both timing and congestion efficiently. The congestion reduction is evaluated by the number of edge overflows.

3.4.1 Buffer Insertion based Interconnect Optimization

Interconnect wires are gradually dominating the performance of deep sub-micron chips. To mitigate the impact of the interconnects, designers have shifted their focus to interconnect centric designs, where the wire is the center of the chip design flow [Cong96]. Traditionally, delay and routing have been the focus of most interconnect optimization efforts. However, the power consumption of the interconnects is becoming a crucial factor in determining the overall chip performance [Bane02]. To address the multi-objectives of the interconnect problem, researchers have developed several subproblems that deal with the various aspects of the interconnects. These problems begin with the simple problem of buffer insertion, determining the number and the positions of buffers to minimize delay and increase in complexity up to power-aware interconnect optimization methods, where the effort is mainly focus on the dual optimization of the power and delay for the interconnects.

Many techniques are employed to reduce interconnect delay. Among these techniques, buffer insertion is widely recognized as an essential technique for interconnect optimization [Bako90]. The importance of buffer insertion has resulted in numerous algorithms and methodologies. All these algorithms can be classified into two classes: *Net based Buffer Insertion* (i.e. consider buffer insertion problem only

on a single net) and *Circuit based Buffer Insertion* (i.e. perform buffer insertion on all critical nets).

3.4.1.1 Net based Buffer Insertion

Broadly characterizing the net-based interconnect optimization efforts shows two major techniques: *analytical* and *dynamic programming* approaches. Analytical approaches [Chu01, Otte03, Li03] focus on finding a closed form expression that minimizes one of the major objectives (i.e. delay, power or routing topology) while keeping the other objectives well within bounds. In [Chu01], the transitional analytical approach is extended to include the wire sizing problem, where the wire lines are divided into a discrete set of segments, and each segment is sized independently. To consider buffer insertion and wire sizing problem while minimizing the power consumption of the buffer, more advanced techniques are proposed in [Otte03, Li03]. These techniques employ a delay relaxed formula ($D = \kappa t_{crit}$), where t_{crit} is the minimum achievable delay on a global net, and $\kappa > 1$. Hence, κ is used as a control knob trading off the delay of the interconnect to its power consumption. The work in [Otte03] shows that 15% relaxation in delay can save up to 33% of the power consumed by the interconnect buffers. These efforts were used to plot the delay-power tradeoff curve in order to determine the optimal number of buffers and their respective positions.

In general, analytical methods lack the ability to accommodate the buffer blockage, which is defined as the presence of macro blocks that prevent buffer insertion. The fact that the closed form solutions provide the optimal number of buffers and their separation as a fixed wire length limits their capability to deal with this

blockage. In fact, analytical solutions may even prevent an optimal solution, if the blockage area is too big. Moreover, analytical approaches assume that the buffer widths and locations can be continuously changed. However, in the real designs buffer widths are discrete due to layout design rules.

To address the limitation of analytical approaches, van Ginneken [vG90] proposed a buffer insertion algorithm based on dynamic programming (DP). Given a Steiner tree spanning a signal net and candidate buffer locations on the tree, van Ginneken 's algorithm can find a delay optimal solution within the frame work of graph theoretic representation of the problem. This algorithm has been modified for interconnect power reduction [ea96, Lill96, Shah96]. In [ea96, Shah96], buffer locations are fixed, and in [Lill96] an $O(n^5)$ algorithm is employed to find the optimal setup.

Motivated by the drawbacks of previous methods, a power optimal maze routing methodology was devised in [Yous05]. This work extends the efforts of delay optimal maze routing and buffer insertion in [Lai02], by using the shortest paths algorithm to find the power optimal path. The limitation of this work is its inability to find the routing tree for multi-pin nets. These nets approximately represent 30-40% of the interconnects on a modern chip [Cong01b]. This issue was addressed in [Yous07] on the expense of a major increase in the total runtime. In [Liu05], an net-based hybrid buffer insertion technique for low-power global interconnect design is proposed. This hybrid method combines an analytical buffer insertion solver with a dynamic programming based approach to produce a high quality solution efficiently. Apart from that, the algorithm can handle buffer blockages and therefore is applicable to the real design.

When applying the net based approach to a real combinational circuit, buffer insertion is performed on sequentially. This may lead to sub-optimal path delay or unnecessary buffer usage due to the lack of global view [Sze05]. Since global routing is performed on a fixed placement, the large number of buffers often increases the congestion and therefore, make the achievement of routability/timing solution become difficult.

3.4.1.2 Circuit based Buffer Insertion

To overcome the weakness of the net based approaches, circuit based approaches have been developed to perform buffer insertion at the circuit level.

In [Liu99, Liu00], Lagrangian relaxation based algorithms for circuit level buffer insertion were proposed. However, the work in [Liu99] assumes that buffers are placed equidistant from each other. In practice, the availability of space decides whether a buffer can be inserted in a particular location. Although these approaches usually produces good results, they do not scale very well. In fact, in [Liu99], the CPU time explodes for the larger testcases.

A path based buffer insertion algorithm was proposed in [Sze05]. This algorithm is built on the dynamic programming approach [vG90]. Starting with a buffer aware static timing analysis, a list of critical paths is determined in the order of their criticality and the over-consuming of buffer resources is avoided. Then, the dynamic programming approach is applied to one path at a time. When there are many critical or near critical paths, the quality of solution largely depends on the ordering of critical paths. Therefore, the optimal solution it produces may be much different from the global optimality.

In [Chen05a], A network flow based algorithm is suggested. To overcome the disadvantages of path based methods, it tries to identify the ordering of critical nets by using the min-cut idea in network flow problems. However, it assumes that the layout space is always available for buffer placement and the buffers are placed equidistantly in the interconnect. In [Xu05], Max-flow Min-cut theorem is applied to find the critical network. By reducing the delay of every edges within the critical network, an overall delay reduction of the circuit is achieved. In addition, the routability is considered through an edge weight that indicates the cost of routability deterioration per unit delay improvement. Initially, the timing/congestion information is computed and nets which have greatest impact on circuit timing are identified. The routing tree selected for buffer insertion of these critical nets improves the the delay at the least cost of routability deterioration. Following that, one buffer is inserted to the selected routing tree of each critical net based on the theoretical optimal buffer location presented in [Xu05]. Though this algorithm can handle timing optimization and buffer insertion simultaneously with routability considerations. It assume that only one buffer is placed for each critical net, which may not be able to satisfy the timing requirement of a real design. Furthermore, the experimental results does not show that by considering the routability during the buffer insertion process, the proposed algorithm produces a better solution in terms of overflowed edges than other method.

By combining a fast greedy net-level dynamic programming based buffer insertion algorithm with a look-ahead and back-off strategy, Shi et al [Wagh06] presents an algorithm that optimizes the buffer usage for the whole circuit while trying to meet the timing constraints. The algorithm starts from inserting buffers for critical

nets using the look-ahead technique. In between the insertion step, the back-off technique is used to reconsider some of the past decisions that prove less effective later on. After that, the critical nets is determined for next buffer insertion step. The proposed look-ahead combined with back-off technique provides an effective solution search strategy. However, as mentioned in this paper, there are parameters need to be tuned to balance its runtime and quality of solution and the improved runtime largely depends on the proposed speed-up techniques.

3.5 Summary

In this chapter, several approaches for the global routing problem are reviewed. Three classes of techniques, namely Sequential Approaches, Meta Heuristic Based Methods and Concurrent Approaches are identified and discussed.

Based on the review, the advantages and disadvantages of these approaches are summarized as follows:

- For most global routing techniques (e.g. sequential and meta heuristics), it is impossible to claim how close the produced solutions are to the optimal solution.

- Sequential approaches are able to solve global routing problems efficiently. However, the quality of the solution depends heavily on the order in which nets are processed. Under any net ordering, it is difficult to route the nets that are considered later since they are subject to more blockages.

- To overcome the net ordering problem and make the routing solution more

predictable, mathematical programming based global routing algorithms (e.g. ILP based method and multicommodity flow based method) have been developed to route all the nets simultaneously.

- ILP based method has flexible objective function which can optimize several objectives simultaneously. When solving the ILP based global routing problem, the general LP solvers are usually used. However, these solvers are slow for the sizes of problems encountered in global routing.
- Unlike the ILP based method, the objective function of multicommodity flow based method is restrained to the maximum edge density minimization or total cost minimization. Nonetheless, it can achieve the multi-objective global routing by considering the other objectives as constraints. The latest proposed approximation algorithm for multicommodity flow based global routing has shown this method produce good results even for very large circuits.

The increasing size of problem makes all the flat global routing approaches unfit in today's VLSI circuit design. Thus, different hierarchical methods are proposed to improve the scalability of flat approaches for handling larger designs. Among these approaches, multilevel hierarchical technique provides better quality solution with less computation time. As the interconnect delay becomes the dominant factor in determining circuit speed, traditional total-wirelength-driven and routability-driven global router can not achieve good performance in the design of high-speed ICs. Hence, many global routing algorithms start to take interconnect delay into account. Combined with the multicommodity flow method, current performance-

driven global routing algorithm can optimize both timing and congestion efficiently.

Among the techniques used to reducing interconnect delay, buffer insertion is widely recognized as an essential technique for interconnect optimization [Bako90]. Different buffer insertion techniques can be classified into two classes: *Net based Buffer Insertion* and *Circuit based Buffer Insertion*. For a real combinational circuit, the net based buffer insertion method may lead to sub-optimal path delay or unnecessary buffer usage due to the lack of global view. Therefore, net based buffer insertion techniques are developed to perform buffer insertion at the circuit level. As the number of buffers increase, the power consumed by the interconnect buffers is becoming an important performance metric. the need for a technique that considers delay, power and routing simultaneously is clear. Several algorithms that based either on the closed form expression or on the framework of a graph theoretic representation has been developed.

According to the literature review conducted in this chapter, in chapter 4, 5 and 6, the proposed global routing techniques and the corresponding primary experimental results are presented.

Chapter 4

Flat Multi-objective Global Routing

According to the literature review presented in the previous chapter, it is clear that although sequential approaches may be effective in practice, they can not give an upper bound on how far away the produced solution is from the optimal solution. Further more, the sequential nature of these techniques make them heavily dependent on the net ordering, which is hard to decide [Abel72]. To overcome these problems, the mathematical programming (concurrent) based methods are used to formulate the global routing problem as an ILP problem. However, the increasing size of global routing problem makes these concurrent based methods unsuitable for practical circuit design. As a result, the different hierarchical methods and combinatorial approximation algorithms have been proposed. Motivated by the success of mathematic programming based methods in global routing problem, in this chapter, an ILP based global routing method is proposed and investigated.

4.1 Rectilinear Spanning Tree Construction

The research work presented in [Behj02] only takes short nets into account because short nets consist of about 80% of the circuits. In this research, both short and long nets are considered for the global routing. To construct Rectilinear Steiner Minimum Trees (RSMTs) for long nets efficiently, a software package, called "GeoSteiner 3.1" [Warm01], which is based on the techniques presented in [Warm97, Zach97] is used. GeoSteiner 3.1 uses a two-phase scheme to solve the rectilinear Steiner minimum tree problem. First, a set of Rectilinear Full Steiner (FST) trees [1] that contains at least one rectilinear Steiner minimum tree are generated. A Steiner minimum tree is then constructed from this set by solving an integer programming problem. This package can compute optimal Euclidean and rectilinear Steiner trees as well as minimum spanning trees in hypergraphs. For a net with n terminals, the output of GeoSteiner 3.1 package is a set of segments. Each segment consists of a start point and an end point. These points may be the net terminals or the Steiner points. An example of the input and output of GeoSteiner 3.1 is shown in Figure 4.1 (a) and (b), respectively. Based on the output, Figure 4.1(c) and (d) show two possible RSMT.

In addition to the Minimum Steiner trees, a limited number of spanning trees are also constructed for each long net based on the RSMT generated by GeoSteiner 3.1. The procedure to build the spanning trees for a long net is implemented as follows: for a long net with n terminals, there are $n-1$ segments between these terminals. Each segment can be assumed as a 2-terminal net and four sub-trees are

[1] FST is a spanning tree between a subset of nodes of a graph where any node is a terminal node

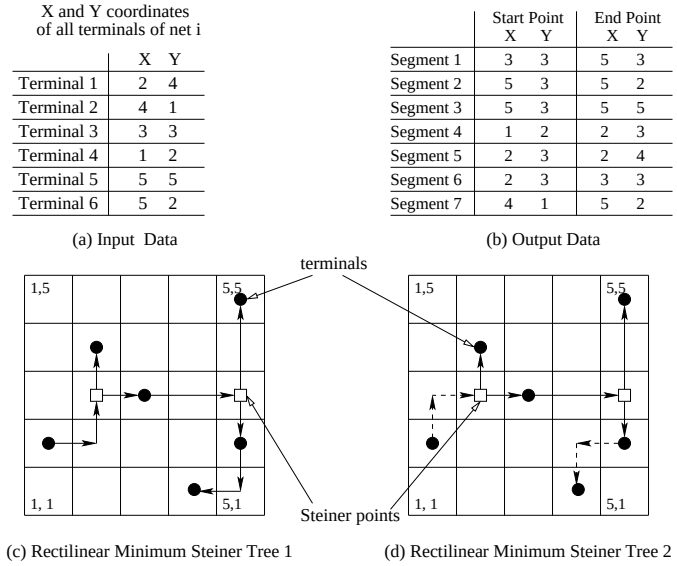


Figure 4.1: Generate Rectilinear Minimum Steiner Tree

constructed for this segment based on the additional tree construction algorithm proposed in [Behj02]. The sub-trees of all segments are then combined to form the spanning trees for the original long net. Figure 4.2 shows four spanning trees constructed for a 4-terminal net.

Table 4.1 lists the ILP relaxation (i.e. ILPR) results comparison between different tree construction approaches for long nets. In the first part of this table, only minimum Steiner trees are built for each long net. For the second part "Steiner+Spanning Trees", both spanning trees and minimum steiner trees are generated for each long net using the techniques described previously. Columns "Max_{rd}" and "MCong." represent the maximum routing density and the maximum estimated congestion, respectively. The maximum estimated congestion is calculated as: $Mcong = max\{r(e_i) \mid e_i \in E\}$ where $r(e_i)$ is the estimated routing demand of edge i ($r(e_i) = \sum_{j=1}^{t} a_{ij} p(x_j)$ and $p(x_j) = 1$ / number of trees produced

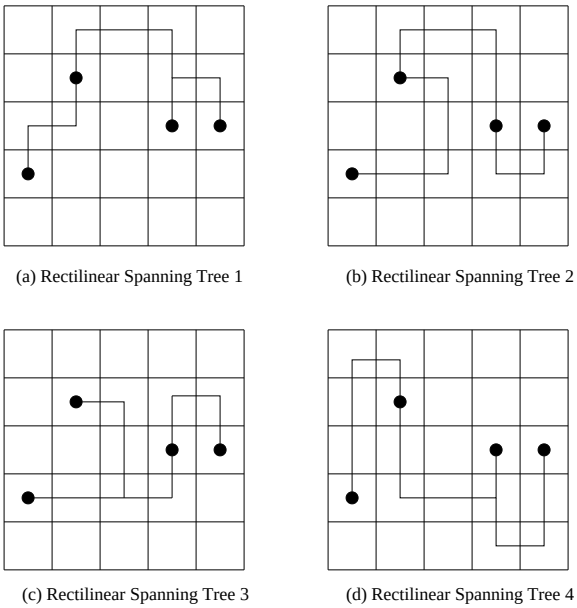


Figure 4.2: Spanning Tree Construction for Long Nets

for net k). From the table it can be observed that the global routing with spanning and minimum Steiner tree construction for long nets produces better results than that of global routing with only minimum Steiner tree construction in terms of maximum routing density and maximum estimated congestion.

4.2 Additional Tree Construction

4.2.1 Additional Tree Construction for 3-terminal Nets

In [Behj02], additional trees are constructed for nets that pass through the congested areas based on the congestion estimation. However, the proposed technique only build trees for 2-terminal nets. In this research, the additional tree construction technique for 2-terminal nets presented in [Behj02] is extended to 3-terminal

ILPR Results for Tree Construction of Long Nets (tot-iter:4)								
Bench	Only Minimum Steiner Trees				Steiner + Spanning Trees			
Marks	Tree	Max_{rd}	MCong	Time	Tree	Max_{rd}	MCong	Time
Fract	192	7	8.33	2	257	**5**	8.07	2
Prim1	1265	10	14.75	13	1747	**8**	14.53	18
Struct	2099	9	9.50	6	2099	9	9.50	9
Ind1	2055	20	27.32	28	3067	**16**	26.60	37
Prim2	3073	13	23.75	71	6253	**12**	22.45	91
Ind2	36422	14	21.77	1826	41523	**13**	20.24	2623
avq.large	50549	13	18.25	3844	55548	**12**	18.06	5058

Table 4.1: Tree Construction for Long Nets

nets. First, each 3-terminal net is separated into two 2-terminal sub-nets. Additional trees are then generated for these two sub-nets using the additional tree construction technique described above. Finally, the additional trees of two sub nets are combined to generate the spanning trees for the original 3-terminal net.

ILPR Results of Different Adding Tree Methods (tot-iter:4)												
Bench	Pure-Global-Routing				Additional-Tree-1				Additional-Tree-2			
Marks	Tree	Max_{rd}	MCong	Time	Tree	Max_{rd}	MCong	Time	Tree	Max_{rd}	MCong	Time
Fract	245	6	8.58	3	257	5	8.07	2	284	5	7.62	2
Prim1	1416	10	15.53	16	1747	8	14.53	18	2125	8	14.53	17
Struct	1582	9	9.50	8	2099	9	9.50	9	2390	9	9.5	7
Ind1	2946	18	26.60	35	3067	16	26.60	37	3073	16	26.60	31
Prim2	5469	13	22.47	85	6253	12	22.45	91	6517	12	22.45	89
Ind2	21564	15	21.24	1006	41523	14	20.24	2623	71820	14	20.18	4403
avq.large	33309	14	18.69	1861	55548	12	18.06	5058	78281	12	18.02	5908

Table 4.2: Comparison of Different Adding Trees Methods

In Table 4.2, a comparison of the results for three different adding tree implementations is given. These implementations are: (1)Pure-Global-Routing: In this implementation, the global routing is performed without adding trees for the nets that pass through congested areas. (2) Additional-Tree-1: In this part, additional

trees are made only for 2-terminal nets that pass through the congested edges to reduce congestion. (3) Additional-Tree-2. Extension to (2), additional trees are constructed for both 2-terminal and 3-terminal nets. All the experimental results are based on the Combined ILP model (WVEM) described next in section 4.3. Similar to Table 4.1, columns "Tree", "Max_{rd}" and "MCong." list the total number of trees generated, the maximum routing density, and the maximum estimated edge congestion. Comparing the results in first two parts, it can be observed that construction of additional trees can reduce the estimated congestion and the maximum routing density for almost all the circuits. However, the computation time is also increased due to the increasing number of trees. It should be noted that although the total number of trees increased, adding trees for both 2-terminal and 3-terminal nets does not improve the quality of solution too much compared with adding trees for only 2-terminal nets. One reason of that is: in [Behj02], at each adding tree iteration, the additional trees that pass through the congested edges, are discarded. This strict condition could limit the additional tree construction in highly congested areas because most of the additional trees will pass through at least one congested edge. However, without adding trees for the highly congested regions to relieve the congestion, it is impossible to decrease the maximum routing density.

4.2.1.1 Relaxation on Additional Tree Condition

Based on the results analysis in the previous section, the additional tree condition used in [Behj02] is relaxed such that additional trees that do not pass through the **highly** congested edges, are accepted. In other word, only the trees that via the

highly congested edges are discarded. The new condition allows more trees to be constructed for the congested nets, therefore, reduce the maximum estimated edge congestion more effectively. The pseudocode of the relaxed condition is illustrated in Figure 4.3. In this figure, "r(k)" is the routing demand of edge k.

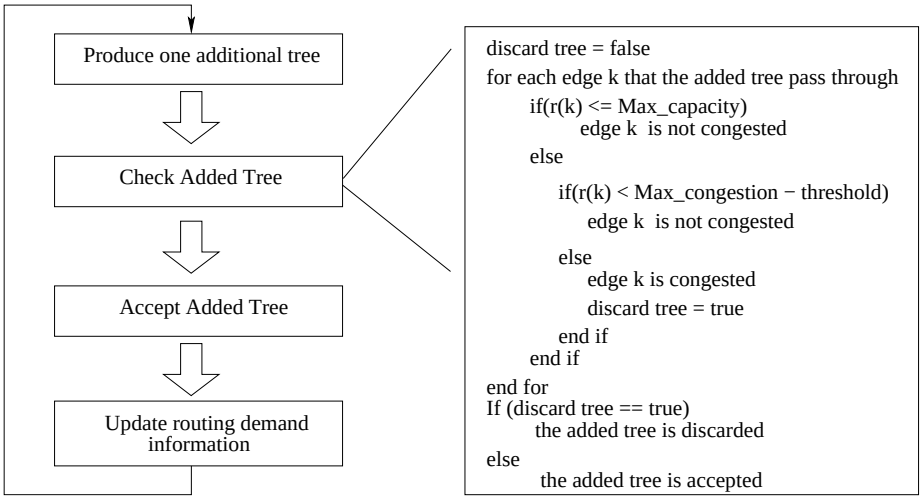


Figure 4.3: Pseudocode of Relaxed Adding Tree Condition

Table 4.3 shows the results comparison between strict condition and relaxed condition with adding tree in 4 and 3 iterations, respectively. Clearly, the relaxed condition reduces the maximum estimated congestion of all the circuits and the maximum routing density of almost all the circuits is also decreased.

LP Relaxation Results of Additional Tree Conditions								
Adding Trees for 2-terminal and 3-terminal Nets								
Bench	Strict condition tot-iter: 4				Relaxed condition tot-iter: 3			
Marks	Tree	Max_{rd}	MCong	Time	Tree	Max_{rd}	MCong	Time
Fract	284	5	7.62	2	397	5	6.54	3
Prim1	2125	8	14.53	17	2516	8	10.04	20
Struct	2390	9	9.5	7	2507	7	7.95	9
Ind1	3073	16	26.60	31	3974	15	21.17	35
Prim2	6517	12	22.45	89	8350	12	19.83	114
Ind2	71820	14	20.38	4403	63930	13	15.97	4125
avq.large	78281	12	18.02	5908	89306	11	13.69	9174

Table 4.3: Comparison of Different Additional Tree Conditions

4.3 Model Formulation

4.3.1 Wirelength Model (WLM)

For the Wirelength Minimization model (WLM) [Vann91, Behj02], the global routing problem is formulated as a 0/1 BIP problem by associating a variable x_i with each tree which connects a net. The variable x_i equals 1 if that particular tree is selected and equals 0 otherwise. The selection of one and only one tree for each net is imposed by the following constraints:

$$\sum_{x_j \in N_k} x_j = 1, \quad k \in \{1, ..., n\}$$

where n is the number of nets and N_k is the set of steiner trees created for net k. All the possible ways to connect nets in the two routing layers are represented by a (0,1) matrix $[a_{ij}]$, with the i^{th} row corresponding to the i^{th} edge in the grid graph and each column corresponding to different ways of connecting a net. The element

in a_{ij} is expressed as:

$$a_{ij} = \begin{cases} 1 & \text{if tree j passes through edge i;} \\ 0 & \text{otherwise.} \end{cases}$$

The edge capacity of each edge is represented by the following constraints:

$$\sum_{j=1}^{t} a_{ij} x_j \leq c_i, \quad i \in \{1, ..., p\}$$

where p is the number of edges on the grid graph, t is the total number of trees produced for all the nets, and c_i is the edge capacity of the i^{th}edge. In this thesis, the edge capacity c_i is represented by a single variable "z_{max}" for all the edges. The global routing problem with the objective of Minimizing the Total Wire Length is then formulated as:

$$\text{Minimize} \sum_{j=1}^{t} w_{lj} x_j \tag{4.1}$$

Subject to

$$\sum_{x_j \in N_k} x_j = 1, \quad k \in \{1, ..., n\}$$

$$\sum_{j=1}^{t} a_{ij} x_j \leq z_{max}, \quad i \in \{1, ..., p\}$$

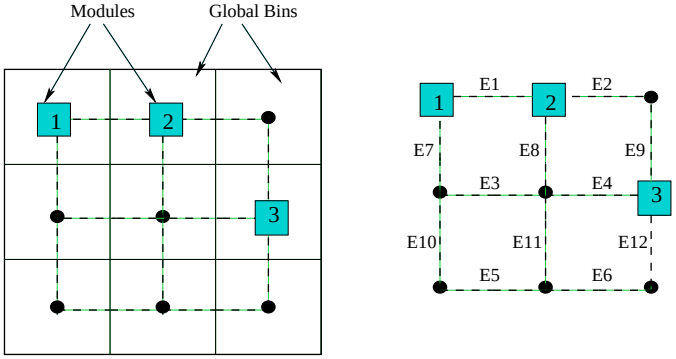
$$x_j \in \{0, 1\} \quad j \in \{1, ..., t\}$$

$$z_{max} \in \{0, C\}$$

where w_{lj} represents the weight associated with the length of tree j. This weight is calculated as the normalized length of tree j with respect to the rest of the trees in net k:

$$w_{lj} = \frac{length\ of\ tree\ j}{max\ length\ of\ trees\ constructed\ for\ net\ k} \tag{4.2}$$

Figure 4.4 shows a global bin graph and the corresponding grid graph of a simple circuit. The circuit has 3 modules and 2 nets. $Net1$ connects modules 1, 2, and 3, and $Net2$ connects modules 1 and 2.



(a) 3 by 3 Global Bin Graph (b) 3 by 3 Grid Graph

Figure 4.4: Example of Wirelength Minimization Model

Net1 can be routed by four trees:

Tree1 : (x_1) passes through edges $E1, E3, E4, E7$, the wirelength is 4;

Tree2 : (x_2) passes through edges $E1, E2, E9$, the wirelength is 3;

Tree3 : (x_3) passes through edges $E1, E8, E4$, the wirelength is 3;

Tree4 : (x_4) passes through edges $E3, E4, E7, E8$, the wirelength is 4;

Net2 can be routed by two trees:

Tree5 : (x_5) passes through edges $E1$, the wirelength is 1;

Tree6 : (x_6) passes through edges $E3, E7, E8$, the wirelength is 3;

The value for w_j can be calculated using equation 4.2. Thus, the wirelength minimization model for the example circuit can be formulated as:

$$\text{Minimize } x_1 + 0.75x_2 + 0.75x_3 + x_4 + 0.67x_5 + x_6$$

subject to

$$
\begin{aligned}
Net1: &\quad x_1 + x_2 + x_3 + x_4 \leq 1 \\
Net2: &\quad x_5 + x_6 \leq 1 \\
E1: &\quad x_1 + x_2 + x_3 + x_5 \leq z_{max} \\
E2: &\quad x_2 \leq z_{max} \\
E3: &\quad x_1 + x_4 + x_6 \leq z_{max} \\
E4: &\quad x_1 + x_3 + x_4 \leq z_{max} \\
E7: &\quad x_1 + x_4 + x_6 \leq z_{max} \\
E8: &\quad x_3 + x_4 + x_6 \leq z_{max} \\
E9: &\quad x_2 \leq z_{max} \\
&\quad x_j \in \{0,1\}, \quad z_{max} \in \{0, C\}
\end{aligned}
$$

4.3.2 Edge Capacity Model (ECM)

In [Ragh87, Karp87] , the Edge Capacity Minimization model (ECM) is proposed to minimize the maximum capacity of the edges.

$$\text{Minimize } z_{max} \tag{4.3}$$

Subject to

$$
\begin{aligned}
&\sum_{x_j \in N_k} x_j = 1, \quad k \in \{1, ..., n\} \\
&\sum_{j=1}^{t} a_{ij} x_j - z_{max} \leq 0, \quad i \in \{1, ..., p\} \\
&x_j \in \{0,1\}, \quad j \in \{1, ..., t\} \\
&z_{max} \in \{0, ..., C\},
\end{aligned}
$$

where z_{max} represents the maximum capacity of the edges. Based on the same simple circuit example shown in Figure 4.4, the edge capacity minimization model can be formulated as follows:

$$\text{Minimize } z_{max}$$

$$\begin{aligned}
\text{Subject to} \\
Net1: &\quad x_1 + x_2 + x_3 + x_4 \leq 1 \\
Net2: &\quad x_5 + x_6 \leq 1 \\
E1: &\quad x_1 + x_2 + x_3 + x_5 - z_{max} \leq 0 \\
E2: &\quad x_2 - z_{max} \leq 0 \\
E3: &\quad x_1 + x_4 + x_6 - z_{max} \leq 0 \\
E4: &\quad x_1 + x_3 + x_4 - z_{max} \leq 0 \\
E5: &\quad x_1 + x_4 + x_6 - z_{max} \leq 0 \\
E6: &\quad x_3 + x_4 + x_6 - z_{max} \leq 0 \\
E7: &\quad x_2 - z_{max} \leq 0 \qquad x_j \in \{0,1\}, \quad z_{max} \in \{0, C\},
\end{aligned}$$

4.3.3 Via Minimization Model (VMM)

As the needs of interconnect increases, over-the-cell multi-layer routing becomes essential for today's VLSI circuit design. However, more metal layers do not necessarily imply less interconnect area and die size because of the space taken up by the vias on the lower layers. In addition to the increased area usage, vias also increase manufacturing cost and cause reliability problem. To consider the via minimization in the global routing problem, a via minimization model (VMM) is formulated as

follows:

$$\text{Minimize } \sum_{j=1}^{t} w_{bj} x_j \tag{4.4}$$

$$\text{Subject to}$$

$$\sum_{x_j \in N_k} x_j = 1, \quad k \in \{1, ..., n\}$$

$$\sum_{j=1}^{t} a_{ij} x_j \leq z_{max}, \quad i \in \{1, ..., p\}$$

$$x_j \in \{0, 1\}, \quad j \in \{1, ..., t\}$$

$$z_{max} \in \{0, ..., C\},$$

where w_{bj} is the weighting factor associated with the number of vias (bends) in the path of tree j and is calculated as:

$$w_{bj} = \frac{number\ of\ vias\ in\ tree\ j}{max\ number\ of\ vias\ in\ trees\ produced\ for\ net\ k} \tag{4.5}$$

Based on the same circuit example shown in Figure 4.4, the number of vias of each tree is:

$Net1$:

$Tree1 : (x_1)$ passes through edges $E1, E3, E4, E7$, the number of vias is 2;

$Tree2 : (x_2)$ passes through edges $E1, E2, E9$, the number of vias is 1;

$Tree3 : (x_3)$ passes through edges $E1, E8, E4$, the number of vias is 2;

$Tree4 : (x_4)$ passes through edges $E3, E4, E7, E8$, the number of vias is 2;

$Net2$:

$Tree5 : (x_5)$ passes through edges $E1$, the number of vias is 0;

$Tree6 : (x_6)$ passes through edges $E3, E7, E8$, the number of vias is 2;

For trees that have zero number of vias, the number of vias is set to 0.1 such that

the vias-weight of these trees is less than that of all the other trees that have at least one bend. For example, for $Tree1$ of $Net2$, which has zero number of vias, the w_{bj} is calculated as:

$$\frac{0.1}{max\ number\ of\ vias\ in\ trees\ produced\ for\ Net\ 2}$$

The objective function of VMM model for this example circuit is:

$$\text{Minimize } \frac{2}{2}x1 + \frac{1}{2}x2 + \frac{2}{2}x3 + \frac{2}{2}x4 + \frac{0.1}{2}x5 + \frac{2}{2}x6$$

4.3.4 Combined Model-1 (WVEM)

The WLM model only takes the total wire-length of all the nets into consideration when choosing trees for each net in the objective function. However, many other design factors, such as congestion and number of vias in the global routing phase are also considered very important. Therefore, a combined model (WVEM) that incorporates wire-length, vias and maximum routing density into the objective function as penalty terms is proposed.

$$\text{Minimize } \sum_{j=1}^{t}(\beta_l w_{lj} + \beta_b w_{bj})x_j + \beta_z z_{max} \tag{4.6}$$

Subject to

$$\sum_{x_j \in N_k} x_j = 1, \quad k \in \{1, ..., n\}$$
$$\sum_{j=1}^{t} a_{ij}x_j - z_{max} \leq 0, \quad i \in \{1, ..., p\}$$
$$x_j \in \{0, 1\}, \quad j \in \{1, ..., t\} \quad z_{max} \in \{0, ..., C\},$$

where w_{lj} and w_{bj} are the weighting factors associated with wire-length and number of vias respectively. The scalars β_l, β_b and β_z are used to change the emphasis of the respective weighting factors. The upper bound of z_{max} is set experimentally.

LP Relaxation + Rounding Results for WLM Model and WVEM Model										
Circuit	WLM Model					WVEM Model				
	Wire	Vias	M_{rd}	OF	Time	Wire	Vias	M_{rd}	OF	Time
ibm01	60112	12646	30	3048	58	60242	12022	24	3006	39
ibm02	165469	27738	57	6120	52	166175	26258	48	6146	54
ibm03	145072	21967	50	3206	69	145803	20808	35	2847	77
ibm04	161380	25882	40	5742	133	162928	23996	29	4997	136
ibm05	410000	50621	66	147	74	410080	48682	66	220	77
Tot	942033	138854	243	18263	386	945228	131766	202	17216	383
Imp	-	-	-	-	-	-0.3%	5%	17%	6%	0%
ibm06	275803	40655	70	7368	107	276046	38956	54	7518	111
ibm07	362218	54104	50	6357	346	363896	51044	39	4883	363
ibm08	403090	67888	50	8443	450	403591	64775	40	7338	461
ibm09	410232	61444	50	11594	830	411573	58190	35	8923	851
ibm10	574234	91799	60	9802	1306	574822	86689	46	8233	1324
Tot	2025577	315890	280	43564	3039	2029928	299654	214	36895	3110
Imp	-	-	-	-	-	-0.2%	5%	23%	15%	-2%
TotImp	-	-	-	-	-	-0.25%	5%	20%	10%	-1%

Table 4.4: Comparison of WLM and WVEM model

Table 4.4 compares the Wire-length Model (WLM) with the combined model 1 (WVEM). Column "Wire", "Bends", "M_{rd}" and "OF" list the total wire-length, total number of vias, maximum routing density and total overflow. Row "Imp" shows the average percentage improvement. As shown in Table 4.4, the WVEM model improves the total number of vias, maximum routing density and total overflow by about 5%, 20% and 10% on average with almost the same total wire-length, and computation time.

Similarly, in Table 4.5 and 4.6, the VMM model and ECM models are compared with the WVEM model. From the comparison, we can see that the combined model WVEM optimizes several objectives simultaneously and effectively.

LP Relaxation + Rounding Results for VMM Model and WVEM Model										
Circuit	VMM Model					WVEM Model				
	Wire	Vias	M_{rd}	OF	Time	Wire	Vias	M_{rd}	OF	Time
ibm01	60209	11993	30	3156	45	60242	12022	24	3006	39
ibm02	166169	26236	59	6340	52	166175	26258	48	6146	54
ibm03	145777	20762	50	3224	70	145803	20808	35	2847	77
ibm04	162831	23814	40	5870	133	162928	23996	29	4997	136
ibm05	410080	48679	68	299	74	410080	48682	66	220	77
Tot	945066	131484	247	18889	374	945228	131766	202	17216	383
Imp	-	-	-	-	-	0%	-0.2%	18%	9%	0%
ibm06	276014	38921	70	7639	107	276046	38956	54	7518	111
ibm07	363668	50826	50	6506	345	363896	51044	39	4883	363
ibm08	403551	64686	50	8842	449	403591	64775	40	7338	461
ibm09	411531	58072	50	11579	828	411573	58190	35	8923	851
ibm10	574792	86527	60	10903	1303	574822	86689	46	8233	1324
Tot	2029556	299032	280	45469	3032	2029928	299654	214	36895	3110
Imp	-	-	-	-	-	0%	-0.2%	23%	19%	-2%
TotImp	-	-	-	-	-	0%	-0.2%	20%	14%	-1%

Table 4.5: Comparison of VMM and WVEM model

LP Relaxation + Rounding Results for ECM Model and WVEM Model										
Circuit	ECM Model					WVEM Model				
	Wire	Vias	M_{rd}	OF	Time	Wire	Vias	M_{rd}	OF	Time
ibm01	61403	13862	24	2847	48	60242	12022	24	3006	39
ibm02	167389	28938	49	5887	53	166175	26258	48	6146	54
ibm03	148043	22728	36	2846	72	145803	20808	35	2847	77
ibm04	168511	27529	30	5206	142	162928	23996	29	4997	136
ibm05	411113	51655	66	103	76	410080	48682	66	220	77
Tot	956459	144712	205	16889	391	945228	131766	202	17216	383
Imp	-	-	-	-	-	1%	9%	1%	-2%	0%
ibm06	286310	41985	54	6657	109	276046	38956	54	7518	111
ibm07	369211	56779	39	5025	361	363896	51044	39	4883	363
ibm08	409375	68193	40	7090	458	403591	64775	40	7338	461
ibm09	417349	63102	35	8942	845	411573	58190	35	8923	851
ibm10	578792	93296	47	7988	1316	574822	86689	46	8233	1324
Tot	2061037	353355	215	35702	3089	2029928	299654	214	36895	3110
Imp	-	-	-	-	-	1.5%	15%	0%	-3%	0%
TotImp	-	-	-	-	-	1%	12%	1%	-2%	0%

Table 4.6: Comparison of ECM and WVEM model

4.3.5 Combined Model-2 (WVZM)

In order to produce a global routing solution with appropriate wire density distribution, minimizing the maximum routing density and the total overflow are both important. However, the WVEM model only considers the maximum routing density. Thus, another combined model WVZM is proposed to optimize the total wire-length, the number of via, maximum routing density and total overflow simultaneously.

$$\text{Minimize} \sum_{j=1}^{t} (\beta_l w_{lj} + \beta_b w_{bj}) x_j + \beta_z z_i \tag{4.7}$$

Subject to

$$\sum_{x_j \in N_k} x_j = 1, \quad k \in \{1, ..., n\}$$

$$\sum_{j=1}^{t} a_{ij} x_j - Cap_i \leq z_i, \quad i \in \{1, ..., p\}$$

$$x_j \in \{0, 1\} \quad j \in \{1, ..., t\}$$

$$z_i \in \{0, C\}$$

where Cap_i is the routing supply of each edge and z_i is a variable associated with the routing overflow of each edge. The upper bound of variable Z_i must be set as low as possible such that the maximum routing density is minimized.

In Table 4.7 the ECM model is compared with WVZM model. It is clear that the WVZM model reduces the total overflow and total number of vias about 53% and 10% on average, but at the cost of increasing maximum routing density and computation time. The comparison of WLM, VMM, ECM, WVEM and WVZM

LP Relaxation + Rounding Results for ECM Model and WVZM Model										
Circuit	ECM Model					WVZM Model				
	Wire	Vias	M_{rd}	OF	Time	Wire	Vias	M_{rd}	OF	Time
ibm01	61403	13862	24	2847	48	60609	12663	25	1792	46
ibm02	167389	28938	49	5887	53	166472	27106	48	3845	56
ibm03	148043	22728	36	2846	72	146116	21593	36	769	102
ibm04	168511	27529	30	5206	142	163333	24953	31	2437	220
ibm05	411113	51655	66	103	76	410082	48709	66	6	77
Tot	956459	144712	205	16889	391	946612	135024	206	8849	501
Imp	-	-	-	-	-	1%	7%	0%	48%	-22%
ibm06	286310	41985	54	6657	109	276681	40077	54	3758	114
ibm07	369211	56779	39	5025	361	364048	52037	42	2152	512
ibm08	409375	68193	40	7090	458	404328	66488	44	2507	525
ibm09	417349	63102	35	8942	845	412261	60094	38	3250	986
ibm10	578792	93296	47	7988	1316	575071	88061	53	3312	1500
Tot	2061037	353355	215	35702	3089	2032389	306759	231	14979	3637
Imp	-	-	-	-	-	1%	13%	-7%	58%	-15%
TotImp	-	-	-	-	-	1%	10%	-7%	33%	-18%

Table 4.7: Comparison of ECM and WVZM model

models for all the IBM benchmarks is shown in Figure 4.5. It is clear that the combined WVZM model can reduce the total overflow effectively without deteriorating the total wire-length and total number of vias.

4.4 A New Global Routing Framework (RNWO)

By incorporating several design factors into a single objective function, the combined WVZM model tends to reduce the total overflow effectively without sacrificing the total wire-length and total number of vias.

Comparing the WVZM model with a sequential maze router [Kast], Table 4.8 indicates that the proposed ILP based router produces better results than the maze router [Kast] in terms of wire-length, total number of vias and computation time. However, the sequential router achieves better solutions in terms of maximum routing density and total overflow. To improve the congestion, a pure IP model that **R**outes as many **N**ets **W**ithout generating **O**verflow (RNWO) is proposed and

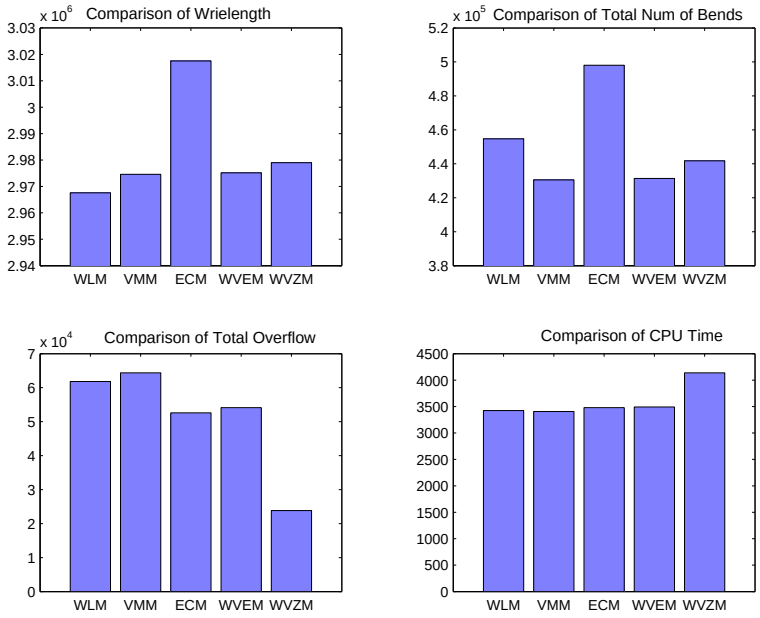


Figure 4.5: Comparison of Different Models

Results Comparison for Maze Router & ILP Router (WVZM Model)										
Circuit	Maze Router					ILP Router (WVZM)				
	Wire	Vias	M_{rd}	OF	Time	Wire	Vias	M_{rd}	OF	Time
ibm01	75915	26802	17	394	109	60609	12663	25	1792	46
ibm02	212841	70535	38	1004	167	166472	27106	48	3845	56
ibm03	189595	59796	32	342	174	146116	21593	36	769	102
ibm04	196582	61311	27	875	376	163333	24953	31	2437	220
ibm05	420581	102388	63	0	229	410082	48709	66	6	77
Tot	1095514	320832	177	2615	1055	946612	135024	206	8849	501
Imp	-	-	-	-	-	13%	58%	-14%	-70%	52%
ibm06	343571	105391	35	613	385	276681	40077	54	3758	114
ibm07	450746	133866	37	718	767	364048	52037	42	2152	512
ibm08	484068	151952	35	981	811	404328	66488	44	2507	525
ibm09	484090	138099	29	485	1242	412261	60094	38	3250	986
ibm10	678502	207133	42	414	1417	575071	88061	53	3312	1500
Tot	2440977	736414	178	3211	4622	2032389	306759	231	14979	3637
Imp	-	-	-	-	-	17%	58%	-23%	-78%	21%

Table 4.8: Results Comparison of Different Routers

formulated as follows:

$$\text{Maximize } \sum_{j=1}^{t} x_j \tag{4.8}$$

Subject to

$$\sum_{x_j \in N_k} x_j \leq 1, \quad k \in \{1, ..., n\} \tag{4.9}$$

$$\sum_{j=1}^{t} a_{ij} x_j \leq Cap_i, \quad i \in \{1, ..., p\} \tag{4.10}$$

$$x_j \in \{0, 1\}, \quad j \in \{1, ..., t\} \tag{4.11}$$

Results of RNWO Model for IBM Benchmark							
Circuit	T-Net	U-Net	Wire	Vias	M_{rd}	OF	Time
ibm01	9128	317/3%	51249	11369	14	0	7
ibm02	14270	408/3%	149951	26117	34	0	13
ibm03	15289	152/1%	139419	21468	30	0	18
ibm04	19738	399/2%	149908	24363	23	0	16
ibm05	21135	0/0%	410083	50469	63	0	21
ibm06	25797	283/1%	253159	38845	33	0	26
ibm07	34378	327/1%	348960	53124	36	0	64
ibm08	35195	275/1%	382489	65459	32	0	67
ibm09	39592	574/1%	379968	58789	28	0	67
ibm10	49491	424/1%	548172	90508	40	0	94

Table 4.9: Results of RNWO Model

The objective of the RNWO model is to maximize the total number of routed nets. Constraint (4.9) allows one or zero tree selected for each net, which means some nets may not be routed due to the limited routing capacity of each edge. he edge capacity of each edge is represented by constraint (4.10) and p is the number of edges on the grid graph. Table 4.9 shows the experimental results of the proposed RNWO model. Column "T_Net" lists the total number of nets, while

column "U_Net" gives the total number of unrouted nets (the left side shows the total number of unrouted nets and the right side shows the percentage of unrouted nets). Clearly, the RNWO model is unable to route all the nets based on the trees constructed for each net. To route the remaining unrouted nets without overflow or with low overflow, we need to either build trees with larger detour or combine the pure IP router with a sequential router. According to the above analysis, a new combined concurrent/sequential global routing framework is proposed as shown in Figure 4.6. Initially, all the nets are decomposed into 2-terminal nets and the rectilinear minimum steiner trees (RMSTs) are constructed for each 2-terminal net. Following that, the pure IP model (RNWO model) is used to route as many nets as possible without generating any overflow. Based on the routing demand consumed by the routed nets (about 98% nets are routed), the additional trees are added to further reduce congestion. The RNWO model is invoked again to route the remaining unrouted nets. Due to the detour limitation of the additional trees, there are still some nets that may not be able to be routed by the ILP router. Therefore, a maze router is finally invoked to route the remaining unrouted nets.

In Table 4.10, the proposed combined router is compared with the pure sequential maze router [Kast]. It is clear from the results in Table 4.10 that the combined router outperforms maze router by a wide margin. For the wire-length and total number of vias, the combined router can reduce both by about 11% and 49% respectively compared with the maze router [Kast]. For the congestion, the combined router produces much lower total overflow than that of the pure maze router. More important, the combined router is 6.0x faster than the maze router. The combined router is also compared with Fengshui 5.1 [Hads03] and multi-commodity flow-

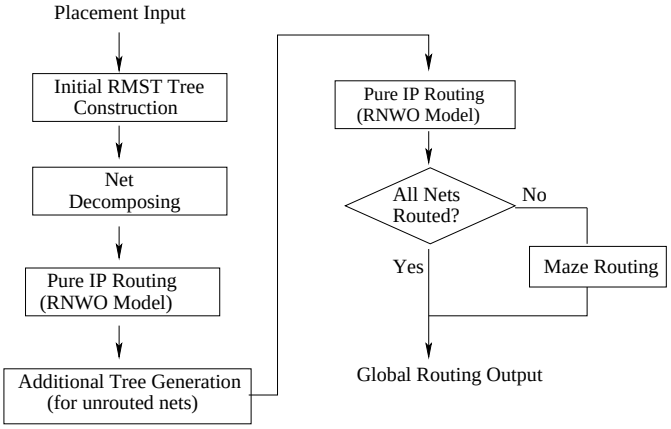


Figure 4.6: New Framework of ILP based Router

based global router [Albe01], as shown in Table 4.11. It is clear that the combined router produces 48% less total overflow and 19% less CPU time than Fengshui with similar total wire-length. Also, the combined router is 8x faster, and produces 3% shorter wire-length on average compared to the multi-commodity flow-based router [Albe01].

Results Comparison for Maze Router & Combined Router										
Circuit	Maze Router					RNWO + Maze				
	Wire	Vias	M_{rd}	OF	Time	Wire	Vias	M_{rd}	OF	Time
ibm01	75915	26802	17	394	109	70052	18880	16	158	34
ibm02	212841	70535	38	1004	167	177707	36101	35	11	46
ibm03	189595	59796	32	342	174	150460	25345	30	3	27
ibm04	196582	61311	27	875	376	176259	34389	26	447	72
ibm05	420581	102388	63	0	229	410083	50469	63	0	21
Tot	1095514	320832	177	2615	1055	984561	165184	170	619	200
Imp	-	-	-	-	-	10%	49%	4%	76%	81%
ibm06	343571	105391	35	613	385	284713	49202	33	0	36
ibm07	450746	133866	37	718	767	386994	65750	36	9	127
ibm08	484068	151952	35	981	811	416445	76956	32	15	118
ibm09	484090	138099	29	485	1242	425324	73084	28	17	109
ibm10	678502	207133	42	414	1417	595515	106668	41	51	157
Tot	2440977	736414	178	3211	4622	2108991	371660	170	92	547
Imp	-	-	-	-	-	13%	49%	4%	97%	88%

Table 4.10: Results Comparison of Different Routers

Results Comparison for Feingshui5.1 & Multicommodity & ILP+Maze Router									
Circuit	Fengshui 5.1			Multicommodity Router			RNWO + Maze		
	Wire	OF	Time	Wire	OF	Time	Wire	OF	Time
ibm01	66006	189	15.1	68981	43	151.2	70052	158	34
ibm02	178892	64	47.9	190418	3	494.5	177707	11	46
ibm03	152392	10	35.2	160755	0	329.8	150460	3	27
ibm04	173241	465	54.1	176610	225	326.6	176259	447	72
ibm05	412197	0	104.8	410954	0	28.2	410083	0	21
Tot	982728	728	257.1	1006618	271	1330.3	984561	619	200
Imp	-	-	-	-2%	63%	-81%	0%	15%	22%
ibm06	289276	35	80.1	296981	0	951.8	284713	0	36
ibm07	378994	309	122.2	408510	0	1229.0	386994	9	127
ibm08	415285	74	113.8	429913	0	865.7	416445	15	118
ibm09	427556	52	125.1	442514	0	726.7	425324	17	109
ibm10	599937	51	212.9	634247	0	1068.4	595515	51	157
Tot	2111048	521	654.1	2212165	0	4841.6	2108991	92	547
Imp	-	-	-	-4.5%	100%	-86%	0%	82%	16%

Table 4.11: Results Comparison for Different Routers

4.5 Summary

In this chapter, the tree construction is performed for both short nets and long nets. For long nets, a software package [Warm01] is used to generate the rectilinear minimum Steiner tree for each net. The experimental results indicate that generating both rectilinear minimum Steiner trees and spanning trees for long nets can lower the maximum routing density for some circuits with the expense of increasing computation time. The additional tree construction technique for 2-terminal nets presented in [Behj02] is also e extended to 3-terminal nets. However, the numerical results show that with the strict additional tree condition, adding trees for both 2 and 3-terminal nets can not reduce the congestion effectively. By relaxing the additional tree condition, the maximum estimated congestion of all the circuits is reduced and the maximum routing density is also decreased for most circuits.

In addition to the route construction, the modeling of different global routing objectives are investigated in detail. As can be seen from the experimental results,

the wirelength minimization model (WLM), maximum edge capacity minimization model (ECM) and the via minimization model (VMM) can improve the corresponding objective effectively.

To solve the global routing problem with several competing objectives, WVEM model and WVZM model incorporate these design factors (i.e. wirelength, the total number of vias and the congestion) into a single objective function as penalty terms. Each design factor has an associated weighting parameter and the selection of the value of weighting parameter determines the importance of the corresponding design factor. One advantage of such an approach is that it tends to produce somewhat more predictable solutions that optimize all different objectives. The results illustrate that the proposed WVZM model provides comparable speed to the sequential router [Kast] and better wirelength and total number of vias.

To further improve the congestion a hybrid framework (i.e. RNWO) is proposed. Experimental results obtained indicate that the combined router outperforms the pure sequential router [Kast]. In addition, the combined router produces 48% less total overflow than Fengshui 5.1 [Hads03] with similar wire-length and 19% less CPU time.

In the next chapter, two hierarchical global routing methods are introduced and compared with the flat ILP based global router developed earlier.

Chapter 5

Hierarchical ILP Based Global Routing

Some of the flat ILP based global routing models described in Chapter 4 (e.g. WVEM and WVZM models) tend to encounter a scaling problem as circuits increase in size. Consequently, a hierarchical global routing becomes essential for the design flow. Inspired by the successful application of multilevel methods in circuit partitioning and placement a top-down hierarchical routing algorithm would attempt to use a hierarchy on the routing graph to decompose the large routing problem into smaller and more manageable pieces. There are several approaches to hierarchical global routing [Yang06]: top-down hierarchical routing, bottom-up hierarchical routing and multilevel global routing [Cong05, Chen05b, Chen06]. All of these methods attempt to solve the scaling problem of large designs by building multilevel hierarchical representations of the routing regions.

5.1 Top-down Global Routing With Net Refinement

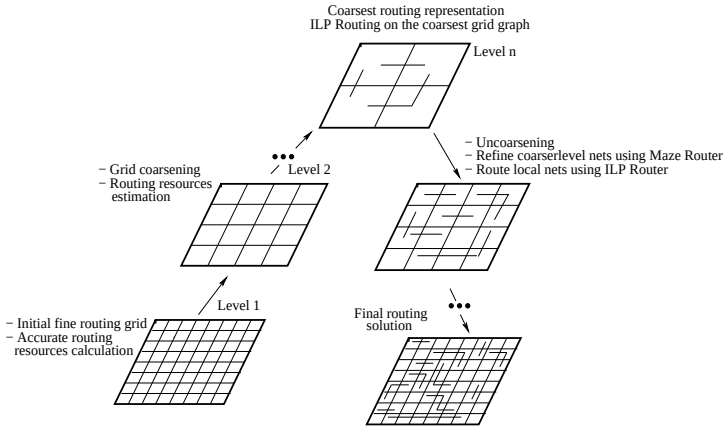


Figure 5.1: Top-down Global Routing with Net Refinement

5.1.1 Coarsening Process

Figure 5.1 shows a flow chart of the overall top-down global routing approach. The first step in this top-down global routing is to build up the hierarchical levels of routing region representations. At the flat level (i.e. level 1), the routing region is partitioned into an array of fine grid cells. During the coarsening process, the flat level grid cells are merged to build the coarser level representations. In addition to the merging of grid cells, the new netlist is also generated based on the newly created routing region. At each coarsening step, the routing resource of each grid cell edge is estimated.

Merging Grid Cells

At each coarsening level, a maximum of four adjacent grid cells from the previous level are merged to build a new grid cell in the current level. Based on the newly merged grid, the coordinates of each terminal cell are also updated. For example in Figure 5.2, by merging neighboring grid cells, the flat level 8 x 7 grid graph is transformed to a coarser 4 x 4 level grid graph. For two-terminal net "Net1", the coordinates of the terminal cells are also changed from (3,5) (6,3) to (2,3) (3,2).

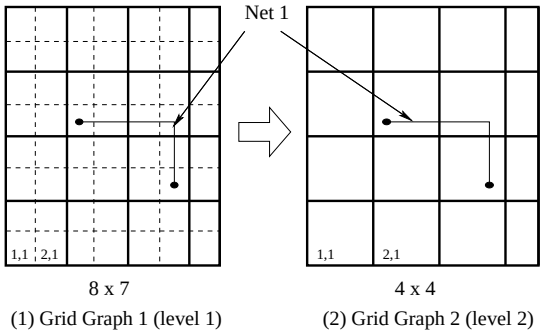


Figure 5.2: Merging of Grid Cells

New Net-list Generation

Based on the coarser grid graph constructed, a new net-list is generated such that all the terminal cells of each net are located in different grid cells. During this process, some local nets are absorbed since all the terminal cells of the nets are relocated to a single grid cells, as shown in Figure 5.3. The time complexity of building the hierarchical levels is $O(b)$, where b is the number of global bins.

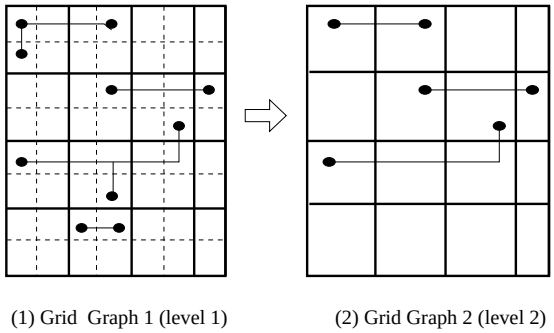


Figure 5.3: New Net-list Generation

Estimating routing resources

At the flat level, the routing resource of each grid edge is given. During the coarsening process, the routing resource of the merged grid edge is obtained by adding the routing resource of edges that consist of the merged grid edge. For example, in Figure 5.4, the capacity of edge "E1" in grid graph 2 sums to the capacity of edge "E1" and "E2" in grid graph 1.

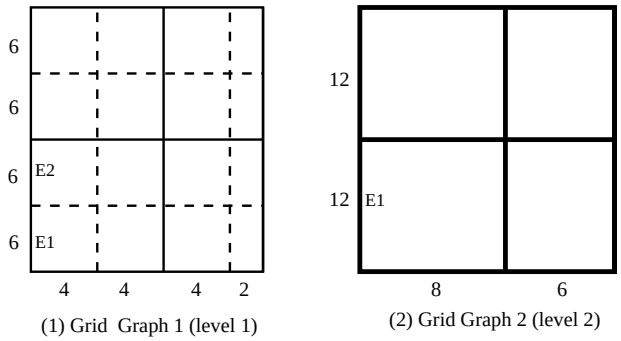


Figure 5.4: Routing Resource Estimation

5.1.2 Coarsest Level

At the coarsest level, an ILP based router [Yang05] is used to route all the nets.

5.1.3 Uncoarsening Process

During the uncoarsening process, two types of nets appear. (1) Local Nets: the nets that just appear at the current level (e.g. Net4 shown in Figure 5.5(2)). (2) Coarser-level Nets: the nets that are carried over from the previous coarser level routing to current level (e.g. Net1, Net2, Net3 shown in Figure 5.5(1) and Figure 5.5(2)). The coarser level nets are refined using a maze-searching algorithm and

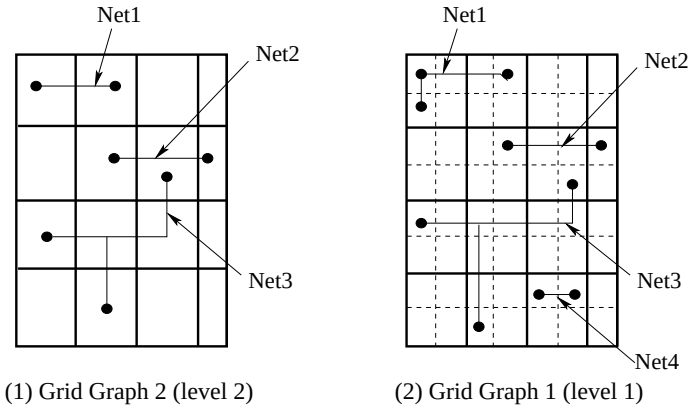


Figure 5.5: Different type of nets at refinement stage

the local nets are routed by an ILP based router.

Refinement of Coarser-level Nets

In order to take advantage of the coarser level routing results, a net-by-net maze routing based refinement algorithm is implemented. The algorithm uses the path

obtained at coarser level routing as a guide to search for a path at the current level. A preferred region is first defined as a set of grid cells that the coarser level path goes through. The path searching is then performed within the preferred region. Figure 5.6 shows an example of the path refinement. In this example, the refinement algorithm finds a path for Net 3 that is totally within its preferred region. Currently, the path refinement algorithm only takes the total wirelength into consideration. In the future version of path refinement algorithm, both wirelength and congestion will be considered.

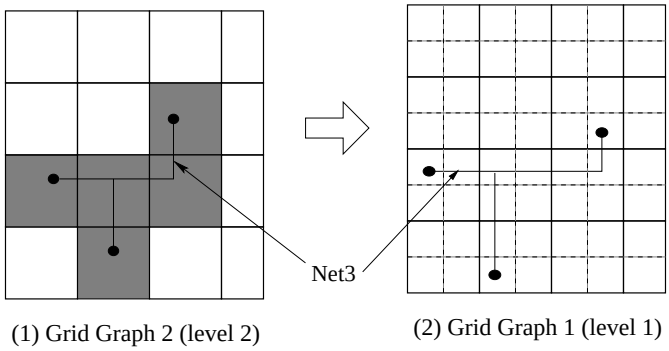


Figure 5.6: Refinement of coarser level nets

Routing of Local Nets

Following the refinement of coarser level nets, the routing resource of each grid edge is updated and the local nets that just appear at the current level are routed by an ILP router.

Revised ILP Models for Top-down Global Routing

The ILP models used in this thesis are similar to those proposed by [Yang05]. The only difference is that at each coarsening level, the updated routing demand of each edge (RD_i) is considered in the edge capacity constraints of the formulation. As an example, the revised formulation of ECM model for top-down routing is as follows:

$$\text{Minimize } z_{max} \tag{5.1}$$

Subject to

$$\sum_{x_j \in N_k} x_j = 1, \quad k \in \{1, ..., n\}$$
$$\sum_{j=1}^{t} a_{ij} x_j + RD_i - z_{max} \leq 0, \quad i \in \{1, ..., p\}$$
$$x_j \in \{0, 1\}, \quad j \in \{1, ..., t\}$$
$$z_{max} \in \{0, ..., C\},$$

where z_{max} represents the maximum capacity of the edges and RD_i is the routing demand of edge i, which is updated after each coarsening level routing.

5.1.4 Experimental Results

Table 5.1 shows the results comparison between the flat level routing and the top-down routing with net refinement. To show the effectiveness of the method, the Edge Capacity Model (ECM) is randomly selected as the ILP model for the top-down ILP routing. Row "Imp" shows the average percentage improvement achieved

Flat Level Routing and Top-down Routing with Maze Refinement (ECM model)																
Bench	Flat Routing				Top-down (2 levels)				Top-down (3 levels)				Top-down (4 levels)			
Marks	Wire	Vias	M_{rd}	Time	Wire	Vias	M_{rd}	Time	Wire	Vias	M_{rd}	Time	Wire	Vias	M_{rd}	Time
Prim1	669328	692	8	15	834513	1119	20	19	951402	1302	20	20	1075537	1508	21	25
Struct	337900	633	7	7	338024	734	10	11	449265	1242	13	20	635847	1732	17	39
Imp	–	–	–	–	-13%	-39%	-96%	-42%	-58%	-92%	-118%	-59%	-74%	-145%	-153%	-241%
Ind1	1020282	1314	17	31	1245054	2210	30	30	1325084	2430	32	28	1374243	2528	34	29
Prim2	3253505	2737	15	81	3776549	4412	25	133	4764674	6058	41	189	5264670	6774	43	212
Bio	1046962	2160	9	118	1165905	3221	14	187	1270664	3651	18	236	2960436	9078	23	1126
Imp	–	–	–	–	-16%	-59%	-66%	-42%	-32%	-92%	-120%	-74%	-93%	-186%	-141%	-335%
Ind2	12832210	12137	15	4327	13330628	21510	24	7137	14890534	25129	25	8825	20334033	35517	39	14160
Ind3	50311413	21164	32	4084	50773430	33150	57	7162	52771093	34865	57	8570	54659434	36392	57	9304
Avq.s	9884393	16848	13	8664	9571927	24450	22	11311	9706695	25080	22	12572	9991100	26236	23	13187
Avq.l	11121285	17198	13	14159	10871420	28405	20	16114	10977861	28817	20	18092	11323124	30037	21	19665
Imp	–	–	–	–	0%	-61%	-65%	-46%	-5%	-72%	-67%	-95%	-17%	-99%	-94%	-111%

Table 5.1: Comparison of Flat Level Routing and Top-down Routing

by the top-down routing. From the results, it can be seen that as the number of coarsening level increases, the quality of solution produced by the top-down routing is deteriorated with respect to the flat level routing. Clearly, the net refinement process destroys the high quality results obtained by the coarser level ILP routing.

Also, the computation time comparison shown in Table 5.1 indicates that the top-down router can not improve the efficiency of the flat level global routing. Table 5.2 shows the computation time consumed at each coarsening level for a 4-level top-down routing with net refinement. The total number of routed nets is shown in column "Total Routed-N". Column "Ref-T" lists the time used for net refinement at each coarsening level, while column "ILP-T" lists the time used for ILP routing. The results shows that the longer computation time of top-down routing is due to the maze searching based net refinement process.

Computation Time of Top-down Routing with Maze Refinement (Level 4)									
Bench	Total	Level 4		Level 3		Level 2		Flat Level	
Marks	Routed-N	Ref-T	ILP-T	Ref-T	ILP-T	Ref-T	ILP-T	Ref-T	ILP-T
Prim1	678	-	3	1	1	2	1	16	1
Struct	1296	-	3	1	1	4	1	28	1
Ind1	1412	-	2	1	1	3	2	19	1
Prim2	2043	-	10	6	1	25	3	164	3
Bio	3460	-	17	13	2	89	3	984	18
Ind2	10542	-	68	236	5	1281	17	12467	86
Ind3	18187	-	38	323	7	1325	15	7552	44
Avq.s	16649	-	25	221	7	1172	25	11630	107
Avq.l	18666	-	34	310	8	1722	28	17438	125

Table 5.2: Computation Time of Top-down Routing with Refinement (4 Levels)

5.2 Top-down Global Routing Without Net Refinement

The reason of performing the maze refinement process is because the global routing at each coarser level is based on different grid graphs. In order to have an accurate routing information, the routes generated at the coarser level must be refined according to the current level grid graph. One possible way to overcome the net refinement problem is to solve the global routing problem of each level based on the flat level grid graph. Initially, a coarsening process is performed to build the hierarchical representations. After the grid cells are coarsened to a certain level, the coarsening process terminates. During the uncoarsening process, at each coarser level (including the highest level), an ILP based router is invoked to route all the coarser level nets based on the flat level grid graph. Figure 5.7 shows the proposed top-down routing algorithm without coarser level nets refinement.

Top-down Routing

1. Set i=1;
2. **FOR** level i and $i <$ top_level
   Generate grid graph for level i+1 by merging bins of level i;
   Update grid cell coordinates for level i+1;
   Regenerate global nets for level i+1;
   $i = i + 1$;
   END FOR
3. Set i = top_level;
4. **FOR** level i and $i >$ flat_level
   Perform ILP routing for current level global nets;
   Update routing demand of each grid edge;
   Perform grid uncoasrening;
   $i = i$ -1 ;
   END FOR
5. Perform ILP routing for flat level unrouted nets.
6. Stop.

Figure 5.7: Top-down Hierarchical Global Routing

5.2.1 Experimental Results

Top-down Routing for ECM model

Flat Level Routing and Top-down Routing without Refinement (ECM model)																
Bench	Flat Level				Top-down (2 levels)				Top-down (3 levels)				Top-down (4 levels)			
Marks	Wire	Vias	M_{rd}	Time	Wire	Vias	M_{rd}	Time	Wire	Vias	M_{rd}	Time	Wire	Vias	M_{rd}	Time
Prim1	669328	692	8	15	684322	732	9	17	712464	779	10	16	762935	872	11	16
Struct	337900	633	7	7	346422	654	8	9	375869	830	9	9	375288	790	8	10
Imp	-	-	-	-	-2.5%	-4.5%	-13.5%	-21%	-8.5%	-22%	-27%	-18%	-12.5%	-25%	-26%	-51%
Ind1	1020282	1314	17	31	1084308	1490	19	33	1129066	1610	19	33	1168014	1695	20	33
Prim2	3253505	2737	15	81	3318922	2788	15	89	3497030	3059	16	91	3657190	3344	16	103
Bio	1046962	2160	9	118	1071117	2283	8	133	1135777	2569	10	152	1195730	2849	11	152
Imp	-	-	-	-	-3%	-7%	-0%	-10%	-9%	-18%	-10%	-12%	-11%	-27%	-16%	-21%
Ind2	12832210	12137	15	4327	12908353	12322	13	5098	13213170	13156	14	4653	13717463	14596	15	3577
Ind3	50311413	21164	32	4084	49151563	19785	32	2861	49895008	20305	35	3139	51819487	22204	38	3175
avq.s	9884393	16848	13	8664	10121431	17619	14	12705	10379917	18779	15	10269	10570610	19533	17	8206
avq.l	11121285	17198	13	14159	11230938	17858	13	12402	11538289	19237	14	13032	11612625	19803	15	9484
Imp	-	-	-	-	-0%	-1%	1%	-6%	-3%	-13%	-6%	1%	-5%	-14%	-16%	19%

Table 5.3: ECM Model: Flat Level Routing vs Top-down Routing (2,3,4 levels)

In Table 5.3, the results of top-down routing without net refinement are compared for different hierarchical levels (the highest level is 4). Column "M_{rd}" shows

the maximum routing density after final routing and row "Imp" shows the average percentage improvement produced by the top-down routing. Comparing tables 5.3, with table 5.1, it can be seen that the top-down routing without nets refinement produces much better results than that of top-down routing with nets refinement. In addition, the top-down routing can reduce the computation time of large size circuits by about 19% compared with the flat level routing. However, for all the benchmarks, the quality of solutions produced by the top-down routing is worse than that of the flat level routing.

Flat Level Routing and Top-down Routing without Refinement (ECM model)																
Bench	Flat Level				Top-down (4 levels)				Top-down (5 levels)				Top-down (6 levels)			
Marks	Wire	Vias	M_{rd}	Time	Wire	Vias	M_{rd}	Time	Wire	Vias	M_{rd}	Time	Wire	Vias	M_{rd}	Time
Ind2	12832210	12137	15	4327	13717463	14596	15	3577	14179394	15608	17	3277	14274916	15802	17	3186
Ind3	50311413	21164	32	4084	51819487	22204	38	3175	52678398	22996	39	3174	53002808	23416	40	3133
avq.s	9884393	16848	13	8664	10570610	19533	17	8206	10453827	18882	16	6060	10387073	18686	15	6287
avq.l	11121285	17198	13	14159	11612625	19803	15	9484	11647939	19930	16	7452	11749173	20529	16	7639
Imp	-	-	-	-	-5%	-14%	-16%	19%	-7%	-17%	-20%	31%	-7%	-18%	-19%	31%

Table 5.4: ECM Model: Flat Level Routing vs Top-down Routing (4,5,6 levels)

Table 5.4 shows the experimental results of 6-level top-down routing for large size circuits. For better comparison, the results of 4-level and 5-level routing are also listed in the table. From the table, we can see that the 6-level top-down routing can reduce the computation time of large size circuits largely by about 31% with the increasing total wirelength, total number of vias and maximum routing density. The above experimental results indicates that for ECM model, the top-down routing can improve the computation time of flat level routing at the expense of the quality of solution.

Flat Level and Top-down Routing without Refinement (WLM model)																
Bench	Flat Level				Top-down (2 levels)				Top-down (3 levels)				Top-down (4 levels)			
Marks	Wire	Vias	M_{rd}	Time	Wire	Vias	M_{rd}	Time	Wire	Vias	M_{rd}	Time	Wire	Vias	M_{rd}	Time
Ind2	12091689	9490	21	2170	12091688	9457	22	2863	12091688	9524	22	3006	12092082	9643	20	3264
Ind3	47205901	16457	33	2131	47132250	16104	50	2652	47131757	16228	56	3048	47130249	16174	53	3139
Avq.s	9096280	12146	21	4163	9096280	12024	21	4912	9096741	12061	23	5427	9096856	12077	21	5614
Avq.l	10411364	12994	23	4515	10411594	13020	23	5750	10411374	13110	23	6700	10411710	13179	23	6854
Imp	-	-	-	-	-	-	-	-	-	-	-	-	0%	1%	-9.5%	-46%

Table 5.5: WLM Model: Flat Level vs Top-down Routing (2,3,4 levels)

Flat Level and Top-down Routing without Refinement (WLM model)																
Bench	Flat Level				Top-down (4 levels)				Top-down (5 levels)				Top-down (6 levels)			
Marks	Wire	Vias	M_{rd}	Time	Wire	Vias	M_{rd}	Time	Wire	Vias	M_{rd}	Time	Wire	Vias	M_{rd}	Time
Ind2	12091689	9490	21	2170	12092082	9643	20	3264	12092102	9710	23	3192	12094050	9674	23	3110
Ind3	47205901	16457	33	2131	47130249	16174	53	3139	47128728	16042	53	3071	47131701	15980	55	2905
Avq.s	9096280	12146	21	4163	9096856	12077	21	5614	9096511	12239	21	5738	9095934	13028	24	6137
Avq.l	10411364	12994	23	4515	10411710	13020	23	5750	10411374	13186	23	7292	10411369	13240	22	7496
Imp	-	-	-	-	0%	1%	-9.5%	-46%	0%	1%	-13%	-47%	0%	-2%	-7.5%	-48%

Table 5.6: WLM Model: Flat Level vs Top-down Routing (4,5,6 levels)

Top-down Routing for Other models

Table 5.5, 5.6, 5.7, 5.8, 5.9 and 5.10 show a comparison between flat level routing and top-down routing without refinement for wirelength minimization model (WLM), via minimization model (VMM), and combined model (WVEM).

From the results comparison, it can be seen that for different models, the top-down routing produces almost the same results as the flat level routing in terms of the total wirelength and the total number of vias. However, the maximum routing

Flat Level and Top-down Routing without Refinement (VMM model)																
Bench	Flat Level				Top-down (2 levels)				Top-down (3 levels)				Top-down (4 levels)			
Marks	Wire	Vias	M_{rd}	Time	Wire	Vias	M_{rd}	Time	Wire	Vias	M_{rd}	Time	Wire	Vias	M_{rd}	Time
Ind2	12114767	8272	25	2167	12116430	8272	25	2653	12114472	8272	25	3010	12118731	8272	25	3260
Ind3	47395661	14023	33	2123	47275328	13841	54	2647	47271369	13841	55	3051	47292195	13841	55	3084
Avq.s	9122104	10076	20	4163	9123488	10076	20	4912	9128446	10076	20	5422	9135709	10076	21	5611
Avq.l	10446320	10831	18	4509	10446597	10831	18	5758	10447588	10831	18	6709	10454086	10831	18	6897
Imp	-	-	-	-	-	-	-	-	-	-	-	-	0%	0%	-20%	-45%

Table 5.7: VMM Model: Flat Level vs Top-down Routing (2,3,4 levels)

Flat Level and Top-down Routing without Refinement (VMM model)																
Bench	Flat Level				Top-down (4 levels)				Top-down (5 levels)				Top-down (6 levels)			
Marks	Wire	Vias	M_{rd}	Time	Wire	Vias	M_{rd}	Time	Wire	Vias	M_{rd}	Time	Wire	Vias	M_{rd}	Time
Ind2	12114767	8272	25	2167	12118731	8272	25	3260	12116892	8272	23	3197	12129296	8272	21	3109
Ind3	47395661	14023	33	2123	47292195	13841	55	3084	47332016	13841	54	3079	47358929	13841	54	2859
Avq.s	9122104	10076	20	4163	9135709	10076	21	5611	9148506	10076	21	5833	9155885	10076	21	6046
Avq.l	10446320	10831	18	4509	10454086	10831	18	6897	10466679	10831	18	7226	10474403	10831	19	7474
Imp	-	-	-	-	0%	0%	-20%	-45%	0%	0%	-17%	-48%	0%	0%	-16%	-47%

Table 5.8: VMM Model: Flat Level vs Top-down Routing (4,5,6, levels)

Flat Level and Top-down Routing without Refinement (WVEM model)																
Bench	Flat Level				Top-down (2 levels)				Top-down (3 levels)				Top-down (4 levels)			
Marks	Wire	Vias	M_{rd}	Time	Wire	Vias	M_{rd}	Time	Wire	Vias	M_{rd}	Time	Wire	Vias	M_{rd}	Time
Ind2	12100759	8299	13	2190	12105726	8311	14	2655	12111315	8335	14	3013	12111059	8341	15	3263
Ind3	47308679	14037	32	2127	47288911	13985	33	2662	47299464	13994	34	3051	47245991	13941	36	3068
Avq.s	9112536	10109	13	4183	9116571	10125	13	4928	9118069	10139	14	5432	9115648	10117	16	5600
Avq.l	10432967	10876	12	4547	10433879	10864	13	5788	10433990	10874	14	6720	10435008	10870	15	6880
Imp	-	-	-	-	-	-	-	-	-	-	-	-	0%	0%	-19%	-44%

Table 5.9: WVEM Model: Flat Level vs Top-down Routing (2,3,4 levels)

Flat Level and Top-down Routing without Refinement (WVEM model)																
Bench	Flat Level				Top-down (4 levels)				Top-down (5 levels)				Top-down (6 levels)			
Marks	Wire	Vias	M_{rd}	Time	Wire	Vias	M_{rd}	Time	Wire	Vias	M_{rd}	Time	Wire	Vias	M_{rd}	Time
Ind2	12100759	8299	13	2190	12111059	8341	15	3263	12105402	8311	16	3155	12112091	8317	16	3049
Ind3	47308679	14037	32	2127	47245991	13941	36	3068	47265591	13991	39	3071	47237705	13931	41	2865
Avq.s	9112536	10109	13	4183	9115648	10117	16	5600	9117147	10159	16	5816	9117147	10150	16	6145
Avq.l	10432967	10876	12	4547	10435008	10870	15	6880	10438068	10894	16	7190	10437025	10896	16	7447
Imp	-	-	-	-	0%	0%	-19%	-44%	0%	0%	-25%	-46%	0%	0%	-27%	-46%

Table 5.10: WVEM Model: Flat Level vs Top-down Routing (4,5,6 levels)

density and computation time are worse than that of the flat level routing.

Comparison of Different Models for Top-down Routing

Top-down Routing without Refinement for Different Models																
Bench	ECM (6 levels)				WLM (6 levels)				VMM (6 levels)				WVEM (6 levels)			
Marks	Wire	Vias	M_{rd}	Time	Wire	Vias	M_{rd}	Time	Wire	Vias	M_{rd}	Time	Wire	Vias	M_{rd}	Time
Ind2	14274916	15802	17	3186	12094050	9674	23	3110	12129296	8272	21	3109	12112091	8317	16	3049
Ind3	53002808	23416	40	3133	47131701	15980	55	2905	47358929	13841	54	2859	47237705	13931	41	2865
avq.s	10387073	18686	15	6287	9095934	13028	24	6137	9155885	10076	21	6046	9117147	10150	16	6145
avq.l	11749173	20529	16	7639	10411369	13240	22	7496	10474403	10831	19	7474	10437025	10896	16	7447
Imp	-	-	-	-	12%	34%	-42%	3%	12%	46%	-29%	4%	12%	45%	-3%	4%

Table 5.11: Comparison of Top-down Routing for Different Models

In Table 5.11, the results of 6-level top-down routing for WLM, VMM and WVEM models are compared with the ECM model. As shown in this table, the combined model (WVEM) produces better results in terms of total wirelength, number of bend, maximum routing density and computation time with respect to other models.

5.3 Bottom-up Global Routing Without Net Refinement

In [L02], the authors suggests that a shorter net has a higher priority than a longer net since a shorter net enjoys less freedom when searching for a path to route it. Therefore, in this section, we attempt to route local nets at each level of coarsening. The algorithm as shown in Figure 5.8 starts by coarsening the finest grid cells of level 1. At level 2, the ILP based global router finds routing paths for the local

nets (i.e. those nets that entirely located inside a grid bin). The routing demand of each grid edge is then updated according to the routing results of local nets and considered in the next level routing. Coarsening continues until the ideal coarse level is reached. At the coarsest level, the local nets are first routed followed by the routing of the rest unrouted global nets. As illustrated by Figure **??**, the bottom-up global routing attempts to solve the ILP formulation at level 1 and terminates at the coarsest level (i.e. level l). The time complexity of bottom-up routing is $O(l \times n)$, where l is the total number of hierarchical levels and n is the number of nets.

Bottom-up Routing

1. Set i=1;
2. **FOR** level i and $i <$ top_level
   Generate grid graph for level i+1 by merging bins of level i;
   Update grid cell coordinates for level i+1;
   Regenerate local nets for level i+1;
   Perform ILP routing for local nets of level i+1;
   Update routing demand of each grid edge;
   $i = i + 1$;
   END FOR
3. Perform ILP routing for top level unrouted nets.
4. Stop.

Figure 5.8: Bottom-up Hierarchical Global Routing

5.3.1 Experimental Results

Bottom-up Routing for ECM model

Table 5.12 shows the experimental results of 6-level bottom-up hierarchical routing for large size circuits. The revised ECM model (5.1) is used at each coarsening

Flat Level Routing and Bottom-up Routing (ECM model)																
Bench	Flat Level				Bottom-up (4 levels)				Bottom-up (5 levels)				Bottom-up (6 levels)			
Marks	Wire	Vias	M_{rd}	Time	Wire	Vias	M_{rd}	Time	Wire	Vias	M_{rd}	Time	Wire	Vias	M_{rd}	Time
Ind2	12832210	12137	15	4327	12271927	10458	14	2156	12172848	9958	15	2122	12171589	9767	15	2262
Ind3	50311413	21164	32	4084	47713352	17635	33	2133	47554903	16985	35	1843	47599964	16640	41	1775
avq.s	9884393	16848	13	8664	9134326	12626	16	3795	9172372	12481	15	3324	9210763	12409	14	3533
avq.l	11121285	17198	13	14159	10482986	13843	13	4097	10503129	13420	13	3715	10523934	13141	13	4055
Imp	-	-	-	-	5%	19%	-5%	56%	5%	20%	-5%	60%	5%	22%	-5%	58%

Table 5.12: ECM Model: Flat Level Routing vs Bottom-up Routing (4,5,6 levels)

level. Row "Imp" shows the average percentage improvement produced by the bottom-up routing. It is clear from this table that the 6-level bottom-up hierarchical routing can reduce the computation time of large size circuits largely by about 58% with a slight increase of the maximum routing density. Also, the total wirelength and total number of vias are improved by 5% and 22%, respectively. The results comparison indicates that the bottom-up hierarchical routing can solve the large size of routing problem efficiently and effectively. Figure 5.9 shows the effects of different hierarchical levels on solution quality using bottom-up routing for large size circuits. The quality of solution is measured by the total wirelength, total number of vias, maximum routing density and computation time. The results clearly show that, 5-level bottom-up routing gives the best results in terms of computation time, while 6-level routing gives best results in terms of total number of vias and maximum routing density. For the total wirelength, the bottom-up routing with levels greater than three produce good results.

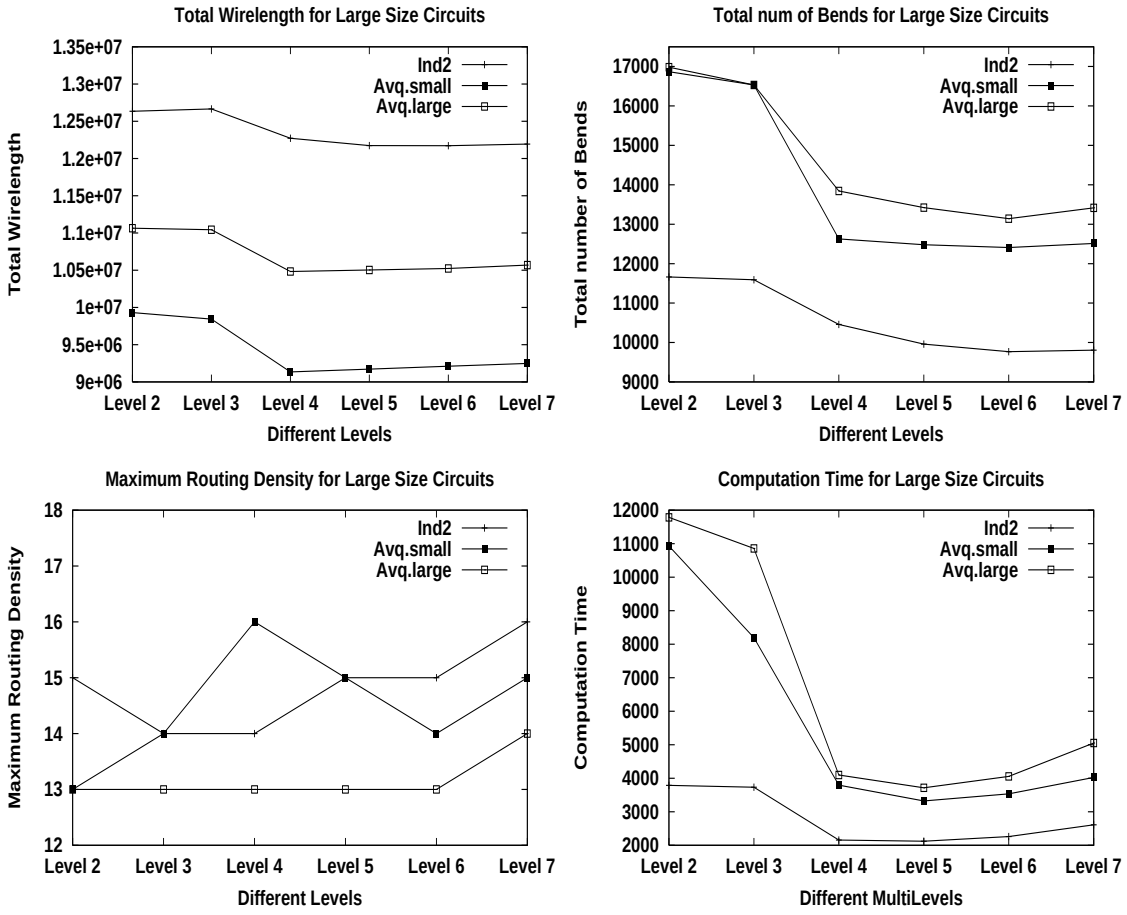


Figure 5.9: Bottom-up Routing of Different Levels for ECM Model

Flat Level and Bottom-up Routing without Refinement (WLM model)																
Bench	Flat Level				Multi-Levels (2 levels)				Multi-Levels (3 levels)				Multi-Levels (4 levels)			
Marks	Wire	Vias	M_{rd}	Time	Wire	vias	M_{rd}	Time	Wire	Vias	M_{rd}	Time	Wire	Vias	M_{rd}	Time
Ind2	12091689	9490	21	2170	12091688	9470	21	2149	12091492	9501	20	2153	12091679	9464	22	2140
Ind3	47205901	16457	33	2131	47131757	16202	40	2180	47132250	16224	40	2284	47134209	16166	40	2109
Avq.s	9096280	12146	21	4163	9096280	12105	21	4061	9096395	12101	22	3841	9096741	11975	23	3728
Avq.l	10411364	12994	23	4515	10411033	13067	25	4595	10411018	13088	25	4877	10412929	12826	24	4195
Imp	-	-	-	-	-	-	-	-	-	-	-	-	0%	1.5%	-12%	5%

Table 5.13: WLM Model: Flat Level vs Bottom-up Routing (2,3,4 levels)

Flat Level and Bottom-up Routing without Refinement (WLM model)																
Bench	Flat Level				Multi-Levels (4 levels)				Multi-Levels (5 levels)				Multi-Levels (6 levels)			
Marks	Wire	Vias	M_{rd}	Time	Wire	Vias	M_{rd}	Time	Wire	Vias	M_{rd}	Time	Wire	Vias	M_{rd}	Time
Ind2	12091689	9490	21	2170	12091679	9464	22	2140	12089514	9752	22	2163	12089514	9275	29	2281
Ind3	47205901	16457	33	2131	47134209	16166	40	2109	47142142	15843	40	1866	47145191	15668	40	1812
Avq.s	9096280	12146	21	4163	9096741	11975	23	3728	9096741	12289	21	3389	9097548	11679	24	3564
Avq.l	10411364	12994	23	4515	10412929	12826	24	4195	10413616	13409	20	3741	104113626	12729	19	4010
Imp	-	-	-	-	0%	1.5%	-12%	5%	0%	0%	-5%	13%	0%	2%	-16%	12%

Table 5.14: WLM Model: Flat Level vs Bottom-up Routing (4,5,6 levels)

Bottom-up Routing for Other models

Table 5.13, 5.14, 5.15, 5.16, 5.17 and 5.18 show the results comparison between flat level routing and bottom-up routing without refinement for wirelength minimization model (WLM) and via minimization model (VMM). and combined model (WVEM). From these tables, it is clear that bottom-up routing produces almost the same results as the flat level routing in terms of total wirelength and total number of vias. For WLM model, the 6-level bottom-up routing reduces the computation time by about 12% while the maximum routing density is also increased by about 16%. For both VMM model and WVEM model, the 6-level routing improves the computation time by about 9% with an increase of maximum routing density by about 14% and 11%, respectively.

In Table 5.19, the results of 6-level bottom-up routing for WLM, VMM and

Flat Level and Bottom-up Routing without Refinement (VMM model)																
Bench	Flat Level				Multi-Levels (2 levels)				Multi-Levels (3 levels)				Multi-Levels (4 levels)			
Marks	Wire	Vias	M_{rd}	Time	Wire	Vias	M_{rd}	Time	Wire	Vias	M_{rd}	Time	Wire	Vias	M_{rd}	Time
Ind2	12114767	8272	25	2167	12116754	8272	25	2142	12117629	8272	25	2131	12135682	8272	24	2139
Ind3	47395661	14023	33	2123	47261505	13841	40	2170	47290194	13841	40	2268	47304017	13841	40	2117
Avq.s	9122104	10076	20	4163	9122681	10076	21	4030	9159228	10076	21	3844	9155194	10076	20	3722
Avq.l	10446320	10831	18	4509	10452766	10831	18	4587	10478611	10831	18	4850	10491263	10831	23	4134
Imp	-	-	-	-	-	-	-	-	-	-	-	-	0%	0%	-13%	5%

Table 5.15: VMM Model: Flat Level vs Bottom-up Routing (2,3,4 levels)

Flat Level and Bottom-up Routing without Refinement (VMM model)																
Bench	Flat Level				Multi-Levels (4 levels)				Multi-Levels (5 levels)				Multi-Levels (6 levels)			
Marks	Wire	Vias	M_{rd}	Time	Wire	Vias	M_{rd}	Time	Wire	Vias	M_{rd}	Time	Wire	Vias	M_{rd}	Time
Ind2	12114767	8272	25	2167	12135682	8272	24	2139	12120414	8272	26	2116	12122844	8272	27	2278
Ind3	47395661	14023	33	2123	47304017	13841	40	2117	47360340	13841	40	1840	47378417	13841	40	1835
Avq.s	9122104	10076	20	4163	9155194	10076	22	3722	9164648	10076	22	3323	9168569	10076	20	3536
Avq.l	10446320	10831	18	4509	10491263	10831	20	4134	10480292	10831	24	3655	10492315	10831	21	4016
Imp	-	-	-	-	0%	0%	-13%	5%	0%	0%	-20%	14%	0%	0%	-14%	9%

Table 5.16: VMM Model: Flat Level vs Bottom-up Routing (4,5,6 levels)

Flat Level and Bottom-up Routing without Refinement (WVEM model)																
Bench	Flat Level				Multi-Levels (2 levels)				Multi-Levels (3 levels)				Multi-Levels (4 levels)			
Marks	Wire	Vias	M_{rd}	Time	Wire	Vias	M_{rd}	Time	Wire	Vias	M_{rd}	Time	Wire	Vias	M_{rd}	Time
Ind2	12100759	8299	13	2168	12103897	8301	14	2152	12103110	8301	14	2144	12109387	8306	14	2153
Ind3	47308679	14037	32	2116	47312780	14053	31	2198	47320347	14072	31	2298	47260137	13931	33	2179
Avq.s	9112536	10109	13	4189	9113689	10105	13	4055	9113112	10119	13	3896	9111613	10086	16	3775
Avq.l	10432967	10876	12	4539	10432520	10877	12	4622	10432169	10878	13	4950	10434652	10857	12	4190
Imp	-	-	-	-	-	-	-	-	-	-	-	-	0%	0%	-9%	4.5%

Table 5.17: WVEM Model: Flat Level vs Bottom-up Routing (2,3,4 levels)

Flat Level and Bottom-up Routing without Refinement (WVEM model)																
Bench	Flat Level				Multi-Levels (4 levels)				Multi-Levels (5 levels)				Multi-Levels (6 levels)			
Marks	Wire	Vias	M_{rd}	Time	Wire	Vias	M_{rd}	Time	Wire	Vias	M_{rd}	Time	Wire	Vias	M_{rd}	Time
Ind2	12100759	8299	13	2190	12109387	8306	14	2153	12109387	8306	14	2133	12099814	8291	15	2295
Ind3	47308679	14037	32	2127	47260137	13931	33	2179	47290996	13976	34	1875	47297111	13964	36	1798
Avq.s	9112536	10109	13	4183	9111613	10086	16	3775	9112305	10095	16	3369	9114265	10103	14	3552
Avq.l	10432967	10876	12	4547	10434652	10857	12	4190	10435455	10857	14	3710	10436468	10863	13	4040
Imp	-	-	-	-	0%	0%	-9%	4.5%	0%	0%	-10%	13%	0%	0%	-11%	9%

Table 5.18: WVEM Model: Flat Level vs Bottom-up Routing (4,5,6 levels)

Bottom-up Routing without Refinement for Different Models																
Bench	ECM (5 levels)				WLM (5 levels)				VMM (5 levels)				WVEM (5 levels)			
Marks	Wire	Vias	M_{rd}	Time	Wire	Vias	M_{rd}	Time	Wire	Vias	M_{rd}	Time	Wire	Vias	M_{rd}	Time
Ind2	12172848	9958	15	2122	12089514	9752	22	2163	12120414	8272	26	2116	12109387	8306	14	2133
Ind3	47554903	16985	35	1843	47142142	15843	40	1866	47360340	13841	40	1840	47290996	13976	34	1875
avq.s	9172372	12481	15	3324	9096741	12289	21	3389	9164648	10076	22	3323	9112305	10095	16	3369
avq.l	10503129	13420	13	3715	10413616	13409	20	3741	10480292	10831	24	3655	10435455	10857	14	3710
Imp	-	-	-	-	1%	3%	-39%	0%	0%	19%	-55%	1%	0%	19%	0%	0%

Table 5.19: Comparison of Bottom-up Routing for Different Models (ILP results)

WVEM models are compared with the ECM model. The results clearly indicate that for bottom-up routing, the WVEM model produces the best quality of solution in terms of total wirelength, total number of bend, maximum routing density and computation time.

Comparison of Bottom-up Routing & Top-down Routing

Flat Level Routing & Top-down Routing & Bottom-up Routing (ECM model)												
Bench	Flat Level				Top-down 5-level				Bottom-up 5-level			
Marks	Wire	Vias	M_{rd}	Time	Wire	Vias	M_{rd}	Time	Wire	Vias	M_{rd}	Time
Prim2	3253505	2737	15	81	3689653	3355	16	95	3223003	2581	16	81
Bio	1046962	2160	9	118	1214006	2951	10	154	1072457	2294	10	147
Imp	-	-	-	-	-14.5%	-14.5%	-9%	-19%	-0.5%	0%	-5%	-12%
Ind2	12832210	12137	15	4327	14179394	15608	17	3277	12172848	9959	15	2122
Ind3	50311413	21164	32	4084	52678398	22996	39	3174	47554903	16985	35	1843
avq.s	9884393	16848	13	8664	10453827	18882	16	6060	9172372	12481	15	3324
avq.l	11121285	17198	13	14159	11647939	19930	16	7452	10503129	13420	13	3715
Imp	-	-	-	-	-7%	-17%	-20%	31%	5%	20%	-5%	60%

Table 5.20: Comparison of Flat Level, Top-down and Bottom-up routing

In Table 5.20, the 5-level top-down routing is compared with the 5-level bottom-up routing. The results of flat level routing is also listed for the purpose of comparison. Row "Imp" shows the average percentage improvement achieved by the

top-down and bottom-up routing compared with the flat level routing. The results clearly indicate that for the large benchmarks, the bottom-up routing achieves better results than that of top-down routing with much less computation time. From the comparison, we also can see that the bottom-up routing not only improves the computation time but also the total wirelength and total number of vias. However,‘ the maximum routing density is only slightly increased by about 5%.

Table 5.21 and Table 5.22 list the total overflowed edges and maximum routing density at each coarsening level for the 5-level top-down routing and bottom-up routing. The column “CNets” shows the total number of nets routed in current level, while column “T_OF” and column “M_{rd} shows the accumulated value on total number of overflowed edges and maximum routing density after current level routing. The results of flat level routing is also listed in the last column “Flat Routing” for comparison purpose. In Table 5.22, the column “Level5 Local” shows the results after the routing of all local nets at level5, while column “Level5 Final” shows the results after the routing of all global nets at level5. By comparing Table 5.21 and Table 5.22, it is clear that the total overflowed edges produced by bottom-up routing is much less than that of the top-down routing, which indicates that the bottom-up routing gives a better solution than that of the top-down routing in terms of routability.

In the bottom-up routing, the shorter nets that have less freedom and therefore, less admissible routes are routed first. As the coarsening continues, the routing resource is consumed and the grid edges become congested. However, at the coarser level, most of the nets are longer nets, which have more freedom and thus, more admissible routes can be built for them to enhance the congestion. Figure 5.10

Flat Routing & Top-down Routing without Refinement (ECM model)																		
Bench	Top-down Routing															Flat Level		
Marks	Level5 Global			Level4 Global			Level3 Global			Level2 Global			Level1 Final			Routing		
	CNets	T_OF	M_{rd}	CNets	T_OF	M_{rd}	CNets	T_OF	M_{rd}	CNets	T_OF	M_{rd}	CNets	T_OF	M_{rd}	Nets	T_OF	M_{rd}
Prim2	334	40	13	468	350	13	398	791	14	425	1030	16	418	**1157**	**16**	2043	299	15
Bio	378	8	6	509	305	9	703	1022	9	830	1759	9	1040	**2390**	**10**	3460	195	9
Ind2	2759	2	12	1575	2932	17	1896	5769	17	2169	7620	17	2143	**8623**	**17**	10542	662	15
Ind3	4775	0	27	2944	31631	31	3893	3334	38	3541	4286	38	2884	**4814**	**39**	18037	750	32
avq.s	3798	5	11	2633	3917	12	3335	7767	14	3568	9977	15	3315	**11294**	**16**	16649	2342	13
avq.l	4696	24	10	2725	6196	13	3635	10190	14	3818	12390	15	3792	**13404**	**16**	18666	3053	13

Table 5.21: Flat Routing & Top-down Routing (5-level)

Flat Routing & Bottom-up Routing without Refinement (ECM model)																		
Bench	Bottom-up Routing															Flat Level		
Marks	Level2 Local			Level3 Local			Level4 Local			Level5 Local			Level5 Final			Routing		
	CNets	T_OF	M_{rd}	CNets	T_OF	M_{rd}	CNets	T_OF	M_{rd}	CNets	T_OF	M_{rd}	CNets	T_OF	M_{rd}	Nets	T_OF	M_{rd}
Prim2	418	0	5	425	0	8	398	0	10	468	59	14	334	**256**	**16**	2043	299	15
Bio	1040	0	4	830	0	5	703	23	7	509	130	10	378	**236**	**10**	3460	195	9
Ind2	2143	0	4	2169	0	8	1896	0	9	1575	0	10	2759	**690**	**15**	10542	662	15
Ind3	2884	0	10	3541	0	13	3893	0	20	2944	0	23	4775	**545**	**35**	18037	750	32
avq.s	3315	0	5	3568	0	7	3335	0	10	2633	15	11	3798	**1257**	**15**	16649	2342	13
avq.l	3792	0	4	3818	0	8	3635	0	8	2725	0	9	4696	**2257**	**13**	18666	3053	13

Table 5.22: Flat Routing & Bottom-up Routing (5-level)

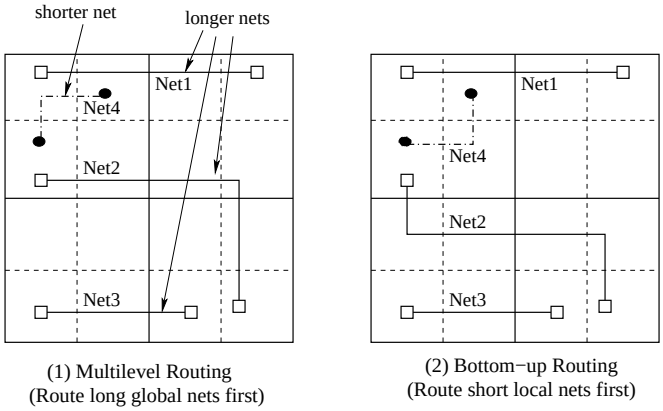


Figure 5.10: Bottom-up & Top-down Routing

illustrates this phenomena. There are three longer nets (Net1, Net2 and Net3) and one shorter net (Net4) in Figure 5.10. In the top-down routing, the original 4x4 grid graph is first transferred to the 2x2 grid graph. Then at the coarser level, the three longer global nets (Net1, Net2 and Net3) will be routed by the ILP router without considering the routing resource consumed by the finer level shorter nets. Therefore, at the subsequent finer level routing, it's difficult to find a route for the shorter net (Net4) such that the maximum routing density is one due to the limited searching freedom of shorter nets. However, in bottom-up routing, the shorter local net (Net4) is first routed followed by the routing of the longer global nets. Since the longer net (Net2) enjoys more freedom when searching for a path to route it, the maximum routing density produced by the bottom-up routing is better than that of the top-down routing. Obviously, the better net ordering scheme of the bottom-up routing makes it more efficient and effective than the top-down routing.

5.4 Summary

To minimize the computation time of the proposed flat ILP based global router, several hierarchical methods were proposed and investigated. For the top-down global routing with net refinement, the quality of solution for all the benchmarks are worser than that of the flat ILP based routing and the maze searching based net refinement process deteriorates the computation efficiency achieved by the top-down routing. To overcome the disadvantage brought by the net refinement process, the top-down global routing without net refinement method was proposed. The experimental results show that the 6-level top-down routing without net refinement

can reduce the computation time of large size circuits by about 31% for the ECM model at the expense of the quality of solution.

In addition to the top-down methods, a bottom-up global routing without net refinement method was also presented. By comparing the 6-level bottom-up global routing with flat ILP based routing for ECM model, the total wirelength and total number of vias are improved by 5% and 22% for large size circuits. The computation time was also improved by 58% with a slight increase in maximum routing density (on average by 5%). For the performance comparison, the bottom-up routing achieves better quality of solution than that of top-down routing with much less computation time.

In Chapter 6 another important objective of the global routing (power optimization) is introduced and a power-efficient multi-pin ILP based global routing methodology is proposed to optimize the routing and interconnect delay as well as mitigate the power consumed by the interconnect buffers.

Chapter 6

Power-efficient ILP based Global Routing

As VLSI technologies moves to the deep submicron regime, interconnect delay becomes a dominant factor in determining circuit performance. It is very common to consider the buffer insertion technique during the routing process to reduce the interconnect delay. In addition, to achieve more functionality, die size are constantly increasing which leads to an increase in the average global interconnect length. As a result, more buffers are required to achieve the timing closure. This increase in buffers will have a huge impact on the chip power consumption. In order to optimize the routing and interconnect delay as well as mitigate the power consumed by the interconnect buffers, a power-efficient multi-pin ILP based global routing technique is proposed in this Chapter. The problem is based on a graph representation of the routing possibilities, including buffer insertion and identifying the least power path between the interconnect source and set of sinks.

6.1 Preliminaries

In this section, some of the fundamental concepts related to the new formulation are discussed.

6.1.1 Interconnect Modeling

In this work, the Elmore delay model is chosen for its simplicity [Elmo48] for modeling the delay of the interconnects. The model states that the delay D_i from the source node i (as shown in Fig. 6.1) is calculated as follows:

$$D_i = \sum_{k=1}^{N} C_k R_{ik}, \tag{6.1}$$

where R_{ik} represents the resistance, shared among the paths from the source node s to nodes i and k, C_k is the capacitance of each wire segment between the nodes (Fig. 6.1), and N is the number of nodes. To estimate the interconnect wire ca-

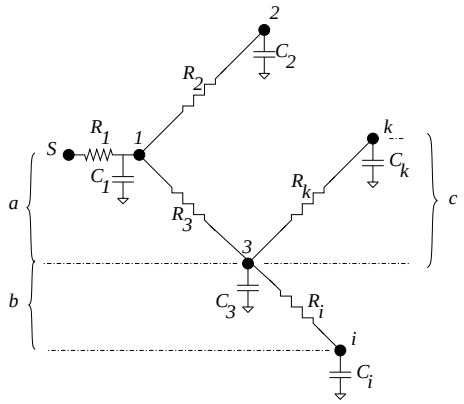


Figure 6.1: RC Tree [Raba03a].

pacitance, the model introduced in [Wong00] is used. It accounts for the fringing,

coupling, and plate capacitances for an interconnect according to the structure in Fig. 6.2. The unit length capacitance (c_{wire}) of the wire is[1],

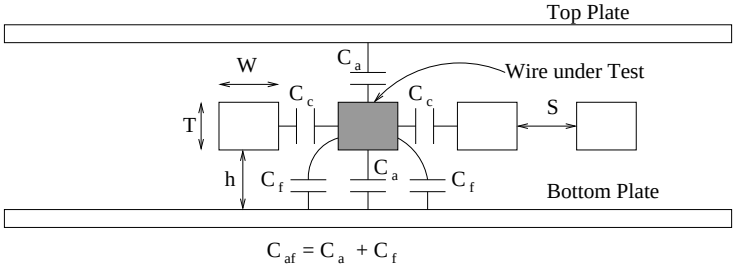


Figure 6.2: Wire structure for capacitance extraction.

$$c_{wire} = 2C_{af} + 2C_c, \tag{6.2}$$

where C_{af} is the total wire fringing and plate capacitance, and C_c is the coupling capacitance between the metal line and the adjacent metal lines. C_{af} is given by (6.3) [Wong00],

$$\begin{aligned} \frac{C_{af}}{\epsilon_{ILD}} &= \frac{W}{h} + 2.04\left(\frac{S}{S+0.5355h}\right)^{1.773} \\ &\quad .\left(\frac{T}{T+4.532h}\right)^{0.071} \end{aligned} \tag{6.3}$$

where ϵ_{ILD} is the interlayer-dielectric primitivity, W is the wire width, T is the wire thickness, S is the spacing between the wires, and h is the spacing between the wire and the ground plans. C_c is given by the following:

$$\frac{C_c}{\epsilon_{ILD}} = 1.411\frac{T}{S}exp\left(\frac{-4S}{S+8.014h}\right) \tag{6.4}$$

[1]The equations were used and evaluated by the Berkeley Predictive Technology Model (BPTM) which is provided by the Device Group at UC Berkeley [BPTM].

$$+2.3074(\frac{W}{W+0.3078S})^{0.25724}$$
$$.(\frac{h}{h+8.691S}).exp(\frac{-2S}{S+6h}).$$

The ITRS [Inte02] predictions are employed to estimate the interconnect wire resistance. Note that although an analytical model is chosen for the capacitance and resistance calculations, without the loss of generality, more exact extraction techniques can be used with no modification to the problem formulation.

In practice, the number of sinks, connected to the same driver without a buffer in between is small [Tang01]. In fact, connecting too many sinks to a single source will result in excessive delays that can not be recovered by buffer insertion. Accordingly, if the degree of this net configuration is limited, the wire lengths of a, b and c in Fig. 6.1 can be computed by generating a limited set of steiner trees between the driver and the set of sinks connected to it.

On the other hand, to estimate the total power, consumed by the driver and sink gates and the driven interconnect, the power consumption model employed in [Bane02] is used. The switching power (P_{active}) is hence calculated as follows:

$$\begin{aligned} P_{active} &= \alpha V_{DD}^2.f_{clk} \times \\ &\quad (W_{driver}.C_o + c_{wire}.l + \\ &\quad W_1.C_{L_{sink\ 1}} + \ldots + W_p.C_{L_{sink\ p}}), \end{aligned} \tag{6.5}$$

(6.6)

where V_{DD} is the supply voltage, f_{clk} is the clock frequency, W_{driver} and $W_{1\ \ldots\ p}$ are the widths of the driver and all the sink gates connected to it, respectively,

α is the switching factor which is assumed to be 0.15 [Chan95], c_{wire} is the wire capacitance per unit length, C_o is the output capacitance of the driving gate and C_{L_i} is the loading capacitance of sink gate i. In addition, l is the total Steiner tree wire length.

The following section presents the performance and power driven ILP-based global routing technique followed by the experimental results.

6.2 Power-Efficient Multi-pin ILP Based Global Routing

In order to optimize the routing and interconnect delay, as well as to reduce the power consumed by the interconnect buffers, a **P**ower-efficient multi-pin **I**LP based global **R**outing **T**echnique (PIRT) is proposed in this section. This performance and power aware routing algorithm is based on the Integer Linear Programming (ILP)-based global routing model in [Yang07].

The goal of the proposed PIRT is to find an efficiently powered buffer path for each net without violating the delay constraint. The routing area is modeled by using a routing grid that is similar to the one in Fig. 2.7(b). In this grid, $G = (V, E)$, each vertex u represents a buffer possible location, and each edge (u, v) represents a unit length wire segment. In this case, the problem can be formally described as follows:

PIRT: *Given a predefined routing grid* $(G = (V, E))$*, a buffer library (B), a buffer function* $(f(u) = 1)$ *if a buffer is allowed at vertex u, and* $(f(u) = -1)$ *otherwise, and a set of nets* $(net1, net2, ..., neti)$*, find the minimum power path for*

each net, subject to a delay constraint $Max\_Delay = max_{k \in N}\{d_k\}$*, where* d_k *is the delay of net K.*

6.2.1 PIRT Phases

The PIRT can be illustrated by dividing it into two distinct phases, as shown in Fig. 6.3. First, an initialization phase (phase I) where the initial minimum Steiner trees are constructed for each net. To reduce the global routing congestion, additional detoured trees are also built for each net. In order to create enough routing alternatives considering the delay requirements several buffered tree (i.e., trees with buffers inserted) are built for each net, if possible. Next, a power optimization phase (phase II) is invoked. This phase attempts to find a low power route (i.e., tree) for each net so that the total power of all the nets is minimized while satisfying the delay requirement of the chip.

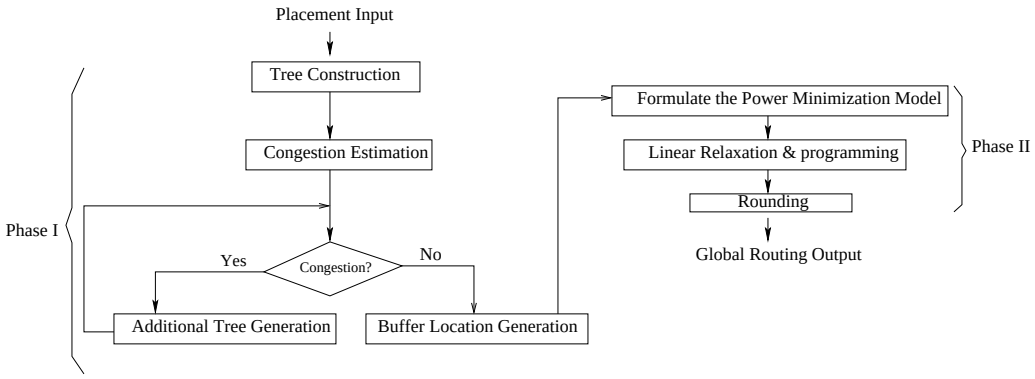


Figure 6.3: PIRT flow chart.

6.2.2 Phase I (Initialization)

In order to perform the interconnect power optimization in phase II, several routing alternatives are needed for each net. Creating routing alternatives allows the ILP model in phase II to pick a globally optimum solution that covers all the nets on the chip simultaneously. The routing alternatives are either unbuffered or buffered routes that connect the interconnect source to its sinks.

Unbuffered Tree Construction

The first step in this phase is to produce a set of admissible routes for each net. In a practical circuit, the terminals in a net are connected by horizontal and vertical wires. Therefore, only the rectilinear spanning trees or rectilinear Steiner trees are considered in the tree construction process. These trees become the unknown variables of the ILP problem. Obviously, the number of trees for each net should not be too large, since the time complexity of the ILP problem is a function of the number of trees. However, a number of trees should be built for each net to guarantee the feasibility of the problem. To remedy this problem, an additional tree generation step is proposed in [Yang07] to reduce the number of trees created for each net, while ensuring that the constructed trees will likely result in a promising (feasible and optimal) solution. Initially, the potentially congested areas in the routing graph are predicted by a heuristic technique [Yang07] in the congestion estimation stage. This a priori congestion information is then used in the additional tree generation stage to eliminate the congested areas by adding trees to the nets passing through these areas iteratively. The congestion of the circuit is re-estimated

after each additional tree generation step.

Buffered Tree Construction

Since unbuffered nets are not guaranteed to achieve the timing requirements of the chip, a set of buffered routes has to be added to the unbuffered trees generated earlier to extend the alternatives presented to the optimization step at phase II.

However, due to the preplaced modules there is usually a limited number of available buffer locations. In this formulation, each location is expected to allow for a limited number of buffers to be inserted.

Accordingly, in our current implementation, for each tree of a two-terminal net, at most two buffered trees are produced: one with one buffer inserted and the other with two buffers inserted, as denoted in Fig. 6.4. The buffer insertion process of three-terminal nets begins by identifying the Steiner point or middle point of the net in Fig. 6.5 The buffer insertion process of three-terminal nets begins by identifying the Steiner point or middle point of the net in Fig. 6.5.Following this, each net is divided into two or three segments, based on the presence of either a Steiner point or a middle point. If one segment is much longer than the other segments, the buffer insertion is considered for only the longest segment. Otherwise, all the segments are sorted in descending order, based on their length, and one buffer is inserted in each segment. For the generated buffered two and three terminal nets, only the trees, where the addition of a buffer results in the reduction of the delay are added to routing alternatives for Phase II.

Limiting the alternatives to two buffered trees per unbuffered tree assists in limiting the runtime of the algorithm. However, this limit can be removed at the

expense of longer runtime. Also, limiting the number of buffers to at-most two is a runtime trade-off that can be tuned. In addition, due to the limited availability of buffer locations, all the nets are first sorted in descending order according to their half-perimeter wirelength such that the long nets have a higher priority for buffered routes generation. However, the priority of short critical or highly loaded nets can be easily elevated to allow for buffered routes to be created for these nets. It is important to note that the buffer route generation priority does not affect the non-ordering nature of the formulation specially for the congestion consideration in phase II.

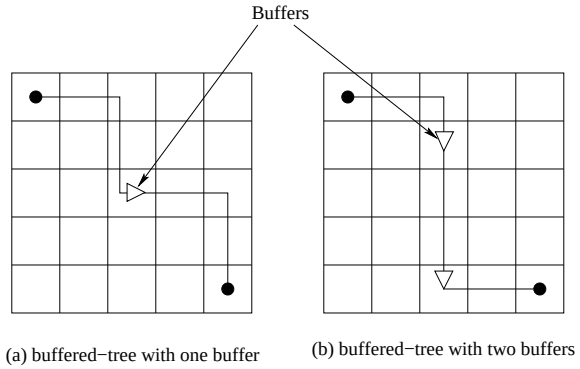


Figure 6.4: Buffer insertion for two-terminal nets (a) buffered-tree with one buffer. (b) buffered-tree with two buffers.

Complexity Analysis of Buffered Tree Generation

Fig. 6.6 illustrates the pseudocode of the buffer insertion algorithm. The time complexity of step 1 (i.e., sorting all the nets) is $O(N\log N)$, where N is the total number of routed nets. For step 2, the time complexity of "find buffer location and insert buffer" is $O(M)$, where M is the number of grid cells traversed by a tree. The complexity of "calculate the power and delay" is $O(P)$, where P is the number

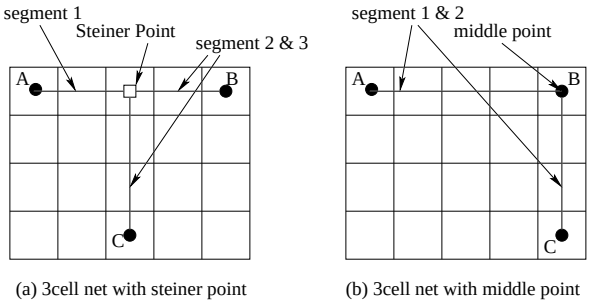


Figure 6.5: Buffer insertion for three terminal nets (a) 3cell net with Steiner point. (b) 3cell net with middle point.

Buffer Insertion Algorithm

1. Sort all the nets based on the half-perimeter wirelength.
2. Insert buffer for each sorted net.
   For each net i
      For each tree j of net i
         find buffer loaction and insert buffer;
         calculate the power and delay;
         if(inserted buffer reduces delay)
            connect the buffered tree to the tree list of net i ;
      End For
   End For
3. Stop.

Figure 6.6: Buffer Insertion Algorithm

of grid cells traversed by a segment of the tree. Since $P \ll M$. the complexity of the second step (buffer insertion process) is $O(NTM)$, where T is the number of trees built for each net. Since T is a constant the complexity is $O(NM)$. Since the maximum value of $M \ll$ than N, the complexity of the whole algorithm is $O(N\log N)$.

6.2.3 Phase II (Power Minimization)

To achieve PIRT's objective to minimize the power consumption of the interconnect while satisfying the chips delay constraints a constant Max_Delay is needed. Max_Delay is the maximum acceptable delay for all the nets which is derived from the clock frequency set by the product specifications. Since, any net having shorter delay than Max_Delay is unnecessarily fast, PIRT strives to trade this delay slack with the power consumption. Accordingly, PIRT power minimization under Max_Delay constraint is presented as follows:

$$\text{Minimize } \sum_{j=1}^{t} w_{pj} x_j, \tag{6.7}$$

subject to

$$\sum_{x_j \in N_k} x_j = 1, \quad k \in \{1, ..., n\}, \tag{6.8}$$

$$\sum_{x_j \in N_k} D_{x_j} x_j \leq Max\_Delay, \quad k \in \{1, ..., n\}, \tag{6.9}$$

$$\sum_{j=1}^{t} a_{ij} x_j - Cap_i \leq Z_i, \quad i \in \{1, ..., p\}, \tag{6.10}$$

$$x_j \in \{0, 1\} \quad j \in \{1, ..., t\} \quad Z_i \in \{0, C\},$$

In this model, the global routing problem is formulated as a 0/1 ILP problem by associating a variable x_j with each tree which connects a net. The variable x_j equals "1" if that particular tree is selected and "0" otherwise. The first constraint ensures that only a single tree among all the possible trees generated for net k is selected. w_{pj} is the weight associated with the power of tree j and is calculated as:

$$w_{pj} = \frac{power\ of\ tree\ j}{max\ power\ of\ trees\ constructed\ for\ net\ k} \tag{6.11}$$

The power of tree j[2] is modeled and calculated by equation (6.5).

For the second constraint, D_{xj} is the delay of tree j calculated using equation (6.1). This constraint ensures that for all the selected nets, the delay of each net does not exceed the Max_Delay.

For the third constraint, all the possible tree combinations created for each net in the two routing layers are represented by a (0,1) matrix $[A_{ij}]$, with the i^{th} row corresponding to the i^{th} edge in the grid graph and each column corresponding to the possible tree combinations for each net. The element a_{ij} is expressed as:

$$a_{ij} = \begin{cases} 1 & \text{if tree j passes through edge i;} \\ 0 & \text{otherwise.} \end{cases}$$

Since the routing resources are finite, each grid edge has a capacity. The capacity of each edge is represented by constraint (6.10), where p is the number of edges on the grid graph, t is the total number of trees produced for all the nets, and Cap_i

[2]Low activity nets can be pruned quickly from the search space to eliminate unnecessary runtime overhead.

is the edge capacity of the i^{th}edge. In order to achieve a global routing solution a slight allowance on the overflow is set by Z_i which is a variable associated with the routing overflow of each edge. C is the upper bound on Z_i. It is a positive value obtained from experimental results. The value of C represents the minimum number of extra tracks needed by the detailed router to achieve a fully routed chip.

Solving the minimization problem set by equation (6.7) ensures that the final net selection simultaneously achieves the delay constraints and minimize the power consumption.

6.3 Experimental Results

Based on the work in [Yous05], the chosen 130nm technology requires a buffer library that spans the range between 5 and 15 times the minimum sized buffer of Wp/Wn = 260/130nm to efficiently buffer the nets. Accordingly, the buffer library for this thesis represents three types of buffers; weak, medium and strong. They are chosen to correspond to three different buffer sizes: $5, 10, 15\times$, the minimum buffer size of the $130nm$ technology. This allows the verification of the technique for different buffer driving capabilities. Table 6.1 lists the parameters of the $130nm$ technology used for the power and delay calculation.

To fully quantify the performance of the proposed technique, the power savings, routing quality, and runtime need to be discussed. Consequently, the following sections introduce the results for these parameters when the PIRT is applied to the multi-pin netlists of different benchmarks.

Parameters for 130nm Technology	
Oxide thickness $T_{ox}(nm)$	2.3
Gate relative dielectric constant ϵ_r	3.9
ILD relative dielectric constant ϵ_r	3.0
Transistor length $L_{min}(nm)$	130
Supply voltage $V_{DD}(V)$	1.2
Clock frequency $f_{clk}(GH_z)$	1.6
Saturation current $I_{DD}(\mu A/\mu m)$	900
Subthreshold leakage $I_{off}(\mu A/\mu m)$	0.01
Parasitic capacitance percent of ideal gate capacitance	19%
Wire width $W_{wire}(nm)$	670
Wire width / thickness A/R	2.0
Wire height $h_{wire}(nm)$	670
Wire spacing $s_{wire}(nm)$	670
Wire resistance $r_{wire}(k\Omega/m)$	24.5
Wire capacitance $c_{wire}(pF/m)$	211.4

Table 6.1: Technology & Equivalent Circuit Model Parameters for Global Interconnets

6.3.1 Power Savings

To calculate the power savings achieved by applying PIRT the value of Max_Delay needs to be identified. In addition, a baseline that does not perform power optimization has to be established for comparison purposes. In the absence of clock frequency constraints in the IBM and ISPD benchmarks the Max_Delay value has to be identified using the available routing information. Since Max_Delay is the longest net delay when the chip is properly routed, i.e. all nets are optimized for their minimum delay, a delay minimization model that uses the data from PIRT's phase I and solves for the minimum delay instead of minimum power will properly route the chip and the longest net delay can be easily extracted. In addition, the power consumed by the final routing solution of the delay minimization model is the perfect baseline to identify the power savings achieved by PIRT in comparison to this delay minimization model.

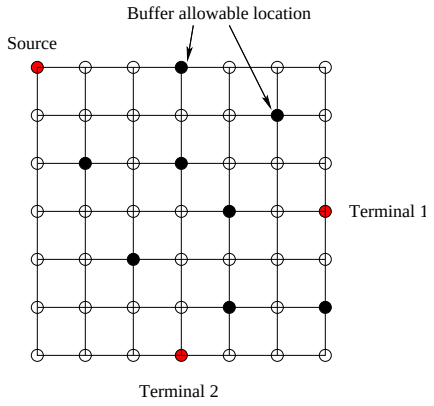


Figure 6.7: Buffer location generation.

6.3.1.1 Delay Minimization

In order to define the Max_Delay and the power consumption of delay optimal routed chip the following delay minimization model is solved.

$$\text{Minimize } \sum_{j=1}^{t} w_{dj} x_j, \tag{6.12}$$

subject to

$$\begin{aligned}
&\sum_{x_j \in N_k} x_j = 1, \quad k \in \{1, ..., n\}, \\
&\sum_{j=1}^{t} a_{ij} x_j - Cap_i \leq Z_i, \quad i \in \{1, ..., p\}, \\
&x_j \in \{0, 1\} \quad j \in \{1, ..., t\}, \\
&Z_i \in \{0, C\},
\end{aligned}$$

where x_j represents tree j built for net k, Cap_i is the routing supply of each edge, and Z_i is a variable associated with the routing overflow of each edge. w_{dj} represents the weight, associated with the delay of tree j and is calculated by the following:

$$w_{dj} = \frac{delay\ of\ tree\ j}{max\ delay\ of\ trees\ constructed\ for\ net\ k} \tag{6.13}$$

The delay of tree j is modeled and calculated by (6.1). In order solve the delay minimization model a number of allowable buffer locations (about 10% of the total vertices) are generated, as shown in Fig. 6.7. Following the buffer location generation in phase I.

After the delay minimization problem is solved, one tree is selected for each net k such that the total delay of all the nets is minimized under the routing overflow constraint. The delay of net k equals the delay of the selected tree. The maximum delay of a circuit is defined as $Max\_Delay = max_{k \in N}\{d_k\}$, where d_k is the delay of net K after the delay minimization.In addition, the power consumed by the final solution represents the baseline power consumption for the calculation of the power savings.

6.3.1.2 Power Savings Comparison

Fig. 6.8 depicts the power savings achieved when the PIRT is applied to the various benchmarks in Table 2.2 in comparison to the power consumption baseline established in 6.3.1.1. In addition to the average power savings for the IBM and ISPD benchmarks, respectively. This figure also compares the power savings when considering only the two and three terminal nets optimized in phase II to the total power savings when all the nets are included.

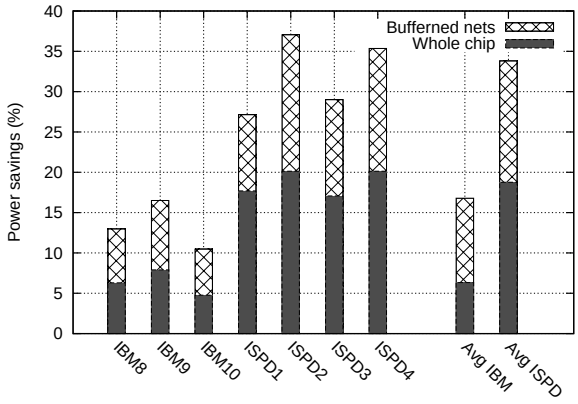


Figure 6.8: Power savings by PIRT and the average for IBM and ISPD benchmarks (strong buffer size.)

For the two and three terminal nets, the average savings for the IBM benchmarks and ISPD2007 benchmarks are 16% and 32%, respectively. These values drop to 6.5% and 19%, when the remaining unoptimized nets are factored in. A comparison of the results for the IBM and ISPD benchmarks indicates a significant increase in savings for the ISPD benchmarks. This can be attributed to the fact that the larger ISPD benchmarks have a higher number of long buffered nets which are the primary target of the PIRT. Since the trend for future chips is an increase in size,[3] this enhanced performance of the PIRT for the ISPD benchmarks emphasizes the importance of the PIRT as a power management technique.

Since the PIRT specifies the delay constraint on the chip to be the delay of the worst net, it is interesting to see how the PIRT affects the delay of the second worst net on the chip. Fig. 6.9 shows the percentage difference between the delay of the worst and the second worst nets. It is evident that the effect of the PIRT is to reduce the slack between these two nets and translate it into power savings. Consequently, the average reduction of the slack from 29% to 7% explains the savings in Fig. 6.8.

Finally, a comparison of the PIRT average power savings with the different buffer sizes in Fig. 6.10 shows that PIRT has more potential for larger buffer libraries.This is explained by noting that the removal of a large power hungry-buffer from the strong library results in higher savings, compared to the weak buffers, less power hungry, from the same library. Although it is tempting to try to route the nets by using the weakest buffer from the beginning, a significant hit to the signal integrity of the routed nets occurs. Accordingly, as the use of a mixed buffer library ensures that the nets are optimally performing over a wide range of chip sizes.

[3]Although the area might not change, the number of nets and buffers are exponentially growing.

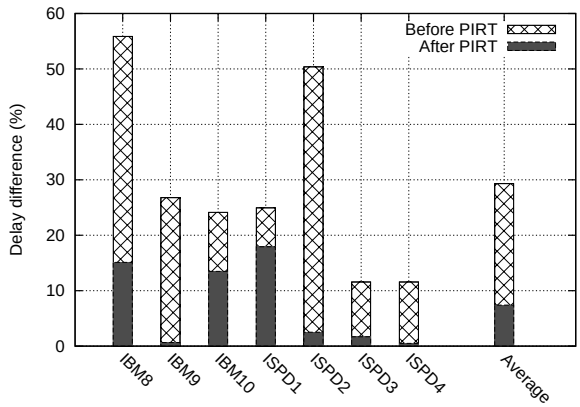


Figure 6.9: Delay difference for the second longest net before and after PIRT.

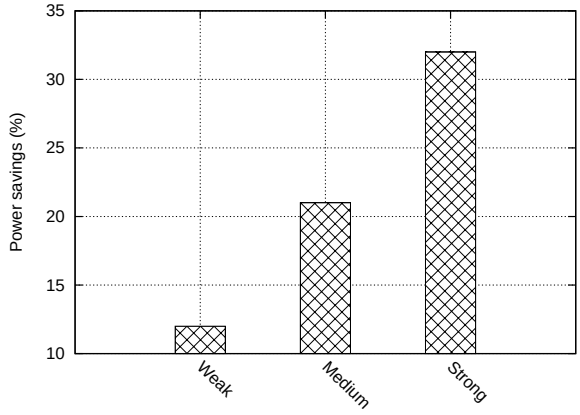


Figure 6.10: Average power reduction over different buffer sizes (5,10 and 15 times the minimum sized buffer).

6.3.2 Routing Quality

Table 6.2 provides the total wirelength, number of vias, and overflow of the delay minimization model (PIRT Phases I-III), and power minimization model (PIRT Phases I-IV) for the buffer size 15 x 130nm. Row "Imp" denotes the average percentage enhancement of each parameter. As shown, the power minimization model reduces the total overflow significantly without increasing the total wirelength and total number of vias for both the IBM and ISPD benchmarks.

Routing Results for Multi-pin Nets						
Circuit	Delay Minimization (15x130nm)			Power Minimization (15x130nm)		
	Wirelength	Vias	OverFlow	Wirelength	Vias	OverFlow
ibm08	1147507	9761	17	1147495	9592	6
ibm09	1043231	10264	634	1042778	10119	436
ibm10	1988162	13531	1180	1985537	13246	946
Avg	1393088	11051	610	1391999	10969	464
Imp	-	-	-	0%	1%	24%
ispd01	1845740	61029	5	1845718	60962	0
ispd02	1698413	87202	53	1698048	86560	15
ispd03	2887420	106369	13	2887350	105656	3
ispd04	2950644	139298	4	2950627	138948	0
Avg	2345554	98470	19	2345435	98039	4
Imp	-	-	-	0%	0%	76%

Table 6.2: Comparison of Delay Minimization and Power Minimization Models

By extending the results for the different buffer sizes in the library, PIRT results in an average reduction of 24%, 24%, and 28% for the weak, medium and strong sizes, respectively. The reason for this is that "power minimization" tends to minimize the total power by selecting trees with shorter wirelength, since the power is primarily proportional to the total wirelength of a tree, whether a buffer is inserted or not. Consequently, the congestion is globally reduced and hence, the power minimization model can improve the total overflow globally and simultaneously.

6.3.3 Computation Time

Table 6.3 displays a comparison of the computation time of the different phases of the PIRT. Columns "Tr-Time", "B-Time", "S-Time", and "Tot-Time" reveal the computation times of the tree construction phase, buffer insertion phase, power or delay model solving time, and the total time, respectively. It is clear that most of the computation time is consumed by the tree construction phase which is common to most other routing algorithms. The buffer insertion and power minimization phases consume only a very small percentage of the total computation time. Due to this small overhead, the PIRT manages to achieve its goal of power reduction without affecting the total runtime. This enables many existing routing techniques to benefit from the inclusion of the PIRT for power minimization.

Computation Time of the PIRT (Routing Multi-pin Nets)				
Circuit	PIRT Method (buffer size:15 x 130nm)			
	Tr-Time(s)	B-Time(s)	S-Time(s)	Tot-Time(s)
ibm08	95	31	1	127
ibm09	167	46	9	222
ibm10	182	57	15	254
ispd01	5412	1850	28	7290
ispd02	12713	3290	50	16053
ispd03	23482	8197	880	32559
ispd04	24071	11007	980	36058

Table 6.3: CPU Time Comparison of PIRT Method for Buffer Size 15 x 130nm.

6.4 Summary

Timing optimization and low power are important goals in global routing, especially in deep submicron designs. Previous efforts that focused on power optimization for

global routing are hindered by excessively long run times or the routing of a subset of the nets. Accordingly, in this chapter, a power efficient multi-pin global routing technique (PIRT) is proposed. This integer linear programming based techniques strives to find a power efficient global routing solution. The results indicate that an average power savings as high as 32% for the 130-nm technology can be achieved with no impact on the maximum chip frequency.

Chapter 7

Conclusions

One of the main objectives of global routing is to route all the nets coarsely in an area that is conceptually divided into small regions called global cells without violating resource constraints. With the continuous and rapid increase in complexity of VLSI designs and fabrication technologies, the global routing problem is becoming more difficult and important problem to solve. In addition to the primary objectives (i.e. wire area and routability), other design factors (i.e. vias, interconnect delay, power and coupling effects) also need to be considered in the global routing stage. On the other hand, the traditional flat global routing methods that can solve small size problems efficiently may take very long time to solve today's very large designs. Therefore, The overall approaches presented in this research tend to effectively model and integrate several objectives of the global routing problem and solve the problem efficiently.

7.1 Flat Multi-objective Global Routing

In this research, Integer Programming (ILP) based global routing models are fully investigated and explored. As can be seen from the experimental results, the wire-length minimization model (WLM), maximum edge capacity minimization model (ECM) and the via minimization model (VMM) can improve the corresponding objectives effectively.

To solve the global routing problem with several competing objectives, WVEM model and WVZM model incorporate these design factors (i.e. wirelength, the total number of bends and the congestion) into a single objective function as penalty terms. Each design factor has an associated weighting parameter and the selection of the value of weighting parameter determines the importance of the corresponding design factor. One advantage of such an approach is that it tends to produce somewhat more predictable solutions that optimize all different objectives. The results illustrate that the proposed WVZM model provides comparable speed to the sequential router [Kast] and better wirelength and total number of bends. To further improve the congestion a hybrid framework is proposed. Experimental results obtained indicate that the combined router outperforms the pure sequential router [Kast]. In addition, the combined router produces 48% less total overflow than Fengshui 5.1 [Hads03] with similar wire-length and 19% less CPU time.

7.2 Hierarchical ILP based Global Routing

To shorten the computation time of proposed flat ILP based global router, in this research, several hierarchical methods are combined with the flat ILP based global

routing approach. For the top-down global routing with net refinement, the quality of solution for all the benchmarks are worse than that of the flat ILP based routing and the maze searching based net refinement process deteriorates the computation efficiency achieved by the top-down routing. To overcome the disadvantage brought by the net refinement process, the top-down global routing without net refinement method is proposed. The experimental results show that the 6-level top-down routing without net refinement can reduce the computation time of large size circuits by about 31% for the ECM model at the expense of the quality of solution.

In addition to the top-down methods, a bottom-up global routing without net refinement method was also investigated in this thesis. By comparing the 6-level bottom-up global routing with flat ILP based routing for ECM model, the total wirelength and total number of bends are improved by 5% and 22% for large size circuits. The computation time is also improved by 58% with a slight increase on maximum routing density (on average by 5%). For the performance comparison, the bottom-up routing achieves better quality of solution than that of top-down routing with much less computation time.

7.3 Power Aware ILP based Global Routing

In their effort to achieve timing closure, chip designers have focused on achieving the lowest possible interconnect delay through buffer insertion and routing techniques. This approach, though, taxes the power budget of modern ICs, especially those intended for wireless applications. Also, in order to achieve more functionality, die sizes are constantly increasing. This trend is leading to an increase in the

average global interconnect length which, in turn, requires more buffers to achieve timing closure. Unconstrained buffering is bound to adversely affect the overall chip performance, if the power consumption is added as a major performance metric. In fact, the number of global interconnect buffers is expected to reach hundreds of thousands to achieve an appropriate timing closure. To mitigate the impact of the power consumed by the interconnect buffers, a power-efficient multi-pin routing technique is proposed in this thesis. The problem is based on a graph representation of the routing possibilities, including buffer insertion and identifying the least power path between the interconnect source and set of sinks. The technique is tested by applying it to the ISPD and IBM benchmarks to verify the accuracy, complexity, and solution quality. The results indicate that an average power savings as high as 32% for the 130-nm technology is achieved with no impact on the maximum chip frequency.

7.4 Future Work

To improve the computation time of the ILP models presented in Chapter 4, preprocessing techniques can be applied to further reduce the sizes of the ILP problems. Another future work involves applying the power-efficient ILP based routing technique to the hierarchical global routing method to further improve the computation time, especially for the tree construction phase.

Since the congestion optimization performance is constrained in the routing stage due to the fixed position of cells, it is of value to consider congestion in the placement stage where the effort on routability improvement would be more

effective. Several research approaches have been investigated for congestion driven placement [Meix90, Chen94, Para98, Hou01, Bren02, Wang99, Wang00b, Wang00c, Yang01a, Wang00a]. . However, for today's large, tight circuit designs, merely relocating cells during the placement stage is not adequate. To achieve a 100% routable design, both cell relocation and wire detouring are important. Therefore, a congestion reduction method that integrates placement with global routing becomes one of the interesting directions for future work.

In order to validate the proposed approaches using different real world circuits, it is necessary to integrate the proposed global routing techniques in industrial tools (e.g. Cadence) in the future. This involves (i) implementing a LEF/DEF parser to translate the circuit information. (ii) interfacing the proposed techniques with industrial EDA tools such that it can read placement solutions from EDA tools and submit the global routing results back to the tools for detailed routing. Apart from validation, the integration also makes the performance comparison between the proposed global routing techniques and the commercial CAD routing tools viable.

Appendix A

Glossary

CAD	: Computer Aided Design
CMOS	: Complementary Metal Oxide Semiconductor
Crosstalk	: a parasitic coupling between neighboring wires due to the mutual capacitances and inductances
DA	: Design Automation
DSM	: Deep sub-micron
EDA	: Electronic Design Automation
ECM	: Edge Capacity Minimization model
FPGA	: Field Programmable Gate Array
Fixed-die	: A layout style that chip area, core area, rows and available cell sites in cell rows are fixed before placement and routing
HPWL	: Half Perimeter Wire Length
IC	: Integrated Circuit
ILP	: Integer Linear Programming
IP	: Intellectual Property

ILPR	: Integer Linear Programming with Relaxation
MCM	: Multi Chip Module
MCNC	: Microelectronics Center of North Carolina
NP-hard	: Non Deterministic Polynomial Hard
OTC	: Over the Cell Routing
RTL	: Register Transfer Logic
RST	: Rectilinear Spanning Tree
RSMT	: Rectilinear Steiner Minimum Tree
RNWo	: Routes as many Nets Without generating Overflow
SoC	: System on Chip (A chip that hold the entire computer system)
VHDL	: Very High Speed Integrated Circuit Hardware Description Language
VLSI	: Very Large Scale Integration
VMM	: Via Minimization Model
WLM	: Wirlength Minimization Model

Appendix B

GeoSteiner & Flute

Initially, the minimal steiner trees for each net are generated by GeoSteiner3.1 package [Warm01], which is the fastest implementation for constructing optimal rectilinear steiner minimal tree. However, in [Chu05], a fast and accurate lookup table based steiner tree algorithm "Flute" is presented. The steiner trees produced by Flute is optimal for nets with up to 9 terminals and it is still very accurate for nets with up to 100 terminals [Chu05]. Table B.1 compares these two packages for RNWO model. In the third part of this table (i.e. "Flute + GeoSteiner"), Flute and GeoSteienr3.1 packages are used to generate minimal steiner trees for nets with less than 10 terminals and nets with more than 9 terminals, respectively. From this table we can see that the Flute package reduces the computation time largely without deteriorating the quality of solution. Since the pure Flute package produces the similar quality of solution as the combined package (Flute + GeoSteiner) with much less CPU time, in this thesis, the Flute package is adopted for steiner trees construction of WLM, VMM, ECM, WVEM, WVZM and RNWO models.

Results Comparison for ILP+Maze Router Without Additional Tree Construction															
Circuit	Only GeoSteiner					Only Flute					Flute+GeoSteiner				
	Wire	Bends	M_{rd}	OF	Time	Wire	Bends	M_{rd}	OF	Time	Wire	Bends	M_{rd}	OF	Time
ibm01	67561	17438	18	351	654	68078	17056	19	356	33	67078	17038	18	340	192
ibm02	179940	36497	36	299	1248	180491	36137	37	293	92	179964	36371	36	235	291
ibm03	151585	26297	31	75	819	151637	25724	30	72	48	152266	26400	31	73	217
ibm04	175151	32971	29	754	1043	175142	32653	28	613	96	175121	32936	29	627	244
ibm05	410292	52798	63	0	1571	410091	50694	63	0	32	409714	52109	63	0	653
Tot	984529	166001	177	1479	5335	988439	162714	177	1334	329	984143	164854	177	1275	1597
Imp	-	-	-	-	-	0%	2%	0%	10%	94%	0%	0%	0%	14%	70%
ibm06	291658	53106	34	77	1307	291541	51634	34	71	85	293000	53250	34	65	487
ibm07	38221	63987	37	338	1873	382235	62574	38	280	184	382760	63865	37	293	679
ibm08	419166	80248	33	132	2009	420359	78822	33	123	177	420533	80173	34	157	767
ibm09	428237	74300	29	135	2003	426903	72368	29	133	261	426371	73572	29	96	892
ibm10	595094	106570	44	488	2995	596624	104778	44	396	348	595778	106000	43	422	1166
Tot	2116371	378211	177	1170	10187	2117442	370176	178	1003	1055	2118442	376860	177	1033	3991
Imp	-	-	-	-	-	0%	2%	0%	14%	90%	0%	0%	0%	12%	61%

Table B.1: Results Comparison for Different Initial Steiner Tree Construction

Bibliography

[Aart85] E. H. L. Aarts and P. J. M. V. Laarhoven, "Statistical Cooling: A General Approach to Combinatorial Optimization Problems," *Phillips Journal Res.,*, vol. 4, pp. 193–226, 1985.

[Abel72] L. C. Abel, "On the ordering of connections of automatic wire routing," *IEEE Transactions on Computers*, vol. 21, pp. 11227–1233, Novermber 1972.

[AD85] B. W. Kernighan A. Dunlop, "A procedure for placement of standard-cell VLSI placement," *IEEE Trans. on CAD of Integ. Circ. and Syst., 4(4)*, vol. 4, no. 4, pp. 92–98, 1985.

[Agni03] A. R. Agnihotri and P. H. Madden, "Congestion Reduction in Traditional and New Routing Architectures," In *Proceedingsedings of The ACM Great Lakes Symposium on VLSI*, pp. 211–214, 2003.

[Aker67] S. B. Aker, "A modification of lee's path connection algorithm," *IEEE Transactions on Computer-Aided Design*, vol. , pp. 97–98, February 1967.

[Albe01] C. Albercht, "Global Routing by New Approximation Algorithms for Multicommodity Flow," *IEEE Transactions on Computer-Aided Design*, vol. 5, pp. 622–632, 2001.

[Algr01] C. Algrecht, "Global routing by new approximation algorithms for multicommodity flow," *IEEE Transactions Computer-Aided Design*, vol. 20, pp. 622–632, 2001.

[Ande99] J. Andersson, "A survey of multiobjective optimization engineering design," Technical Report LiTH-IKP-R-1097, Dept. of Mechanical Engineering, Linkoping University, 1999.

[Aosh83] A. Aoshima and E. Kuh, "Multi-channel optimization in gate array LSI layout ," In *Proceedings of IEEE International Conference on Computer-Aided Design*, pp. 1005–1008, November 1983.

[Asso97] Semiconductor Industry Association, "National Technology Roadmap for Semiconductors," pp. 101, 1997.

[Bako90] H.B. Bakoglu, *Circuits , Interconnections, and Packaging for VLSI*, Addison-Wesley, 1990.

[Bane02] K. Banerjee and A. Mehrotra, "A Power-Optimal Repeater Insertion Methodology for Global Interconnects in Nanometer Designs," *IEEE Transactions on Electron Devices*, vol. 49, No. 11, pp. 2001–2007, Nov. 2002.

[Behj02] L. Behjat, *New Modeling and Optimization Techniques for the Global Routing Problem* PhD thesis, University of Waterloo, Ont. Canada, 2002.

[Bora94] M. Borah, R. M. Owens, and M. J. Irwin, "An edge-based heuristic for Steiner routing," *IEEE Transactions Computer-Aided Design*, vol. 13, pp. 1563–1568, Dec 1994.

[BPTM] "BPTM Provided by the Device Group at UC Berkeley," .

[Bren02] U. Brenner and A. Rohe, "An Effective Congestion Driven Placement Framework," In *Interational Symposium on Physical Design*, pp. 6, April 2002.

[Burs83a] M. Burstein and S. J. Hong, "Hierarchical VLSI layout: Simultaneous wiring and placement," In *proc. of VLSI'83, F. Anceau and E. J. Aas, eds)*, pp. 45–60, Elsevier Science Publishers, B.V. Amsterdam, 1983.

[Burs83b] M. Burstein and R. Pelavin, "Hierarchical Wire Routing," In *IEEE Transactions Computer-Aided Design, Vol. CAD-2*, pp. 223–234, Oct. 1983.

[Cald00] A. Caldwell, A. Kahng, and I. Markov, "Can recursive bisection alone produce routable placement," In *Proceedings of IEEE/ACM Design Automation Conference*, pp. 477–482, 2000.

[Card91] R. Carden and C.K. Cheng, "A Global Router Using An Efficient Approximate Multicommodity Multiterminal Flow Algorithm," In *Proceedings of ACM/IEEE Design Automation Conf.*, pp. 316–321, 1991.

[Chan00] T. Chan, J. Cong, T. Kong, and J. Shinnerl, "Multilevel optimization for large-scale circuit placement," In *Proceedings IEEE International Conf. on CAD*, pp. 171–176, November 2000.

[Chan95] A. P. Chandrakasan and R. W. Brodersen, *Low Power Digital CMOS Design*, Kluwer Academic Publishers, Boston, 1995.

[Chan99] H. Chang, L. Cooks, and M. Hunt, *Surviving the SOC Revolution*, Kluwer Academic Publishers, London, 1999.

[Chen04] L. Cheng, X. Y. Song, G. W. Yang, and Z. W. Tang, "A Fast Congestion Estimator for Routing with Bounded Detours," In *Proceedings of the 2004 conference on Asia South Pacific design automation*, pp. 666–670, 2004.

[Chen05a] R. Chen and H. Zhou, "Efficient Algorithms for Buffer Insertion in General Circuits Based on Network Flow," In *Proceedingsedings of 2005 ICCAD*, pp. 509–514, 2005.

[Chen05b] T. Chen and Y. Chang, "Multilevel Full-Chip Gridless Routing Considering Optical Proximity Correction," In *Proceedings of the 2005 conference on Asia South Pacific design automation*, pp. 1160–1163, 2005.

[Chen06] T. Chen, Y. Chang, and S. Lin, "A Novel Framework for Multilevel Full-Chip Gridless Routing," In *Proceedings of the 2006 conference on Asia South Pacific design automation*, pp. 636–641, 2006.

[Chen89] Y. A. Chen, Y. L. Liu, and Y. C. Hsu, "A new global router for ASIC design based on simulated evolution," In *Proceedings of the International Symposium on VLSI Technology, Systems and Applications*, pp. 261–265, 1989.

[Chen94] C. E. Cheng, "RISA: Accurate and Efficient Placement Routability Modeling," In *Proceedings of 1994 Computer Aided Design*, pp. 690–695, 1994.

[Chia90] C. Chiang, M. Sarrafzadeh, and C. K. Wong, "Global Routing Based on Steiner Min-Max Trees," *IEEE Transactions on Computer-Aided Design*, vol. 9, pp. 1318–1325, 1990.

[Chia94] C. Chiang, C. K. Wong, and M. Sarrafzadeh, "A Weighted Steiner Tree-based Global Router with Simultaneous Length and Density Minimization," *IEEE Transactions on Computer-Aided Design*, vol. 13, pp. 1461–1469, 1994.

[Cho06] M. Cho and D. Z. Pan, "BoxRouter: A New Global Router Based on Box Expansion and Progressive ILP," In *Proceedingsedings of the Design Automation Conference*, pp. 373–378, IEEE/ACM, San Francisco, California, USA, 2006.

[Chu01] C. Chu and D.F.Wong, "Closed Form Solution to Simultaneous Buffer Insertion/Sizing and Wire Sizing," *ACM Transactions on Design Automation of Electronic Systems*, vol. 6, No. 3, pp. 343–371, July 2001.

[Chu04] C. Chu, "Flute: Fast Lookup Table Based Wirelength Estimation Technique," *IEEE Transactions Computer-Aided Design*, pp. 696–701, 2004.

[Chu05] C. Chu and Y. C. Wong, "Fast and Accurate Rectilinear Steiner Minimal Tree Algorithm for VLSI Design," In *Proceedings of the 2005 International Symposium on Physical Design*, pp. 28–35, 2005.

[Coel00] C. A. Coello, "An Updated Survey of GA-Based Multiobjective Optimization," *ACM Computing Surveys*, vol. 32, pp. 109–143, June 2000.

[Cong00] J. Cong, S. Lim, and C. Wu, "Performance Driven Multi-Level and Multiway Partitioning with Retiming," In *ACM/IEEE Design Automation Conference*, pp. 274–279, June 2000.

[Cong01a] J. Cong, J. Fang, and Y. Zhang, "Multilevel Approach to Full-Chip Gridless Routing," In *Proceedings of the 2001 International Conference on Computer Aided Design*, pp. 396–403, 2001.

[Cong01b] J. Cong and Z. Pan, "Interconnect Performance Estimation Models for Design Planning," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 20, No. 6, pp. 739–752, June 2001.

[Cong02] J. Cong, M. Xie, and Y. Zhang, "An Enhanced Multilevel Routing System," In *Proceedings of the 2002 International Conference on Computer Aided Design*, pp. 51–58, 2002.

[Cong05] J. Cong, J. Fang, and Y. Zhang, "MARS-A Multilevel Full-Chip Gridless Routing System," *IEEE Transactions Computer-Aided Design*, vol. 24, pp. 382–394, March 2005.

[Cong92] J. Cong and B. Preas, "A New Algorithm for Standard Cell Global Routing," *Integration: the VLSI Journal*, vol. 14, pp. 49–65, November 1992.

[Cong96] J. Cong, L. He, C. Koh, and P. H. Madden, "Performance Optimization of VLSI Interconnect Layout," *Integration, the VLSI Journal*, vol. 21, No. 1-2, pp. 1–94, Nov. 1996.

[Cong97] J. Cong and P. Madden, "Performance Driven Global Routing for Standard Cell Design," In *Proceedingsedings of the ACM International Symposium on Physical Design*, pp. 73–80, 1997.

[Dai87] W. W. Dai and E. S. Kuh, "Simultaneous floorplanning and global routing for hierarchical building block layout," *IEEE Transactions Computer-Aided Design*, vol. 6, pp. 828–837, 1987.

[Dant63] G. B. Dantzig, "Linear Programming and Extensions," *Princeton, NJ: Princeton University*, 1963.

[Deng94] A. C. Deng, "Power Analysis For CMOS/BiCMOS Circuits," In *Proceedings of the International Workshop on Low Power Design*, pp. 3–8, 1994.

[ea96] J. Cong et al., "Simultaneous Buffer and Wire Sizing for Performance and Power Optimization," *Proceedings of the International Symposium on Low Power Electronics and Design*, pp. 271–276, 1996.

[Elmo48] W. C. Elmore, "The Transient Response of Damped Linear Networks," *Journal of Applied Physics*, vol. 19, No. , pp. 55–63, Jan. 1948.

[Esbe94] H. Esbensen, "A macro-cell global router based on two genetic algorithms," In *Proceedings of the European Design Automation Conference*, pp. 428–433, 1994.

[Gare77] M. R. Garey and D. S. Johnson, "The rectilinear steiner tree problem is np-complete," *SIAM Journal of Applied Mathematics*, vol. 32, pp. 826–834, 1977.

[Garg98] N. Garg and J. Kunemann, "Performance Driven Global Routing for Standard Cell Design," In *Proceedings of 39th Annu, Symposium on Foundations of Computer Science*, pp. 300–309, 1998.

[Gegu00] T. Geguchi, T. Koide, and S. Wakabayashi, "Timing-driven hierarchical global routing with wire-sizing and buffer-insertion for VLSI with multi-routing," In *Proceedings of the 2000 conference on Asia South Pacific design automation*, pp. 99–104, 2000.

[Grif94] J. Griffith, G. Robins, J.S. Salowe, and T. Zhang, "Closing the gap: Near-optimal Steiner trees in polynomial time," *IEEE Transactions Computer-Aided Design*, vol. 13, pp. 1351–1365, Nov 1994.

[Hach89] G. Hachtel and C. Morrison, "Linear Complexity Algorithms for Hierarchical Routing," *IEEE Transactions on Computer Aided Design*, vol. 8, No. 1, pp. 64–80, 1989.

[Hadl75] F. Hadlock, "Finding a Maximum Cut of a Planar Graph in Polynomial Time," *SIAM Journal of Computing*, vol. 11, pp. 885–892, 1975.

[Hads03] R. T. Hadsell and P. H. Madden, "Improved Global Routing through Congestion Estimation," In *Proceedingsedings of the Design Automation Conference*, pp. 28–34, IEEE/ACM, Anaheim, CA, 2003.

[Hana66] M. Hanan, "On Steiner's Problem with Rectilinear Distance," *SIAM Journal of Applied Mathematics*, vol. 14, pp. 255–265, 1966.

[Heis91] J. Heistermann and T. Lengauer, "The efficient solution of integer programs for hierarchical global routing," *IEEE Transactions Computer-Aided Design*, vol. 10, pp. 748–753, 1991.

[High69] D. W. Hightower, "A solution to the line routing problem on a continous plane," In *Proceedings of 6th Design Automation Workshop*, pp. , 1969.

[Ho04] T. Ho, Y. Chang, and S. Chen, "Multilevel Routing with Antenna Avoidance," In *Proceedings of the 2004 International Symposium on Physical Design*, pp. 34–40, 2004.

[Ho90] J. M. Ho, G. Vijayan, and C. K. Wong, "A new approach to the rectilinear steiner tree problem," *IEEE Transactions on Computer-Aided Design*, vol. 9, pp. 185–193, February 1990.

[Hong97] X. L. Hong, T. X. Xue, J. Huang, C. K. Cheng, and E. S. Kuh, "TIGER: An Efficient Timing-Driven Global Router for Gate Array and Standard Cell Layout Design," *IEEE Transactions on Computer-Aided Design*, vol. 16, pp. 1323–1331, November 1997.

[Hou01] W. Hou, H. Yu, Y. Cai, W. Wu, J. Gu, and W. Kao, "A New Congestion-Driven Placement Algorithm Based on Cell Inflation," In *Proceedings of Interational Conference on ASP-DAC*, pp. 605–608, 2001.

[Hu02] J. Hu and M. T. Shing, "A Timing-Constrained Simultaneous Global Routing Algorithm," *IEEE Transactions on Computer-Aided Design*, vol. 21, pp. 1025–1036, 2002.

[Hu85] T. C. Hu and M. T. Shing, "A decomposition algorithm for circuit routing," In *T. C. Hu and E. S. Kuh, editors, VLSI Layout: Theory and Design*, pp. 144–152, IEEE Press, New York, 1985.

[Huan93] J. Huang, X. L. Hong, C. K. Cheng, and E. S. Kuh, "An efficient timing driven global routing algorithm," *In Proceedings of ACM/IEEE International Conference, Design Automation*, pp. 596–600, 1993.

[Hwan76] F. K. Hwang, "On Steiner Minimal Trees with Rectilinear Distance," *SIAM Journal of Applied Mathematics*, vol. 30, pp. 104–114, January 1976.

[Inte02] "International Technology Roadmap for Semiconductors," 2002.

[ISPD07] ISPD2007, "http://www.ispd.cc/ispd07_contest.html," 2007.

[ISPD98] ISPD98/IBM, "http://www.ece.ucsb.edu/ kastner/labyrinth/benchmarks/," 1998.

[Jing04] T. Jing, X. L. Hong, J. Y. Xu, , C. K. Cheng, and J. Gu, "UTACO: A Unified Timing and Congestion Optimization Algorithm for Standard Cell Global Routing," *IEEE Transactions Computer-Aided Design*, vol. 23, pp. 358–365, 2004.

[Kahn92] A.B. Kahng and G. Robins, "A new class of iterative Steiner tree heuristics with good performance," *IEEE Transactions Computer-Aided Design*, vol. 11, pp. 893–902, July 1992.

[Kang03] S. Kang and Y. Leblebici, *CMOS Digital Integrated Circuits*, McGraw-Hill, 2003.

[Karm84] N. Karmarkar, "A New Polynomial-Time Algorithm for Linear Programming," *Combinatorica*, vol. 4, pp. 373–395, 1984.

[Karp87] R.M. Karp, F.T. Leighton, and R.L. Rivest, *Global wire routing in two-demensional arrays.Algorithmica, 2:113-129*, 1987.

[Kary97] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar, "Multilevel Hypergraph Partioning: Application in VLSI Design," In *Proceedings of the Design Automation Conference*, pp. 526–529, ACM/IEEE, Las Vegas, Nevada, June 1997.

[Kast] R. Kastner and M. Sarrafzadeh, "Labyrinth: A global router and routing development tool. http://www.ece.ucsd.edu/ kastner/labyrinth," .

[Kast00] R. Kastner, E. Bozorgzadeh, and M. Sarrafzadeh, "Predictable Routing," In *Proceedings of IEEE International Conference on Computer-Aided Design*, pp. 110–113, 2000.

[Kast01] R. Kastner, E. Bozorgzadeh, and M. Sarrafzadeh, "An Exact Algorithm for Coupling-Free Routing," In *Proceedingsedings of International Symposium on Physical Design*, pp. 10–15, 2001.

[KDB94] B. A. McCoy K. D. Boese, A. B. Kahng and G. Robins, "Rectilinear Steiner Trees with Minimum Elmore Delay," In *Proceedings of ACM/IEEE Design Automation Conf.*, pp. 381–386, Jun. 1994.

[Klei91] J. Kleinhans, G. Sigl, F. Johannes, and K. Antreich, "GORDIAN: VLSI Placement By Quadratic Programming and Slicing Optimization," *IEEE Transaction on Computer Aided Design*, vol. 10, No. 3, pp. 356–365, March 1991.

[Kuca04] D. Kucar, S. Areibi, and A. Vannelli, "Hypergraph Partitioning Techniques," *Special Issue of Continuous, Discrete and Impulsive System Journal*, vol. 11, pp. 341–369, Feb 2004.

[L02] S. L and Y. Chang, "A Novel Framework for Multilevel Routing Considering Routability and Performance," In *Proceedings of the 2002 International Conference on Computer Aided Design*, pp. 44–50, 2002.

[Lai02] M. Lai and D.F. Wong, "Maze Routing with Buffer Insertion and Wiresizing," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 21, No. 10, pp. 1205–1209, Oct. 2002.

[Lee61] C. Y. Lee, "An Algorithm for Path Connection and its Application," *IRE Transactions on Electronic Computers*, vol. 10, pp. 346–365, 1961.

[Lee91] K. W. Lee and C. Sechen, "A global router for sea-of-gate circuits," *IEEE Transactions on Computer-Aided Design*, vol. , pp. 242–247, 1991.

[Leng90] T. Lengauer, *Combinatorial Algorithms for Integrated Circuit Layout*, John Wiley & Sons, New York, 1990.

[Leng93] T. Lengauer and R. Muller, "Robust and accurate hierarchical floorplanning with integrated global wiring," *IEEE Transactions Computer-Aided Design*, vol. 12, pp. 802–809, 1993.

[Li03] R. Li, D. Zhou, J. Liu, and X. Zeng, "Power-Optimal Simultaneous Buffer Insertion/Sizing and Wire Sizing," *Proceedings of the International Conference on Computer-Aided Design*, pp. 581–586, Nov. 2003.

[Lill96] J. Lillis, C. K. Cheng, and T. Y. Lin, "Simultaneous Routing and Buffer Insertion for High Performance Interconnect," *Proceedings of the Great Lakes Symposium on VLSI*, pp. 148–153, 1996.

[Liu00] I-M. Liu, A. Aziz, and D. F. Wong, "Meeting delay constraints in DSM by minimal repeater insertion," In *Proceedingsedings of DATE*, pp. 436–441, 2000.

[Liu05] X. Liu, Y. Peng, and M. C. Papaefthymiou, "RIP: an efficient hybrid repeater insertion scheme for low power," *Proceedings of the Design Automation and Test in Europe 2005*, vol. 2, No. , pp. 1330–1335, March 2005.

[Liu99] I-M. Liu, A. Aziz, D. F. Wong, and H. Zhou, "An Efficient Buffer Insertion Algorithm for Large Networks Based on Lagrangian Relaxation," In *Proceedingsedings of the International Conference on Computer Design*, pp. 210–215, 1999.

[Lou01] J. Lou, S. Krishnamoorthy, and H. S. Sheng, "Estimating Routing Congestion using Probabilistic Analysis," In *Interational Symposium on Physical Design*, pp. 112–117, April 2001.

[Luk87] W. K. Luk, P. Sipal, M. Tamminen, D. Tang, L. S. Wong, and C. K. Wong, "A hierarchical global wiring algorithm for custom chip design," *IEEE Transactions Computer-Aided Design*, vol. 6, pp. 518–533, 1987.

[Lund86] M. Lundy, "Convergence of an Annealing Algorithm," *Mathematical programming*, vol. 34, pp. 111–124, 1986.

[MCNC91] 1991.

[Meix90] G. Meixner and U. Lauther, "Congestion Driven Placement Using a New Multi-Partitioning Heuristic," In *Proceedings of International Conference on Computer-Aided Design*, pp. 332–335, November 1990.

[Mika68] K. Mikami and K. Tabuchi, "A computer program for optimal routing of printed circuit connectors," In *Proceedings of IFIPS Conf.*, pp. 1475–1478, 1968.

[MS84] M. Marek-Sadowska, "Global router for gate array," In *Proceedings of the IEEE International Conference on Computer Aided Design*, pp. 332–337, 1984.

[Nair87] R. Nair, "A simple yet effective technique for global wiring," *IEEE Transactions on Computer-Aided Design*, vol. 6, pp. 165–172, March 1987.

[Ng87] A. P. Ng, P. Raghavan, and C. D. Thompson, "Experimental results for a linear program global router," *Computer Artificial Intelligence*, vol. 6, pp. 130–143, 1987.

[Osyc85] A. Osyczka, "Multicriteria Optimization for engineering design," *In Design Optimization, J. S. Gero, Ed. Academic Press Inc.*, pp. 193–227, 1985.

[Otte03] R. Otten and G. S. Garcea, "Simultaneous Analytic Area and Power Optimization for Repeater Insertion," *Proceedings of the International Conference on Computer-Aided Design*, pp. 568–573, 2003.

[Pan06] M. Pan and C. Chu, "FastRoute: A Step to Integrate Global Routing into Placement," In *Proceedings of ICCAD*, pp. 464–471, Nov. 2006.

[Para98] P.N. Parakh, R.B. Brown, and K.A. Sakallah, "Congestion Driven Quadractic Placement," In *Proceedings of Design Automation Conference*, pp. 275–278, June 1998.

[Raba03a] J. Rabaey, A. Chandrakasan, and B. Nikolic', *Digital Integrated Circuits*, Prentice Hall, NJ, 2003.

[Raba03b] J. Rabaey, A. Chandrakasan, and B. Nikolic, *DIGITAL INTEGRATED CIRCUITS*, Pearson Education Publishing Company, Inc, 2003.

[Ragh87] P. Raghavan and C. D. Thompson, "Randomized Rounding: A Technique for Provably Good Algorithms and Algorithmic Proofs," *Combinatorica*, vol. 7(4), pp. 365–374, 1987.

[Ragh91] P. Raghavan and C. D. Thompson, "Multi-Terminal Global Routing: A Deterministic Approximation Scheme," *Algorithmica*, vol. 6, pp. 73–82, 1991.

[Rard98] R. L. Rardin, *Optimization in Operations Research*, Prentice Hall, Inc., 1998.

[Sait95] S. M. Sait and H. Youssef, *VLSI Physical Design Automation*, McGraw-Hill, London, England, 1995.

[Sarr96] M. Sarrafzadeh and C. K. Wong, *An Introduction to VLSI Physical Design*, McGraw-Hill, 1996.

[Sche86] C. Schen, *VLSI Placement and Routing Using Simulated Annealing*, Kluwer Academic, Boston, MA, 1986.

[Sech85] C. Sechen and A. Sangiovanni-Vincentelli, "The TimberWolf Placement and Routing Package," *IEEE Journal of Solid-State Circuit*, vol. sc-20, pp. 510–522, Apr. 1985.

[Shah90] F. Shahrokhi and D. W. Matula, "The maximum concurrent flow problem," *Journal of the ACM*, vol. 37, pp. 318–334, 1990.

[Shah91] K. Shahookar and P. Mazumder, "VLSI Cell Placement Techniques," *ACM Computing Surveys*, vol. 23, No. 2, pp. 143–220, 1991.

[Shah96] J. C. Shah and S. S. Sapatnekar, "Wiresizing with Buffer Placement and Sizing for Power-Delay Tradeoffs," *Proceedings of VLSI Design*, pp. 346–351, 1996.

[Sher95] N. Sherwani, S. Bhingarde, and A. Panyam, *Routing in the Third Dimension*, IEEE Press, New York, NY, 1995.

[Sher99] N. Sherwani, *Algorithms for VLSI Physical Design Automation*, Kluwer Academic, Boston, MA, 1999.

[Shra87] E. Shragowitz and S. Kell, "A global router based on a multicommodity flow model," *Integration, the VLSI Journal*, vol. 5, pp. 3–16, 1987.

[Souk78] J. Soukup, "Fast Maze Router," In *Proceedings of the Design Automation Conference*, pp. 100–102, 1978.

[Sun93] Wern-Jieh Sun and Carl Sechen, "Efficient and effective placement for very large circuits," In *Proceedings of IEEE/ACM ICCAD*, pp. 170–177, 1993.

[Sze05] C. N. Sze, C. J. Alpert, J. Hu, and W. Shi, "Path Based Buffer Insertion," In *Proceedingsedings of the Design Automation Conference*, pp. 509–514, IEEE/ACM, 2005.

[Tang01] X. Tang, R. Tian, H. Xiang, and D. F. Wong, "A New Algorithm for Routing Tree Construction with Buffer Insertion and Wire Sizing under Obstacle Constraints," *Proceedings of the International Conference on Computer-Aided Design*, pp. 49–56, 2001.

[Tarj83] R. Tarjan, "Data Structures and Network Algorithms," *Society for Industrial and Applied Mathematics*, 1983.

[Thom00] M. D. Thompson, *A Clustering Utility-based Approach for ASIC Design* PhD thesis, University of Waterloo, Ont. Canada, 2000.

[Ting83] B. Ting and B. Tien, "Routing Techniques for Gate Array," *IEEE Transactions Computer-Aided Design*, vol. 2, pp. 301–312, Oct. 1983.

[Vann91] A. Vannelli, "An Adaptation of the Interior Point Method for Solving the Global Routing Problem," *IEEE Transactions on Computer-Aided Design*, vol. 10, pp. 193–203, February 1991.

[Vecc83] M. P. Vecchi and A. Kirkpatrick, "Global Wiring by Simulated Annealing," *IEEE Transactions Computer-Aided Design*, vol. 2, pp. 215–222, 1983.

[vG90] L. P. P. P. van Ginneken, "Buffer Placement in Distributed RC-Tree Networks for Minimal Elmore Delay," *Proceedings of ISCAS*, vol. , No. , pp. 865–868, 1990.

[Wagh06] M. Waghmode, Z. Li, and W. Shi, "Path Based Buffer Insertion," In *Proceedingsedings of the Design Automation Conference*, pp. 296–301, IEEE/ACM, 2006.

[Wang00a] M. Wang and M. Sarrafzadeh, "Modeling and minimization of routing congestion," In *Proceeding of the 2000 conference on Asia and South Pacific deisgn automation*, pp. 185–190, 2000.

[Wang00b] M. Wang, X. Yang, and M. Sarrafzadeh, "Congestion Minimization During Placement," *IEEE Transactions on Computer Aided Design*, vol. 19, No. 10, pp. 1140–1148, 2000.

[Wang00c] M. Wang, X. Yang, and M. Sarrafzadeh, "Multi-Center Congestion Estimation and Minimization During Placement," In *Proceedings of Interational Symposium on Physical Design*, pp. 147–152, 2000.

[Wang96] D. Wang and E. S. Kuh, "Performance-driven interconnect global routing," In *Proceedings of Great Lake Symp. VLSI*, pp. 132–136, 1996.

[Wang99] M. Wang and M. Sarrafzadeh, "On The Behavior of Congestion Minimization During Placement," In *Proceedings of Interational Conference on ASP-DAC*, pp. 145–150, 1999.

[Warm01] D. M. Warme, P. Winter, and M. Zachariasen, "GeoSteiner 3.1," *www.diku.dk/geosteiner/*, 2001.

[Warm97] D. M. Warme, "A New Exact Algorithm for Rectilinear Steiner Trees," *International Symposium on Mathematical Programming*, 1997.

[Wong00] S. C. Wong, G-Y. Lee, and D-J. Ma, "Modeling of Interconnect Capacitance, Delay, and Crosstalk in VLSI," *IEEE Transactions on Semiconductor Manufacturing*, vol. 13, No. 1, pp. 108–111, Feb. 2000.

[Xu05] J. Xu, X. Hong, and T. Jing, "Timing-Driven Global Routing with Efficient Buffer Insertion," *IEEE International Symposium on Circuits and Systems*, vol. 19, pp. 2449–2452, May 2005.

[Yan04] J. T. Yan and S. H. Lin, "Timing-Constrained Congestion-Driven Global Routing ," In *Proceedings of the 2004 conference on Asia South Pacific design automation*, pp. 683–686, 2004.

[Yang01a] X. Yang, R. Kastner, and M. Sarrafzadeh, "Congestion Reduction During Placement Based on Integer Programming," In *Proceedings of Interational Conference on Computer-Aided Design*, pp. 573–576, 2001.

[Yang01b] X. Yang, R. Lauther, and M. Sarrafzdeh, "Congestion Estimation During Top-down Placement," In *Proceedings of International Symposium on Physical Design*, pp. 164–169, April 2001.

[Yang05] Z. Yang, *A Multiple-objective based Hierarchical Global Routing Approach for VLSI ASIC Design*, Ph.D. Proposal, ECE Department, 2005.

[Yang06] Z. Yang, S. Areibi, and A. Vannelli, "An ILP Based Hierarchical Global Routing Approach for VLSI ASIC Design," University of Waterloo, Technical Report, University of Waterloo, Waterloo, Ontario, 2006.

[Yang07] Z. Yang, S. Areibi, and A. Vannelli, "An ILP Based Hierarchical Global Routing Approach for VLSI ASIC Design," *Optimization Letters*, vol. 1, pp. 281–297, June 2007.

[Yous05] A. Youssef, M. Anis, and M. Elmasry, "POMR: A Power-Aware Interconnect Optimization Methodology," *IEEE Transactions on VLSI Systems*, vol. 13, pp. 297–307, March 2005.

[Yous07] A. Youssef, T. Myklebust, M. Anis, and M. Elmasry, "A Low-Power Multi-Pin Maze Routing Methodology," *Proceedings of the IEEE International Symposium on Quality Electronic Design*, pp. 153–158, March 2007.

[Yous99] H. Youssef and S. M. Sait, "Timing-driven global routing for standard-cell VLSI design," *Computer Systems: Science and Engineering*, vol. 14, pp. 175–185, May 1999.

[Zach97] M. Zachariasen, "Rectilinear Full Steiner Tree Generation," University of Copenhagen, Technical Report, University of Copenhagen, Denmark, 1997.

[Zhou04] H. Zhou, "Efficient Steiner Tree Construction Based on Spanning Graphs," *IEEE Transactions Computer-Aided Design*, vol. 23, pp. 704–710, May 2004.

[Zhu98] K. Zhu, Y. W. Chang, and D. F. Wong, "Timing-driven routing for symmetrical-array-based FPGAs," In *Proceedingsedings of ACM/IEEE International Conference on Computer Design*, pp. 628–633, 1998.

[Zube05] P. Zuber, W. Stechele, and A. Herkersdorf, "Reduction of CMOS Power Consumption and Signal Integrity Issues by Routing Optimization," In *Proceedings of the 2005 Design Automation and Test in Europe*, pp. 986–987, 2005.