AN ANALOG TEMPLATE ROUTER BASED ON LAYOUT DESCRIPTION SCRIPT (LDS)

by

Cem Sümengen

B.S., Industrial Engineering, Istanbul Technical University, 2011

B.S., Electronics Engineering, Istanbul Technical University, 2010

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Electrical and Electronics Engineering

Boğaziçi University

2014

AN ANALOG TEMPLATE ROUTER BASED ON
LAYOUT DESCRIPTION SCRIPT (LDS)

APPROVED BY:

Prof. Günhan Dündar (Thesis Supervisor)	.....
Assoc. Prof. Alper Şen	.....
Assist. Prof. İ. Faik Başkaya	.....

DATE OF APPROVAL: 26.09.2014

ACKNOWLEDGEMENTS

First and foremost, I would like to express my appreciation to my thesis advisor, Prof. Günhan Dündar for his guidance and support throughout my research and thesis. I would like to thank Assist. Prof. İ. Faik Başkaya and Assoc. Prof. Alper Şen for sharing their knowledge and taking part in my thesis jury.

I owe many thanks to my dearest friends for their friendship and support; Ata Sarrafinezhad, İslam Tete, Arda Varılsüha, Bilgin Kızıltaş, Erhan Kıroğlu, A. Burak Tahiroğlu, Yusuf Oktay, Fatih Gençoğlu, Y. Umut Kuş, Necati Cihan Camgöz, Erşan Aykut and Yunus Gündüz. I also would like to thank Ahmet Unutulmaz and all the Beta Lab. members for their friendship and helping me throughout my thesis. Besides, I especially would like to express the deepest appreciation to my colleagues in Arçelik A.Ş; Hürriyet Keskin, Çağatay Sönmez, Hamdican Ölçücü, Dilek Kayahan, Gizem Diril and Anıl Üzümcüoğlu for their friendship, support and encouragement.

In addition, I would like to thank my family for their love and support throughout my life.

Finally, I would like to thank TÜBİTAK BİDEB for supporting me in my graduate studies.

ABSTRACT

AN ANALOG TEMPLATE ROUTER BASED ON LAYOUT DESCRIPTION SCRIPT (LDS)

Long design-test cycles in full-custom design process leads to high design costs in analog circuit design. The objective of design automation is to decrease the design cost and the time to market by reducing the time spent in the design process. Template-based tools have an advantage of requiring lower computational time by utilizing stored design knowledge. This property makes template based design techniques very appealing for some applications such as porting a design to different technologies or making minor changes on the design in the same technology. Unutulmaz et al. (2011) presented a declarative language to define layout templates for analog circuits, named Layout Description Script (LDS). In this study, a template router is proposed which outputs routing information for layout templates coded in LDS. The output of the router is flexible routing information also coded in LDS. It is shown that the layout instances created from the output template with routing information can adapt itself to the changes in the dimensions of the circuit elements. Therefore, the template can be used by design automation tools in a closed optimization loop. The router proposed in this thesis differs from the other routers by the nature of it's input and output. The input file holds the relative placement information of the circuit elements rather than providing fixed coordinates in the layout. Likewise, the output file consists of relative positions of the interconnect and is self-adapting for the variations in dimensions of the circuit elements.

ÖZET

LDS TABANLI ANALOG ŞABLON YÖNLENDİRİCİSİ

Analog devre tasarım süreçlerindeki uzun tasarım-test döngüleri yüksek tasarım maliyetlerine yol açmaktadır. Analog tasarım otomasyon araçlarının kullanılmasıyla tasarım süreçlerinde harcanan zamanı azaltarak tasarım maliyetlerinin ve pazara giriş süresinin düşürülmesi amaçlanmaktadır. Şablona dayalı tasarım araçları, önceden depolanmış tasarım bilgilerini kullanarak hesaplama zamanını düşürmektedir. Bu özelliği, şablona dayalı tasarım araçlarını bir teknolojideki tasarımı başka teknolojilere aktarmak veya aynı teknolojide bir tasarım üzerinde ufak değişiklikler yapmak gibi uygulamalarda oldukça çekici kılmaktadır. Unutulmaz ve diğ. (2011) analog devreler için devre planı şablonları tanımlamada kullanılan LDS adında tanımsal bir dil sundular. Bu çalışmada çıktı olarak LDS sablonları için elektriksel bağlantı bilgisini üreten bir şablon yönlendirici tasarlanmıştır. Yönlendiricinin çıktısı da aynı zamanda LDS ifadelerinden oluşmaktadır ve esnek bir yapıdadır. Yönlendiricinin çıktısı olarak verilen bağlantı bilgisini kapsayan şablon ile örneklenen devre planının, devre elemanlarının ebatlarındaki değişimlere uyum sağlayabildiği gösterilmiştir. Bu sayede üretilen şablon, tasarım otomasyon araçları tarafından kapalı bir döngüde kullanılmaya uygundur. Çalışmada önerilen yönlendirici, aldığı girdinin ve çıktısının doğası gereği diğer yönlendiricilerden ayrılmaktadır. Girdi dosyası, devre elemanlarının sabit koordinatları yerine göreli yerleşim bilgisini içermektedir. Aynı şekilde çıktı bilgisi de elektriksel bağlantıların devre elemanlarının ebatlarındaki değişimlere uyum sağlayabilecek şekilde göreli yerleşimlerini içermektedir.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF SYMBOLS

c_{ij}	Coupling capacitance between the wires i and j
d_{ij}	Distance between the wires i and j
d_{Layer}	Thickness of a routing layer
h_c	Thickness of a wire segment
I_{eq}	The root mean square (RMS) current on a path
I_{max}	Maximum current on a path
$J_{max}(T_{ref})$	Maximum current density allowed by a process for temperature T_{ref}
J_{peak_Layer}	Layer dependent peak current density
l_c	Length of a wire segment
l_{ij}	Overlapping length of the wires i and j
n	Total number of vertices in a graph
R	Resistance
s	Safety factor
w_c	Width of a wire segment
$w_{min\_process}$	Minimum wire width determined by manufacturing process
ρ	Resistivity of a conducting material

LIST OF ACRONYMS/ABBREVIATIONS

2-D	Two-Dimensional
CAD	Computer Aided Design
CIG	Channel Intersection Graph
IC	Integrated Circuit
LDS	Layout Description Script
LP	Linear Programming
MRST	Minimum Rectilinear Steiner Tree
MST	Minimum Spanning Tree
OPAMP	Operational Amplifier
RMS	Root Mean Square
RTL	Register Transfer Level
SMT	Steiner Minimal Tree
VLSI	Very Large Scale Integration

1. INTRODUCTION AND BACKGROUND

Contrary to digital design, analog circuit design automation has not shown a significant development towards replacing manual design process. Long design-test cycles in full-custom design process leads to high design costs in analog circuit design. The objective of design automation is to reduce the time spent on the design process by moving the design effort from the human side to the computer side resulting in a decrease in the design cost and time to market [1].

Performance of analog integrated circuits (IC) is easily affected from layout parasitics. For this reason, powerful optimization techniques are carried out by design automation tools in the physical design process to get sufficient accuracy [2]. In most cases, layout synthesizers are used with commercial parasitic extractors in iterative design automation loops. These layout synthesizers can be categorized as optimization-based and template-based tools with respect to the layout generation procedures. In general, optimization-based tools generate higher quality results with a drawback of higher time complexity. On the contrary, template-based tools which are generally known as resulting lower quality layouts, have an advantage of having lower computational time by utilizing stored design knowledge. Additionally, some research showed that it is possible to increase design quality of template-based layout synthesizers considerably by using optimization-based synthesis approaches [3].

A declarative language to define layout templates for analog circuits, named Layout Description Script (LDS), and a methodology based on linear programming (LP) to instantiate a layout from a set of LDS statements are presented in [3]. In addition, tools for generating layout [4] and routing [2] templates are introduced. Once a layout template is created, time consumption of the automated iterative design loop dramatically decreases. The main steps to generate a layout template are module generation, placement, and routing [3].

Routing is the final step of template generation. The main objective of routing is to electrically connect terminals of the modules and input/output ports. Contrary to digital circuit routing, analog circuit routing has strict performance constraints. Therefore, more precise and performance oriented methods are required [1].

In this study, an analog template router which performs the routing step of a given layout template based on LDS with given constraints is introduced. The rest of this first chapter is focused on the general background of template based design. In Chapter 2, a general discussion on circuit routing is provided. In Chapter 3, the software tool developed for generating routing templates is presented. Finally, in Chapter 4, conclusions are drawn and some future research directions are discussed.

1.1. Computer-Aided-Design and Physical Design Automation

The design of integrated circuits (IC) facilitated the development of many high technology systems. Major improvements in manufacturing technologies of integrated circuits enabled production of highly complex designs. The rapid development in this field is ongoing. Minimum feature sizes are getting smaller which yields higher speeds and lower power consumption. Additionally, transistor sizes lead to more complex designs. As a consequence of this trend, a massive field called Very Large Scale Integration (VLSI) has emerged [5].

Smaller feature sizes cause many problems such as lower yield, reduced accuracy and higher power dissipation. Furthermore, parasitic effects which are caused by the non-ideal features of manufacturing materials have become more and more important. Considering the complicated nature of VLSI systems, it is a difficult job to handle all problems sufficiently and requires extensive amount of design knowledge and experience. For this reason, using computers in design operation becomes considerably appealing. The involvement of computers in the design process is called Computer Aided Design (CAD). CAD tools support the designer throughout the design process [5]. VLSI Physical Design Automation is

the research, development, and utilization of algorithms and data structures associated with the physical design process. Its main task is to search for the optimal positions of the electronic components and interconnect geometries on a 2-D plane while providing desired functionality and performance [6].

The massive progress on the VLSI field is fueled by advanced design tools. VLSI tools need to offer high computational performance to handle millions of components. The continuity of the progress on VLSI systems would not be possible without the improvement in physical design automation tools. Designers extensively use CAD tools in nearly all steps of physical design. Many of these steps are automated either partly of completely. Considering this relation, it is not surprising that physical design automation became a highly popular research field in the last twenty years. [6].

The main objective of VLSI CAD is to obtain a physical design geometry for production from high-level system specifications. The main phases of VLSI CAD are listed below (Figure 1.1).

- Design Specification
- Functional Design
- Logic Design
- Structural Design
- Physical Design [7]

In the first phase, a high level system description is formed including the methodology and design constraints from real world requirements. After that, the system behavior is modeled in functional design phase which defines the input-output relation of the system. In logic design phase, the functional description of the system is developed. This functional description is called Register Transfer Level (RTL) description and usually characterized with Boolean expressions. Logic design can be used for testing and verification. Next step is the structural design phase where the circuit design is performed based on

functional design considering speed, power, and other requirements of the system. In the physical design phase, the structural design is converted into a geometric representation ready for fabrication. This geometric information is called layout and represents all logic components in the system. Physical design is mainly divided in two steps: placement and routing. Placement involves establishing the positions of the circuit modules on the layout whereas routing, which is our main interest, concerns itself with determining the geometry of electrical interconnects between terminals [7].

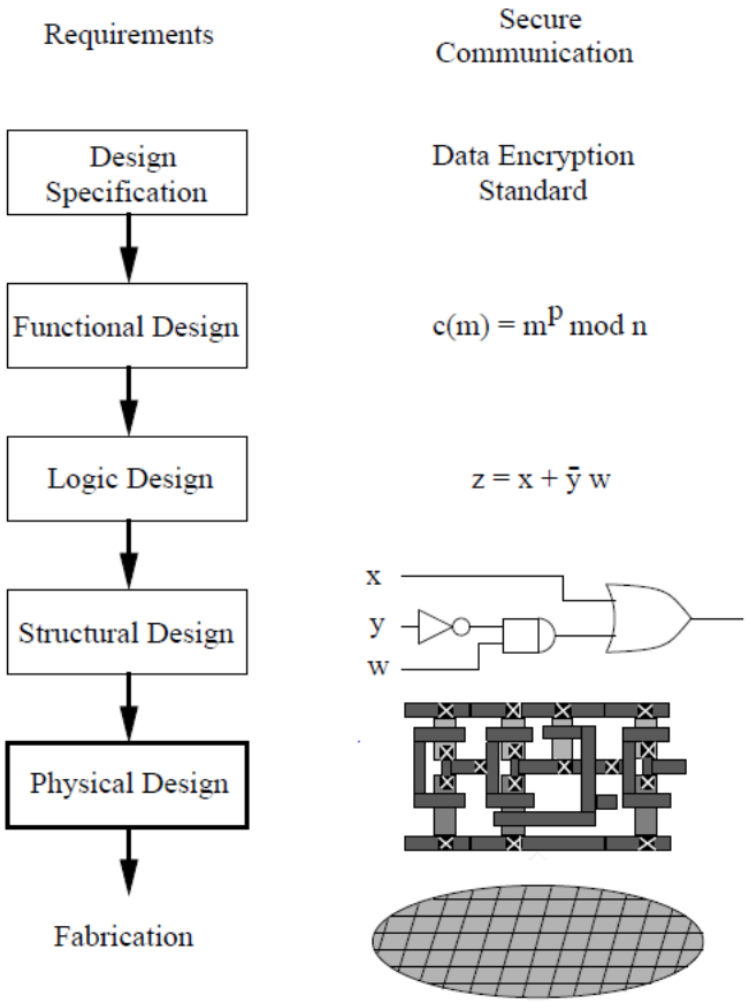

Figure 1.1. The VLSI Design Process [7].

1.2. Template Based Design

In some layout generation tools the layout is generated with the assistance of previously stored information or the information provided by the designer rather than creating the design from scratch. This information can be provided as a rough drawing of a layout or a code block written in a specific language. There are a number of reasons to adopt this approach. Many analog circuit designers do not prefer to give full control to the design tool over the design process. Hence it is very appealing to designers to use the computer software as a design aid tool generating the layout in their guidance. Moreover this approach is very suitable for porting a design to different technologies or making minor changes on the design in the same technology. For these types of problems, creating a new design is disadvantageous both in terms of cost and circuit performance. Once the designer has a good layout, it is more desirable to use the existing design template to update the design rather than creating a new one [8].

In [9] one of the first instances of template based physical design is presented. A sample layout is geometrically analyzed and is used as a template for both placement and routing in the proposed approach.

BALLISTIC [10] is an example of layout languages for representing a layout. Relative positions of modules with the interconnections can be described with this language and the design can be ported across technologies.

CAIRO [11,12] is another language for representing layouts and is a superset of C functions. A layout can be generated by compiling a code block written in this language.

A template based IC layout generation tool LAYGEN is presented in [13,14]. The layout generation process starts with a high level description of the technology independent layout template. The layout is created through this information based on the constraints defined in the template.

Another language for defining layout templates, named Layout Description Script (LDS) and a tool for generating layout instances based on linear programming is introduced in [3]. More information about LDS is given in the next section.

1.3. Layout Description Script (LDS)

Layout Description Script is a declarative language for defining analog layout templates. It is allowed to define rectangular structures and their dimensions along with placement constraints, fabrication layers, and design rules in LDS with simple statements. There are tools available to interpret LDS statements and instantiate a layout by using a linear programming based methodology. Solving a design template defined in LDS yields an optimized layout with respect to area and wire length [3].

LDS commands to define the dimensions of rectangular structures are; top(i), bottom(i), left(i) and right(i) where i is the name of the structure or '*BOUNDARY*' to refer to layout boundaries (Figure 1.2a).

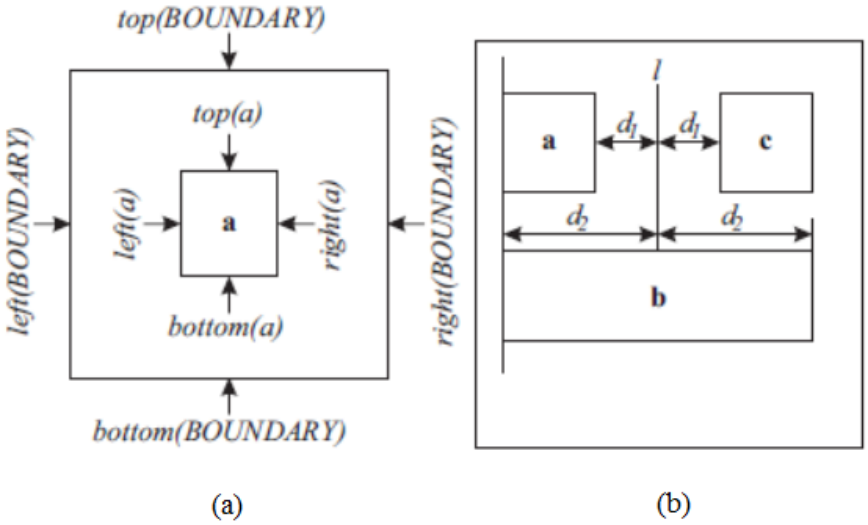

Figure 1.2. LDS (a) Structure Boundaries (b) Placement Constraints [3].

Furthermore, it is possible to form statements by using linear combinations of LDS commands and mathematical operators. A general statement is as follows [3]:

$$\alpha_1 * command_1\,(i_1) + \cdots + \alpha_n * command_n\,(i_n) \leq \beta$$

where α_k and β are real numbers and $command\,(i_k)$ is an LDS command. Accepted comparison operators are equal to (=), less than or equal to (<=), and greater or equal to (>=) [3].

It is required to control the relative placement of modules since symmetry and mismatch constraints are highly critical for analog circuits. LDS provides a full control on the placement. It is possible to construct LDS statements to deal with relative placement constraints. Statements to meet an alignment constraint and a symmetry constraint are given below:

$$left\,(a) = left(b)$$

$$\frac{right\,(b) + left(b)}{2} = \frac{left\,(c) + right(a)}{2}$$

The first statement implies placement of the left edges of modules a and b to the exact same coordinate in the layout. Likewise, the second statement results in modules a and b to be placed vertically symmetrical with respect to the axis in the middle of module b. The effect of these two statements in the layout is visualized in Figure 1.2b [3].

To instantiate a layout from a set of LDS statements, the LDS code block is formulated as a Linear Programming problem and solved with a cost function which minimizes the total width/height of the circuit. As the LP in 2-D space is a NP-hard problem, an approach to solve the problem where vertical and horizontal solutions are found separately is adopted in the optimizer [3].

2. ROUTING

Routing process is the part of the physical design step with the task of determining the position of the interconnects between preplaced geometrical structures, as defined in a point-to-point manner in the circuit netlist. The interconnect consisting of wires and vias of a certain net, is also called routing of the net [5].

A routing model and a routing algorithm together forms a routing approach. The routing model defines how the information is stored and represented whereas the routing algorithm is the technique for searching the solution. Choosing the routing model and the algorithm are strongly related and the best alternative depends on the problem definition, performance criteria and the limitations [5].

2.1. Classification of Routing Approaches

2.1.1. Routing Hierarchy

Routing approaches can be classified into single-step routing and two-step routing according to the routing strategy.

2.1.1.1. Single-Step Routing. Single-step routing approach is where the actual geometric layout of the wire segments are determined in the first place. This strategy can also be referred as area routing algorithm. The major disadvantage of single-step routing is the absence of a global perspective which may cause many problems to arise, especially in large scale circuits [5].

When performing a single-step routing approach, previously routed nets constitute obstacles for the other nets. Therefore, it becomes difficult to find a near-optimal solution. Computational complexity of such approaches are very

high and they require rip-up and re-route strategies. Furthermore, it is hard to estimate the output quality of the routing algorithm and algorithmic parameters are often determined experimentally. Although using single-step routing approaches are not very likely to produce high quality layouts for large scale circuits, the outcome can be quite satisfactory when the size of the routing area is limited in a single phase [5].

2.1.1.2. Two-Step Routing. In the two-step routing approach, the problem is divided into two sub-problems; a global routing phase and a succeeding detailed routing phase. In this strategy, the routing regions are identified in the chip area. In the global routing phase, the nets are assigned to the routing regions on the interconnect path and a coarse routing solution is obtained (Figure 2.1a,b). In the detailed routing phase, the exact geometrical positions of wire segments are determined for each routing region (Figure 2.1c) [5, 15].

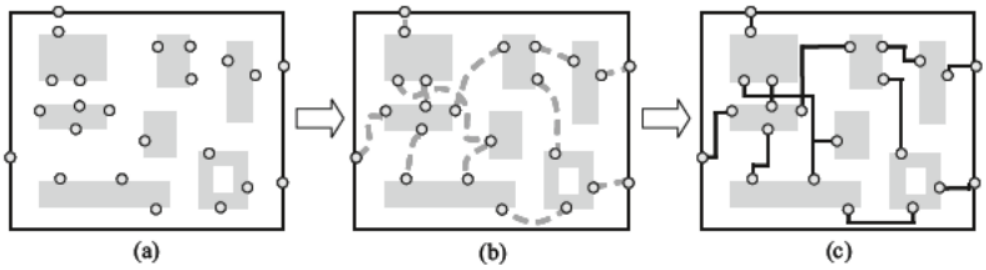

Figure 2.1. (a) A Given Placement with Blocks and Pins (b) Global Routing (c) Detailed Routing [16].

Splitting the complex routing problem into two conceptually simpler problems comes out as a significant advantage of this strategy. The impact of the algorithmic features on the layout quality is easier to evaluate and the design of the algorithm becomes simpler [5].

2.1.2. Layout Representation

2.1.2.1. Grid Representation. It is required to represent the layout with proper data structures to apply any routing algorithm with the purpose of solving the routing problem. In general, layouts can be represented with regular and irregular grid models [5].

Regular grid model, also referred to as uniform grid is the most basic form of grid representation, where the routing area is represented with an evenly formed grid. However, this approach has certain disadvantages. Using a coarse grid (Figure 2.2b) may be insufficient to represent the layout adequately resulting in a lower optimization performance, whereas using a fine grid (Figure 2.2a) may require too much memory [8].

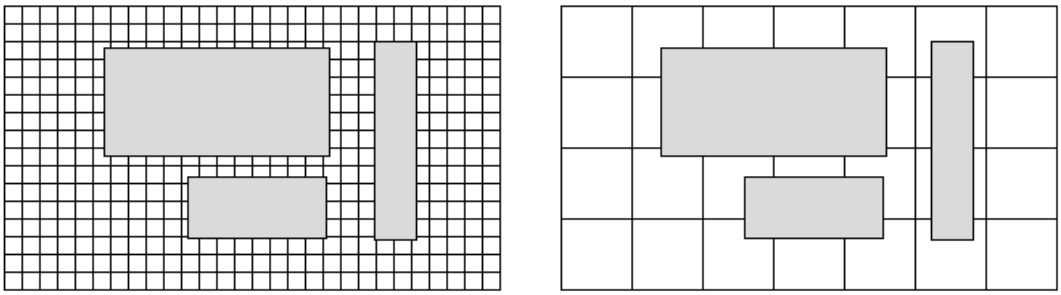

Figure 2.2. (a) Fine Grid (b) Coarse Grid.

The grid representation of the layout may be constructed non-uniformly (Figure 2.3). This model is much more efficient in terms of memory management compared to the regular-grid model. This approach allows using a smaller grid size in the dense regions. An irregular grid is formed by extending the module boundaries in the layout in 2-D plane [8].

2.1.2.2. Graph Representation. Due to excessive memory requirements and algorithmic disadvantages of grid models, a graph based routing model is highly favorable. Graph theory is widely used in electronic design automation. It is a

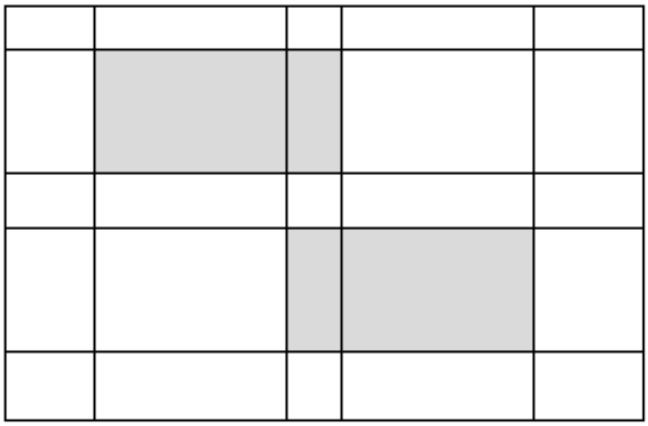

Figure 2.3. Non-uniform Grid.

fundamental concept in computational science and has the advantage of having numerous computationally effective algorithms. It has been shown that graph theory based approaches are very advantageous in terms of memory efficiency and optimization performance for routing problems [5].

It is possible to construct a grid graph from grid representation of a layout which enables to use of graph theory techniques in the routing problem. A grid graph is represented with G(V,E) where V is the set of routing regions and E is the set of edges connecting routing regions (Figure 2.4) [16].

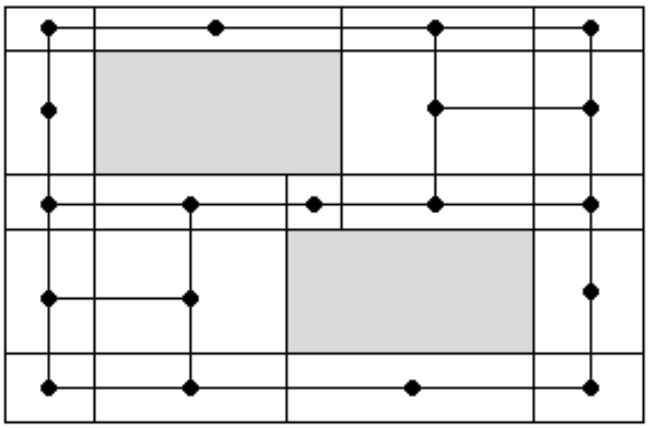

Figure 2.4. Grid Graph.

Channel Intersection Graph (CIG) (Figure 2.5a) is another graph representation of a layout. In CIG, the spaces between adjacent modules correspond to routing channels. Crossing points of the routing channels are regarded as vertices and the edges are the routing channels forming an undirected weighted graph [7]. Channel Intersection graph is a general and a very accurate model for global routing. An extended CIG (Figure 2.5b) is constituted by representing the pins as vertices in the graph to utilize them for routing algorithms. [6]

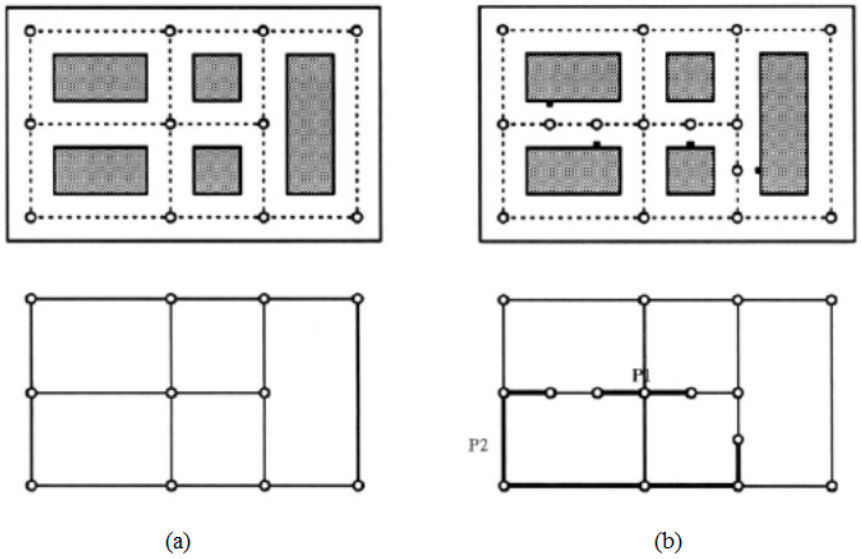

Figure 2.5. (a) Channel Intersection Graph (b) Extended Channel Intersection Graph [6].

2.1.3. Layer Assignments

Routing models are classified into reserved and unreserved layer models with respect to layer assignments. In the unreserved layer model, there is no restriction in placing a wire segment on any layer, whereas in the reserved layer model, certain types of wire segments are restricted to be placed on specific layers. The most common form of the reserved layer model is the VH model where routing only along a particular direction is allowed for each layer. For example, the first and the second metal layers can be reserved to only horizontal and vertical wires,

respectively. Reserved layer model is widely used by existing routers as it reduces the complexity of the routing problem considerably. However using a reserved layer model results in lower quality routing solutions [15, 16].

2.2. Routing Considerations

2.2.1. Symmetric Routing

Analog differential circuits are commonly expected to be designed symmetrically to meet design specifications such as offset, differential gain, and noise [8]. Therefore, it is critically important to match the parasitics of differential signal paths. Symmetrical routing (Figure 2.6) is required to match parasitics on such nets [15].

In order to route two signal nets symmetrically, it is also necessary for the signal terminals to be placed symmetrically. To electrically connect two nets symmetrically, the net pair can be routed simultaneously. Another way to achieve this requirement is routing one net and then mirroring the path with respect to the symmetry axis [8].

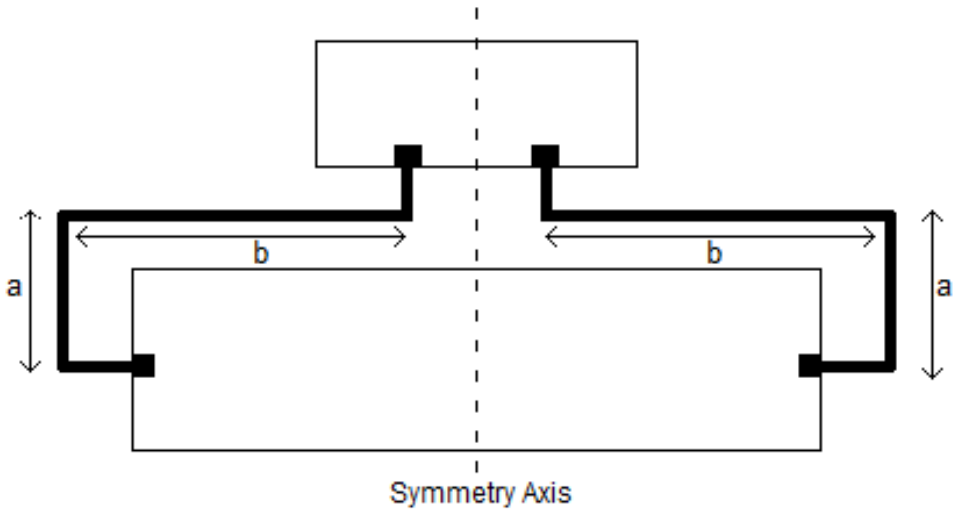

Figure 2.6. Symmetric Routing.

2.2.2. Performance Constraints

Parasitic effects introduced to the circuit by the interconnect have a negative influence on the performance of the circuit. The interconnect parasitics are caused by non-ideal characteristics of circuit elements and added to the circuit as parasitic capacitance, inductance, and resistance. The effect of the interconnect parasitics is calculated using an equivalent circuit model of the wire. Each wire segment adds a series of parasitic resistances and capacitances. Since the influence of the parasitic inductance caused by the interconnect is generally negligible, we will not deal with it in the rest of this paper. The influence of these effects on the performance of the circuit has to remain within the performance specifications [15].

2.2.2.1. Crosstalk. Minimum features sizes decrease as the chip manufacturing technology advances. This results in wires to be placed closer so as to reduce the interconnection delay and area consumption. Therefore, the coupling capacitance becomes larger than self capacitance. For example, it is shown that the coupling capacitance constitutes 70%-80% of the total wiring capacitance even in 0.25μm technology. Consequently, crosstalk effect, which is mostly caused by the coupling capacitance (Figure 2.7), becomes a significant issue for signal integrity [16].

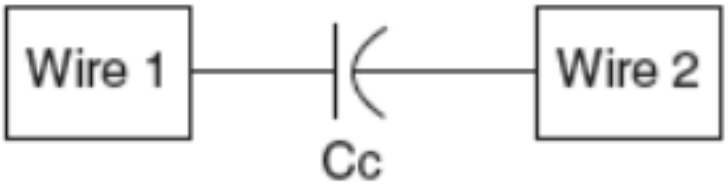

Figure 2.7. Coupling Effect [16].

The crosstalk between two wires is mostly induced by the coupling capacitance and related to the relative positions of the wires. The coupling capacitance between the wires that are perpendicular to each other is insignificant compared to the coupling capacitance between parallel wires.

Actually, it is possible to estimate the crosstalk by only taking neighboring parallel wires into account [16].

An approximation for the coupling capacitance c_{ij} between the wires i and j is given in Equation 2.1 [16].

$$c_{ij} = a\frac{l_{ij}}{(d_{ij})^k} \tag{2.1}$$

Here the crosstalk is proportional to the technology constant a and the overlapping length of wires l_{ij} and inversely proportional to the distance between wires d_{ij}. k is a constant between 1 and 2 and close to 2 (Figure 2.8) [16].

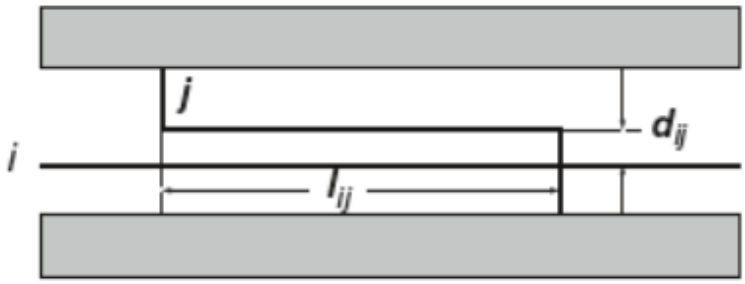

Figure 2.8. Capacitive Crosstalk Computation Between the Wires i and j [16].

Crosstalk effect can originate from the coupling between the wires either in the same layer or in different layers. The parasitic capacitance caused by the coupling within the same layer can be prevented by adding extra space between the wires or using power lines shielding. A strategy for avoiding the parasitic capacitance across different layers is to place the wires along different directions in adjacent layers [6].

2.2.2.2. Wire Resistance. The effect of interconnect delay was regarded as negligible in the past. However RC delay of the interconnect became a vital concern in modern ICs. As the feature size reduces and chips become larger,

interconnect resistance which is a major factor on determining the RC delay, gains significance [6]

The resistance of a wire segment R can be calculated using Equation 2.2.

$$R = \frac{\rho l_c}{h_c w_c} \tag{2.2}$$

where ρ is resistivity of the conducting material, l_c, h_c and w_c are the length, thickness and width of the wire segment, respectively [6].

The power consumption and signal delay usually grow with added wire length. Moreover, area consumption is an important constraint in chip design and the area occupied by routing is roughly proportional to the total wire length. A large chip area increases the total cost by adding extra costs due to fabrication, yield, and reliability issues. Considering these effects, minimizing the total wire length is a fundamental objective in the routing problem [7].

2.2.3. Electromigration Avoidance

Interconnections in analog circuits are subject to a wide range of currents contrary to digital circuits. Some interconnections may carry extremely large currents. Therefore, the current levels have to be considered when determining the width of the interconnections. Inadequate wire widths may result in failures because of electromigration effects [15, 17]. The current levels of the interconnects have to be determined in order to achieve an electromigration reliable design [18]. However, the current level flowing through a wire segment depends on the wiring topology. This information is easy to obtain for two-terminal nets by circuit level simulation. However for multi terminal nets, physical design has to be taken into account in the analysis [17].

The minimum wire width w_{min} is can be found for a given temperature T_{ref} by using Equation 2.3 [17].

$$w_{min} = max \begin{cases} \frac{I_{eq}.s}{d_{Layer}.J_{max}(T_{ref})} \\ \\ \frac{|I_{max}|s}{J_{peak\_Layer}.d_{Layer}} \\ \\ w_{min\_process} \end{cases} \tag{2.3}$$

where I_{eq} is the root mean square (RMS) current on this path, s is safety factor ($s \approx 1.1, 1.2, ...$), d_{Layer} is the thickness of routing layer, $J_{max}(T_{ref})$ is the maximum current density allowed by this process for temperature T_{ref} ($J_{max}(150°C) \approx 1$ .. 2 mA/μm2), I_{max} is the maximum current on this path, J_{peak_Layer} is the layer dependent peak current density (process dependent), $w_{min\_process}$ is the minimum wire width determined by manufacturing process [17].

2.3. Routing Techniques

2.3.1. Maze Routing

Maze routing, which is also known as Lee's algorithm is one of the most widely used routing algorithms. The algorithm consists of wave propagation and succeeding retracing phases (Figure 2.9a,b). In the propagation phase, grid cells are numbered starting from the source node S. In every iteration, grid cells that are adjacent to the previously numbered cells are marked increasingly in a wave propagation manner. Once the wave reaches to the target node T, this phase ends. After that the shortest path between S and T can be found by tracing the path backwards for decreasing numbers [8, 16].

It is guaranteed to find a path which is also the shortest path from source to target by performing maze routing algorithm if such a path exists. Nevertheless, this algorithm has significant drawbacks. The space and time complexity of the algorithm is *O(mn)* where m and n are the number of vertical and horizontal grid

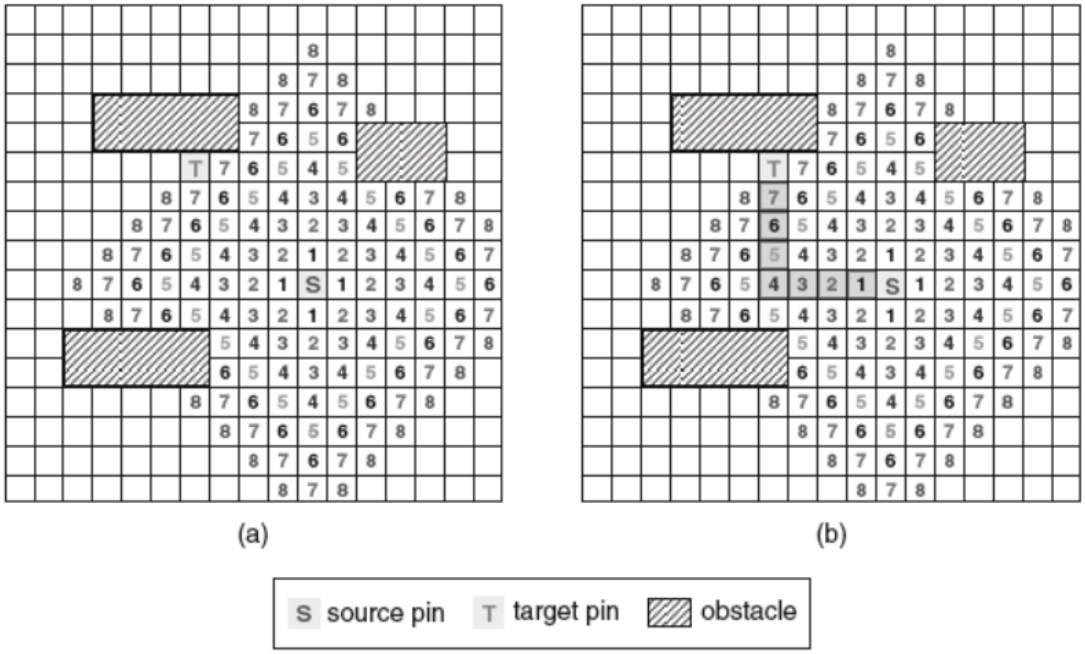

Figure 2.9. (a) Wave Propagation Phase (b) Retracing Phase [16].

cells in the problem space. Additionally the algorithm requires a massive amount of memory which makes it impractical to use with large scale circuits [8, 16].

2.3.2. Line Expansion Router

Line expansion routers provide an improvement in run time and memory requirements by representing the routing space by line segments rather than a grid structure. The line search algorithm (Figure 2.10) proposed by Mikami and Tabuchi [19] creates four level-0 line segments passing through each of the source S and target T nodes vertically and horizontally. The line segments are extended to either an obstacle or the circuit boundary. After that each grid point on the line segments are promoted as base points and new line segments are created passing through each base point perpendicularly. Once a line segment generated from source node crosses a line segment generated from target node, a path between S and T can be found by tracing the line segments [8, 16].

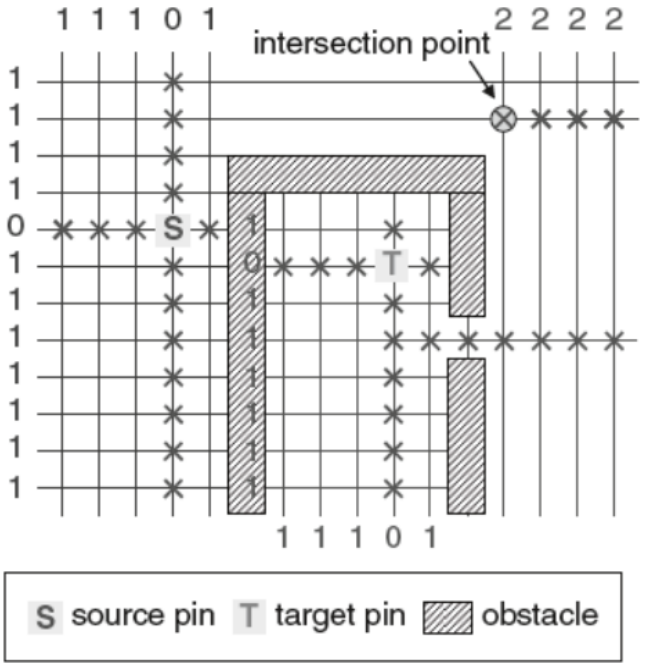

Figure 2.10. Line-Search Algorithm.

Beside the advantages in time and memory consumption, line search algorithm lowers the number of unnecessary bends on the path. It also guarantees to find a path between the source and the target if one exists. However, finding the shortest path is not guaranteed. Furthermore, as the routing process gets more complex, the computation time and the memory requirements of this algorithm increases drastically [8, 16].

2.3.3. Channel Routing

Channel routing is a technique for routing nets in rectangular channels which is commonly used in digital design. Even though it does not directly deal with layout parasitics, channel routing also has been favored by a number of analog routers.

Routing channels are identified by partitioning the routing region. There are different strategies for the decomposition of the routing region (Figure 2.11a,b).

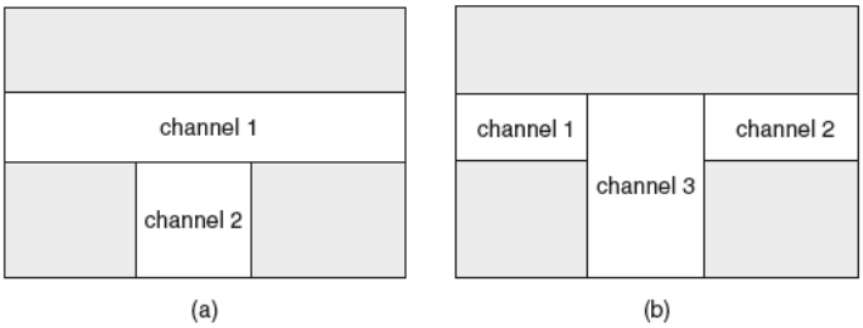

Figure 2.11. Two Ways of Routing Region Decomposition [16].

The channel height is determined by the routing process. The decomposition topology and the order of routing the channels directly influences the performance of the routing. Fixing the wiring in a channel may lead to conflicts with other channels due to congestion constraint. For example in Figure 2.11a after routing channel 1, it is not possible to expand channel 2 which may result in an infeasible solution. Fortunately it is always possible to find a channel ordering while avoiding conflicts [8, 16].

Channel routing technique implies using a reserved layer model where fabrication layers are restricted to be routed along horizontal or vertical directions. There are a variety of effective methods for solving the channel routing problem such as Left-Edge Algorithm [20] and Dogleg technique [21]. A solution to the channel routing problem is shown in (Figure 2.12) [8, 16].

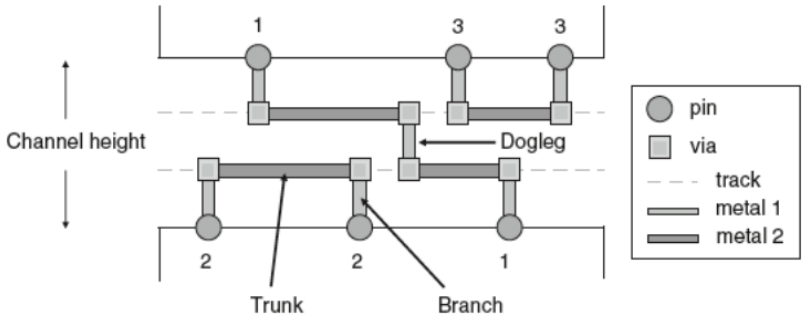

Figure 2.12. Channel Routing [16].

2.3.4. Dijkstra's Algorithm

Dijkstra's algorithm [22] is a methodology for optimally solving a single pair shortest path problem in a graph (Figure 2.13). The algorithm finds a path with the lowest cost from a given source vertex to all other vertices. The algorithm can be stopped as soon as the shortest path to a destination is determined to find the shortest path between a vertex pair. Dijkstra's algorithm reaches the optimal solution in time $O(n^2)$ where n is the total number of vertices in the graph [6].

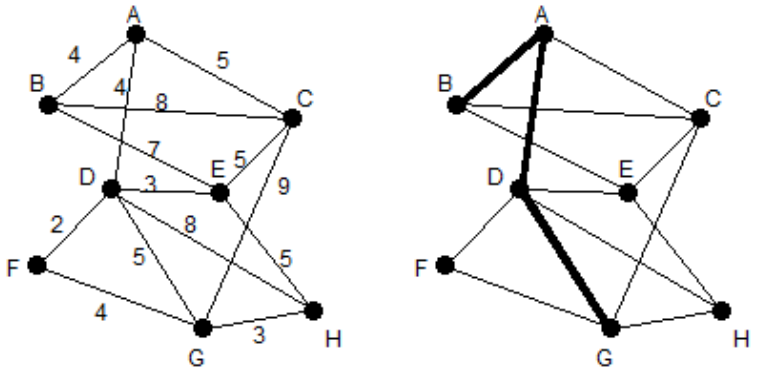

Figure 2.13. Single Pair Shortest Path Between Vertices B and G.

The algorithm starts by assigning initial distance values which represents the distance from the source node to each vertex. The source node is assigned zero and all other nodes are assigned infinity. All nodes except the source node are marked as unvisited. Source node is selected and the distance values of each of its neighbors are updated. To find the new distance; the sum of the distance of the selected node and the weight of the connecting edge is calculated for each neighbor. The new distance is the minimum of the calculated value and the current distance of the vertex. After doing this for each neighbor, the selected node becomes a visited node and its distance becomes fixed. Finally, the unvisited node with the smallest distance value becomes selected and the algorithm continues. When the distance value of the destination node is fixed, the algorithm stops and the shortest distance is found.

2.3.5. A*-Search Routing

A*-Search [23] is a general graph search algorithm. It can be used to solve the single pair shortest path problem more efficiently than Dijkstra's algorithm. In A* algorithm, the cost of a path x is calculated with the function $f(x)=g(x)+h(x)$ where $g(x)$ is the cost from the source node to current node of x, and $h(x)$ is the estimated cost from the current node of x to the target node. In each iteration, the node with the lowest cost becomes selected [16].

A notable property of $h(x)$ is that it never overestimates the minimum cost from current node to the destination node. With this property, optimality of the result is granted. For example it is convenient to select $h(x)$ as the Manhattan distance for rectilinear routing as it is the smallest possible distance [16].

Dijksta's algorithm is a special case of A* algorithm where $h(x)$ is zero. A* routing is a widely used technique for the VLSI global routing problem [16].

2.3.6. Steiner Routing

The route search algorithms discussed so far are concerned with connecting two terminals. Such algorithms have to be performed sequentially for each pair of pins if the net being routed consists of more than two terminals. In such manner, the quality of the solution strongly relies on the order of routing terminal pairs. The problem of connecting more than two terminals optimally with possibly adding intermediate nodes is called the Steiner problem and is an NP-complete problem. The intermediate nodes are referred to as Steiner points and the optimum solution is called a Steiner minimal tree (SMT) (Figure 2.14a). A special case of the Steiner problem is minimum rectilinear Steiner tree problem (MRST) where only horizontal and vertical directions are allowed for connections (Figure 2.14b) [8,16].

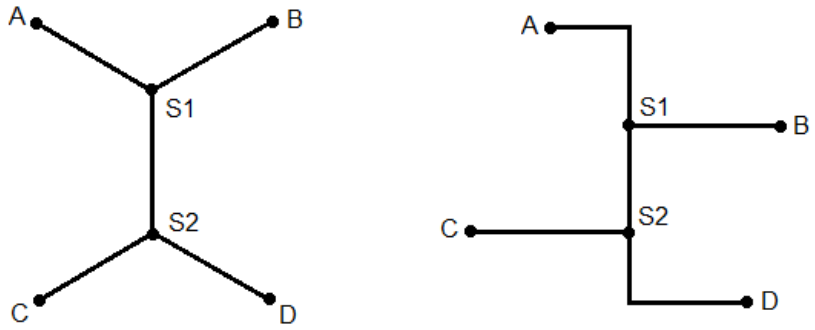

Figure 2.14. (a) SMT and (b) MRST.

Steiner trees have also been studied broadly in the context of weighted graphs apart from Euclidean space. Given G=(V,E) is an edge weighted graph and a set of nodes S, the minimum subset T of the edges connecting all nodes in S constructs a Steiner minimal tree [24].

Minimum spanning tree (MST) problem in a graph is a special case of SMT. Given G=(V,E) is a connected undirected graph, a minimum spanning tree is a subgraph of G connecting all vertices while minimizing the total edge weights (Figure 2.15) [24]. A Minimum Spanning Tree of a graph can be computed in polynomial time by using Prim's [25] or Kruskal's [26] algorithms.

The Steiner minimal tree problem is shown to be NP-complete and no polynomial time algorithm to solve this problem exists. That's why many heuristics are proposed to efficiently obtain near-optimal solutions [27]. A fast heuristic algorithm suggested by Kou, Markowsky, and Berman [28] producing no more than $2\left(2-\frac{1}{l}\right)$ times optimal results where l is the number of leaves in the optimal tree is presented in later in this chapter.

2.3.6.1. Minimum Spanning Tree Problem. Let *G=(V,E)* be an undirected edge weighted graph with real edge costs. The Minimum Spanning Tree is an edge

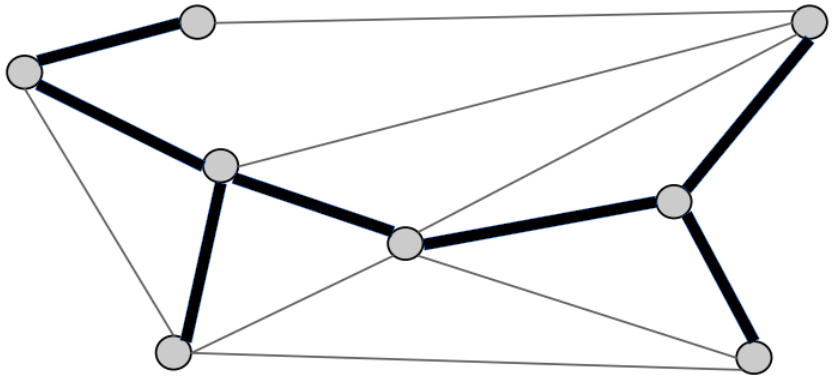

Figure 2.15. Minimum Spanning Tree.

selection problem where the set of edges $E' \subseteq E$ are selected such that E' is a tree spanning all nodes in V and the total cost of the edges in E' is minimum over all such trees [6].

The Jarnik-Prim algorithm is an easy and efficient way to construct a minimum spanning tree. Prim's algorithm starts by selecting an arbitrary node from V. The tree T_{MST} grows by adding nodes and their incident edges to the tree one by one. At each iteration, an edge is selected such that it is the minimum weight edge among the edges with strictly one endpoint in T_{MST}. This procedure prevents cycles in the tree. The time complexity of Prim's algorithm is $O(n \log n + m)$ with the use of Fibonacci heap priority queue [23].

2.3.6.2. Distance Network Heuristic. Let $G=(V,E,d)$ be an undirected distance graph, where $V=\{v_1, v_2,\ldots,v_n\}$ is the set of vertices in G, $E \subseteq \{\{v_i,v_j\} | v_i \in V, v_j \in V \text{ and } v_i \neq v_j\}$ is the set of edges in G and $d:E \rightarrow R$ is a distance function which maps E into the set R of nonnegative numbers. Given a set of Steiner points $S \subseteq V$, consider the complete undirected distance graph $G_1=(V_1,E_1,d_1)$ constructed from G and S in such a way that $V_1=S$ and for every $\{v_i,v_j\} \in E_1$, $d(\{v_i,v_j\})$ is equal to the distance of the shortest path from v_i to v_j in G. The heuristic algorithm is as follows [28]:

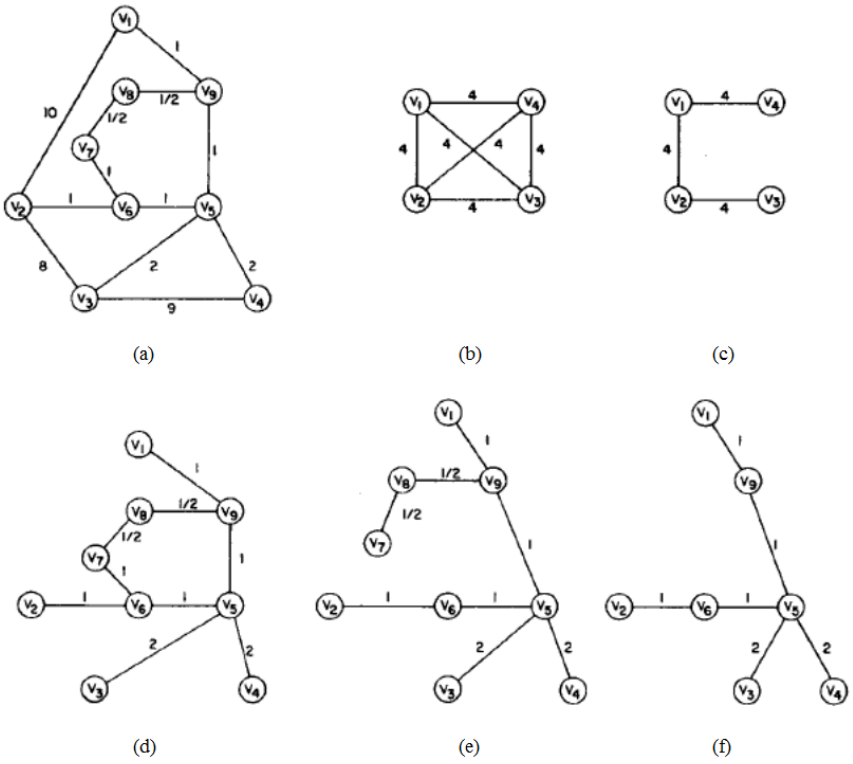

Figure 2.16. Distance Network Heuristic [28].

- Step 1: Given $G=(V,E,d)$ (Figure 2.16a), construct the distance network $G_1=(V_1,E_1,d_1)$ (Figure 2.16b).
- Step 2: Determine a minimum spanning tree T_1 of G_1 (Figure 2.16c).
- Step 3: Replace each edge in T_1 with the corresponding shortest path in G and build the subgraph G_s of G (Figure 2.16d).
- Step 4: Determine a minimum spanning tree T_s of the subgraph G_s (Figure 2.16e).
- Step 5: Remove all degree-1 non-terminal nodes and their incident edges from the tree T_s one by one. The remaining tree is a Steiner tree (Figure 2.16f). [28]

The worst-case time complexity of this algorithm is $O(SV^2)$ and can be reduced to $O(S(e+VlogV))$ for sparse networks if Fibonacci heaps are used [27].

2.3.7. Group Steiner Trees

Steiner trees are widely used in VLSI routing. Most of the research on this subject assumes that each terminal of the net is present as a single port on the layout. Nevertheless, a terminal can consist of many electrically equivalent ports on the layout. The problem of determining a minimum weight tree connecting multi-port multi-terminal nets is called the Group Steiner Problem. The formal definition of the problem is given below [29].

Let $G=(V,E)$ be a weighted graph and $N=\{N_1,...,N_k\}$ be a set of k groups of nodes, finding a minimum-cost Steiner tree in G such that the tree contains at least one node from each group N_i is called the group Steiner problem. The group Steiner problem is NP-complete as it is a direct generalization of the NP-complete Steiner Problem [29].

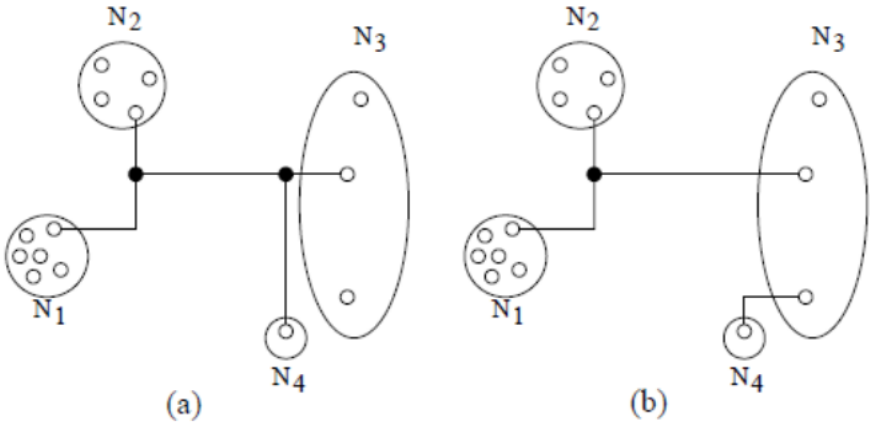

Figure 2.17. Group Steiner Tree Problem (a) Weak-Connectivity Version (b) Strong-Connectivity Version [29].

The group Steiner problem is classified by strong connectivity and weak connectivity versions which shows somehow different characteristics. In strong connectivity version (Figure 2.17a), different ports of the same terminal are implicitly connected. This version allows the solution to consist of distinct trees

forming a forest. It is possible to solve strong group Steiner problem efficiently using an existing graph Steiner tree heuristic within the same error bound. In order to achieve this, a new instance of the graph should be created by setting edge weights between the nodes in the same groups to zero [29].

The weak connectivity group Steiner problem (Figure 2.17b) is where the intra-group edges must be a part of the solution. In this context, the solution must be a tree including at least one node from each group of nodes for all terminals [29]. Contrary to the strongly connected version, it is shown that it is unlikely to solve the weakly connected group Steiner problem with the worst-case error ratio bounded by a constant in polynomial time [27].

3. LDS Template Router

3.1. The Routing Problem

As stated earlier, template based design schemes provide fast and reliable designs for many applications. A template based optimization approach requires generating layout templates. The aim of this project is to develop a software tool to create routing information for layout templates coded in LDS.

The software tool takes an LDS layout template with placement information and electrical connectivity information as input. The output of the software is a flexible routing information for the template coded in LDS. The output template with the routing information has to be able to instantiate layouts with varying transistor dimensions so that the template can be used by design automation tools. This property removes the need for routing the circuit layout in each iteration in a closed optimization loop.

This specific router differs from other routers by the nature of its input and output. The input file holds the relative placement information of the circuit elements rather than providing fixed coordinates in the layout. Likewise, the output file consists of relative positions of the interconnect and is self-adapting for the variations in dimensions of the circuit elements. Therefore, the routing model has to be chosen by considering the specific needs of this approach.

A sample circuit is used to demonstrate the routing process in the rest of this thesis. In Figure 3.1, schematic diagram of the sample OPAMP circuit is depicted. The placement of the modules that is going to be used in the sample template can be seen in Figure 3.2.

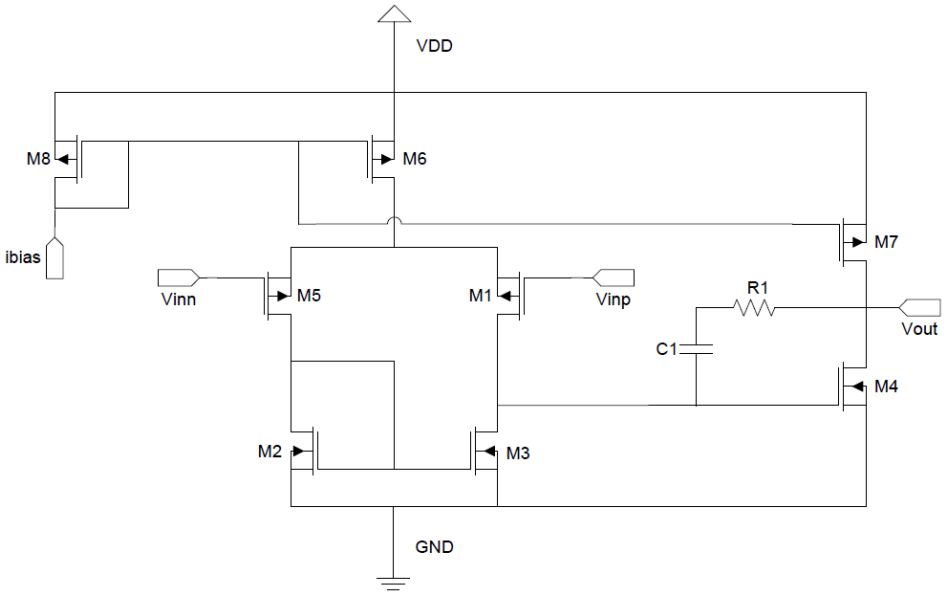

Figure 3.1. Schematic Diagram Of the Sample Circuit.

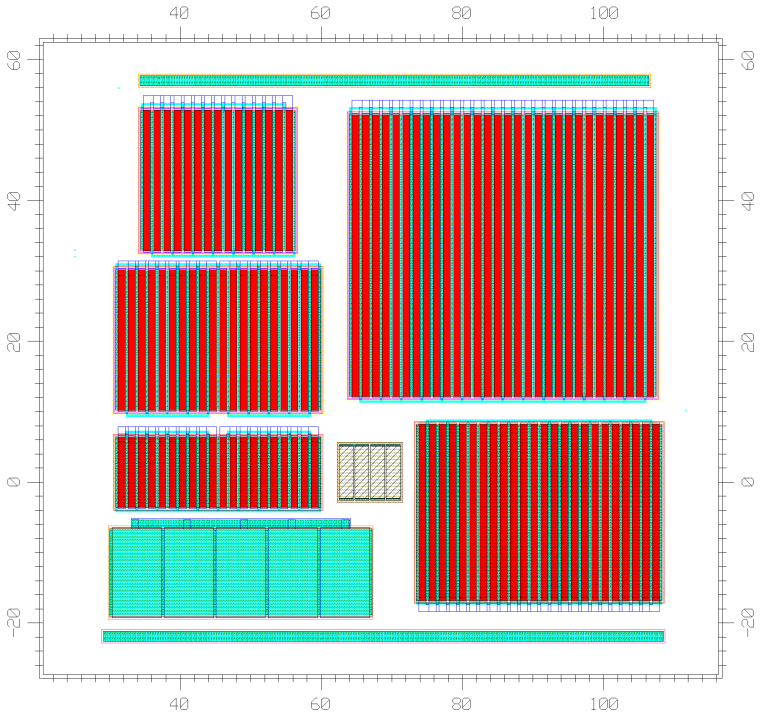

Figure 3.2. Layout Of The Sample Circuit Without Routing.

3.2. Analyzing the Routing Space

In some applications, over-the-cell routing can be acceptable. However, generally speaking, over-the-cell routing is not desirable for the following reasons. Firstly over-the-cell routing is expensive in terms of parasitics as it requires routing in higher level metal layers. In addition, if intellectual property blocks are used in the design, it is best to avoid crossing wires on such blocks as the module characteristics are unknown. In this project, wiring is only allowed between blocks. Layout modules and ports are considered as obstacles in the routing space. First step of the routing process is to analyze the layout geometry and determine the routing channels.

An initial instance of the LDS definition of the layout template without routing has to be created to analyze the layout space. Output of the sample layout template is shown in (Figure 3.3).

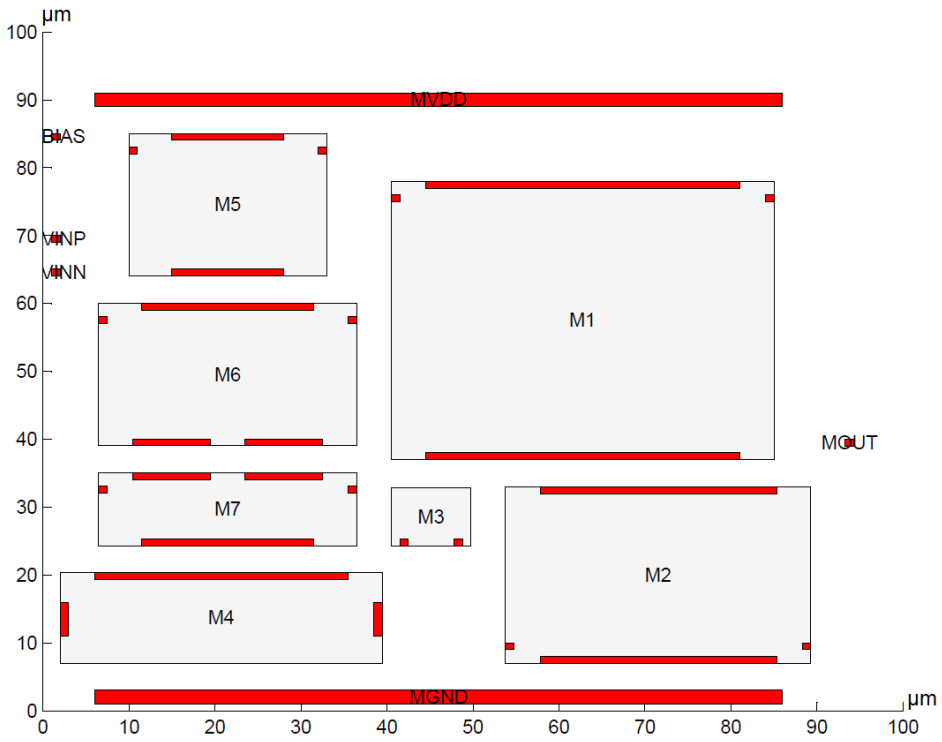

Figure 3.3. Output of the Sample Layout Template.

The input/output ports of the layout are BIAS, VINP, VINN, MVD, MGND and MOUT.

Module M1 is the transistor M7 in circuit schematic.

Module M2 is the transistor M4 in circuit schematic.

Module M3 is the resistor R1 in circuit schematic.

Module M4 is the capacitor C1 in circuit schematic.

Module M5 is the transistor M6 in circuit schematic.

Module M6 is the stacked layout element of transistors M1 and M5 in circuit schematic.

Module M7 is the stacked layout element of transistors M2 and M3 in circuit schematic.

Red sections in the layout represent metal layer-1. Metal layers are considered as ports and eligible for electrical connections.

To be able to work on the layout space, it has to be split into discrete regions. The LDS router forms an irregular grid structure by extending each module and port boundaries along the layout space. Port terminals are also broken down into smaller pieces if they are larger than a threshold in either x or y dimension. Each partition of the port terminals are also contribute to the grid structure, thereby they can be represented properly in the grid.

The irregular grid represents the layout adequately. Each partition of the grid is marked as obstacle, port terminal, or empty region accordingly.

The next objective of the router is to determine routing regions and define a channel intersection graph. For this purpose every corner of the obstacles are marked as nodes. These corner nodes form the basis of creating the CIG. Each corner node is extended until it hits an obstacle or reaches the layout boundary. A new node is created where the extension of a corner node hits an obstacle or the extensions of corner nodes intersect (Figure 3.4).

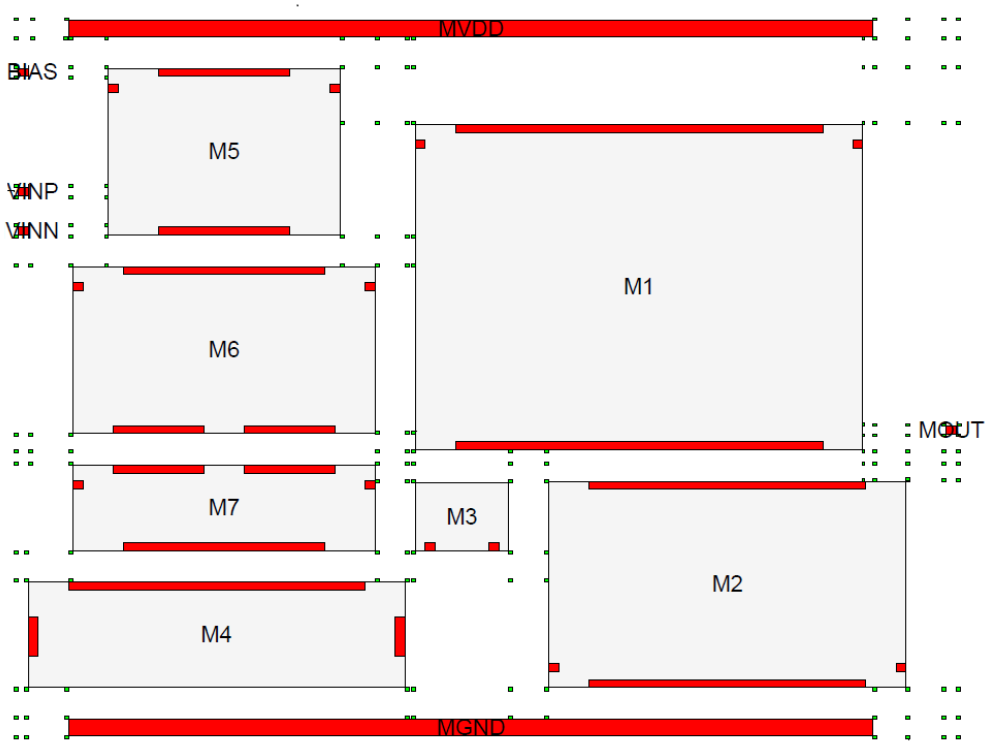

Figure 3.4. Initial Nodes on the Layout.

Two nodes have a connection if there is a direct path between them. Two nodes are neighbors if they have a direct connection. A node can have at most one neighbor to a particular side. The connections define routing channels. However, it can be seen in Figure 3.4 that an actual routing channel is represented by more than one connection. Appropriate nodes have to be merged to represent the layout more conveniently.

Nodes are grouped in merging process. The key point is that a node group cannot have more than one connection to another node group on each of the four

sides at the end of node merging. In other words, for example, if two nodes with the same x coordinates are merged vertically, their neighboring nodes on the right side must also merge if they both have one. Figure 3.5 illustrates this rule, if A and B merge, A' and B' must also merge. The rules of merging process have to guarantee this rectilinearity property.

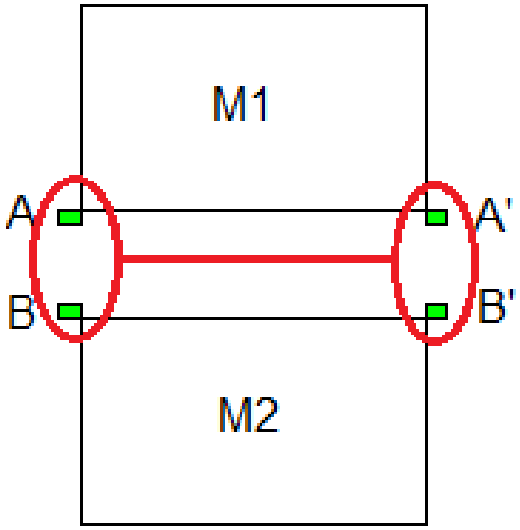

Figure 3.5. Node Merging Illustrated.

Rules of node merging are listed below:

- Rule 1: Every single node is considered a node group in the first step of the merging process and node groups are merged in pairs iteratively. When node groups are merged, every single node in these node groups are grouped together. Node groups are first merged horizontally. When there are no node groups that can merge horizontally, merging continues vertically.
- Rule 2: Node groups can merge if there is a direct connection between them.
- Rule 3: Node groups merge starting from the closest node group pair which has no merging rule violation.
- Rule 4: Node groups cannot merge if the distance between them is larger than a certain threshold.

- Rule 5: Every node originates from extending at least one side of a module. Top and bottom sides extend horizontally, left and right sides extend vertically. When two nodes are merged, we associate the side extensions which these nodes originate from. When node groups are merged vertically, horizontal extensions are associated. When node groups are merged horizontally, vertical extensions are associated. For example let *n1* and *n2* be two nodes originating from extending top side of M1 and bottom side of M2 respectively. If *n1* and *n2* merge vertically, extensions of the top side of M1 and bottom side of M2 are put in the same set permanently.
- Rule 6: When associating side extensions, opposite sides of the same module cannot be combined in the same set.

Rule 6 implies that A and A' in Figure 3.5 cannot merge not to lose a routing channel. This rule also guarantees rectilinearity principle stated previously. Let us examine Figure 3.6a. If Node Group 2 and Node Group 3 are merged together, we lose a routing channel(left side of M4); if not, rectilinearity principle is violated (Figure 3.6b). That's why if nodes in Node Group 1 are merged, we must prevent composing Node Group 3.

In Node Group 1, there are nodes originating from top side of M4 (colored with blue) and top side of M3 (colored with yellow). Rule 5 tells us to associate blue and yellow node lines in the same set. In this case, we would not be able to merge the nodes in Node Group 3 in accordance with Rule 6. Because there are nodes originating from bottom side of M4 in Node Group 3(colored with red). Since blue and yellow node lines are in the same set, red node line cannot be added to the same set with blue node line (Because blue and red node lines are extensions of opposite sides of M4). Appropriate channel intersections graphs can be constructed as in Figure 3.6c and Figure 3.6d.

Resulting nodes from the node merging process and corresponding connections to construct the CIG for the sample layout are illustrated in Figure 3.7 and Figure 3.8 respectively.

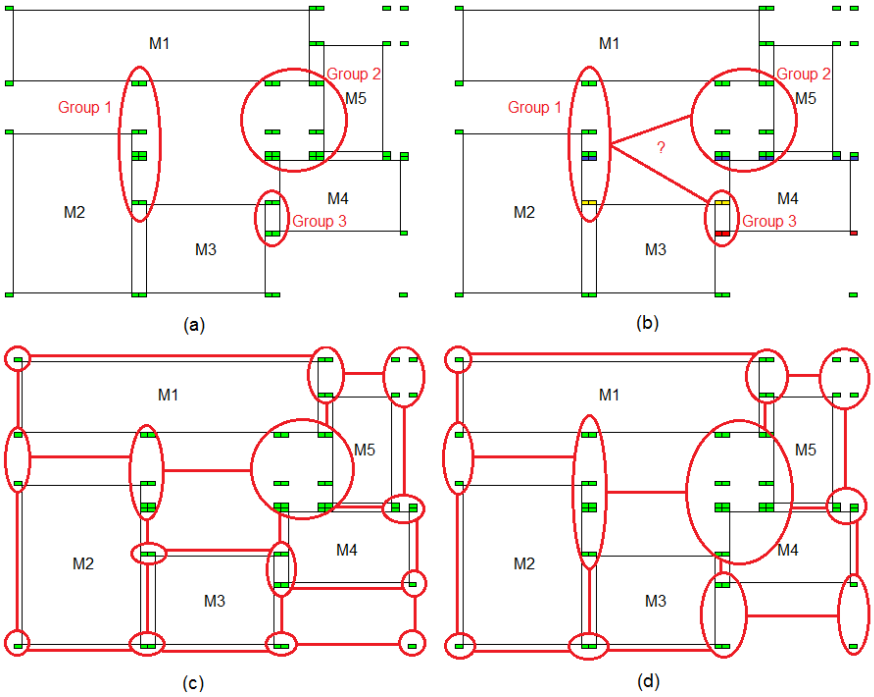

Figure 3.6. Node Merging Rules.

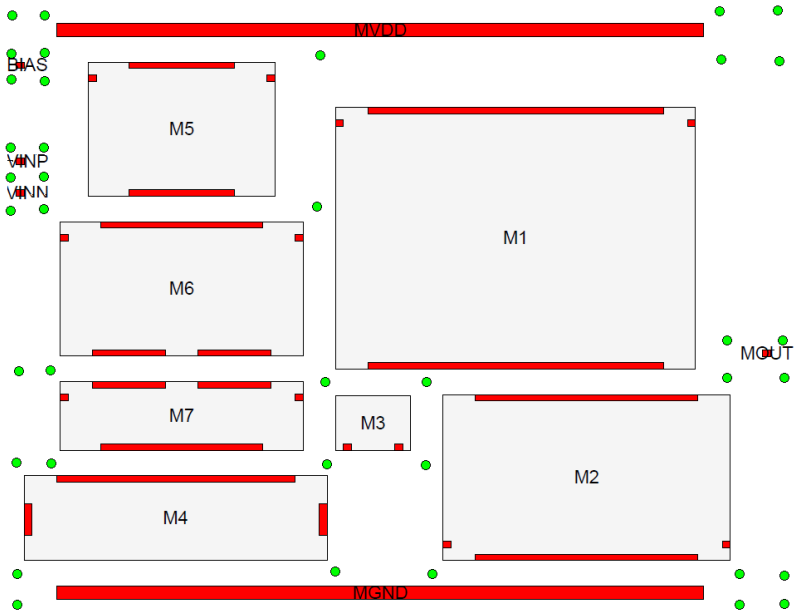

Figure 3.7. Merged Node After Node Merging Process.

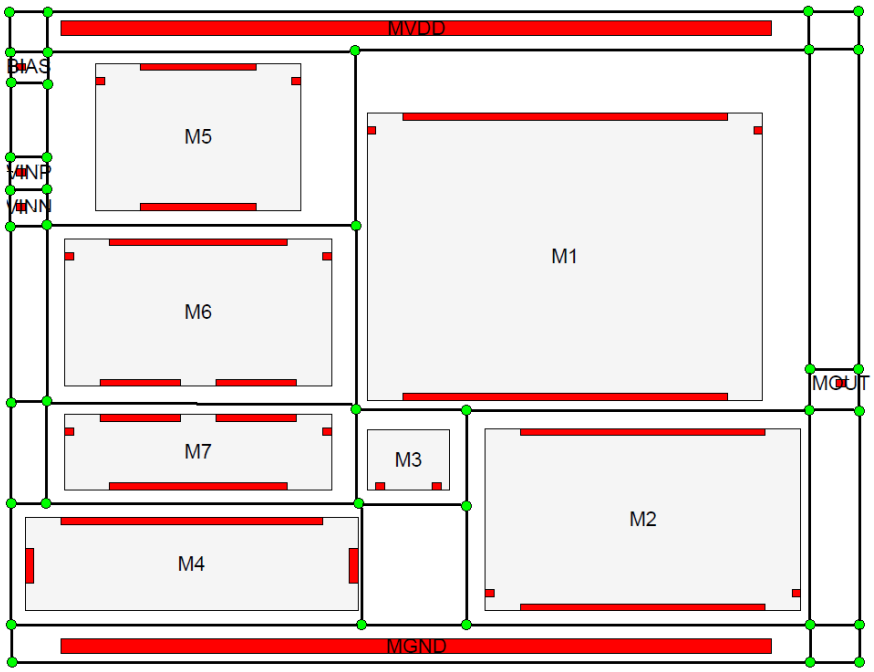

Figure 3.8. Merged Nodes and Resulting CIG.

Nodes in Figure 3.8 represents the vertices and the connections represent the edges in CIG. The relative position of each connection with respect to the modules is also stored for future use. Next step is to construct an extended CIG to use in routing phase.

As stated earlier, ports were split into smaller terminals. Each terminal is added to the graph as a port node. Additional nodes are added dividing the connections to connect port nodes to the graph. Each division has the same relative position information with the original connection with respect to the modules. The layout with the additional nodes is illustrated in Figure 3.9.

3.3. Electrical Connectivity

The template router takes three input files characterizing electrical connectivity information. These are netlist, symmetry and port mapping files.

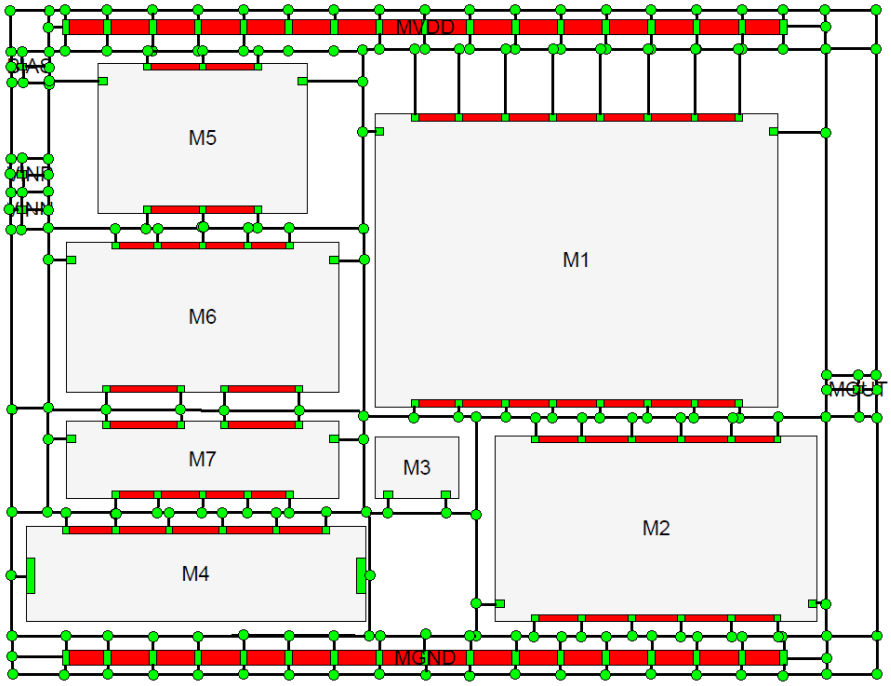

Figure 3.9. Extended CIG.

Netlist file (Figure 3.10) defines the electrical connectivity between the ports at circuit level. Each line of the netlist file starts with the net name and then the ports that belong to that net are listed. The name of the ports not necessarily be the same as they appear in the layout template.

```
netlist.net
n1 M1S MVDDP
n2 M1G MBIASP M5G
n3 M4P1 M3P1
```

Figure 3.10. Netlist File.

Symmetry file (Figure 3.11) indicates the nets to be routed symmetrically. Each line of the symmetry file can be composed of a pair of net names from the netlist file. The nets in the symmetric net pairs must have the same number of ports in the netlist and the symmetric ports have to be in the same order in the netlist file.

```
Symmetry.sym
n3 n4
n6 n7
```

Figure 3.11. Symmetry File.

Port mapping file (Figure 3.12) maps the ports to the layout. Each line of the portmap file starts with the name of a port and then the terminal names in the layout template corresponding to that port are listed. If there is a single terminal of a port in the layout and the port and the terminals names match, this port does not have to be mapped explicitly.

```
portmap.map
M1S M1S1
M1G M1G1 M1G2
M1D M1D1
M2S M2S1
M2G M2G1 M2G2
```

Figure 3.12. Portmap File.

The router processes these files and matches the circuit level ports with the port nodes in the graph. The extended CIG and the electrical connectivity information are prepared together for actual routing procedure.

3.4. Global Routing

Nets are assigned to channels on the wiring path in the global routing phase. An undirected edge weighted graph $G=(V,E)$ derived from the extended CIG of the sample layout template is illustrated in Figure 3.13. $V=\{v_1, v_2,\ldots,v_n\}$ is the set of vertices representing the nodes and $E\subseteq\{\{v_i,v_j\}|v_i\in V,v_j\in V \text{ and } v_i ?v_j\}$ is the set of edges representing the routing channels with the edge weights of the distance between the connected vertices.

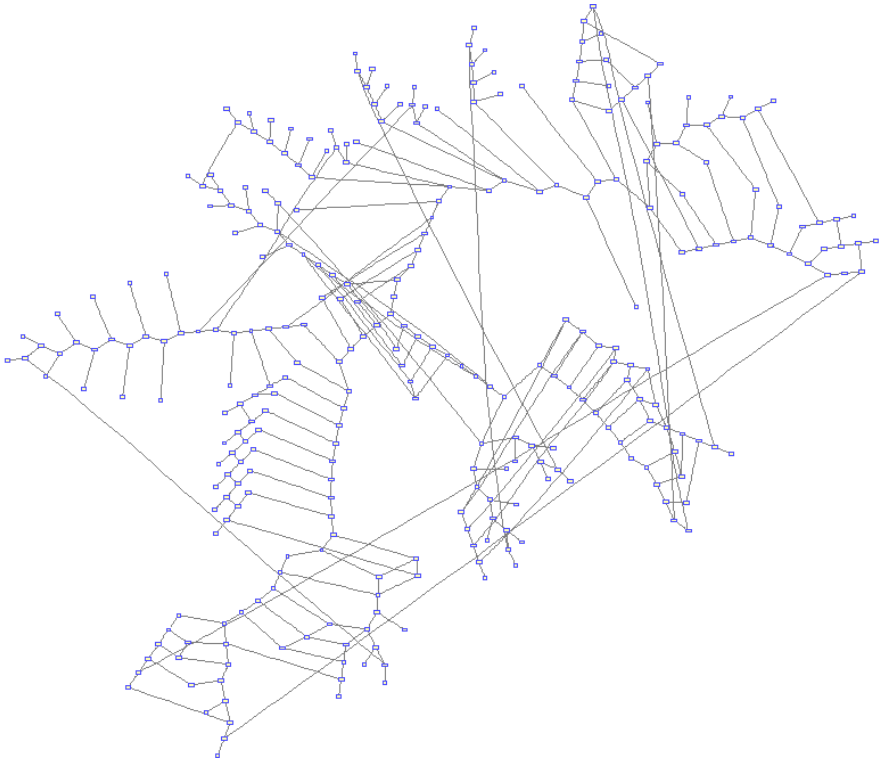

Figure 3.13. Global Routing Graph.

An appropriate global routing graph derived from this master graph is created for each net for global routing. For each net n_i, a global routing graph G_i is constructed such that:

- All terminals of the same ports in n_i are connected to each other by pairs with an edge of minimum weight.
- For the nets that appear in the symmetry file, routing is only allowed around symmetrically placed modules with respect to the symmetry axis of the net. For each symmetrical net pair, symmetrically placed modules are detected and only the routing channels around these modules and connections are left in G_i.

Global routing calculations are done by running a global routing algorithm for each net. For the nets having only two ports in the netlist, Dijkstra's algorithm is used to find the shortest path. For the other multi-port

multi-terminal nets, distance network heuristic is performed with the principles of solving a strong connectivity group Steiner problem. For symmetrical net pairs, the global routing algorithm is only run for a single net in the pair and the routing of the other net is found by mirroring the solution. Global routing calculations are done sequentially. Previously routed nets do not constitute obstacles for the routing algorithm because a restricted VH layer model is used for layer assignment. In this case, any route is possible for all nets with the assumption of flexible channel widths.

3.5. The Output Template With Routing

The main objective of this project is to generate a template with routing information. The actual geometry of the interconnect is determined when instantiating a layout from the template. Only the relative positions of the wire segments are stored in the template. Therefore, there is no need of a detailed routing phase when creating the template.

In LDS generation phase, wire segments and their relative positions with respect to other circuit elements are described in LDS to define paths. As only rectilinear routing is enabled, a wire segment can be either horizontally or vertically placed.

Every wire segment is placed between boxes which correspond to vertices in the circuit graph. A box can either be a via or in the same layer with the wire or segment. The LDS statements to describe the horizontal wire segment *wire1* between the boxes box1 and box2 can be seen in Figure 3.14.

```
//Box dimensions
right ( box1 ) - left ( box1 ) = metWidth;
top ( box1 ) - bottom ( box1 ) = metWidth;
right ( box2 ) - left ( box2 ) = metWidth;
top ( box2 ) - bottom ( box2 ) = metWidth;

//Wire segment dimensions
top ( wire1 ) - bottom ( wire1 ) = metWidth
right ( wire1 ) - left ( wire1 ) >= metWidth

//Aligning the wire segment with the boxes
top ( wire1 ) = top ( box1 )
right ( wire1 ) >= right ( box 1 )
left ( wire1 ) <= left ( box1 )
top ( wire1 ) = top ( box2 )
right ( wire1 ) >= right ( box2 )
left ( wire1 ) <= left ( box2)

//Relative position of the wire segment wrt circuit elements
top ( wire1 ) <= bottom ( M1 )
bottom ( wire1 ) >= top ( M2 )
```

Figure 3.14. Wire Segment Definitions.

Note that this design provides flexibility in the layout by allowing *box1* and *box2* to swap sides. The LDS statements in Figure 3.14 will result in the layout in Figure 3.15 if *wire2* and *wire3* are defined similarly to *wire1*. Also, additional statements are needed to align *box3* and *box4* with the modules.

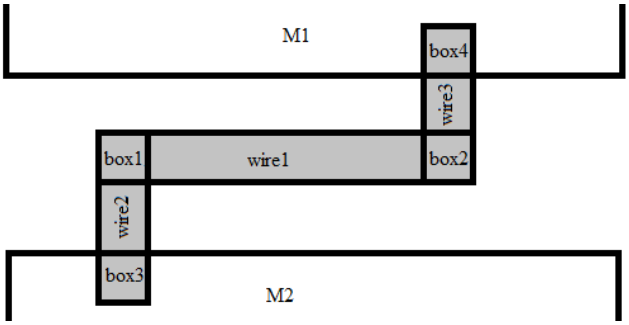

Figure 3.15. Realization of a Wire Segment.

A layout instantiated from the sample layout template after routing is depicted in Figure 3.16. In this figure, first metal layer, second metal layer and vias are colored with red, green, and blue respectively.

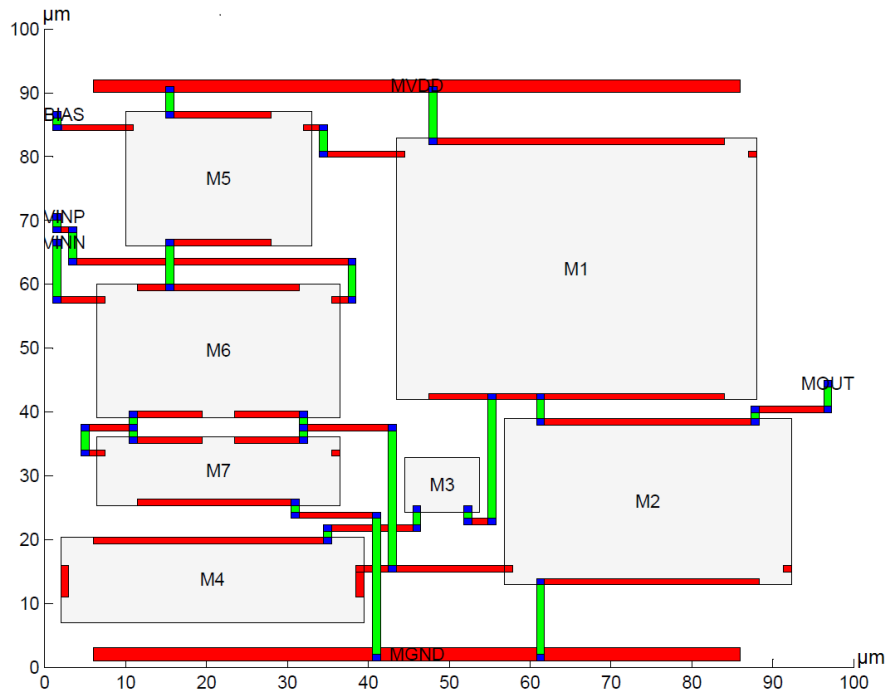

Figure 3.16. Sample Layout with Routing.

The final template with routing can be used in a closed optimization loop by a design automation tool. As the transistor sizes change, a new layout instance can be acquired easily from the template. Figure 3.17, Figure 3.18, Figure 3.19, Figure 3.20 and Figure 3.21 are the instances of the sample layout template obtained by altering the dimensions of the modules randomly.

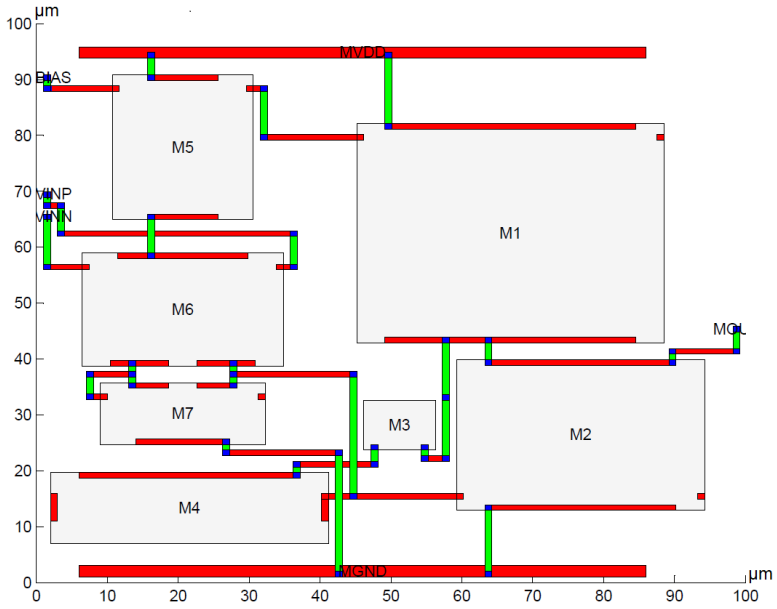

Figure 3.17. An Instance of the Sample Layout Template with Random Module Dimensions - 1.

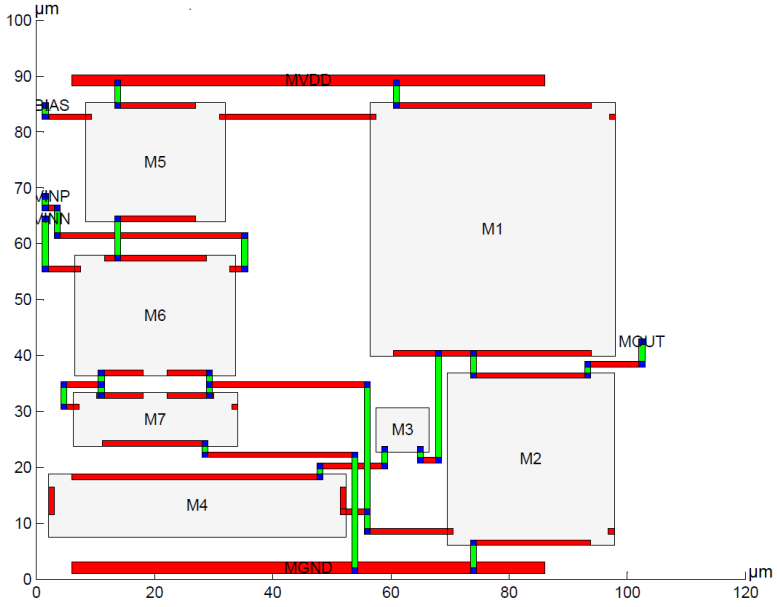

Figure 3.18. An Instance of the Sample Layout Template with Random Module Dimensions - 2.

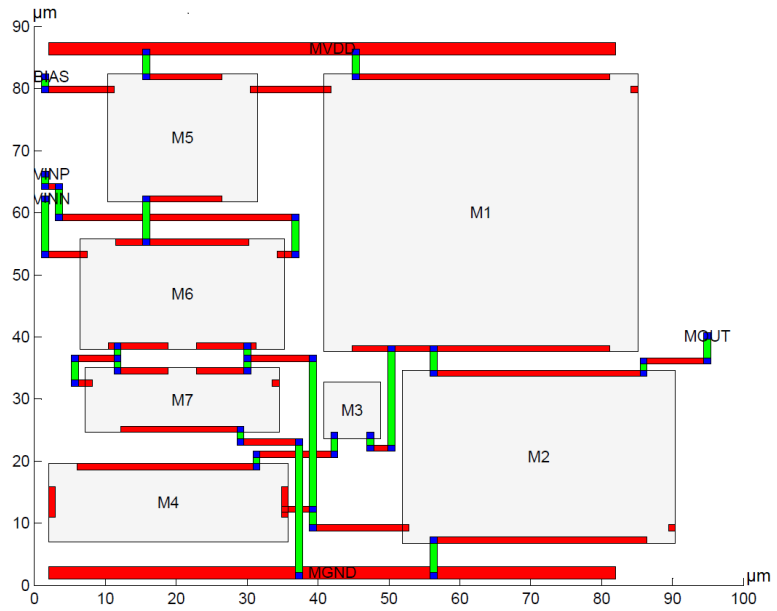

Figure 3.19. An Instance of the Sample Layout Template with Random Module Dimensions - 3.

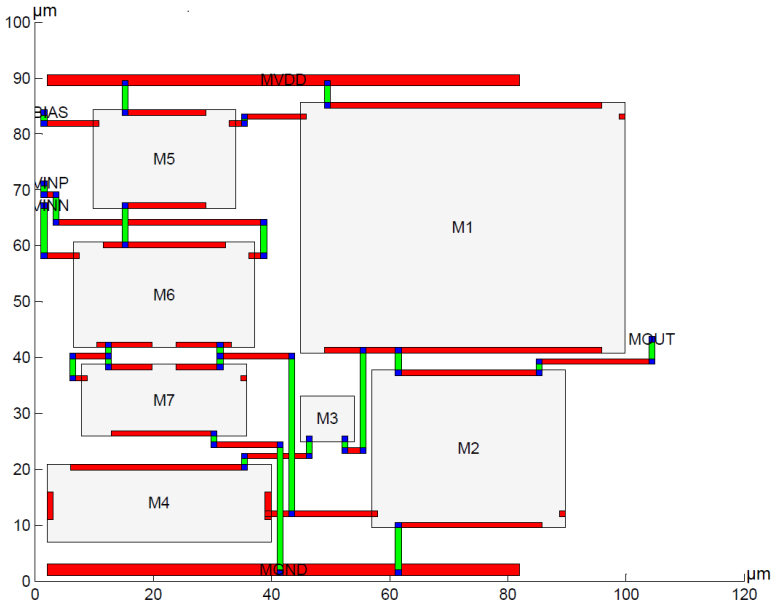

Figure 3.20. An Instance of the Sample Layout Template with Random Module Dimensions - 4.

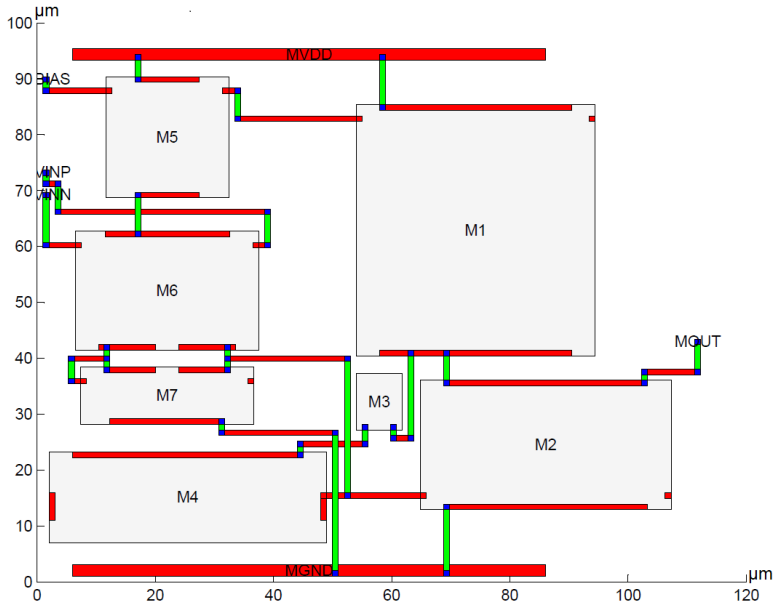

Figure 3.21. An Instance of the Sample Layout Template with Random Module Dimensions - 5.

4. CONCLUSION AND FUTURE WORK

In this thesis, advantages and disadvantages of template based design approaches are discussed. Background information about LDS which is a template based design tool is presented. General information about circuit routing is presented. Classification of routing approaches, main considerations in circuit routing and well-known routing techniques are expressed in detail.

A new template router is proposed for routing LDS templates. The working principle of the template router is explained step by step. The routing approach used in the router is elaborated thoroughly. The routing process and the resulting layout instances are demonstrated on a sample OPAMP circuit template with figures. It has been showed that the resulting template with routing information can adapt itself to the changes in the dimensions of the circuit elements.

The template router proposed in this thesis differs from the other routers by the form of its input and output. The input file holds the relative placement information of the circuit elements rather than providing fixed coordinates in the layout. Likewise, the output file consists of relative positions of the interconnect. Wire segments are defined with their relative positions with respect to other circuit elements in the layout.

The router proposed in this thesis can further be improved. The routing approach used in this project is easily adaptable to new extensions. It is possible to add design considerations such as crosstalk awareness or electromigration avoidance by some modifications in the routing algorithm.

REFERENCES

1. Yilmaz, E. and G. Dundar, "Analog Layout Generator for CMOS Circuits", *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, Vol. 28, No. 1, pp. 32–45, Jan 2009.

2. Unutulmaz, A., G. Dundar and F. V. Fernandez, "A Template Router", *Circuit Theory and Design (ECCTD), 2011 20th European Conference on*, pp. 334–337, Aug 2011.

3. Unutulmaz, A., G. Dundar and F. V. Fernandez, "LDS - A Description Script for Layout Templates", *Circuit Theory and Design (ECCTD), 2011 20th European Conference on*, pp. 857–860, Aug 2011.

4. Unutulmaz, G. D., A and F. V. Fernandez, "LDS Based Tools to Ease Template Construction", *Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD), 2012 International Conference on*, pp. 61–64, Sept 2012.

5. Lin, A. H. M. v. R., C. and D. Leenaerts, *Mixed-Signal Layout Generation Concepts*, The Springer International Series in Engineering and Computer Science, Springer, 2005.

6. Sherwani, N. A., *Algorithms for VLSI Physical Design Automation*, Kluwer Academic Publishers, Norwell, MA, USA, 2nd edn., 1995.

7. Kahng, A. B. and G. Robins, *On Optimal Interconnections for VLSI*, Kluwer international series in engineering and computer science: VLSI, computer architecture, and digital signal processing, Springer, 1995.

8. Graeb, H. E., *Analog Layout Synthesis: A Survey of Topological Approaches*, Springer, 2010.

9. Conway, J. and G. G. Schrooten, “An Automatic Layout Generator for Analog Circuits”, *Design Automation, 1992. Proceedings., [3rd] European Conference on*, pp. 513–519, Mar 1992.

10. Owen, R. D. S. J. C. O. S. R., B.R. and K. Martin, “BALLISTIC: An Analog Layout Language”, *Custom Integrated Circuits Conference, 1995., Proceedings of the IEEE 1995*, pp. 41–44, May 1995.

11. Dessouky, M. and M. Louerat, “A Layout Approach for Electrical and Physical Design Integration of High-Performance Analog Circuits”, *Quality Electronic Design, 2000. ISQED 2000. Proceedings. IEEE 2000 First International Symposium on*, pp. 291–298, 2000.

12. Dessouky, M. L., M. and J. Porte, “Layout-Oriented Synthesis of High Performance Analog Circuits”, *Design, Automation and Test in Europe Conference and Exhibition 2000. Proceedings*, pp. 53–57, 2000.

13. Lourenco, N. and N. Horta, “Laygen - An Evolutionary Approach to Automatic Analog IC Layout Generation”, *Electronics, Circuits and Systems, 2005. ICECS 2005. 12th IEEE International Conference on*, pp. 1–4, Dec 2005.

14. Lourengo, M. V. J. G., N. and N. Horta, “LAYGEN - Automatic Layout Generation of Analog ICs from Hierarchical Template Descriptions”, *Research in Microelectronics and Electronics 2006, Ph. D.*, pp. 213–216, 2006.

15. Lampaert, G. G., K. and W. M. C. Sansen, *Analog Layout Generation for Performance and Manufacturability*, Kluwer international series in engineering and computer science: Analog circuits and signal processing, Springer, 1999.

16. Wang, Y. W. C., L. T. and K. T. Cheng, *Electronic Design Automation: Synthesis, Verification, and Test*, Systems on Silicon, Elsevier Science, 2009.

17. Lienig, G. J., J. and T. Adler, "Electromigration Avoidance in Analog Circuits: Two Methodologies for Current-Driven Routing", *Design Automation Conference, 2002. Proceedings of ASP-DAC 2002. 7th Asia and South Pacific and the 15th International Conference on VLSI Design. Proceedings.*, pp. 372–378, 2002.

18. Martins, N. L. A. C., R. and N. Horta, "Electromigration-Aware Analog Router with Multilayer Multiport Terminal Structures", *Integration, the {VLSI} Journal*, Vol. 47, No. 4, pp. 532 – 547, 2014.

19. K, M. and K. Tabuchi, "A Computer Program for Optimal Routing of Printed Circuit Connectors", *IFIPS Proceedings*, Vol. H47, pp. 1475–1478, 1968.

20. Hashimoto, A. and J. Stevens, "Wire Routing by Optimizing Channel Assignment Within Large Apertures", *Proceedings of the 8th Design Automation Workshop*, DAC '71, pp. 155–169, ACM, New York, NY, USA, 1971.

21. Deutsch, D. N., "A Dogleg Channel Router", *Proceedings of the 13th Design Automation Conference*, DAC '76, pp. 425–433, ACM, New York, NY, USA, 1976.

22. Dijkstra, E. W., "A Note on Two Problems in Connection with Graphs", *Numerische Mathematik*, Vol. 1, No. 1, pp. 269–271, Dec. 1959.

23. Hart, N. J. N., P. E. and B. Raphael, "Correction to "A Formal Basis for the Heuristic Determination of Minimum Cost Paths"", *SIGART Bull.*, , No. 37, pp. 28–29, Dec. 1972.

24. Mehlhorn, K. and P. Sanders, *Algorithms and Data Structures: The Basic Toolbox*, SpringerLink: Springer e-Books, Springer, 2008.

25. Prim, R. C., "Shortest Connection Networks and Some Generalizations", *Bell System Technical Journal*, Vol. 36, No. 6, pp. 1389–1401, 1957.

26. Kruskal, J. B., "On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem", *Proceedings of the American Mathematical Society*, Vol. 7, No. 1, pp. 48–50, Feb. 1956.

27. Hwang, D. S. R., F. K. and P. Winter, *The Steiner Tree Problem*, Annals of Discrete Mathematics, Elsevier Science, 1992.

28. Kou, G. M., L. and L. Berman, "A Fast Algorithm for Steiner Trees", *Acta Informatica*, Vol. 15, No. 2, pp. 141–145, 1981.

29. Alpert, D. P. M., C.J. and S. S. Sapatnekar, *Handbook of Algorithms for Physical Design Automation*, Taylor & Francis, 2008.