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

The design of modern integrated circuits is overwhelmingly complicated due to the enormous number of cells in a typical modern circuit. To deal with this difficulty, the design procedure is broken down into a set of disjoint tasks. Circuit layout is the task that refers to the physical realization of a circuit from its functional description. In circuit layout, a connection-list called *netlist* of cells and nets is given. Placement and routing are subtasks associated with circuit layout and involve determining the geometric locations of the cells within the placement area and connecting cells sharing common nets. In performing the placement and the routing of the cells, minimum placement area, minimum delay and other performance constraints need to be observed.

In this work, we propose and investigate new approaches to placement and routing problems. Specifically, for the placement subtask, we propose new *convex* programming formulations to estimate wirelength and force cells to spread within the placement area. As opposed to previous approaches, our approach is partitioning free and requires no hard constraints to achieve cell spreading within the placement area. The result of the global optimization of the new convex models is a global placement which is further improved using a *Tabu search* based iterative technique. The effectiveness, robustness and superiority of the approach are demonstrated on a set of *nine* benchmark industrial circuits.

With regard to the routing subtask, we propose a hybrid methodology that combines *Tabu search* and *Stochastic Evolution* as a search engine in a new channel router. We also propose a new scheme based on *Utility Theory* for selecting and assigning nets to tracks in the channel. In this scheme, problem-domain information expressed in the form of utility functions is used to guide the search engine to explore the search space effectively. The effectiveness and robustness of the approach is demonstrated on *five* industrial benchmarks.

Acknowledgements

First, all praise to God, the most Gracious and most Merciful, whose help and guidance is ever dominating throughout my life.

I would like to thank my supervisor Dr. Anthony Vannelli for his constant support and for providing me advice when appropriate. His enthusiasm and guidance have proven invaluable to my research.

I would also like to thank my friend and colleague Dr. Shawki Areibi for contributing to this research and for many fruitful discussions and valuable suggestions.

My deep appreciation goes also to the people at the Canadian Bureau for International Education (CBIE) for their understanding and encouragement.

Financial support for this work was provided by the Secretariat of Scientific Research (SSR) of Libya through CBIE, and in part by the Natural Science and Engineering Research of Canada (NSERC). This support is greatly appreciated.

To My Parents

Contents

1 Introduction **1**

1.1 VLSI Design Cycle 2

1.2 Circuit Layout Cycle 6

1.2.1 Partitioning 6

1.2.2 Placement 6

1.2.3 Routing 8

1.3 Design Style 9

1.3.1 Macro/Full-Custom 9

1.3.2 Semi-Custom 10

1.4 Motivations and Objectives 12

1.4.1 Motivations 13

1.4.2 Objectives 13

1.5 Thesis Outline 15

2 Cell Placement **17**

2.1 Constructive Placement 18

2.2 Iterative Improvement Placement 19

2.3 A Combination of Constructive and Iterative Methods 20

2.4 Net Length Estimation 23

2.5 Test Circuits 23

2.6 Summary 26

3 The Cell Repeller Model **28**

3.1 Problem Formulation and the Quadratic Measure 30

3.2 New Convex Models 32

3.2.1 Elements of **v** are Independent 33

3.2.2 Elements of v are Dependent 35

3.3 The Repeller Model 41

3.4 Summary 44

4 The New Generic Placement Method **45**

4.1 Cell Spreading and the Partitioning Approach 46

4.2 New Method for Cell Spreading 49

4.2.1 Identifying Dense and Sparse Regions 50

4.2.2 Cell-Attractor Assignment 52

4.3 The Attractor-Repeller Model 57

4.4 The Attractor-Repeller Placer: Basic Algorithm 59

4.5 Global Placement Legalization 61

4.6 Iterative Improvement Method 62

4.6.1 Tabu Search 62

4.6.2 Tabu Search for Placement Iterative Improvement 64

4.7 Summary 67

5 Application To Standard Cell Placement 68

5.1 Qualitative Analysis 69

5.1.1 Attractors-Repellers and Cell Spreading 70

5.1.2 Cell Attractors and Convergence of the Global Optimization 74

5.2 Numerical Results 76

5.3 Summary 82

6 Routing Problem 84

6.1 Detailed Routing 85

6.1.1 Channel Routing Problem (CRP) 86

6.1.2 SwitchBox Routing Problem (SBRP) 97

6.2 Summary 98

7 Utility Function based Channel Router 100

7.1 Stochastic Evolution (SE) 101

7.2 Hybrid Methodology 105

7.2.1 Utility Functions 105

7.2.2 Channel Routing Problem-domain Information and Utility Functions 107

7.3 Algorithm Description 108

7.3.1 Initial Solution 108

7.3.2 Merging Phase 109

7.3.3 Ripup and Reassign Phase 111

7.4 Implementation and Results 111

7.5 Summary 115

8 Conclusions and Future Directions **117**

8.1 Summary and Contributions 118

8.1.1 Cell Placement 118

8.1.2 Channel Routing 121

8.2 Future Directions 122

Bibliography **125**

List of Tables

2.1 MCNC Benchmarks used as test cases 24

5.1 Wire Length estimates and CPU time for initial placements. 77

5.2 Wire Length Comparison, TW v7.0 flat and hierarchical modes, Gordain/Domino and ARP 79

5.3 Chip width comparison: TimberWolf v7.0 and ARP. Width is measured in *microns*. 80

5.4 Run-time comparison in CPU seconds: TimberWolf v7.0 in flat and hierarchical modes, Gordian/Domino and ARP. 80

5.5 Relative wirelength improvement with respect to other approaches (+ means ARP is better). 81

5.6 Relative chip-width improvement with respect to other approaches (+ means ARP is better). 81

5.7 Relative CPU time improvement with respect to other approaches (+ means ARP is better). 82

7.1 Statistics for the different benchmarks. 112

7.2 For all the benchmarks, average number of generations (AVG-GEN) required to converge to an optimal solution (OPT-SOL) when the search engine is (i)only SE; (ii)SETS hybrid; (iii)SETS hybrid and utility functions (SETS-UTFN). 114

List of Figures

1.1 Design process steps 5

1.2 Design process steps 7

1.3 Macro-cell design topology. 10

1.4 Semi-custom design topologies. (a) Row-oriented standard cells (b) Gate Arrays. 11

2.1 Classification of placement methods. 18

2.2 A typical combination of constructive and iterative improvement methods. 22

2.3 Estimating the length of a net using the half-perimeter of the minimum rectangle enclosing the cells in the net. In this example, HPWL=$H + V$. 24

2.4 Cumulative distribution of number of nets. 26

3.1 Coordinates of a two cell net. 35

3.2 Simple two cell netlist. (a) Both cells are movable. (b) A fixed cell is added to the netlist. 36

3.3 Representation of the regions in which $\sum_{i,j} \eta(\zeta_{ij})$ is convex, quasi-convex and nonconvex. 41

3.4 $f(z_{ij}) = z_{ij} - ln(z_{ij}) - 1$. The portion of the curve marked with "x's" represents $f(z_{ij})$ for $z_{ij} < 1$ and the one marked with "+'s" represents $f(z_{ij})$ when $z_{ij} \geq 1$. $f(z_{ij})$ is not convex for $z_{ij} < 1$. Note that we subtracted 1 so that at convergence, $z_{ij} = 1$, $f(z_{ij}) = 0$. . . 42

3.5 $f(z_{ij}) = z_{ij} + e^{1-z_{ij}} - 2$. The portion of the curve marked with "x's" represents $f(z_{ij})$ for $z_{ij} < 1$ and the one marked with "+'s" represents $f(z_{ij})$ when $z_{ij} \geq 1$. Again, $f(z_{ij})$ is not convex for $z_{ij} < 1$. Again, note that we subtracted 2 from $f(z)$ so that at convergence , $z_{ij} = 1$, $f(z_{ij}) = 0$. 43

4.1 Examples of relative placements [34]. 47

4.2 Region R_1 is dense, while regions R_2, R_3 and R_4 are sparse. 52

4.3 Region R_1 is dense while regions R_2 and R_3 are sparse. Cell C_1 located in R_1 with geometric location (x_1, y_1) is hooked to the dummy fixed cell C with coordinates (x_2, y_3) where x_2 and y_3 are the x and y coordinates of C_2 and C_3 (which are the closest cells to C_1 in the x and y directions respectively). Note that $Dx(ij)$ and $Dy(ij)$ are the distances in the x and y directions between cell i and cell j. . . 56

4.4 An outline of the new global placement method **ARP**. 60

5.1 Benchmark *Primary1*: cell spreading after each pass using no repellers, $\rho(z_{ij}) = 0$ and $d = 1$ in $f(z_{ij})$ ("+" represent locations of movable cells, "x" represent locations of fixed cells (I/O pads), and "o" represent locations of attractors). 69

5.2 Cell spreading using strict repeller model. "+" represents movable cells and "x" represents fixed cells. 70

5.3 Benchmark *Primary1*: cell spreading after each iteration ("+" represents locations of movable cells, "x" represents locations of I/O pads on the chip periphery, and "o" represents locations of attractors). 71

5.4 Variability of average wirelength over the different iterations of the algorithm. 73

5.5 Variability of average wirelength versus the scaling parameter k, (equation (5.1)). 74

5.6 Variability of run time. (a) versus passes (b) versus number of attractors. 75

6.1 Routing: (a) Global routing; (b) Detailed routing. 85

6.2 A channel. 87

6.3 Terminology for CRP 87

6.4 Netlist representation for routing requirements [65]. 88

6.5 Solution of the netlist in 6.4. 89

6.6 (a) VCG (b) HCG for the netlist in Fig 6.4. In HCG, maximal cliques are (1,2,3,4,5), (2,4,6), (4,6,7), (4,7,8,9) and (7,9,10) 90

6.7 Example of a switchbox. 97

7.1 Outline of the SETS-CR. 109

7.2 The routing of Deutsch's difficult example. 113

7.3 The variation of the average cost value as the algorithm progresses over generations for the difficult Deutsch problem. 113

Chapter 1

Introduction

From [1] its humble beginning in the early 1950's to the manufacture of circuits with millions of components today, VLSI design has brought the power of a mainframe computer to a laptop. This tremendous growth in the area of VLSI design is made possible by the development of sophisticated design tools and software. To deal with the complexity of millions of components, VLSI design tools must be computationally fast and generate layouts close to optimality. The future growth of VLSI systems depends critically on the research and development of *Circuit Layout (Physical Design)* automation tools.

The layout of integrated circuits on chips and boards is a complex task. The optimization problems that have to be solved during the circuit layout are intractable [39, 62]. In other words they are usually NP-hard [24]. This implies that, for most of these problems, the optimal solutions cannot be obtained in polynomial time.

[1] Most of the background described in this chapter is well established. For those who are interested in more information, the following references are recommended: [33, 47, 56, 67].

1.1 VLSI Design Cycle

The VLSI design cycle starts with a formal specification of a VLSI chip, follows a series of steps, and eventually produces a packaged chip. A typical design cycle may be represented by the flowchart shown in Figure 1.1. The steps of the VLSI design cycle can be briefly outlined as follows [56, 62]:

1. **System Specification:** As in any design process, the first step is to lay down the specifications of the system to be designed. This necessitates creating a high level representation of the system. The factors to be considered in this process include: performance, functionality, and the physical dimensions. The choice of fabrication technology and design techniques are also considered. The end results are specifications for the size, speed, power and functionality of the VLSI system to be designed.

2. **Functional Design:** In this step, the behavioral aspects of the system are considered. The outcome is usually a timing diagram or other relationships between sub-units. This information is used to improve the overall design process and to reduce the complexity of the subsequent phases.

3. **Logic Design:** In this step, the logic structure that represents the functional design is derived and tested. The achieved design is represented by a textual, schematic or graphic description. The logic design is usually represented by Boolean expressions. These expressions are minimized to achieve the smallest logic design which conforms to the functional design. Logic design of the system is simulated and tested to verify its correctness.

4. **Circuit Design:** The purpose of circuit design is to develop a circuit representation based on the logic design. The boolean expressions are converted into circuit representation by taking into consideration the speed and power requirements of the original design. The electrical behavior of the various components are also considered in this phase. The circuit design is usually expressed in a detailed circuit diagram.

5. **Circuit Layout:** In this step, the circuit representation of each component is converted into geometric representation. This representation is in fact a set of geometric patterns which perform the intended logic function of the corresponding component. Connections between different components are also expressed as geometric patterns. As stated earlier, this geometric representation of a circuit is called a *layout*. The exact details of the layout also depend on design rules, which are guidelines based on the limitations of the fabrication process and the electrical properties of the fabrication materials. Circuit layout is a very complex process, therefore, it is usually broken down into various sub-steps in order to handle the complexity of the problem. In fact, circuit layout is arguably the most time consuming step in the VLSI design cycle.

6. **Design Verification:** The layout is verified in this step to ensure that the layout meets the system specifications and fabrication requirements. Design verification consists of *Design Rule Checking* and *Circuit Extraction*. Design Rule Checking is a process which verifies that all geometric patterns meet the design rules imposed by the fabrication process. After checking the layout for design rule violations and removing the design violations, the functionality of the layout is verified by circuit extraction. This is a reverse engineering process and generates the circuit represen-

tation from the layout. This reverse engineering circuit representation can then be compared with the original circuit representation to verify the correctness of the layout.

7. **Fabrication:** After verification, the layout is ready for fabrication. The fabrication process consists of several steps: preparation of wafer, deposition and diffusion of various materials on the wafer according to layout description. Before the chip is mass produced, a prototype is made and tested.

8. **Packaging, Testing and Debugging:** Finally, the wafer is fabricated and diced in a fabrication facility. Each chip is then packaged and properly tested.

The VLSI design cycle involves several iterations, both within a step and between different steps. The entire design cycle may be viewed as transformations of representation in various steps. In each step, a new representation of the system is created and analyzed. The representation is iteratively improved to meet system specifications. For instance, a layout is iteratively improved so that it meets the timing specifications of the system. Another example may be detection of design rule violations during design verification. If such violations are detected, the circuit layout step needs to be repeated to correct the error. In this thesis, the emphasis is on the **circuit layout** step of the VLSI design cycle. The required steps in this process will be discussed in more detail in the next section.

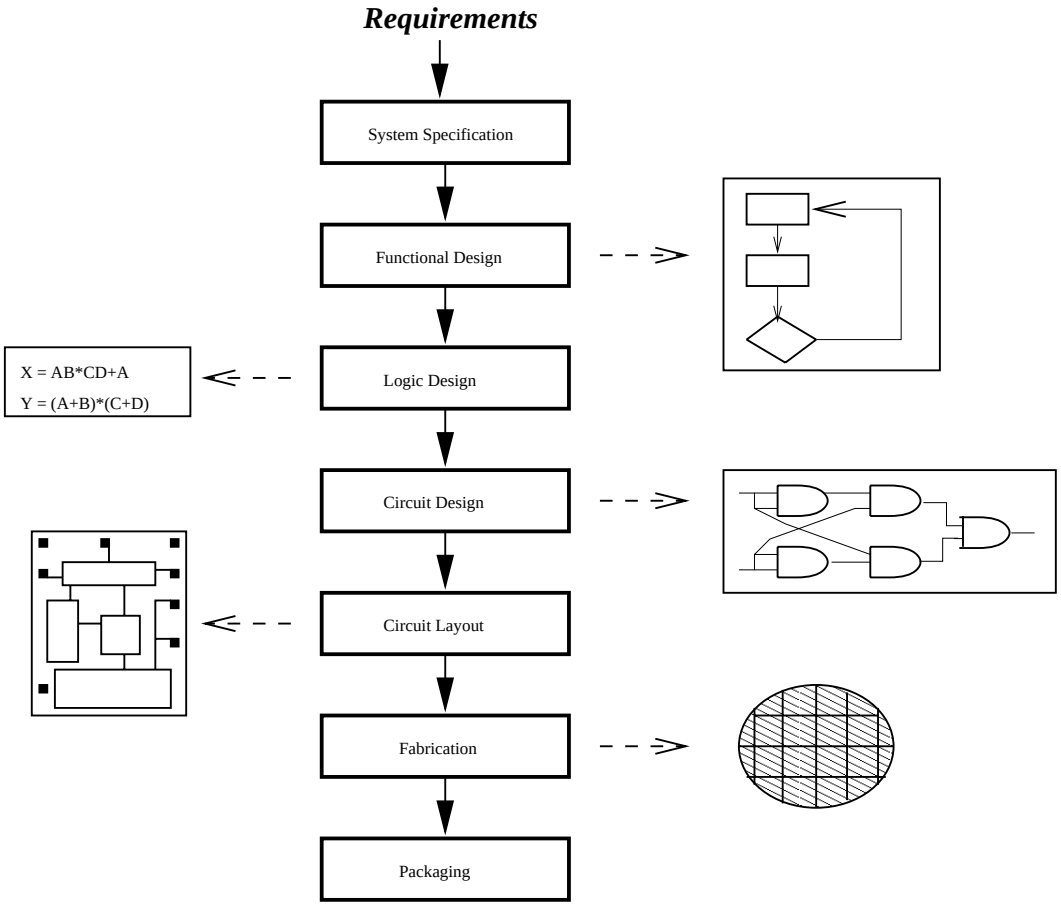


Figure 1.1: Design process steps

1.2 Circuit Layout Cycle

The input to the circuit layout design cycle is a circuit diagram and the output is the layout of the circuit. This is accomplished in several stages such as partitioning, floorplanning, placement, routing and compaction. The different stages of circuit layout are shown in Figure 1.2. To give a global perspective, the following is a description of these stages.

1.2.1 Partitioning

A chip may contain several million transistors. Layout of the entire circuit cannot be handled due to the limitations of memory space as well as computation power available. Therefore, it is normally partitioned by grouping the components into blocks (subcircuits/modules). The actual partitioning process considers many factors such as: size of the blocks, number of blocks and number of interconnections between the blocks. The output of partitioning is a set of blocks along with the interconnections required by blocks. The set of interconnections required is referred to as *netlist*. For more details, refer to [3, 56, 35].

1.2.2 Placement

During placement, the blocks are exactly positioned on the chip. The goal of placement is to find a minimum area arrangement for the blocks that allows completion of interconnections between the blocks. Placement is typically done in two phases. In the first phase, an initial placement is created. In the second phase, the initial placement is evaluated and iterative improvements are made until the layout has

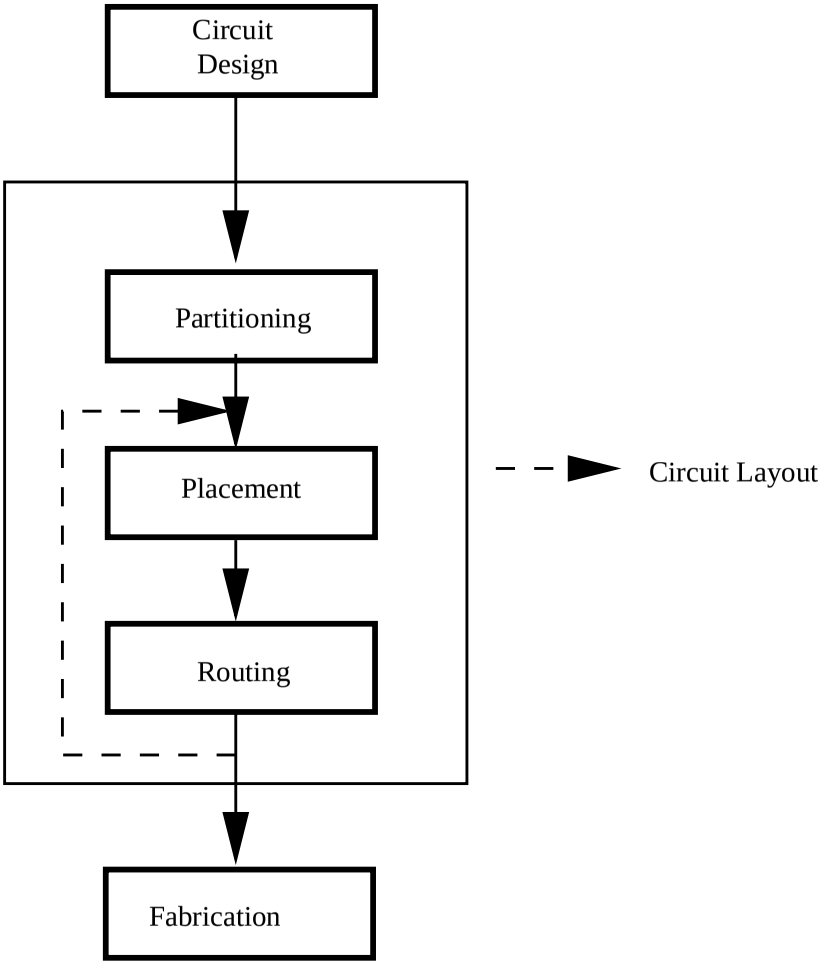


Figure 1.2: Design process steps

minimum area and conforms to design specifications. Space between the blocks is intentionally left empty to allow interconnections between blocks.

The quality of the placement will not be evident until the routing phase has been completed. Placement may not lead to routable design; i.e, routing may not be possible in the space provided. In that case, another placement iteration is required to address this problem. An estimate of the required routing space is employed to limit the number of iterations of the placement algorithm. A good routing and circuit performance heavily hinge on a good placement algorithm. Once the positions of the blocks are fixed, it becomes difficult to improve the routing and the overall performance of the circuit.

1.2.3 Routing

The objective of the routing phase is to complete the interconnections between the blocks according to the specified netlist. The space not occupied by the blocks is partitioned into rectangular regions called *channels* and *switchboxes*. Using the channels and the switchboxes, the aim is to complete all circuit connections using the shortest possible wirelength. The routing problem is difficult and it is usually done in two phases; i.e, *Global Routing* and *Detailed Routing*. In global routing, connections are completed between the blocks of the circuit disregarding the exact geometric details of each wire and pin. Global routing specifies the "loose route" of a wire through different regions in the routing space. In other words, global router finds a list of channels which are to be used as a passageway for each wire. Detailed routing, follows global routing, performs point to point connections between pins and blocks; i.e, loose routing is converted into exact routing by specifying geometric information such as layer assignments of wires. Detailed routing includes **channel**

routing and **switchbox routing**.

1.3 Design Style

Circuit layout is an extremely hard process and even after breaking the entire process into several conceptually easier steps, each step is still computationally very hard. As a consequence, restricted models and design styles are used in order to reduce the complexity of circuit layout. The design styles can be broadly classified into as either *macro/full-custom* or *semi-custom*. In macro/full-custom layout, different blocks of a circuit can be placed at any location on a silicon wafer provided that they do not overlap. In semi-custom layout, some parts of a circuit are predesigned and placed on some specific place on the silicon wafer.

1.3.1 Macro/Full-Custom

In this design style, the circuit is partitioned into a collection of subcircuits according to some criteria such as functionality of each subcircuit. Each subcircuit is called a *block* or *a cell*, see Figure 1.3. Blocks are allowed to be of any size and they are to be placed at any location on the chip surface without any restrictions. This design style allows for very compact designs. However, the process of automating macro/full-custom design has much higher complexity than other restricted models. For this reason, macro/full-custom design style is only used when area of final design must be minimized and designing time is less important.

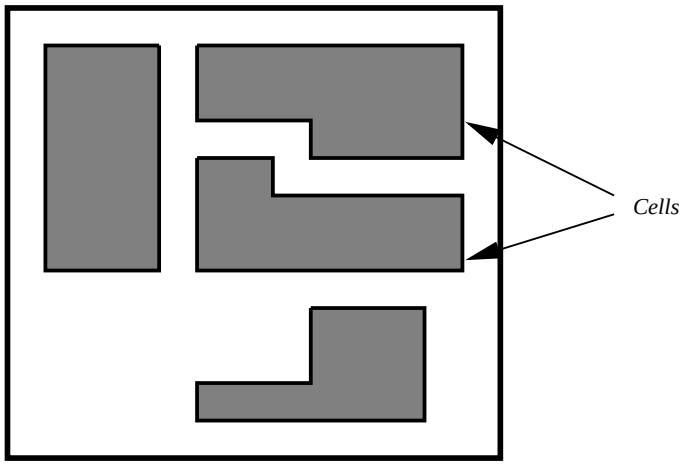


Figure 1.3: Macro-cell design topology.

1.3.2 Semi-Custom

Standard Cell

In standard cell design style, cells are of the same height and not necessarily same width. The circuit is partitioned into several smaller blocks each of which is equivalent to some predefined sub-circuit (cell). A collection of these cells are called a *cell library*. Cells are placed in rows and the space between rows is called a *channel*. The channels are used to connect the cells, see Figure (1.4-a).

This design style is well suited for moderate size circuits and medium production volumes. A standard cell design usually takes more area than macro/full-custom design.

Gate Arrays

Unlike standard cell design style, in gate array design style all cells are identical. In other words, all cells have same height and same width. In this design style, the

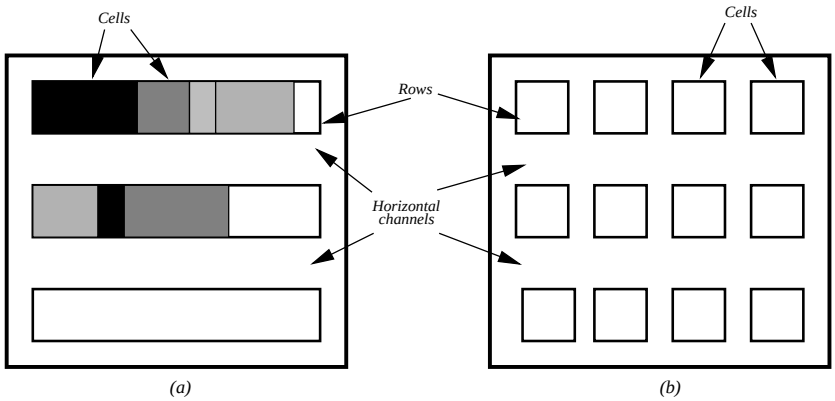


Figure 1.4: Semi-custom design topologies. (a) Row-oriented standard cells (b) Gate Arrays.

entire wafer is prefabricated with an array of identical gates or cells. These cells are separated by both vertical and horizontal spaces called vertical and horizontal channels, see Figure (1.4-b). The channels are used to perform the interconnections between the cells. The name "gate array" signifies the fact that each cell may simply be a gate, such as a three input NAND gate.

Gate array design style is a simplification of standard cell design. Compared to standard cell and full custom design styles, gate array design imposes higher rigidity upon the circuit. However, gate arrays are cheaper to produce.

The gate arrays architecture is the most restricted form of layout. This also means that it is the simplest for layout algorithms to work with.

Field Programmable Gate Arrays (FPGAs)

The Field Programmable Gate Array (FPGA) is a new approach to Application Specific Integrated Circuits (ASIC) [18, 19, 20]. A FPGA consists of horizontal rows of programmable logic blocks which can be interconnected by a programmable

routing network. The typical FPGA logic block is more complex than a gate and much simpler than a cell in the standard cell design. In its simplest form, a logic block is simply a memory block which can be programmed to remember the logic table of a function. Given a certain input, the logic block look up the corresponding output from the logic table and sets its output line accordingly. Thus by loading different look-up tables, a logic block can be programmed to perform different functions. The rows of logic blocks are separated by horizontal routing channels. The channels are not simply area in which metal lines can be arranged for a specific design. Rather, they contain predefined wiring segments of fixed lengths. Each input and output of a logic block is connected to a dedicated vertical segment. Connection between horizontal segments is provided through *antifuses* whereas the connection between a horizontal segment and vertical segment is provided through a *cross fuse*. The customization (programming) of a generic (unprogrammed) FPGA is simple. Given a circuit, it is decomposed into smaller subcircuits such that each subcircuit can be mapped to a logic block. The interconnections between any two subcircuits is achieved by programming the FPGA interconnects between their corresponding logic blocks.

1.4 Motivations and Objectives

In this thesis, we focus on the *cell placement* and *detailed routing* problems. We defer the details to later chapters and presently, we briefly describe the problems to illustrate and justify the motivations and goals for further investigation.

1.4.1 Motivations

Placement methods may be broadly classified as *constructive* or *iterative improvement* methods. The consensus is that certain iterative improvement methods produce high quality placements, but excessive computation time is required to do so. On the other hand, constructive methods yield not as high quality placements as iterative improvement methods, yet good, in a much shorter time. Ideally, both quality and computational efficiency of the solution are crucial for a practical placement method. Quality of solution is important for performance of the circuit and computational efficiency is essential for curtailing the design procedure, especially for large circuits where weeks, months or even years may be required to realize these circuits.

The fact that future placement and routing tasks will be much more complicated (due to the increasing size of the circuits and the growing design objectives) implies that faster placement and routing tools should be developed to handle such immense complexity. Future placement and routing tools must be adequately flexible to handle any modifications in VLSI design styles and design objectives. In summary, they should be (i) effective (ii) efficient (iii) flexible and (iv) robust.

1.4.2 Objectives

Our main objective is to develop and examine new methods and strategies to perform the placement and detailed routing tasks. These new methods should be robust and flexible besides, effective and efficient.

For the placement problem, we propose a combination of constructive and iterative improvement methods in which the constructive method provides a good

initial placement that is further improved by the iterative method. Our constructive method performs the placement of the cells in a global sense; i.e, cell-interconnections are considered simultaneously when the placement is computed. Previous approaches to global placement rely on mathematical programming and partitioning of the placement area to spread the cells on the placement floor. The methodology iterates between repartitioning and global optimization until the placement area is exhausted. The partitioning approach requires the addition of hard constraints to spread the cells and that increases the amount of computation time required. It also suffers from other shortcomings as we will see in chapter 3 and chapter 4.

In this work, we propose new mathematical models to estimate the wirelength (that is, for global placement) and prevent cells from overlapping while minimizing the wirelength estimate. The new formulations are convex programming models and accordingly, any convex programming methodology can be directly applied. In fact, proofs of convexity of the new formulations are presented in detail. In terms of forces, the new formulations correspond to ***cell repellers*** that prohibit connected cells from overlapping while minimizing the wirelength (distance) between their geometric locations. Furthermore, adaptive ***cell attractors*** are also added to the repellers to pull cells to sparse regions without excessively stretching short nets. We refer to the new formulations based model as the ***Attractor-Repeller*** model. Furthermore, we propose a *generic* placement method based on the attractor-repeller approach, and illustrate its competitiveness and effectiveness compared to up-to-date placers.

A significant impact of the new method is on cell placement with no fixed cells (i.e, I/O pads) such as FPGA placement. Specifically, the fact that the new method relies on cell repellers and attractors in spreading the cells within the placement

area, implies that the existence or absence of fixed cells in the original netlist is irrelevant. As we will see later, only one fixed cell is needed to drive the (repeller) objective function to the convexity region. One way to achieve this without affecting the structure of the netlist, is to fix one of the cells or add a dummy fixed cell to the netlist.

With regard to the detailed routing problem, we consider the *channel routing* problem. Specifically, we propose a channel router based on a hybridization of Stochastic Evolution and Tabu search Methods. Moreover, we propose to express the problem-domain information in the form of ***utility functions*** to guide the exploration of the search space. Unlike previous search heuristics based routers, the use of utility functions in our router provides a powerful tool to determine the best moves (swapping and moving nets between tracks) that guarantee convergence in shorter times.

1.5 Thesis Outline

The remaining chapters of this thesis are organized as follows. In chapter 2, the cell placement problem is described in greater details and a taxonomy of existing placement methods is presented. In chapter 3, the new formulations of the proposed *cell repeller* model are presented. Proofs of their convexity are also presented. Chapter 4 presents the *cell attractor* approach to spread the cells within the placement area without causing any excessive stretching of the nets. The *attractor-repeller* model (which combines the cell attractors and cell repellers) for global placement and a new *generic* placement algorithm based on the attractor-repeller model are also presented. Furthermore, an iterative improvement technique based on the Tabu search metaheuristic [3, 58] is presented. In chapter 5, the new placement

method is applied to the standard placement problem. Qualitative analysis of the method and comparisons to up-to-date placers are presented. Chapter 6 describes the routing problem in general and detailed routing in particular (with emphasis on channel routing problem). In chapter 7, the new utility-function based hybrid channel router is presented. Numerical results using a set of benchmarks are also presented. Chapter 8 concludes the thesis and presents recommendations and future directions.

Chapter 2

Cell Placement

Cell placement is the subtask of circuit layout which is concerned with assigning locations to cells within the chip area according to an appropriate cost function. The main objective in the placement problem is to minimize the chip area. This parameter is difficult to estimate and accordingly cost functions based on other parameters are employed. There are two prevalent cost functions: (i) *Wirelength* based cost function and (ii) *Min-cut* based cost function [52, 56]. Minimizing either cost function captures the main objective (minimum chip area). Besides minimum area, other objectives such as minimum delay, minimum clock skewness and minimum power dissipation are crucial in many VLSI applications.

Cell placement is NP-hard [24]. Attempting to evaluate every possible arrangement or configuration of the cells to determine the best one requires time proportional to the *factorial* of the number of cells[3]. Alternatively, researchers employ heuristic algorithms to obtain reasonable solutions in reasonable times. Placement heuristic algorithms or methods can be divided into two major classes: *constructive* methods and *iterative improvement* methods [52, 56]. In a constructive method,

a good placement is built in a global sense. That is, circuit description or more specifically, net-cell connections are used in constructing the placement. In iterative improvement placement, algorithms begin with initial placement and search for better configurations by repeatedly modifying the existing placement. Normally, constructive methods produce reasonable placements in a short time. On the other hand, iterative improvement methods produce high quality placements but require large amount of computational efforts. Figure (2.1) illustrates the taxonomy of the popular placement methods.

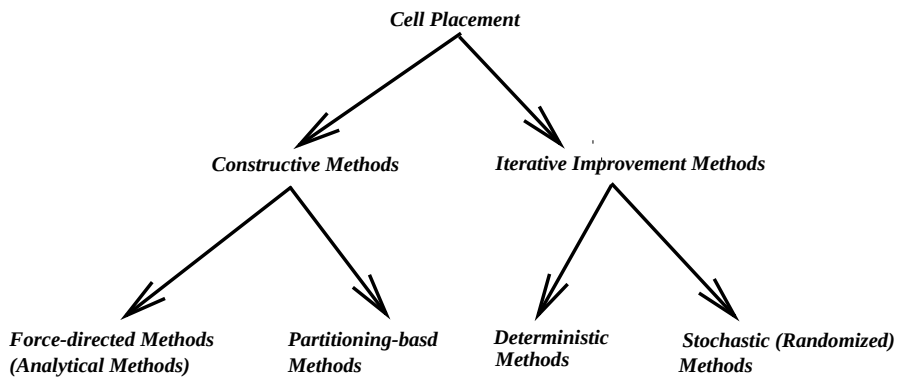


Figure 2.1: Classification of placement methods.

2.1 Constructive Placement

As we mentioned above, constructive methods employ cell-net connections to construct a placement. The resulting placement is referred to as ***global placement*** because all circuit connections are considered simultaneously while constructing the placement. Some researchers [34, 48] referred to the resulting placement as ***relative placement*** because only relative and not the final positions are typi-

cally computed. In the remaining parts of this thesis, we will use the two terms interchangeably whenever this placement is referred to.

For large circuit instances, constructive methods are preferable because (as we mentioned above) they produce reasonable placements in a reasonable time. Intuitively, an approach that combines the advantages of both methods is desirable. In fact hybrids of the two methods have been investigated [30, 17, 34, 9]. The resultant combined method typically produces high quality solutions in a reasonable time. *Analytical* placement [30, 12] and *Min-cut* placement algorithms [38, 1, 29] are the most popular constructive methods. In the analytical approach, the placement problem is formulated as a continuous (linear or quadratic) mathematical program. Continuous optimization techniques are then applied. The result is a relative (global) placement in which a cell is placed in the immediate neighborhood of the cell ideal location. Min-cut approach is based on the recursive application of bipartitioning (or quadpartitioning) algorithms [22, 35, 67]. Basically, the algorithm partitions the set of cells into two or four subsets by either a horizontal or vertical line such that the number of cut-nets between the two subsets is minimized and a certain area-based criterion is satisfied. This procedure is recursively applied to each subset until the subset contains only one cell.

2.2 Iterative Improvement Placement

Iterative improvement methods start with an initial placement (that can be randomly generated) and iteratively modifies the exiting placement in an endeavor to produce a better one. Typically, local changes in the form of moving a cell to a new location, reflecting a cell or swapping two cells are employed to perturb the existing placement and produce a new one. Iterative improvement methods differ

from constructive methods in the sense that the later does not directly use cell-net connections while determining the placement [34]. Iterative improvement methods can be classified into two categories: (i)*deterministic* and (ii) *randomized* methods. In deterministic methods, only local changes that lead to better placements are accepted. This approach works well for small instances, or if the initial solution is really good. Otherwise, it may not produce good quality solutions due to its inability to escape local minima. On the other hand, randomized algorithms accept changes that lead to better placements, and changes that lead to poor (less quality) placements are also accepted with a certain probability. Randomized algorithms are much more powerful compared to deterministic algorithms. The power of the randomized algorithms stems from their capability to escape local minima which is a direct consequence of accepting poor solutions. Simulated Annealing (SA) [10, 54] and Genetic Algorithms (GA) [13, 55] are among the randomized algorithms that have been applied to the placement problem. As a result of their ability to escape local minima, randomized algorithms produce high quality answers, but they require excessive computation time to produce those answers.

2.3 A Combination of Constructive and Iterative Methods

The fact that each of the constructive and iterative improvement methods exhibit strengths and deficiencies concerning quality and efficiency of solutions suggests that a combination (hybrid) of the two methods can be superior to each individual implementation. For constructive methods, the solution quality can still be improved if different modeling of the problem is attempted, or if different solution

methodologies are examined. In a combined method, the overall performance can be perceived as the average of the performance of the constituent methods. Thus a combined method is fast and still produces relatively high quality solutions. Normally, a placement produced by a constructive method is provided to an iterative improvement algorithm as an initial placement for further improvement. The result is saving a large amount of time while achieving good placement. A typical combination of constructive and iterative methods involves optimizing analytical formulation of the total wirelength (sum of wirelength across all nets) combined with adding more constraints and new forces to reduce cell overlap, followed by local improvement of the resulting legal placement [30, 34]. This heuristic is illustrated in Figure (2.2). For instance, in [30, 34] the constructive placement methodology iterates between minimizing a linear or quadratic formulation of wirelength and slicing (partitioning) the placement area. In each iteration, new constraints are added to the formulation to reduce cell overlap and help distribute cells among the regions resulting from the slicing process. The method terminates when the size of a partition (or region) is less than a prespecified threshold. As it has been indicated previously, the result of the global optimization is a global (relative) placement in which the location of a cell relative to where it should eventually reside is determined. The relative placement is unacceptable from physical standpoint because of the overlap among the cells. Depending on the cell design style, overlap is eliminated by *legalizing* the relative placement. In case of gate array, standard cell and FPGA design style, cells are snapped to rows, and different techniques are used for other design styles. Following the *legalization* phase, the iterative improvement begins and further improvement of the initial placement is performed to account for incorrect enforcement of some cells to non-optimal locations during the computation of the relative placement and the legalization phases.

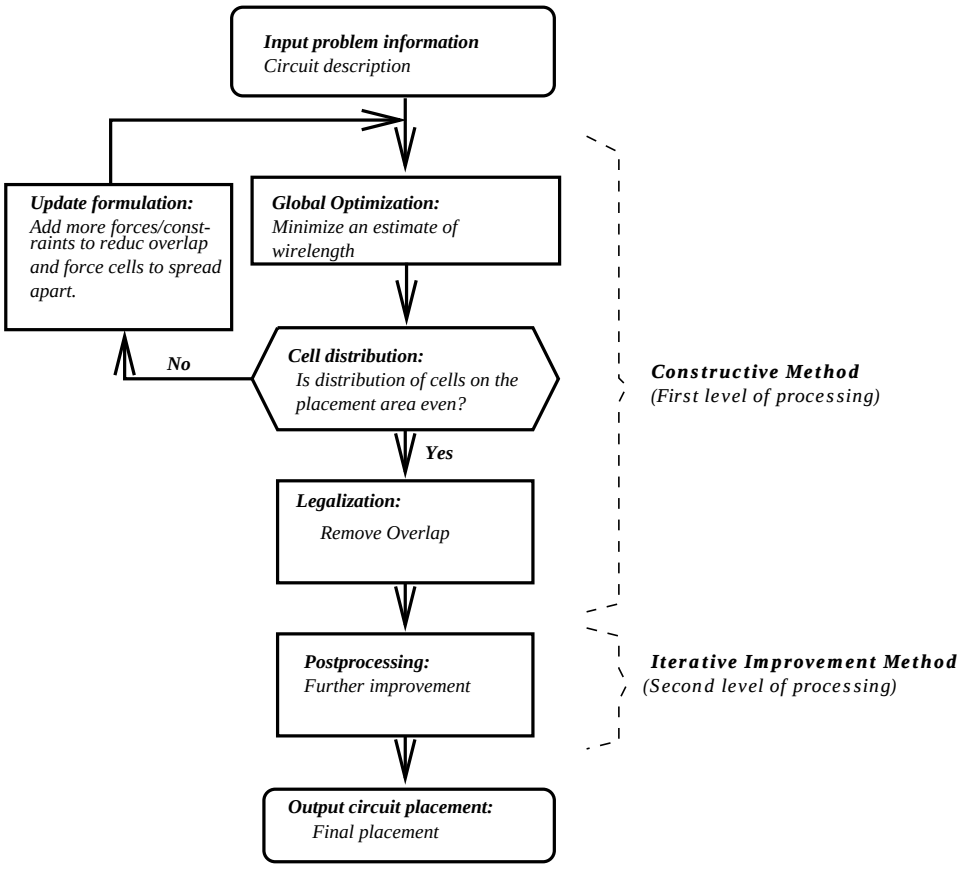


Figure 2.2: A typical combination of constructive and iterative improvement methods.

2.4 Net Length Estimation

The typical wirelength measure in VLSI placement and routing is Manhattan [52, 67]. This implies that wire segments connecting the different cells in actual placement are parallel to the x and y axes. Furthermore, the minimum wirelength of a net (in actual placement) is given by the minimum *Steiner tree* connecting the pins of the net. However, finding the minimum Steiner tree is an intractable problem owing to the large number of net-associated possible combinations of Steiner trees that need to be searched. An approximation to Steiner tree is minimum spanning tree which (like Steiner tree) has minimum length based on linear distance [67]. Again, finding the minimum spanning tree is still expensive given the fact that the number of nets in a typical modern circuit is quite large. In order to spare computational efforts, simpler approximations are introduced. The wirelength of an individual net is approximated by the *half-perimeter* of the minimum rectangle enclosing all the cells in the net. The resulting approximation is referred to as the Half Perimeter Wire Length (HPWL) of the net. Figure (2.3) shows an example of estimating the length of a net as the half-perimeter of the smallest enclosing rectangle. The total HPWL $\mathcal{H}_t$ is computed as the sum of the HPWL of the individual nets comprising the netlist. That is,

$$\mathcal{H}_t = \sum_{i=1}^{N} (H_i + V_i) \tag{2.1}$$

where H_i and V_i are the horizontal and vertical spans of net i respectively.

2.5 Test Circuits

The benchmarks (test cases) used in this work to evaluate the performance of the new placement method are presented in Table (2.1).

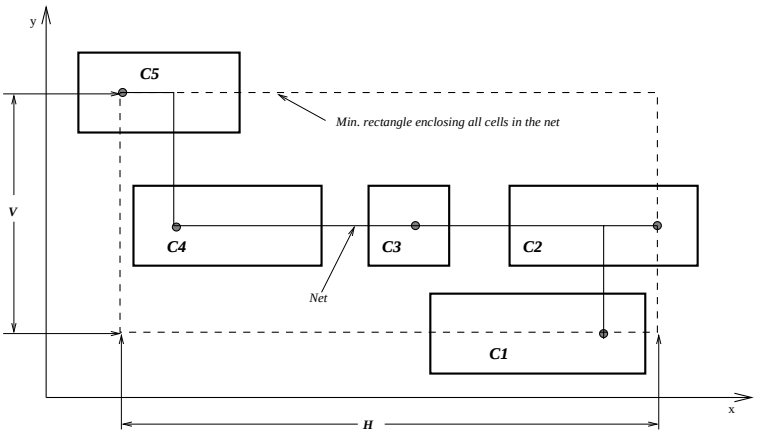


Figure 2.3: Estimating the length of a net using the half-perimeter of the minimum rectangle enclosing the cells in the net. In this example, HPWL=$H + V$.

Circuit	Cells	Pads	Nets	Pins	Rows
Fract	125	24	147	462	6
Prim1	752	81	904	5526	16
Struct	1888	64	1920	5471	21
Ind1	2271	580	2478	8513	15
Prim2	2907	107	3029	18407	28
Bio	6417	97	5742	26947	46
Ind3	15059	374	21940	176584	54
Avq.small	21854	64	22124	82601	80
Avq.large	25114	64	25384	82751	86

Table 2.1: MCNC Benchmarks used as test cases

In Table (2.1), circuit identifier, numbers of (movable) cells, fixed cells (I/O pads), nets, pins and rows are presented. All circuits are taken from the MCNC benchmark test suite [44]. Clearly, the set of test cases covers a large spectrum (or range) of circuits as long as circuit size is concerned. The motivation behind choosing such test cases with a wide range of size variability is to be able to assess the robustness of our method. Also, it is so that we will be able to draw a general conclusion when we compare our results to those reported in the literature [59, 60, 30, 32].

A useful statistic to help understand the structure of a circuit is the distribution of nets with respect to the number of cells connected by a net. To draw a general conclusion about the structure of the test circuits presented above (Table (2.1)), we assume that the test circuits constitute a single circuit with a number of nets and number of cells equal to the sum of the individual number of nets and cells in each of the constituent circuits. Figure (2.4) demonstrates the *cumulative distribution* function of the number of nets with respect to the number of cells connected by a net for the combined circuit. It is clear that nets connecting 10 to 12 cells or less, overwhelmingly dominate the set of nets. Specifically, nets connecting 4 cells or less represent around 93% of the total number of nets, and nets connecting 12 cells or less account for around 98% of the total number of nets. This indicates that typically circuits are sparse. It also indicates that when solving the placement problem, nets connecting more than 12 cells can be ignored with no major risk of inferior solution quality. Furthermore, ignoring these longer nets may save a significant amount of computational effort.

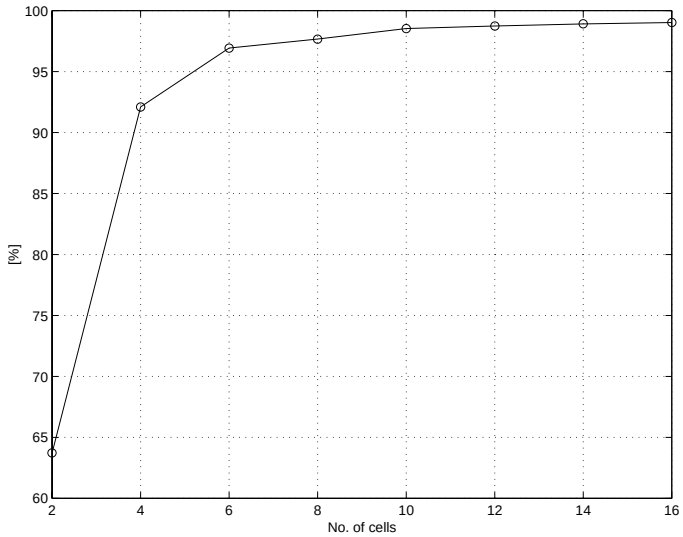


Figure 2.4: Cumulative distribution of number of nets.

2.6 Summary

Without loss of generality, placement techniques can be broadly divided into constructive and iterative improvement techniques. Constructive methods are fast, but the quality of solution they attain is not as good as that attained by iterative improvement techniques. On the other hand, iterative improvement techniques produce better solution quality, but require larger amount of computation time to produce such good quality solutions.

A combination of methods is, therefore, desirable since it makes use of the merits of the different methods. A constructive method can be used to generate an initial placement which is further improved using iterative improvement method. In such a scenario, the quality of the final placement will not be limited by the constructive method, and the computation burden of the iterative method will be reduced as a result of the quality of the initial placement.

As we indicated previously, the first step in a constructive method is typically an initial placement. In our generic (constructive) placement method, the initial placement is generated by the proposed attractor-repeller model. In the next chapter, the new convex objective functions for the *cell repeller* model are presented.

Chapter 3

The Cell Repeller Model

Global placement is the first step in a constructive placement method. Global placement entails determining the relative locations of cells while minimizing a prespecified objective. Typically, the prespecified objective is an estimate of the total *interconnecting wirelength* , or number of *cut-nets* according to whether *analytic* or *min-cut* method is used. By simultaneously accounting for all cell interconnections while determining the relative cell positions, relative placement takes a *global view* of the cell geometric locations. Legalizing (removing overlap from) global placement yields an initial placement in which each cell is placed in the vicinity (immediate neighborhood) of the location where the cell should eventually reside.

In the global placement, cells are allowed to overlap and are not restricted to certain positions as long as they fall within the bounds of the placement area. As a result, a legal solution cannot be obtained by only solving global placement. Cell positions provided by global placement give insight about the ideal positions of the cells. An extra step is essential to remove overlap and satisfy placement restrictions.

As we mentioned previously, in this work, the idea is to develop and examine

new analytical formulations for computing global placement. Therefore, in this and the subsequent chapters, we will limit our discussion to analytical methods while describing global placement.

Traditionally, analytical placement has been formulated as a mathematical program with either linear or quadratic wirelength objective function [15, 30, 34, 8, 17, 64, 43]. In this chapter we present new wirelength objective functions such that, upon minimization, overlap between pairs of connected cells is diminished if not entirely prevented. Our main contributions can be summarized as follows.

- we propose a cell repeller model to estimate the wirelength that, upon minimization, a prespecified *target distance* (TD) is maintained between each pair of connected cells.
- we propose different classes of convex functions that generate the desired repelling force.
- the fact that our models are convex suggests that any solution methodology for convex optimization can be applied.
- besides being convex, our new models are easy to tune and do not impose any restrictions on the value of the desired distance between the cells. For instance, the the target distance can be adaptively computed during the optimization process based on the information in the problem-domain.

In the next section, the placement problem and the traditional wirelength objective functions are presented.

3.1 Problem Formulation and the Quadratic Measure

A circuit is represented by a hypergraph $G(V, E)$, where the vertex set $V = \{v_1, v_2, \cdots, v_N\}$ represent the nodes of the hypergraph; i.e, set of cells to be placed, and $E = \{e_1, e_2, \cdots, e_M\}$ represents the set of edges of the hypergraph; i.e, set of nets connecting the cells. The two dimensional placement region is represented as an array of legal placement locations. The hypergraph is transformed into a graph (a hypergraph with all hyperedge sizes equal to 2) via clique model for each net. Each edge e_j is an unorder pair of vertices with a nonnegative weight w_j assigned to it. The placement task seeks to assign all cells of the circuit to legal locations such that cells do not overlap. Each cell i is assigned a location (x_i, y_i) on the XY-plane. The cost of an edge connecting two cells i and j with locations (x_i, y_i) and (x_j, y_j) is computed as the product of the squared l_2 norm of the difference vector $(x_i - x_j, y_i - y_j)$ and the weight of the connecting edge w_{ij}. The total cost $\phi(x, y)$ can then be given as the sum of the cost over all edges,

$$\phi(x, y) = \sum_{1 \leq i < j \leq N} w_{ij}[(x_i - x_j)^2 + (y_i - y_j)^2] \tag{3.1}$$

Formulation (3.1) can be written in matrix form:

$$\phi(x, y) = \frac{1}{2}\mathbf{x}^T\mathbf{C}\mathbf{x} + \mathbf{d}_x^T\mathbf{x} + \frac{1}{2}\mathbf{y}^T\mathbf{C}\mathbf{y} + \mathbf{d}_y^T\mathbf{y} + t \tag{3.2}$$

Vectors $\mathbf{x}$ and $\mathbf{y}$ denote the coordinates of the N movable cells; matrix $\mathbf{C}$ is the Hessian matrix; vectors $\mathbf{d}_x^T$ and $\mathbf{d}_y^T$ and the constant term t result from the contributions of the fixed cells. Normally the first moment constraints are added to force the distribution of the cells to be uniform around the center of the placement area. It follows that the quadratic placement model is given as:

$$\begin{aligned} \text{Min } & \phi(x, y) \\ \text{s.t. } & A_x x = b_x \\ & A_y y = b_y \\ & l_x \leq x_i \leq u_x \\ & l_y \leq y_i \leq u_y \end{aligned}$$

where A_x and A_y are $q \times n$ matrices; q is the number of regions into which the placement area has been partitioned. The $q \times 1$ vectors b_x and b_y represent the centers of the q regions. The parameters l_x, u_x, l_y and u_y are lower and upper bounds on the x and y coordinates of the cells. Clearly, the above optimization problem can be split into two 1-dimensional subproblems and each subproblem can then be solved independently.

As pointed out previously, minimizing the quadratic model for wirelength as given by (3.1) yields a placement where cells overlap. Consequently, extra efforts need to be done to remove the overlap. In all previous attempts [15, 30, 34, 8, 64, 43], the overlap problem is handled by partitioning the placement area and adding new constraints to the formulation to restrict the movement of the cells before solving another optimization problem. As a result, partitions are refined in each iterate and overlap between the cells is reduced.

Overlap between connected cells can be prevented if a target distance between each connected pair of cells is maintained. In other words, place the cells such that a lower bound on the distance between their respective locations is maintained. For example the following formulation accomplishes this aim [16].

$$\psi(x, y) = \sum_{1 \leq i < j \leq N} w_{ij}[(x_i - x_j)^2 + (y_i - y_j)^2 - d_{ij}]^2 \tag{3.3}$$

Minimizing (3.3) yields a placement with no overlap between connected cells. However, this model lacks convexity and, therefore, convergence to a global optimal

answer is not guaranteed. Besides, nonconvex optimization is not studied as well as convex optimization and identifying a suitable solution methodology is not an easy task. Another difficulty is that even if a solution methodology is found and convergence obtained, there is still no guarantee that the solution is a global minima. These difficulties motivated us to seek a different formulation for the repeller model. Such a model should be convex in the first place, besides it should be flexible enough to incorporate other performance issues (delays, clock, power dissipation, etc).

3.2 New Convex Models

In an endeavor to reduce overlap among cells through maintaining a target distance between their geometric locations, we discovered a class of functions that achieves this aim. The following theorem summarizes what exactly we have discovered.

Theorem 1 *Let* $\zeta : \Re^m \longrightarrow \Re$ *and given by* $\zeta = \| \mathbf{v} \|_2^2$ *where* $\mathbf{v} = (v_1, v_2, \cdots, v_m)$ *is m-dimensional vector, then* $\eta(\zeta) = \zeta + \rho(\zeta)$ *is convex for* $\zeta \in [1, \infty)$*, provided that the function* $\rho(\zeta) \in \pi = \{-\mathbf{ln}(\zeta), \mathbf{e}^{1-\zeta}\}$.

To prove this theorem, we use the fact that if a function $f(x) \in C^2$, then $f(x)$ is convex over a convex set Θ if the Hessian matrix of $f(x)$ is *positive semidefinite* through Θ [41]. First, we consider the case when the elements of $\mathbf{v}$ are independent, then we consider the case when the elements of $\mathbf{v}$ are dependent; specifically when $\mathbf{v} = \mathbf{u} - \mathbf{r}$ and $\mathbf{u}$ and $\mathbf{r}$ are also m-dimensional vectors.

3.2.1 Elements of v are Independent

We start by defining

$$\dot{\rho}(\zeta) = \partial\rho(\zeta)/\partial\zeta$$

$$\ddot{\rho}(\zeta) = \partial^2\rho(\zeta)/\partial\zeta^2$$

The gradient of $\eta(\zeta)$ is given by

$$\nabla\eta(\zeta) = \begin{bmatrix} 2v_1(1+\dot{\rho}(\zeta)) \\ 2v_2(1+\dot{\rho}(\zeta)) \\ \vdots \\ 2v_m(1+\dot{\rho}(\zeta)) \end{bmatrix}$$

subsequently, the Hessian matrix $\nabla^2\eta(\zeta)$ can be computed as follows:

$$\nabla^2\eta(\zeta) = B + C$$

where B and C are given as follows:

$$B = 4\ddot{\rho}(\zeta)\mathbf{v}^T\mathbf{v}$$

$$C = diag\{2(1+\dot{\rho}(\zeta))\} \tag{3.4}$$

Clearly [1] when $\ddot{\rho}(\zeta) > 0$, matrix B is a rank one matrix and, therefore, has $m-1$ zero eigenvalues and only one positive eigenvalue. Hence, matrix B is a *positive semidefinite* matrix. At $\dot{\rho}(\zeta) = -1$, the diagonal matrix C vanishes; i.e, $C = 0$. Thus, the Hessian matrix $\nabla^2\eta(\zeta)$ is a *positive semidefinite* matrix when $\dot{\rho}(\zeta) = -1$. On the other hand, when $\dot{\rho}(\zeta) > -1$, C is a positive matrix because $C > 0$. In this case C adds to the positiveness of the Hessian matrix $\nabla^2\eta(\zeta)$. As a result, the Hessian matrix $\nabla^2\eta(\zeta)$ is a *positive definite* matrix when $\dot{\rho}(\zeta) > -1$.

[1] Note that function $\ddot{\rho}(\zeta) \geq 0$ for $\rho(\zeta) \in \pi$.

We can now examine if $\eta(\zeta)$ satisfies theorem (1). In fact we only need to examine $\dot{\rho}(\zeta)$.

Case 1: $\rho(\zeta) = -\mathbf{ln}(\zeta)$

In this case $\dot{\rho}(\zeta) = -1/\zeta$. Matrix C is 0 when $\dot{\rho}(\zeta) = -1$ or $\zeta = 1$, thus in this case $\nabla^2\eta(\zeta) = B$ is *positive-semidefinite*. When $\dot{\rho}(\zeta) > -1$ or $\zeta > 1$, matrix $C > 0$ and as a consequence $\nabla^2\eta(\zeta)$ is *positive-definite*. Thus Theorem (1) is satisfied and accordingly $\eta(\zeta)$ is convex for $\zeta \geq 1$.

Case 2: $\rho(\zeta) = \mathbf{e}^{1-\zeta}$

Function $\dot{\rho}(\zeta) = -\mathbf{e}^{1-\zeta} \geq -1$ implies that $1 - \zeta \leq 0$ or $\zeta \geq 1$. Again when $\dot{\rho}(\zeta) = -1$ or $\zeta = 1$ matrix C is 0 and $\nabla^2\eta(\zeta) = B$ is *positive semidefinite*. When $\dot{\rho}(\zeta) > -1$ or $\zeta > 1$ C is *positive* and $\nabla^2\eta(\zeta)$ is *positive definite*. Thus, Theorem (1) is satisfied and $\eta(\zeta)$ is convex for $\zeta \geq 1$.

The following corollary demonstrates the effect on $\nabla^2\eta(\zeta)$ as $\dot{\rho}(\zeta) \longrightarrow 0$ or equivalently as $\zeta \longrightarrow \infty$.

Corollary 1 *As $\dot{\rho}(\zeta) \longrightarrow 0$, matrix $B \longrightarrow 0$, matrix $C \longrightarrow diag\{2\}$, and consequently the Hessian matrix $\nabla^2\eta(\zeta) \rightarrow C$. That is, $\nabla^2\eta(\zeta) \approx 2I$, where I is the identity matrix.*

Corollary (1) implies that as $\dot{\rho}(\zeta) \longrightarrow 0$ (which implies that $\zeta \longrightarrow \infty$), the Hessian matrix $\nabla^2\eta(\zeta)$ become less dependent on the variables $v_1, v_2, \cdots, v_m$. In other words, the Hessian matrix approaches the Hessian of the quadratic function ζ. Since the quadratic function is a perfectly convex function, we conclude that as $\zeta \longrightarrow \infty$, the curvature of $\eta(\zeta)$ resembles that of a quadratic function and it is fairly reasonable to assume $\eta(\zeta)$ is a perfectly convex function.

3.2.2 Elements of v are Dependent

We now turn to investigate the case where the elements of vector **v** are *not independent* variables. In particular, we are interested in the case where **v** is given as the difference between m-dimensional vectors p_i and p_j. We limit our analysis to the 2-dimensional case (see Figure(3.1)) as distances in 2-dimensional space constitute the objective function in VLSI placement, and also to simplify the analysis.

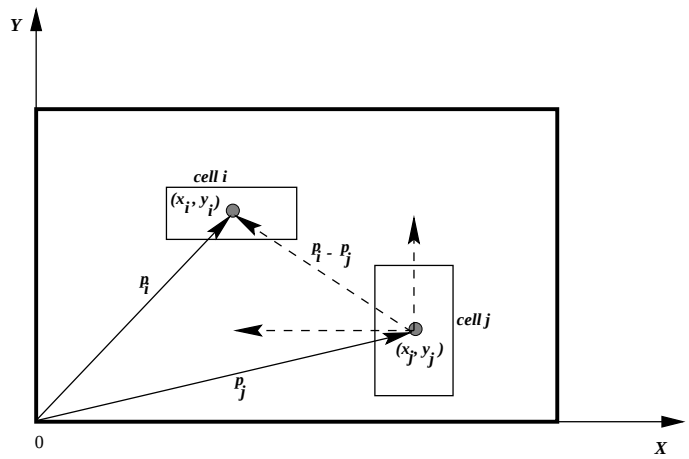


Figure 3.1: Coordinates of a two cell net.

It is well known that the standard quadratic formulation given by (3.2) is *positive semidefinite*, and is *positive definite* if one (or more) variables is (are) fixed [30, 34]. Now the question to be answered: is the function $\eta(\| p_i - p_j \|_2^2)$ convex? We will answer this question and use VLSI placement as our framework. In the context of VLSI placement p_i and p_j represent the geometric locations (x_i, y_i) and (x_j, y_j) of cells i and j. We can then be more specific and ask: is $\eta(\| p_i - p_j \|_2^2)$ convex if cell i and cell j are movable (free) cells, and what would be the effect of fixing one cell (or adding a fixed cell to the group) on the convexity of $\eta(\| p_i - p_j \|_2^2)$? Each interconnection between a pair of cells i and j contributes $\eta(\| p_i - p_j \|_2^2)$ units

to the overall objective function. In theory, the number of $\eta(\| p_i - p_j \|_2^2)$ terms constituting the objective function can be ∞. Practically, the number of terms may be very large. In both cases it is not feasible to examine such instances. However, a reasonable alternative is to limit our investigation to small instances and draw a general conclusion about larger ones. Without loss of generality and to keep the analysis simple, we consider a two cell netlist (the simplest netlist that can ever exist) and look at different scenarios with regard to whether all cells are movable or some of them are fixed. In the first scenario, we consider the case where the two

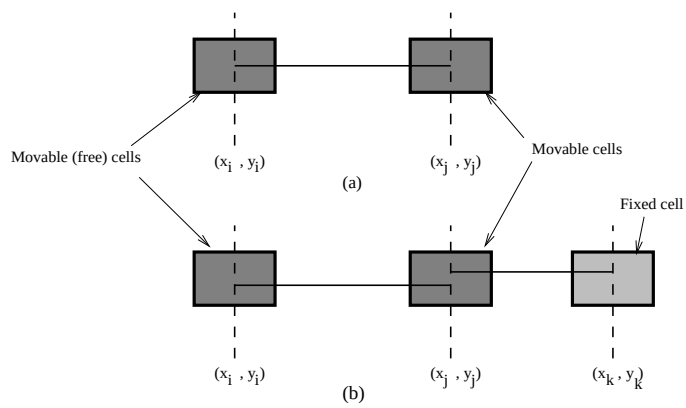


Figure 3.2: Simple two cell netlist. (a) Both cells are movable. (b) A fixed cell is added to the netlist.

cells are movable. In the second scenario, we consider the case where a fixed cell is connected to one of the two movable cells.

Scenario 1: we consider the case of two interconnected movable cells i and j, see Figure 3.2(a). The quadratic formulation of wirelength is given by

$$\zeta_{ij} = \| p_i - p_j \|_2^2 = (x_i - x_j)^2 + (y_i - y_j)^2 \tag{3.5}$$

and the objective function to be minimized $\eta(\zeta_{ij}) = \zeta_{ij} + \rho(\zeta_{ij})$. It follows that the

Hessian matrix $\nabla^2\eta(\zeta_{ij})$ is given as

$$\nabla^2\eta(\zeta_{ij}) = M_1 + M_2$$

where

$$M_1 = \ddot{\rho}(\zeta_{ij})\theta^T\theta$$

$$\theta = \begin{bmatrix} \frac{\partial\zeta_{ij}}{\partial x_i} & \frac{\partial\zeta_{ij}}{\partial x_j} & \frac{\partial\zeta_{ij}}{\partial y_i} & \frac{\partial\zeta_{ij}}{\partial y_j} \end{bmatrix}$$

and

$$M_2 = \begin{bmatrix} 1+\dot{\rho}(\zeta_{ij}) & -(1+\dot{\rho}(\zeta_{ij})) & 0 & 0 \\ -(1+\dot{\rho}(\zeta_{ij})) & 1+\dot{\rho}(\zeta_{ij}) & 0 & 0 \\ 0 & 0 & 1+\dot{\rho}(\zeta_{ij}) & -(1+\dot{\rho}(\zeta_{ij})) \\ 0 & 0 & -(1+\dot{\rho}(\zeta_{ij})) & 1+\dot{\rho}(\zeta_{ij}) \end{bmatrix}$$

Clearly matrix M_1 is a rank one matrix (with all zero's but one positive eigen value). Matrix M_1 is, therefore, positive semidefinite matrix. Matrix M_2 is positive semidefinite if $\dot{\rho}(\zeta_{ij}) \geq -1$. As a consequence, the Hessian matrix $\nabla^2\eta(\zeta_{ij})$ is positive semidefinite for $\dot{\rho}(\zeta_{ij}) \geq -1$. For $\dot{\rho}(\zeta_{ij}) < -1$, M_2 is an indefinite matrix and consequently $\nabla^2\eta(\zeta_{ij})$ can be indefinite matrix. Note that when $\rho(\zeta_{ij}) \in \pi$, $\dot{\rho}(\zeta_{ij}) \geq -1$ implies that $\zeta \geq 1$.

Scenario 2: we consider the case where one of the cells (cell j for instance) is connected to cell k with a fixed geometric location $p_k = (x_k, y_k)$, see Figure 3.2(b). For the sake of simplicity, let $(x_k, y_k) = (2, 2)$. The objective function f is then given as

$$f = \eta(\zeta_{ij}) + \eta(\zeta_{jk})$$

where ζ_{ij} is given by equation (3.5) and

$$\zeta_{jk} = \| p_j - p_k \|_2^2 = (x_j - 2)^2 + (y_j - 2)^2$$

The Hessian matrix $\nabla^2 f$ is given by

$$\nabla^2 f = M_1 + M_2 + M_3$$

where

$$M_1 = \ddot{\rho}(\zeta_{ij})\theta^T\theta$$

$$M2 = \ddot{\rho}(\zeta_{jk})\lambda^T\lambda$$

$$\ddot{\rho}(\zeta_{ij}) = \frac{\partial^2\rho(\zeta_{ij})}{\partial\zeta_{ij}^2}$$

$$\ddot{\rho}(\zeta_{jk}) = \frac{\partial^2\rho(\zeta_{jk})}{\partial\zeta_{jk}^2}$$

$$\lambda = \begin{bmatrix} 0 & \frac{\partial\zeta_{jk}}{\partial x_j} & 0 & \frac{\partial\zeta_{jk}}{\partial y_j} \end{bmatrix}$$

and

$$M_3 = \begin{bmatrix} 1+\dot{\rho}(\zeta_{ij}) & -(1+\dot{\rho}(\zeta_{ij})) & 0 & 0 \\ -(1+\dot{\rho}(\zeta_{ij})) & 2+\dot{\rho}(\zeta_{ij})+\dot{\rho}(\zeta_{jk}) & 0 & 0 \\ 0 & 0 & 1+\dot{\rho}(\zeta_{ij}) & -(1+\dot{\rho}(\zeta_{ij})) \\ 0 & 0 & -(1+\dot{\rho}(\zeta_{ij})) & 2+\dot{\rho}(\zeta_{ij})+\dot{\rho}(\zeta_{jk}) \end{bmatrix}$$

where $\dot{\rho}(\zeta_{ij}) = \frac{\partial\rho(\zeta_{ij})}{\partial\zeta_{ij}}$ and $\dot{\rho}(\zeta_{jk}) = \frac{\partial\rho(\zeta_{jk})}{\partial\zeta_{jk}}$.

Evidently for $\ddot{\rho}(\zeta_{ij}) \geq 0$ and $\ddot{\rho}(\zeta_{jk}) \geq 0$, M_1 and M_2 are rank one matrices and therefore they are *positive semidefinite matrices*. Matrix M_3 is *positive semidefinite* when $\dot{\rho}(\zeta_{ij}) = -1$ and $\dot{\rho}(\zeta_{jk}) = -1$. Also it is unmistakable that the determinant of every principal submatrix of M_3 is *positive* if $\dot{\rho}(\zeta_{ij}) > -1$ and $\dot{\rho}(\zeta_{jk}) > -1$. Matrix M_3 is therefore *positive definite* when $\dot{\rho}(\zeta_{ij}) > -1$ and $\dot{\rho}(\zeta_{jk}) > -1$. It follows that $\nabla^2 f$ is also *positive definite* when $\dot{\rho}(\zeta_{ij}) > -1$ and $\dot{\rho}(\zeta_{jk}) > -1$. Again, $\dot{\rho}(\zeta_{ij}) \geq -1$ and $\dot{\rho}(\zeta_{jk}) \geq -1$ imply that $\zeta_{ij} \geq 1$ and $\zeta_{jk} \geq 1$ (provided that $\rho(\zeta_{ij}) \in \pi$).

Explanation: Theorem 1 and corollary 1 manifest that adding the nonlinear function $\rho(\zeta)$ where $v_1, v_2, \ldots, v_n$ are independent variables, causes the Hessian matrix to be dependent on the problem variables $v_1, v_2, \ldots, v_n$. Specifically, the function $\dot{\rho}(\zeta)$ rectifies the positive definiteness of matrix C (given by equation (3.4)) which in turn regulates the positive definiteness of the Hessian matrix $\nabla^2\eta(\zeta)$. Based on what we have seen from scenarios (1) and (2), this is applicable to the case where $v_1, v_2, \ldots, v_n$ are not independent. In Scenario (1), the positive semidefiniteness of matrix M_2 is regulated by the term $\dot{\rho}(\zeta_{ij})$ which in turn regulates the semidefiniteness of the associated Hessian matrix $\nabla^2\eta(\zeta_{ij})$. Similarly, in scenario (2) the positive definiteness of M_3 is regulated by the terms $\dot{\rho}(\zeta_{ij})$ and $\dot{\rho}(\zeta_{jk})$ which in turn regulates the positive definiteness of the associated Hessian matrix $\nabla^2 f$. These results ascertain the following points:

- As in the case of traditional quadratic formulation, the existence of fixed cells adds definiteness to the Hessian matrix. Such augmentation forces the Hessian matrix to be positive definite. The existence of, at least, one fixed cell attached to, at least, one free cell is sufficient to provide adequate positiveness to the Hessian matrix. This can be explained in light of the fact that the netlist hypergraph G (presented in section 3.1) is a *connected* graph. Consequently, cells are globally interconnected and the effect of a fixed cell propagates and accordingly sustains the positive definiteness of the associated Hessian matrix. In general, the amount of positiveness added to the associated Hessian matrix is proportional to the number of fixed cells in the netlist.

- The positive definiteness of the Hessian matrix is controlled by the terms $\dot{\rho}(\zeta_{ij})$, $i, j \in \{1, 2, \ldots, N\}$, where N is the total number of cells in the netlist.

We now state the following theorem which in essence a generalization of theorem (1)

Theorem 2 *Let* $\zeta_{ij} : \Re^m \longrightarrow \Re$ *and given by* $\| p_i - p_j \|_2^2$*;* $p_i \in \Re^m$ *and* $p_j \in \Re^m$*;* $i \in \mathcal{S}_1$ *and* $j \in \mathcal{S}_2$*;* $\mathcal{S}_1 \subseteq \mathcal{S}$ *and* $\mathcal{S}_2 \subseteq \mathcal{S}$*;* $\mathcal{S}_1 \cap \mathcal{S}_2 \neq \phi$*;* $\mathcal{S} = \{1, 2, \ldots, N\}$ *, then* $\sum_{i,j} \eta(\zeta_{ij})$ *is a convex function provided that* $\zeta_{ij} \geq 1$ $\forall i$ *and* $\forall j$*, and at least one vector* p_h*,* $h \in \mathcal{S}$*, is constant.*

Geometrically, each ζ_{ij} corresponds in essence to a circle of radius $r_{ij} \in \Re$. According to Theorem (2), function $\sum_{i,j} \eta(\zeta_{ij})$ is convex if the loci of each ζ_{ij} occur outside *unit* circle. It is quasiconvex if these loci fall on the contour of the unit circle, and nonconvex if some of these loci fall inside the unit circle. Figure (3.3) depicts the different regions in $\Re^2$ for which $\sum_{i,j} \eta(\zeta_{ij})$ is convex, quasiconvex and nonconvex with respect to the unit circle. Clearly, the nonconvex region is insignificant compared to the convex region. This implies that the model can be applied to real world problems without expecting any major difficulties.

Besides Theorem (2), the following Corollary (analogous to Corollary 1) emphasizes the relationship between the Hessian matrix and ζ_{ij} as $\zeta_{ij} \to \infty$.

Corollary 2 *As* $\zeta_{ij} \to \infty$*,* $\forall i$ *and* $\forall j$*, the Hessian matrix of* $\sum_{i,j} \eta(\zeta_{ij})$ *become less dependent on the problem variables, and eventually approaches the Hessian matrix of quadratic function.*

Generally the elements of the Hessian matrix of $\sum_{i,j} \eta(\zeta_{ij})$ are combinations of variable and fixed terms. Each variable term is comprised in part by $\dot{\rho}(\zeta_{ij})$. Corollary 2 demonstrates that ζ_{ij} become large, the variable terms vanish, and the Hessian matrix become independent on the problem variables (behaves like quadratic function).

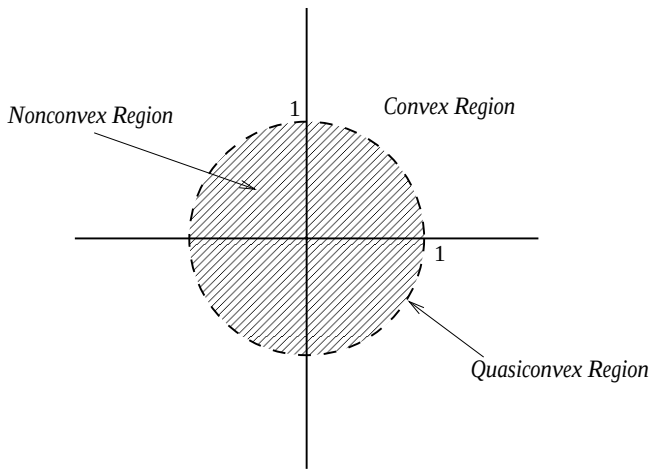


Figure 3.3: Representation of the regions in which $\sum_{i,j} \eta(\zeta_{ij})$ is convex, quasiconvex and nonconvex.

Before we conclude this section, we would like to stress again that in real circuits, the number of fixed cells is normally enough to add sufficient positiveness to the Hessian matrix.

3.3 The Repeller Model

Since $\eta(\zeta_{ij})$ is convex in the interval $[1, \infty)$, it is quite obvious to choose $[1, \infty)$ as our working interval. Function $\eta(\zeta_{ij})$ is a convex and monotonically increasing function for each $\zeta_{ij} \in [1, \infty)$.

If we let

$$z_{ij} = \frac{\zeta_{ij}}{d} = \frac{(x_i - x_j)^2 + (y_i - y_j)^2}{d} \tag{3.6}$$

where $d > 0$ is a constant, we can then present the convex repeller model for the

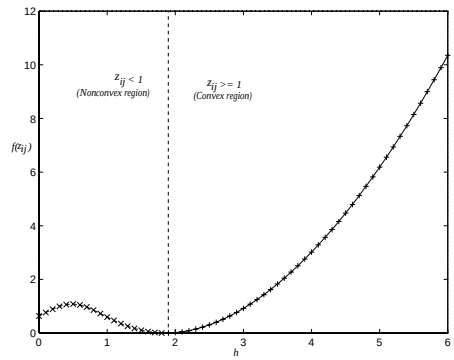


Figure 3.4: $f(z_{ij}) = z_{ij} - ln(z_{ij}) - 1$. The portion of the curve marked with "x's" represents $f(z_{ij})$ for $z_{ij} < 1$ and the one marked with "+'s" represents $f(z_{ij})$ when $z_{ij} \geq 1$. $f(z_{ij})$ is not convex for $z_{ij} < 1$. Note that we subtracted 1 so that at convergence, $z_{ij} = 1$, $f(z_{ij}) = 0$.

pair of connected cells i and j as:

$$f(z_{ij}) = \begin{cases} \eta(z_{ij}) & \text{if } z_{ij} \geq 1 \\ 0 & \text{otherwise} \end{cases} \tag{3.7}$$

where, again from Theorem (1), $\eta(z_{ij}) = z_{ij} + \rho(z_{ij})$ and $\rho(z_{ij}) \in \{-\mathbf{ln}(z_{ij}) - 1, \mathbf{e}^{1-z_{ij}} - 2\}$.

Figure (3.4) and Figure (3.5) illustrate, in 1-dimension, examples of $f(z_{ij})$ for $\rho(z_{ij}) = -\mathbf{ln}(z_{ij}) - 1$ and $\rho(z_{ij}) = \mathbf{e}^{1-z_{ij}} - 2$ respectively. In these examples, $z_{ij} = \| \bar{x} + h\Delta\bar{x} - \bar{y} + h\Delta\bar{y} \|_2^2$, $h \in \{0, 0.1, 0.2, \ldots, 6\}$, and vectors $\bar{x}, \Delta\bar{x}, \bar{y}$ and $\Delta\bar{y}$ are randomly generated. Note that the nonconvex part of $f(z_{ij})$ is also shown.

The fact that $f(z_{ij})$ is flat in the interval $z_{ij} \in [0, 1]$ implies that $f(z_{ij})$ has multiple solutions in this region. However, line search methods concludes the search when the first optimal answer is encountered. Thus, if the initial solution is outside the flat region, the search will be concluded when $z_{ij} = 1$. It follows that at

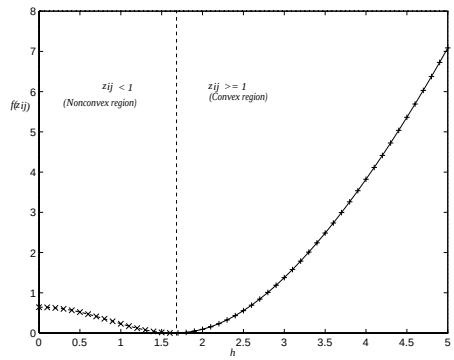


Figure 3.5: $f(z_{ij}) = z_{ij} + e^{1-z_{ij}} - 2$. The portion of the curve marked with "x's" represents $f(z_{ij})$ for $z_{ij} < 1$ and the one marked with "+'s" represents $f(z_{ij})$ when $z_{ij} \geq 1$. Again, $f(z_{ij})$ is not convex for $z_{ij} < 1$. Again, note that we subtracted 2 from $f(z)$ so that at convergence , $z_{ij} = 1$, $f(z_{ij}) = 0$.

convergence ($z_{ij} = 1$) the square of the distance equals a target estimate of d units. The cell repeller based placement model can then be given as

$$Min \quad \mathcal{F}(\mathbf{z}) = \sum_{1 \leq i < j \leq N} w_{ij} f(z_{ij}) \tag{3.8}$$

$$l_x \leq x_i \leq u_x$$

$$l_y \leq y_i \leq u_y$$

where $\mathbf{z} = \{z_{ij} : i, j = 1, 2, \ldots, N\}$ and w_{ij} is the connectivity weight between cells i and j.

3.4 Summary

In this chapter, new formulations for estimating wirelength in global placement have been proposed. The new model is referred to as the cell repeller model since, upon minimization, pairs of connected cells are placed such that their geometric locations are spatially separated. Complete theoretical proofs for the convexity of the new class of formulations have been presented. The fact that the new formulations are convex guarantees convergence to a global minima using any convex optimization methodology. The cell repeller model prevents overlap between connected cells, but cells sharing no common nets still overlap as they are not accounted for in the repeller model. Thus, uniform distribution of the cells on the placement area cannot be obtained using only the repeller model.

In the next chapter, a new mathematical scheme to attain uniform distribution of the cells within the placement area without causing any stretching of the nets is presented. The new mentioned mathematical scheme is what we, previously, referred to as cell attractors. Furthermore, a combined model (previously, referred to as the Attractor-Repeller model) for global placement and a new placement method based on the combined model are presented. An iterative improvement technique to further improve the initial global placement is also presented.

Chapter 4

The New Generic Placement Method

In this chapter, we present our new generic placement method. Additionally, we describe heuristics to legalize the global placement, and to further improve the legal global placement.

Before we present the details of the new method, we would like first to shed some light on the partitioning approach and how it has been used to spread the cells within the placement area. The rationale is that, previous approaches for analytic placement [30, 34, 15, 2, 63] rely on exhaustive partitioning and hard constraints to uniformly distribute the cells and fully utilize the placement area . However, the partitioning approach suffers from several major deficiencies. Our aim is to disclose the inherent deficiencies of the partitioning approach and demonstrate how these deficiencies have been addressed in the new placement method.

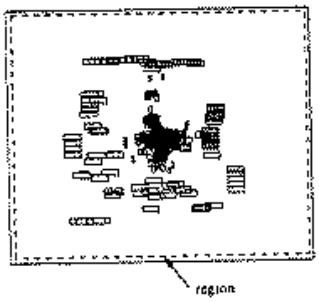
The method Gordian [30] has been the most successful analytic placer that can handle large design instances [30, 34, 15, 60, 43, 32, 2, 63]. Accordingly, we

consider the method of Gordian as our framework while describing the partitioning approach.

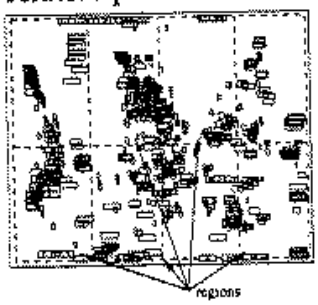
4.1 Cell Spreading and the Partitioning Approach

In Gordian, the placement problem is formulated as a sequence of quadratic programming derived from the net-cell interconnections (netlist) according to equation (3.2). The method iterates between minimizing a quadratic wirelength and partitioning of the placement area. In each iteration, the cell set and placement area are recursively partitioned. The subsets of cells are assigned to the different partitions (regions) such that the size (in terms of the total area of cells in a partition) is the same. Moreover, new constraints are imposed on each subset of cells. The new constraints restrict the freedom of cells in the subsequent global optimization iterations and eventually drive cells near their final locations. The decision of how cells are partitioned and assigned to the different regions on the placement floor is based on the concurrent positioning of the cells. It is also based on the number of cut-nets between the different regions; i.e, number of nets crossing cut-lines between the different partitions. After each iteration of global optimization, a refined global placement is obtained.

In the first iteration, cells are only required to fall within the placement area, Figure (4.1-a). In other words, the overall placement area is the only region available for cell positioning. Although cells are allowed to fall anywhere within the placement area, there is a natural tendency for cells to cluster around the *center* of the placement area. This is chiefly because connected cells are pulled together as a result of the wirelength minimization process. Furthermore, some of the cells are moved away from the center of the region as a result of the attraction forces



(a) Typical relative placement with one region.



(b) Typical relative placement with eight regions.

(c) Typical relative placement with > 100 regions.

Figure 4.1: Examples of relative placements [34].

originated by the I/O pads (fixed cells) located on the periphery of the placement area. It is quite obvious that the placement obtained in the first iteration is not adequate for creating legal placement as the amount of overlap among the cells is substantial and cells are not essentially near their final locations, Figure (4.1-a). In the second iteration, the placement area is divided into 4 disjoint regions and cells are distributed equally among them. Generally, in the nth iteration, the placement area is divided into 2^{2n} regions [1] and, again cells are distributed equally among them, Figure (4.1-c). As we pointed out previously, a global optimization is performed in each iteration resulting in a reduction of cell overlap and better utilization of the placement area. The slicing process is terminated if the size of a partition is less than a prespecified threshold [30].

Despite the fact that the partitioning approach usually accomplishes decent cell spreading and utilization of the placement floor, it suffers from many drawbacks. The following are among the several major shortcomings of the approach.

- solution quality may be deteriorated as a consequence of possible erroneous assignment of cells to the different regions (partitions).
- the infeasibility of correcting any erroneous assignments as a result of restricting the movement of cells in a partition within the partition.
- the need for hard constraints to force cells to spread within the placement area.
- the number of hard constraints increases as the number of partitions increases. The result is a much harder problem.

[1] Provided that iteration index n is given as $n = 0, 1, 2 \cdots$. In other words, first iteration index is 0, second iteration index is 1, and so on.

- since determining the best size for a partition is a hard problem, sizes of various partitions are assumed equal. However, equal partition-size may result in a major deterioration in the final answer.

These deficiencies motivated us to seek an alternative to the partitioning approach. An approach is desired that is robust in terms of quality of cell spreading, area utilization, and efficient in accomplishing these objectives. Specifically in our approach, new repelling forces have been added to prevent overlap among connected cells. Unfortunately, cells sharing no common nets still fold on top of each other as they are not accounted for in the repeller model. In fact, the repeller engine can achieve cell spreading if the target distance parameter d in equation (3.6) is chosen to be relatively large with respect to the average cell width or average cell height. But, cell spreading achieved this way tends to stretch short nets and accordingly deteriorate the total wirelength.

Cell spreading needs to be done delicately over the course of several iterations so that excessive stretching of short nets is avoided. In each iteration, highly connected cells need to be displaced to fill some sparse regions on the placement area. In the following section, we describe our new approach to cell spreading.

4.2 New Method for Cell Spreading

As we just have stated, hinging on the repeller engine to spread cells achieves the goal at the expense of stretching short nets and accordingly deteriorating total wirelength. To prohibit this, we propose adding new forces to the repeller model to regulate cell spreading such that excessive net stretching is avoided and adequate distribution of cells is attained. The basic idea of our approach is to force cells

to spread over the placement area without restricting their movement. In other words, no constraints are imposed on cell positioning, and cell spreading is entirely driven by the natural movement of cells during the global optimization of the total wirelength. Specifically, the new forces attract cells to sparse regions within the placement area. As opposed to the partitioning approach, these forces encourage a cell to move to a sparse region in a direction conforming with the direction of the cell movement. The approach involves adding *dummy fixed* cells to the less dense regions of the placement floor and establishing connections between these new fixed cells and the movable cells in the dense regions. The new connections are, in fact, new nets added to the netlist. It follows that during the computation of the global placement, the dummy fixed cells exert additional forces on the movable cells and force them to move towards the less dense regions where the dummy fixed cells reside. Thus, movable cells fill the sparse regions, and accordingly the intensity of cell spreading increases in each iteration.

The first step involves identifying dense and sparse regions on the placement area. In the following section, we present how these regions are identified.

4.2.1 Identifying Dense and Sparse Regions

The procedure of identifying dense and sparse regions depends on the cell distribution and the variability of cell area (as cell width is not the same for all cells). Specifically, for each cell i with geometric location (x_i, y_i), an $\ell_w \times \ell_h$ rectangular window w_i centered at (x_i, y_i) is imposed on the placement floor and the total area A_a of all cells with centers enclosed by w_i is computed; i.e:

$$A_a = \sum_{k=1}^{r} c_w^k c_h^k$$

where c_w^k and c_h^k are the width and height of cell k and r is the number of cells enclosed by window w_i. Next, region $\mathcal{R}_i$ surrounded by w_i is regarded sparse only if

$$A_a < A_w$$

where

$$A_w = \ell_w \ell_h$$

In our implementation, ℓ_w and ℓ_h have been expressed as linear functions of the chip width W and chip height H and adaptively varied from iteration to another; i.e:

$$\ell_w = \alpha_t W$$

and

$$\ell_h = \alpha_t H$$

where α is a constant that is updated from iteration to iteration according to [2]

$$\alpha_{t+1} = 0.95\alpha_t, \quad t = 1, 2, \cdots, \mathcal{I}$$

where t is the iteration number, $\mathcal{I}$ is the total number of iterations and $\alpha_1 = 0.1$. Cells that have been collapsed in a sparse region are not considered when new sparse regions are being identified. Each sparse region $\mathcal{R}_i$ is then split into 4 quadrants and the center of the sparsest quadrant (in terms of number of cells that fall inside a quadrant) (x_a^i, y_a^i) become a center of a new cell attractor. A cell attractor is, merely, a *dummy fixed* cell located at (x_a^i, y_a^i). Figure (4.2) illustrates an example of sparse and dense regions.

Let

$$\mathcal{A} = \{(x_a^1, y_a^1), (x_a^2, y_a^2), \cdots, (x_a^q, y_a^q)\}$$

[2] We need to stress that this formula is empirical and it is entirely based on experimentation with the benchmarks.

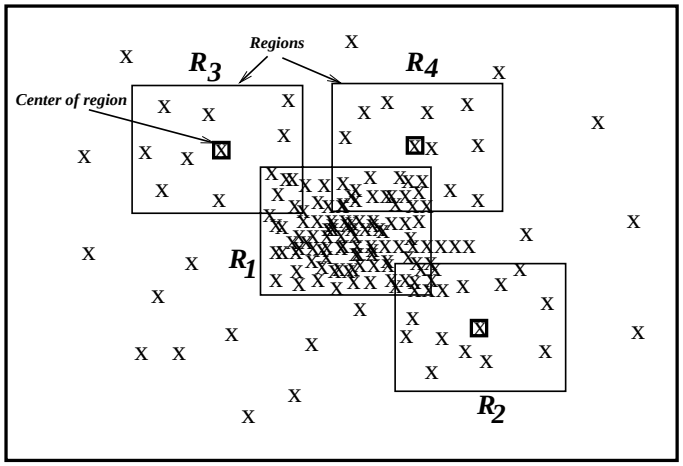


Figure 4.2: Region R_1 is dense, while regions R_2, R_3 and R_4 are sparse.

be the set of locations of the q cell attractors. The attractors in $\mathcal{A}$ are used to divide the set of cells among the different identified sparse regions on the placement floor. Precisely, for each cell i an attractor is selected based on a certain *cell-attractor* assignment criteria. Subsequently, a connection is established between cell i and the selected cell attractor. In other words, a new two-cell net is added to the netlist of the circuit. The details of the cell-attractor assignment are presented in the following section.

4.2.2 Cell-Attractor Assignment

When assigning cells to sparse regions, the aim is to attain cell spreading while preventing any possibility of excessive stretching of short nets in subsequent iterations. As we indicated previously, this objective is difficult to achieve unless spreading is done delicately over several iterations. In general, the deterioration in net wirelength is correlated with the distance between its cells and the attracting dummy cells in the sparse regions. Greater distances normally correspond to excessive net

stretching. Thus, the criteria to assign a cell to a sparse region (or equivalently cell attractor) should be based on how many distance units that separates the geometric location of that cell from the geometric location of the dummy attracting cell in that particular sparse region. Based on this, we propose general *cell-attractor* assignment criteria that can be expressed in terms of the distance between the geometric locations of the cells and the attractors. Specifically, the criteria are based on the well known inequality between the *minimum*, *harmonic mean*, *geometric mean*, *arithmetic mean* and *maximum* of a set or a sequence of positive numbers [45]. To formalize the discussion, let

$$\mathcal{D}^i = \{d_1, d_2, \cdots, d_q\}$$

be the sequence or set of distance between the geometric location of cell i and that of each attractor (dummy cell) in $\mathcal{A}$. To facilitate the analysis, we assume $\mathcal{D}^i$ is the set of distance between the geometric location of cell i and the attractors in the x-direction. For instance

$$d_1 = |x_i - x_a^1|$$

is the distance (in the x-direction) between cell i and the attracting dummy cell with coordinates (x_a^1, y_a^1).

The harmonic mean of the nonnegative sequence of numbers $\mathcal{D}^i$ is defined as

$$H(\mathcal{D}^i) = \frac{q}{1/d_1 + 1/d_2 + \ldots + 1/d_q}$$

The geometric mean of the same sequence is defined as

$$G(\mathcal{D}^i) = (d_1 d_2 \ldots d_q)^{1/q}$$

and the arithmetic mean is given by

$$A(\mathcal{D}^i) = \frac{d_1 + d_2 + \ldots + d_q}{q}$$

We also define

$$D_{min} = min\{d_1, d_2, \ldots, d_q\}$$

and

$$D_{max} = max\{d_1, d_2, \ldots, d_q\}$$

For the finite sequence of positive numbers $\mathcal{D}^i$, we have [45]

$$D_{min} \leq H(\mathcal{D}^i) \leq G(\mathcal{D}^i) \leq A(\mathcal{D}^i) \leq D_{max} \tag{4.1}$$

with equality if and only if

$$d_1 = d_2 = \ldots = d_q$$

Inequality (4.1) provides a set of intact criteria to control the degree of cell spreading and accordingly the extent of net stretching. For instance, if D_{min} is used as a cell-attractor assignment criterion, cell i will be connected to an attractor in the closest zone of sparse regions. In such case, cell i would be displaced by a relatively small distance with respect to its most recent geometric location while computing the relative placement in the subsequent iteration. Thus, excessive stretching in its nets is not highly likely to take place. If $H(\mathcal{D}^i)$ is employed as a cell-attractor assignment criterion, each cell i will be assigned to a cell attractor located in a sparse region that is at least (or it can be at most) $H(\mathcal{D}^i)$ units from its most recent location, depending on whether $H(\mathcal{D}^i)$ is used as lower (or upper) bound on the distance separating cell i from the different sparse regions. Hence, more spreading but more stretching in short nets is expected compared to the previous case.

To make the discussion formal, let $\mathbf{s}_1$, $\mathbf{s}_2$, $\mathbf{s}_3$ and $\mathbf{s}_4$ be disjoint subsets such that

$$\mathcal{D}^i = \cup_{j=1}^{4} \mathbf{s}_j$$

and are given by

$$\mathbf{s}_1 = \{d_k : D_{min} \leq d_k < H(\mathcal{D}^i), k = 1, \ldots, q\}$$

$$\mathbf{s}_2 = \{d_k : H(\mathcal{D}^i) \leq d_k < G(\mathcal{D}^i), k = 1, \ldots, q\}$$

$$\mathbf{s}_3 = \{d_k : G(\mathcal{D}^i) \leq d_k < A(\mathcal{D}^i), k = 1, \ldots, q\}$$

$$\mathbf{s}_4 = \{d_k : A(\mathcal{D}^i) \leq d_k \leq D_{max}, k = 1, \ldots, q\}$$

Each subset or subsequence $\mathbf{s}_j$ corresponds in essence to a group of cell attractors or equivalently a group of sparse regions. In fact, each group of sparse regions represents a sparse zone on the placement floor. The group of sparse regions located at distances given in $\mathbf{s}_1$ represents the closest sparse zone to cell i. Subset or subsequence $\mathbf{s}_2$ corresponds to the next closest sparse zone to cell i and so on.

Clearly, inequality (4.1) provides an ordering of the sparse zones according to how far they are from cell i. It follows that, a cell attractor can be selected based on inequality (4.1) and the corresponding ordering of the sparse regions given by $\mathbf{s}_1, \mathbf{s}_2, \mathbf{s}_3$ and $\mathbf{s}_4$. Dispersion of the distance sequence $\mathcal{D}^i$ gives clues on how to select a cell attractor for cell i. Specifically, the *variability* $\varrho(\mathcal{D}^i)$ of sequence $\mathcal{D}^i$

$$\varrho(\mathcal{D}^i) = \frac{\sigma(\mathcal{D}^i)}{A(\mathcal{D}^i)}$$

where $\sigma(\mathcal{D}^i)$ is the standard deviation of sequence $\mathcal{D}^i$ given by

$$\sigma(\mathcal{D}^i) = \sqrt{\frac{1}{q}\sum_{k=1}^{q}(d_k - A(\mathcal{D}^i))^2}$$

can be utilized to characterize the dispersion of the distance sequence. Accordingly $\varrho(\mathcal{D}^i)$ provides clues on a suitable zone of sparse regions where cell i can be displaced to. If $\varrho(\mathcal{D}^i)$ is small, then any group of cell attractors can be chosen without expecting any major change in the solution[3]. On the other hand, if $\varrho(\mathcal{D}^i)$

[3] if $\varrho(\mathcal{D}^i)$ is zero or equivalently $\sigma(\mathcal{D}^i)$ is zero then cell i is at equidistant from all cell attractors.

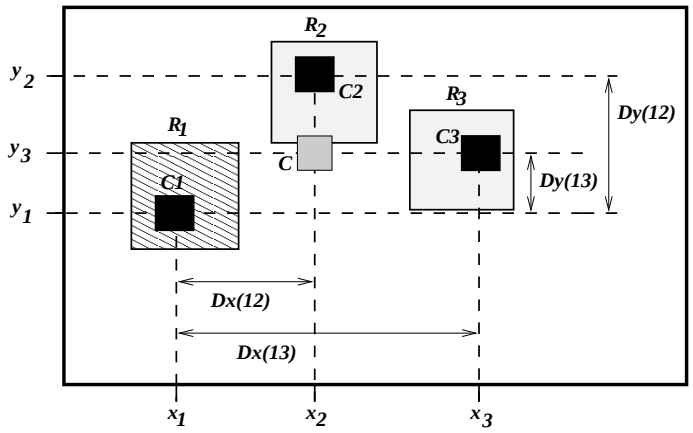


Figure 4.3: Region R_1 is dense while regions R_2 and R_3 are sparse. Cell C_1 located in R_1 with geometric location (x_1, y_1) is hooked to the dummy fixed cell C with coordinates (x_2, y_3) where x_2 and y_3 are the x and y coordinates of C_2 and C_3 (which are the closest cells to C_1 in the x and y directions respectively). Note that $Dx(ij)$ and $Dy(ij)$ are the distances in the x and y directions between cell i and cell j.

is relatively large, then a group can be selected depending on the circuit design style, size and netlist sparsity. For instance, thresholds of $\varrho(\mathcal{D}^i)$ can be designated for each group of cell attractors based on experimentation with different problem size and design style.

In our implementation, we applied the new placement method to *standard-cell* design style which is widely used in ASIC (Application Specific Integrated Circuits). In this design style, as we pointed out in previous chapters, cells are rectangular in shape with same height but not necessarily same width. Also, short nets represent the majority of the nets in this design style, see Figure (2.4). Based on our experience, we found that connecting cells to cell attractors corresponding to the distance subsequence $\mathbf{s}_1$ (closest zone of sparse regions) yields the least net stretching and quite good cell spreading. Accordingly, the results reported in this thesis (specifically, in chapter 5) are all based on cell spreading obtained through establishing connections between each cell and the closest dummy fixed cell in the closest zone of cell attractors.

In the following section, we present the complete attractor-repeller model for global placement.

4.3 The Attractor-Repeller Model

We are now in a position to give the full Attractor-Repeller model for the global placement; i.e:

$$Min \quad \mathcal{F}(\mathbf{z}) + g(x) + h(y) \tag{4.2}$$

$$s.t$$

$$l_x \leq x_i \leq u_x$$

$$l_y \leq y_i \leq u_y$$

Parameters l_x, l_y, u_x and u_y are lower and upper bounds on x and y. The first term, $\mathcal{F}(\mathbf{z})$, represents the repelling terms or shortly the *repellers* and is given by equation (3.8). We have already described in details how the cell repeller works when we presented and analyzed the new wirelength convex models in chapter 3. The second and the third terms, $g(x)$ and $h(y)$, represent the attracting terms or simply the *attractors*. Minimizing the distance between two connected cells corresponds to pulling these cells together. If the position of one of the cells is fixed, then the free cell will be pulled towards the fixed cell and (as we pointed out previously), this is why the fixed cell is referred to as cell attractor. We repeat again, for each movable cell, a connection is established with the closest cell attractors in the x and y directions. It follows that, the geometric location of the cell attractor to which a movable cell is hooked with is given by the x and y coordinates of the closest cell attractor in the x and y directions respectively, see Figure (4.3). In other words, Each movable cell i is assigned to an attractor with a coordinate (x_a^μ, y_a^ν) where μ and ν are the closest attractors to cell i in the x and y directions respectively. Accordingly, $g(x)$ and $h(y)$ are given as

$$g(x) = \sum_{1 \leq i \leq N} min\{(x_i - x_a^1)^2, \cdots, (x_i - x_a^q)^2\}$$

$$h(y) = \sum_{1 \leq i \leq N} min\{(y_i - y_a^1)^2, \cdots, (y_i - y_a^q)^2\}$$

Before we conclude this section, we would like to point out that throughout the remaining parts of the thesis, we will refer to formulation (4.2) as the Attractor-Repeller (AR) formulation (or model), and to the new placement method (based on the AR model) as the **A**ttractor-**R**epeller **P**lacer (**ARP**).

4.4 The Attractor-Repeller Placer: Basic Algorithm

Figure (4.4) illustrates the flow of the new placement algorithm ARP. Following the parsing of the circuit information, the quadratic formulation of wirelength is solved to obtain an initial placement. Obviously, the majority of cells are on top of each other with only a small portion shifted towards the boundaries of the placement area as a result of the attracting forces due to the I/O pads. In the subsequent iterations, the algorithm proceeds iteratively via minimizing the AR model. In each iteration, following the termination of the global optimization, the AR model is updated as a result of the new connections between the movable cells and the cell attractors. Specifically, based on the resulting cell positioning, dense and sparse regions are identified according to the window-based technique presented previously. Next, the global placement is legalized through snapping the cells to the rows (section 4.5). The next step involves improving the global placement slightly through a sequence of cell swapping and cell displacement within and among the different rows. The idea behind perturbing the global placement for better positioning of the cells is to increase the likelihood of connecting cells to cell attractors in sparse regions that enclose, or at least, located near their final ideal positions. In other words, by improving the current cell positioning, we hope to displace highly connected cells to the same zone on the placement floor so that in the subsequent iteration they end up hooked to the same cell attractors. In each iteration, new attractors are created and attractors from the previous iteration are deleted. As depicted in Figure (4.4), the sequence of creating attractors, establishing new connections with cell attractors and, solving the updated AR model continues until the termination criteria is met. Specifically, the algorithm stops when the number of iterations

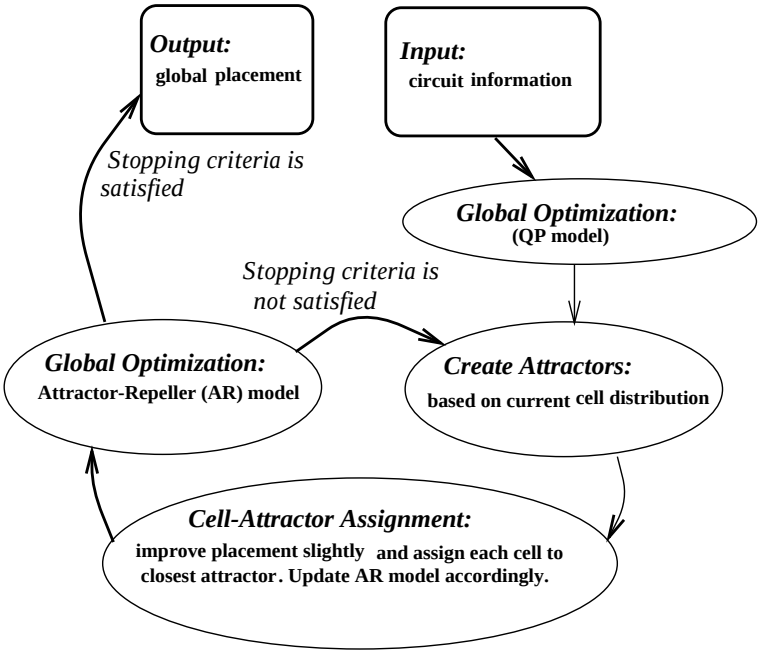


Figure 4.4: An outline of the new global placement method **ARP**.

exceeds an upper bound $\mathcal{K}$, or if the *ratio* of the total area of the identified sparse regions to that of the placement area is $< \kappa\%$. Experimentally, we found that $5 \leq \mathcal{K} \leq 8$ and $\kappa = 10\%$ are quite sufficient to yield uniform cell spreading and accordingly adequate utilization of the placement area.

The solution methodology used is a *quasi-Newton* nonlinear algorithm that calculates a step direction based on the gradient and an approximation to the Hessian [11]. It then performs a line search along that direction to minimize the objective function, generating a new iterate. The optimization stops if the difference in the objective function values over two successive iterations is sufficiently small. The Hessian approximation is generated using the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm. This method has the advantage of having a super-linear convergence rate.

The resulting cell positions do not constitute a valid placement since there is still minor overlap and cells are not positioned within the rows as required. It is therefore essential to "legalize" the placement by further adjusting the cell positions to meet the placement restrictions.

In the next section, we describe a simple heuristic to remove remaining cell overlap and create a legal global placement.

4.5 Global Placement Legalization

The result of the global optimization is a relative placement where each cell is placed at or near its final destination. The next step is to adapt the relative placement to the specific design style (standard cell, macro-cell, ect.). In standard cells or row-based design style, cells are snapped to rows. In macro-cell design style, an optimization of the area utilization is needed and cells need to be packed in a compact slicing structure.

In this thesis, the focus is on standard cells. In this design style, cells are of same height but not necessarily same width. The chip area is determined by the width of the routing channels between the rows and lengths of the rows. The objective is to accomplish a smaller chip area by reducing channel widths and not exceeding the maximum row length. The process of snapping cells to rows is called *legalizing* the relative placement, or shortly *legalization*. In this work, we used a simple heuristic for legalization which was proposed by Dunlop and Kernighan [1]. In this heuristic, cells are sorted based on their y-coordinates and divided into k subsets (k is the desired number of rows). Each subset is then assigned to a row while keeping the row length nearly equal. In fact, cells are allowed to move between the rows in order to keep the row lengths equal. The sequence of the cells within the rows

is determined based on their x-coordinates. In other words, cells are positioned adjacent to each other from left to right within the row based on their sorted x positions. The complexity of this heuristic is $O(n \log n)$ since it is based on the sorted cell positions [1].

The legal (global) placement is a good placement, however, further improvement of the placement is still required to account for any incorrect enforcement of some cells into non-optimal positions during the global optimization or the legalization phases.

In the following section, we describe an iterative improvement technique that was developed by other members [3, 58] in our group, and was successful to a decent extent in enhancing the overall quality of the initial global placement.

4.6 Iterative Improvement Method

The method is based on a meta-heuristic technique known as ***Tabu*** Search [3, 58]. Before we describe the method, we present a summary on Tabu search.

4.6.1 Tabu Search

The basis for Tabu search may be described as follows [25, 27]. An objective function f has to be minimized on a set X of feasible solutions. A neighborhood $N(S)$ is defined for each solution S in X. The set X and the definition of the neighborhood $N(S)$ induce the solution space. Tabu search is basically an iterative procedure which starts from an initial solution and tries to reach an optimal solution by moving step by step in the solution space Ω. Each step consists of generating a collection V^* of solutions in the neighborhood $N(S)$ of the current solution S, and

then moving to the best solution $\hat{S}$ in V^* even if $f(\hat{S}) > f(S)$. V^* is obtained by applying one or several moves to S.

A risk of cycling exists as soon as a solution $\hat{S}$, worse than S is accepted. In order to prevent cycling to some extent, modifications which would bring the search back to a previously visited solution should be prohibited. However, it may sometimes be useful to come back to an already visited solution and continue the search in another direction from there. This is realized in Tabu search by keeping a list T containing the last k modifications (k may be fixed or variable). Whenever a modification is made for moving from S to $\hat{S}$, its reverse (i.e, the modification which would generate S) is considered *tabu*. Sometimes it is desirable to suspend the tabu status of a move if it seems that such a step may be useful. This is called *aspiration* and it is used to temporarily release a solution from its Tabu status. A major feature of aspiration is that: it increases the flexibility of the algorithm while preserving the basic features that allow the algorithm to escape local minima and avoid cycling. Different applications employ different aspiration criteria.

Stopping criteria can be also application dependent. However, in general, if a lower bound f^* on the minimum value of f is known, then the process may be interrupted when the value of the current solution is close enough to f^*. Moreover, the procedure is terminated if no improvement is made during a given number of iterations.

The following is a general description of the Tabu search method:

Tabu search algorithm:

S = initial solution in X;

numOfIter = 0; (current iteration)

$S^* = S$; (best solution)

$T = 0$;

initialize the aspiration level.

while $(f(S) > f^*)$ **and** (numOfIter $<$ maxNumOfIter)

generate a set of neighbor solutions V^* such that

moves are not in the Tabu list or else $f(S_i) < f(S^*)$;

choose best solution $\hat{S}$ in V^*;

update Tabu list and aspiration level;

if $f(\hat{S}) < f(S^*)$ **then**

$S^* = \hat{S}$;

endif

$S = \hat{S}$;

endwhile

Tabu search tends to be very aggressive in regions where good solutions are likely to be found, while spending little time in regions that are less attractive.

4.6.2 Tabu Search for Placement Iterative Improvement

In the context of VLSI placement, the new solution S' is obtained from S by the following procedure. First, cell c_i is randomly selected. Then all target cells $c_j \in \mathcal{N}_\delta(c_i)$ ($\mathcal{N}_\delta(c_i)$ the set of cells located in the neighborhood of c_i) are exhaustively examined for interchange with cell c_i. The gains are estimated from the difference of the objective function $f(S)$ before and after the interchange. The best target cell c_j associating with the maximum gain is selected for the interchange with cell c_i if the pair of cells (c_i, c_j) is not in the Tabu list T. The new solution S' is obtained by interchanging a pair of cells (c_i, c_j), even if the gain is *negative*. In

determining whether a move is Tabu or not, only one Tabu list T containing the $|T|$ last cell pair interchanges is used. Such a list is sufficient to prevent cycling. The Tabu list T is treated as a circular list. Thus, the addition of cell pair removes the cell pair recorded in its position $|T|$ interchanges ago. Note that the key step in the Tabu method for placement is to find a *good solution* rather than to find the *best solution* in the neighborhood $N(S)$. Given the parameter δ and cell c_i, the complexity for examining the cells $c_j \in \mathcal{N}_\delta(c_i)$ is constant. But given the parameter δ, the complexity for examining the cells $c_j \in \mathcal{N}_\delta(c_i)$ for all cells $c_i, i = 1, 2, \cdots, N$ is $O(N)$. Therefore, the complexity for finding the best solution in the neighborhood $N(S)$ is $O(N)$. The complexity of the heuristic to generate cell pair (c_i, c_j) is constant. Although it needs more Tabu Search steps to obtain a good result, the total computation time is still much less than the case of finding the best solution. The Tabu Search heuristic used for the placement consists of two stages. Cell overlap is allowed in the first stage and prohibited in the second stage. The main reason of allowing cells to overlap in the first stage is to increase the solution space being searched. A different objective function is used in each stage.

First Search Stage

The first stage minimizes the total half-perimeter wire length while restricting overlap to a minimum amount besides ensuring equal row length. The objective function is given as:

$$f(S) = c_l(S) + c_o(S) + c_r(S) \tag{4.3}$$

where $c_l(S)$ is the total half-perimeter wire length (HPWL) given by equation (2.1). The second term $c_o(S)$ is the overlap penalty function, and given by:

$$c_o(S) = p_o \sum_j O(i, j) \tag{4.4}$$

where p_o is a penalty parameter. The function $O(i,j)$ returns the total amount of overlap area between cells i and j. Certainly, by checking every other cell on the same row as cell i, it can be determined which of these cells overlap with cell i. However, the complexity is $O(N_i)$, where N_i is the number of cells in the row. The time spent doing overlap computation can be substantial. If W_{max} denotes the maximum cell width, it follows that another way to compute the cost $c_o(S)$ is to search W_{max} units to the left and to the right of cell i to locate all other cells which overlap with cell i. This is an efficient method as long as the ratio between the maximum and minimum cell width is not too large.

The overlap penalty parameter is an empirical parameter. If p_o is too large, the Tabu method will be primarily concerned with the minimization of the overlap penalty function (little attention will be paid to the minimization of the HPWL). If p_o is too small, the Tabu method will concentrate on the minimization of the HPWL (little attention will be paid to the elimination of cell overlap). At the end of the first Tabu Search stage, a large amount of the cell overlap still exists. The HPWL increases when cell overlap is removed. Experimentally, it has been found that $p_o \in (0.1, 0.5)$ yields the least HPWL [3, 58].

The last term $c_r(S)$ is the row length penalty function and is given by

$$c_r(S) = p_r \sum_{i=1}^{R} |L_{ai} - L_{di}| \tag{4.5}$$

where p_r is a row penalty parameter, R is the number of rows, L_{ai} and L_{di} are the actual and desired row length for row i. Experimentally, it has been found that $p_r = 5$ is approximately the smallest value which would yield uniform row lengths without placing excessive emphasis on $c_r(x)$ in the objective function $f(S)$ [53, 3].

Second Search Stage

After the termination of the first search stage, a simple heuristic is used to remove all the overlap among cells by shifting them. At this stage, the placement has no cell overlap, and the row length are changed slightly. The objective here is to minimize the HPWL only; i.e, $f(S) = c_l(S)$. The heuristic avoids causing overlap in this stage by choosing cells having the same width and are close to each other. To further refine the search, a simple neighborhood interchange heuristic is used since Tabu Search does not exhaustively search all possible cells in the neighborhood $N(S)$.

4.7 Summary

In this chapter, we presented our new placement algorithm. The flow of the algorithm in general and details of how it deals with cell spreading were throughly dissected. We highlighted the major deficiencies of the partitioning approach, and we presented means to overcome these major shortcomings. Specifically, we presented in some details the different procedures to identify dense and sparse regions, creating cell attractors in the sparse regions and assigning cells to the different cell attractors. Moreover, different criteria to assign cells to cell attractors have been presented. We also presented a simple heuristic to create a legal placement from the global placement. Finally, we shed some light on the Tabu search method and how it was adapted to perform local improvement of the globally optimized placement.

Application of the new placer (ARP) to standard-cell placement including qualitative analysis of the method and experimental results are the topic of the next chapter.

Chapter 5

Application To Standard Cell Placement

In previous chapters, we indicated that standard cell design is widely used in ASIC (Application Specific Integrated Circuits) and standard cell placement is a challenging problem since cells vary in width and the typical number of cells per circuit is quite large. In this chapter, we consider applying the new placement method ARP to standard cell placement. We illustrate the strength and the other various aspects of the attractor-repeller approach compared to other approaches. Specifically, we present a qualitative analysis in which we examine:

- the individual effect of the attractors and the repellers on the spreading of the cells.
- and the effect of the cell attractors on the speed of convergence of the global optimization.

Moreover, to demonstrate the effectiveness and robustness of the method, final results are compared to the state-of-art placement methods, namely, TimberWolf v6.0, TimberWolf v7.0 [59] and Gordian/Domino [30, 32].

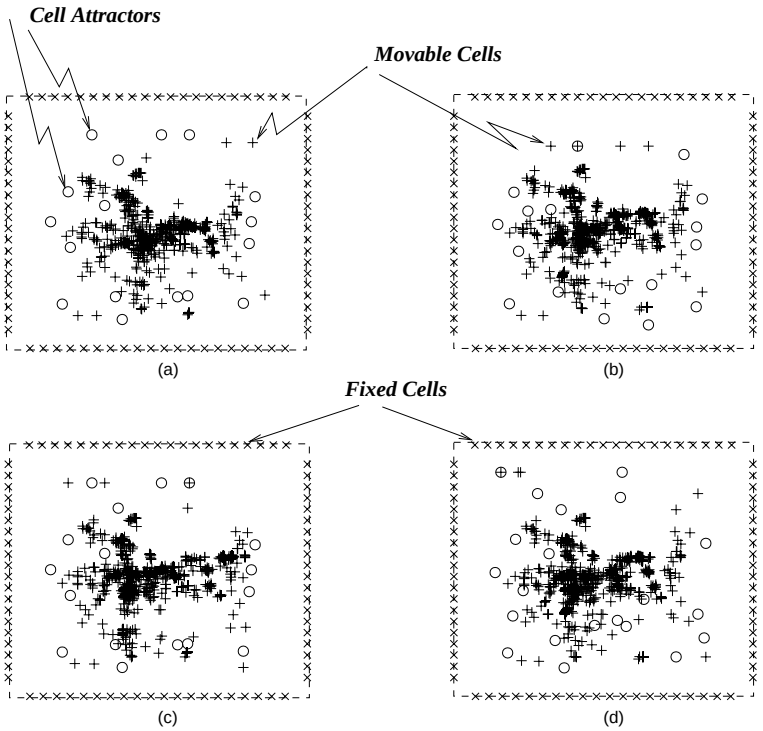


Figure 5.1: Benchmark *Primary1*: cell spreading after each pass using no repellers, $\rho(z_{ij}) = 0$ and $d = 1$ in $f(z_{ij})$ ("+" represent locations of movable cells, "x" represent locations of fixed cells (I/O pads), and "o" represent locations of attractors).

5.1 Qualitative Analysis

We start by examining cell spreading and how it is correlated with cell attractors and repellers.

5.1.1 Attractors-Repellers and Cell Spreading

Cell attractors and repellers complement each other in the sense that cell repellers prevent connected cells from folding on top of each other and cell attractors prevent excessive displacement of connected cells by the cell repellers.

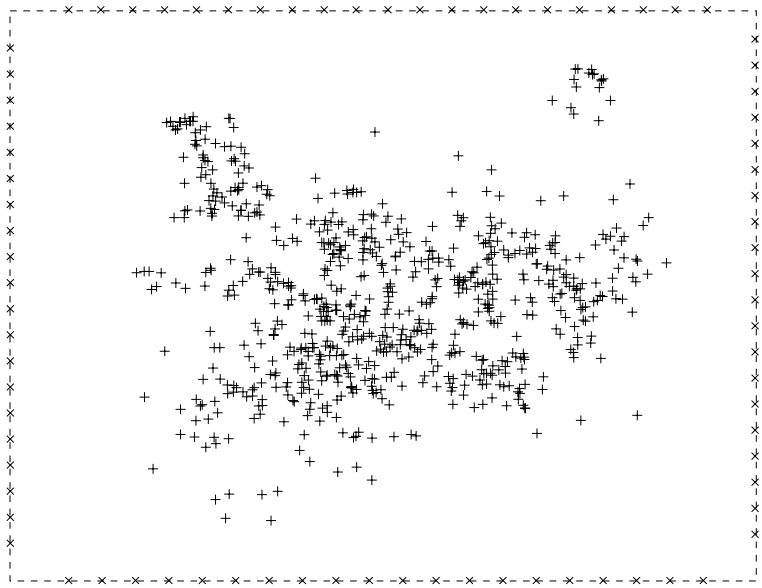


Figure 5.2: Cell spreading using strict repeller model. "+" represents movable cells and "x" represents fixed cells.

To demonstrate this, and to draw a general conclusion on whether a combination of attractors and repellers is better or not, we conducted different scenarios in which

- repellers are inactivated; i.e, $\rho(z_{ij}) = 0$ and $d = 1$ in equation (3.7). That is, the AR model is reduced to a combination of quadratic estimate of wirelength and cell attractors.
- no attractors (i.e, $g(x) = 0$ and $h(y) = 0$ in equation (4.2)) are used and the model reduces to a strict repelling engine.

- the AR model (the combination of attractors and repellers) is used.

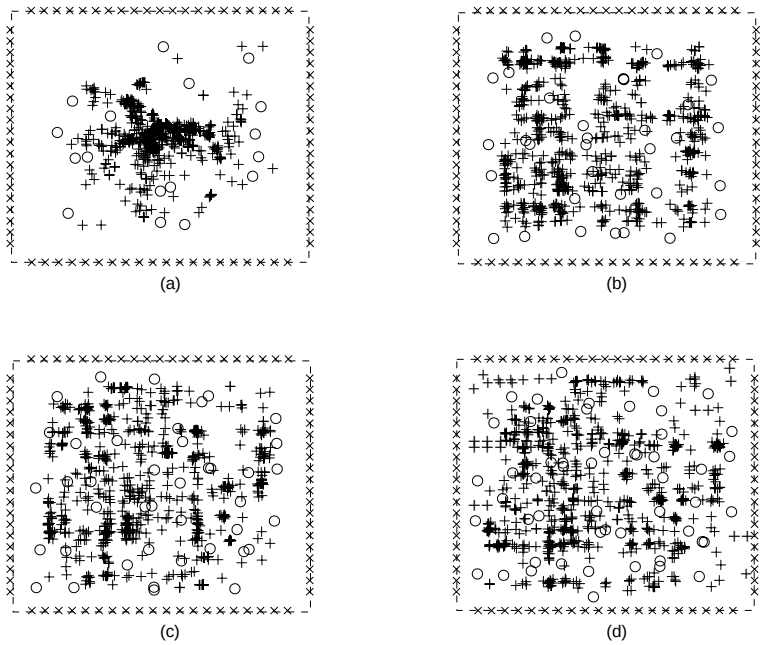


Figure 5.3: Benchmark *Primary1*: cell spreading after each iteration ("+" represents locations of movable cells, "x" represents locations of I/O pads on the chip periphery, and "o" represents locations of attractors).

The industrial circuits listed in Table (2.1) are used in each scenario. To discern between the different scenarios, Figures demonstrating spreading of cells and attractor distribution for benchmark *Primary1* are presented.

Figure (5.1) illustrates cell spreading in the *first scenario*. Part (a) shows the initial solution obtained via minimizing a quadratic wirelength objective function. Parts (b)-(d) illustrates the spreading after including the cell attractors. Clearly, no substantial improvement in cell spreading with respect to the first iteration is noteworthy. This is owing to the fact that in the absence of the repelling forces, the attraction forces between the connected cells outweigh the forces exerted by the

cell attractors. In other words, the attraction forces between the connected cells overwhelmingly dominate the resultant forces acting on the cells.

In contrast, Figure (5.2) demonstrates cell spreading in the *second scenario* in which no attractors are included and the model is strictly repelling. Clearly, the amount of cell spreading is remarkable, but the resultant wirelength is found to be higher and extra efforts by the final placement improver are necessary to offset this undesirable effect. As we suggested previously, this is owing to the fact that, a strictly repelling engine tends to stretch short nets and accordingly deteriorate total wirelength.

Figure (5.3) illustrates cell spreading in the *third scenario* (AR model). Part (a) shows the initial solution obtained from minimizing the quadratic wirelength. Clearly, the majority of the cells are clustered in the center of the placement region and the amount of overlap between the cells is substantial. The cell attractors (shown as circles in the Figure) are created according to the current cell positions. In parts (b)-(c), the global optimization involves minimizing the AR model. Better spreading of the cells can be noticed in each iteration compared to the preceding iterations. In the second iteration, clusters of overlapped cells tend to move to the same sparse regions and in subsequent iterations, some of these clusters tend to flatten out filling existing empty space in their immediate neighborhood on the placement floor. In this scenario, there is a trend of improvement in both cell spreading and wirelength reduction in each subsequent iteration. Figure (5.4) illustrates the *variability* or *relative spread* of the average wirelength as the global optimization is carried out for a finite number of iterations (specifically, the algorithm is executed for each benchmark and the average of wirelength across all benchmarks is computed). Clearly, the wirelength decreases as the algorithm proceeds from one iteration to another. In fact, for some benchmarks, we observed

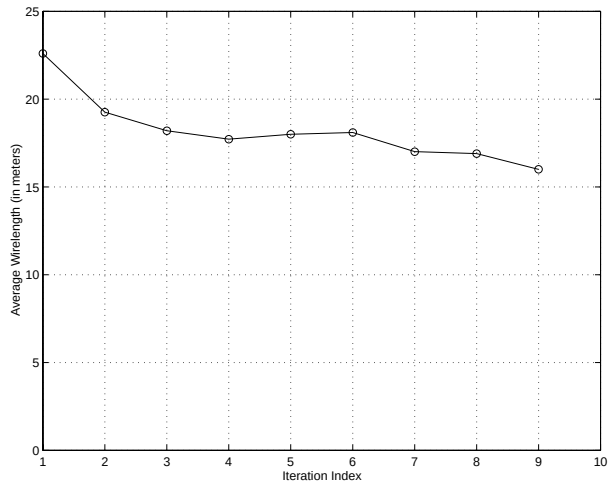


Figure 5.4: Variability of average wirelength over the different iterations of the algorithm.

that if the algorithm is executed for a fairly large number of iterations, the wirelength obtained is quite close to the final answers obtained after improving the initial placement generated from a relatively smaller number of iterations.

Figure (5.5), depicts the variability of the average wirelength as a function of the intensity of the repelling forces (that is, parameter d in equation (3.7)). Based on experimentation, we found that there is correlation between the intensity of the repelling forces required to spread the cells apart (prohibit overlap) and average cell width of a particular circuit. Accordingly, the following empirical formula for estimating d is used:

$$d = \frac{\sqrt{w_a}}{k} \tag{5.1}$$

where w_a is the average cell width (which is constant for a given circuit) and k is scaling factor. In fact formula (5.1) was found quite useful. To examine the correlation between parameter d (which reflect the strength of the repelling terms)

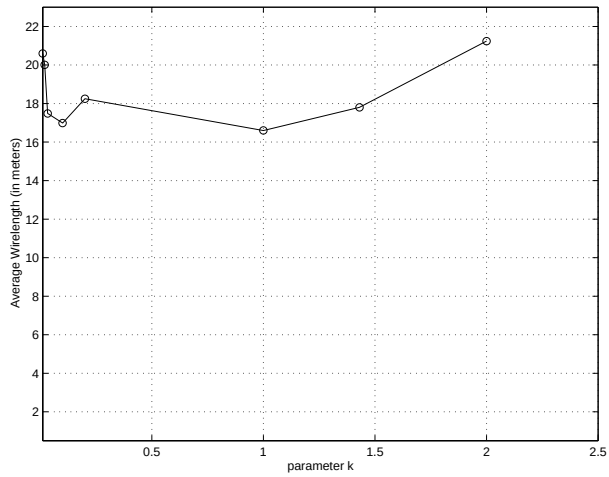


Figure 5.5: Variability of average wirelength versus the scaling parameter k, (equation (5.1)).

and the wirelength, parameter k is varied between 0.01 to 2.5 for each test case. Experimentally we found that, on average, $k = 1$ (or equivalently $d = \sqrt{w_a}$) yields best answers in terms of wirelength. This is demonstrated in Figure (5.5). No correlation between d and the speed of convergence of the global optimization was observed.

5.1.2 Cell Attractors and Convergence of the Global Optimization

As we have shown in chapter 3, the attractor-repeller objective function is convex as long as the whole circuit forms one set; that is the netlist graph is connected and there exist some fixed cells. The convexity and smoothness of the objective function improves as the number of fixed cells increases.

As indicated previously (Figure (4.4)), the AR model is used after the first

iteration (an initial solution is generated in the first iteration via minimizing a quadratic model; i.e, $\rho(z) = 0$ and $d = 1$ and no attractors exist). Figure (5.6-a) shows the variability (*relative spreading*) of the algorithm run time as a function of the number of iterations. The relationship between the algorithm run time and the number of attractors is illustrated in Figure (5.6-b) (precisely, the algorithm is executed for many iterations and the *average run* time as well as the *average number of attractors* are computed and plotted in Figure (5.6)).

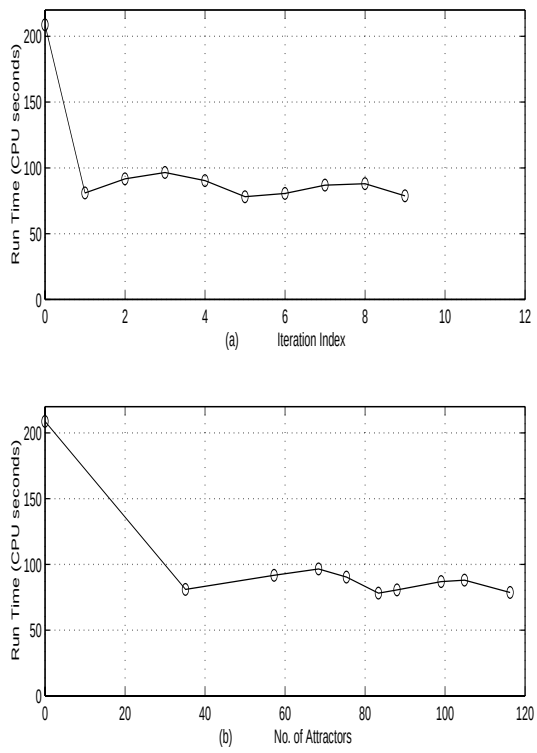


Figure 5.6: Variability of run time. (a) versus passes (b) versus number of attractors.

Examining these Figures, it is clear that there is a substantial decrease in the run time after the first iteration. That suggests a strong correlation between the convergence of the global optimization and the existence of the cell attractors. On

average, the percentage of decrease in run time in the second iteration (that is, when the cell attractors are added) is almost 62%. That is, on average, the run time improves by an average of 62% compared to the first iteration. After the second iteration, the rate of decrease in run time is relatively lower, but its general trend is downward.

Based on these observations, it is evident that there is a strong tendency of the optimization process to converge in a much shorter time when the attractors exist. Again, this indicates that the smoothness and curvature (convexity) of the objective function improve as the number of fixed cells in the circuit increases.

5.2 Numerical Results

The new method is implemented in the C language on a Sun-Ultra1/140 workstation. Half-perimeter wirelength (HPWL) is used in estimating the wirelengths. The HPWL has been used in other results presented in the literature ([54, 59, 60, 30, 32]), since rectilinear wiring is typically used in routing a circuit. Table (5.1) lists the wirelengths and CPU time of the various benchmarks for the initial (global) placements.

As previously mentioned, the final phase in our combined placement method involves further improvement of the initial placement. The Tabu search based iterative improvement technique presented in chapter 4 serves as our final placer.

To assess the new placement method ARP, we consider comparing solution quality and efficiency (computation times) of ARP to the state-of-art placers. Specifically, Simulated Annealing (SA), and a combination of analytic and iterative improvement based placers. Namely, TimberWolf v6.0 (TW v6.0) [54], TimberWolf

Ckt	Wirelength	CPU seconds
Fract	0.12	5
Prim1	1.03	41
Prim2	5.62	241
Struct	0.39	51
Ind1	2.10	101
Bio	2.17	547
Ind3	64.8	1989
Avq.s	9.1	3792
Avq.l	9.6	5010

Table 5.1: Wire Length estimates and CPU time for initial placements.

v7.0 (TW v7.0) [60, 59] and Gordian/Domino [30, 32].

TW v7.0 [60, 59] is the latest release of TimberWolf placement package. It has two modes of operation, namely, a *flat* mode and *hierarchical* mode. In the flat mode, the original netlist is placed. In the hierarchical mode, however, the original netlist is clustered into various levels of netlists. At each level, SA tries to find an optimal placement for the clustered netlist, followed by flattening the netlist. Subsequently, the flattened netlist is admitted to the next level and the process is repeated until the original netlist is placed. The clustering approach has the advantage of greatly reducing the complexity of the circuit and consequently, reducing the computational efforts required to solve the problem.

The method of Gordian [30] has been described when the partitioning approach was introduced in chapter 4. The method of Domino is an iterative improvement technique that has been applied to initial placements generated by Gordian [32].

In Domino, the placement problem is modelled as a *transportation problem* and *network flow* algorithms are employed to solve the resulting optimization problem. Specifically, cells positioned near each other in the existing placement are considered for improvement. The iterative process produces a sequence of intermediate placements. In each iterative step, an improved placement is generated from the current placement. The process terminates when after several generations no significant improvement is obtained.

The metrics used in the comparison of the new method ARP and the other different approaches are *total wirelength* and *longest-row width* which reflect solution quality, and *computation times* which typify the efficiency of the method. Longest row width determines chip width and accordingly total chip area. Thus, longest row width is an important parameter in the assessment of a given placement. Numerical results of the other approaches are taken from the literature [59]. For some benchmarks, no results have been reported in [59] and their entries in the results tables are left empty.

Tables (5.2), (5.3) and (5.4) list the final wirelength, longest row width and computation times for ARP and the other approaches. The reported computation times include times for final placements. ARP outperforms TW v6.0, TW v7.0 and Gordian/Domino on the majority of the benchmarks. For benchmark *Ind3*, ARP outperforms TW v6.0 but lacks compared to TW v7.0 and Gordian/Domino, and for benchmark *Bio* and *Prim2*, ARP lacks insignificantly compared to TW v7.0 in flat mode (FM). We suspect a different positioning of the I/O pads compared to the other methods, or a scaling problem is the reason for this difference in results.

On average, ARP achieves 7.78%, 1.02%, 3.96% and 8.68% reduction in wirelength compared to TW v6.0, TW v7.0-FM, TW v7.0-H (Hierarchical) and Gordian/Domino respectively.

Tables (5.3) and (5.6) show comparisons of longest row width obtained by ARP and the other approaches. On average, ARP outperforms TimberWolf v6.0 and TimberWolf v7.0 by 4.28% and 0.13%. No longest row width was reported for Gordian in [59] to compare to.

Ckt	TW6	TW7.FM	TW7.H	Gord/Dom	ARP
Fract	-	-	-	-	0.034
Prim1	1.0	0.93	0.99	1.08	0.79
Prim2	3.71	3.53	3.72	4.02	3.61
Struct	-	-	-	-	0.34
Ind1	-	-	-	-	1.50
Bio	1.97	1.8	1.88	1.98	1.83
Ind3	48.38	43.08	44.67	44.94	48.12
Avq.s	6.72	6.45	6.13	6.42	6.06
Avq.l	6.93	6.50	6.81	7.16	6.54

Table 5.2: Wire Length Comparison, TW v7.0 flat and hierarchical modes, Gordain/Domino and ARP

As for computation times, the results are illustrated in Tables (5.4) and (5.7) (computation times of the other approaches are scaled). Evidently, ARP consumes less computation times to place each test case compared to the other approaches. On average, ARP outperforms TimberWolf v6.0 by 83%, TimberWolf v7.0 (flat mode) by 87%, TimberWolf v7.0 (Hierarchical mode) by 7.6% and Gordian/Domino by 13.8%.

Ckt	TW6	TW7.0	ARP
Fract	-	-	704
Prim1	5260	5100	5170
Prim2	8380	8210	8201
Struct	-	-	2360
Ind1	-	-	4810
Bio	5114	4936	4928
Ind3	28832	26368	26176
Avq.s	9560	9128	9080
Avq.l	9744	9400	9344

Table 5.3: Chip width comparison: TimberWolf v7.0 and ARP. Width is measured in *microns*.

Ckt	TW6	TW7.FM	TW7.H	Gor/Dom	ARP
Fract	-	-	-		12
Prim1	467	488	130	168	95
Prim2	3127	4307	736	542	504
Struct	-	-	-		116
Ind1	-	-	-	-	376
Bio	8606	12224	1273	1553	1290
Ind3	38619	70873	5156	6087	4253
Avq.s	54681	78248	7657	10261	8534
Avq.l	56802	97612	9175	12403	11202

Table 5.4: Run-time comparison in CPU seconds: TimberWolf v7.0 in flat and hierarchical modes, Gordian/Domino and ARP.

Ckt	TW v6 %impr.	TW7.FM %impr.	TW7.H %impr.	Gor/Dom %impr.
Prim1	+21	+15	+20.2	+26.8
Prim2	+2.7	-2.2	+2.9	10.2
Bio	+7.1	-1.64	+2.9	+7.5
Ind3	+0.53	-10.4	-7.2	-6.6
Avq.s	+9.8	+6.0	+1.1	+5.6
Avq.l	+5.6	-0.6	+3.9	+8.6
avg.	+7.78	+1.02	+3.96	+8.68

Table 5.5: Relative wirelength improvement with respect to other approaches (+ means ARP is better).

Ckt	TW v6.0 %impr.	TW v7.0 %impr.
Prim1	+1.7	-1.3
Prim2	+2.1	+0.11
Bio	+3.6	+0.16
Ind3	+9.2	+0.73
Avq.s	+5.0	+0.53
Avq.l	+4.1	+0.60
avg.	+4.28	+0.13

Table 5.6: Relative chip-width improvement with respect to other approaches (+ means ARP is better).

Ckt	TW v6 %impr.	TW7.FM %impr.	TW7.H %impr.	Gor/Dom %impr.
Prim1	+79	+80	+27	+3.8
Prim2	+83	+88	+31	+7.0
Bio	+85	+89	-1.3	+17
Ind3	+88	+93	+17	+30
Avq.s	+84	+89	-10	+16
Avq.l	+80	+88	-18	+9
avg.	+83	+87	+7.6	+13.8

Table 5.7: Relative CPU time improvement with respect to other approaches (+ means ARP is better).

5.3 Summary

In this chapter, we described the application of the new placement method to a set of standard cell benchmarks. Correlation between quality of answers and the intensity or strength of the repelling and attracting forces acting on cells is presented. We experimentally demonstrated that excessive repelling forces produce good spreading, but tend to deteriorate wirelength. In addition, we experimentally found that the strength of the repelling forces necessary to yield sufficient cell spreading are proportional to the average cell width. Accordingly, we proposed an empirical formula to determine parameter d (which controls the strength of the repelling forces) as a function of the average cell width.

Correlation between cell attractors and speed of convergence of the global optimization of the AR-model is presented. We indicated that this correlation is owing

to the improved smoothness of the model when the cell attractors are added to the repeller model.

Final results of the method are compared to the state-of-art placers. We demonstrated that, on average, the new method outperformed these placers in terms of solution quality and efficiency.

In the next chapter, the other layout problem addressed in this work will be presented. Namely, the cell routing problem. Our focus will be on detailed routing in general and channel routing in particular.

Chapter 6

Routing Problem

In the placement phase [56], the exact locations of circuit cells are determined. A netlist is also generated which specifies the required interconnections. Space not occupied by the cells can be viewed as a collection of regions. These regions are used for routing and are called the *routing regions*. Each routing region has a *capacity*, which is the maximum number of nets that can pass through that region. The capacity of a region is a function of the design rules and dimension of the routing regions and wires. Nets must be routed within the routing region and must not violate the capacity of any routing region. In addition, nets must not short-circuit, that is, nets must not intersect each other.

There are two types of routing regions: **switchbox routing** and **channel routing**. A *switchbox* is a rectangular area bounded on all sides. A *channel* is a rectangular area bounded by two opposite sides. *Capacity* of a channel is a function of the number of layers, channel height, wire length and wire separation.

A VLSI chip may contain several million transistors. As a result, tens of thousands of nets have to be routed to complete the layout. In addition, there may

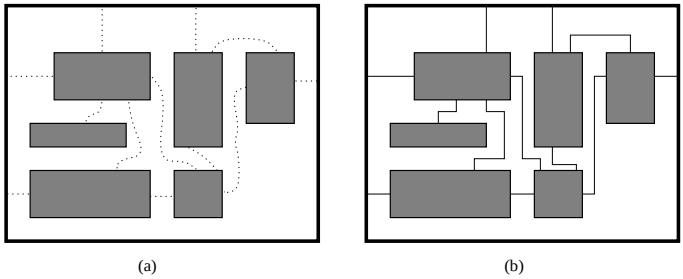


Figure 6.1: Routing: (a) Global routing; (b) Detailed routing.

be several hundreds of possible routes for each net. This makes the routing problem computationally hard. In fact, even when the routing problem is restricted to channels, it cannot be solved in polynomial time; i.e, the *channel routing problem* is *NP-complete* [61]. Therefore, routing has traditionally been divided into two phases. The first phase is called *global routing* and generates a loose route for each net. In fact it assigns a list of routing regions to each net without specifying the actual geometric layout of wires (see Figure 6.1 (a)). The second phase which is called *detailed routing*, finds the actual geometric layout of each net within the assigned routing regions (see Figure 6.1 (b)). Unlike global routing which considers the entire layout, a detailed routing considers just one region at a time. The exact layout is produced for each wire segment assigned to a region and vias are inserted to complete the layout. Detailed routing includes *channel* and *switchbox* routing.

6.1 Detailed Routing

As previously pointed out, the detailed routing problem is typically solved in an incremental manner; that is, by routing one region at a time in a predefined order [56]. The ordering of the regions is determined by several factors including the criticality of routing certain nets and the total number of nets passing through a

region. Characteristics of a routing problem largely depend upon the topology of the routing region. Routing regions consist of one or more layers. In the general case, even single-layer routing problems are NP-complete [49]. In multi-layer routing problems, the wiring can switch adjacent layers at certain locations using *vias*. In many multi-layer models, the layers are restricted to contain either horizontal or vertical segments. This type of model is known as *restricted layer model*. Multi-layer routing problems are also NP-complete [61]. For this reason many of the algorithms for multi-layer problems are heuristic in nature.

A primary objective function of a router is to minimize the total routing area. Various secondary objective functions have also been considered. One such objective is to minimize the number of vias [56]. Other objective functions include minimization of the average length of a net and minimization of the number of vias per net.

6.1.1 Channel Routing Problem (CRP)

The majority of modern automatic IC routing systems are based on channel routers [47]. These systems employ a divide and conquer approach in which the layout problem is divided into several channel routing problems which are each solved separately. In descriptive terms, a channel is a routing region bounded by two parallel rows of terminals [65]. The top and bottom rows are called *top boundary* and *bottom boundary* respectively. Each terminal is assigned a number between 0 and N. Terminals with the same number i ($1 \leq i \leq N$) must be connected by net i, while those with number 0 designate unconnected terminals (see Figure 6.2).

The horizontal dimension of a channel is called the *channel length* and the vertical dimension is called the *channel height*. The horizontal segment of a net is

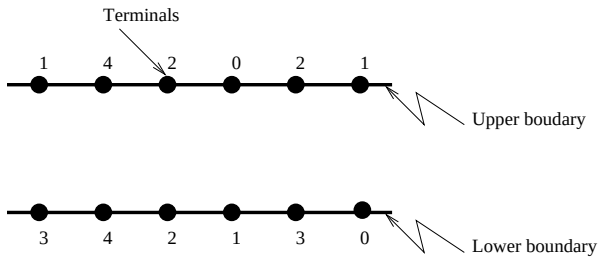


Figure 6.2: A channel.

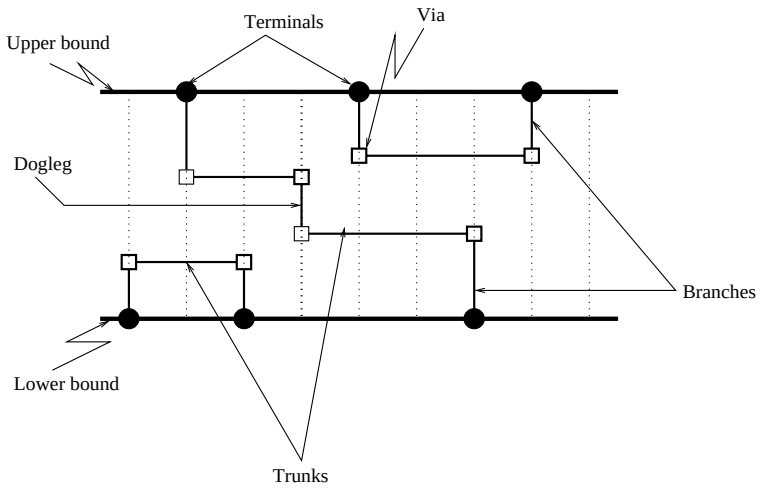


Figure 6.3: Terminology for CRP

called a *trunk* and the vertical segments that connect the trunk to the terminals are called its *branches*. The horizontal line along which a trunk is placed is called *track*. A *dogleg* is a vertical segment that is used to maintain the connectivity of two trunks of a net on two different tracks. Figure (6.3) illustrates the terms mentioned above.

A CRP is specified by four parameters [56]: (i) channel length, (ii) top (bottom) terminal list, (iii) left (right) connection list and (iv) number of layers. Channel length is specified in terms of number of columns. The top and bottom lists specify the terminals in the channel. The top list is denoted by $T = (T_1, T_2, \cdots, T_m)$ and

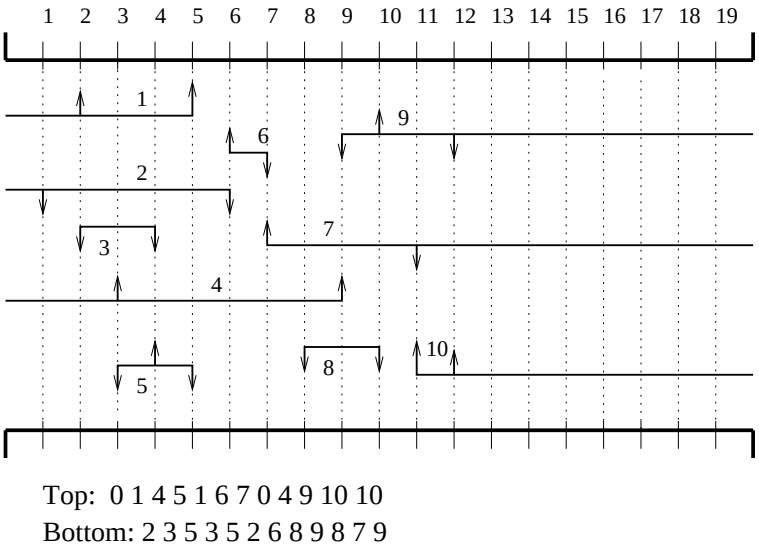


Figure 6.4: Netlist representation for routing requirements [65].

the bottom list by $B = (B_1, B_2, \cdots, B_m)$. T_i (B_i) is the net number for the terminal at the top (bottom) of the ith column or, if 0 the terminal does not belong to any net. The left (right) connection list consist of nets that enter the channel from the left (right) end of the channel. Figure (6.4) illustrates an example of a netlist [65]. Arrows indicate whether nets are to be connected on the top or lower boundary of the channel.

Given the above specifications, the problem is to find the interconnections of all nets in the channel so that the channel uses minimum possible area. A solution to CRP is a set of horizontal and vertical segments for each net. This set of segments must make all terminals of a net electrically equivalent. The solution specifies the channel height in terms of the total number of *tracks* required for routing. Thus, the main objective is minimize the channel height which implies minimizing the number of tracks and consequently the channel area. There are two key constraints which must be satisfied while assigning the horizontal and vertical segments, namely,

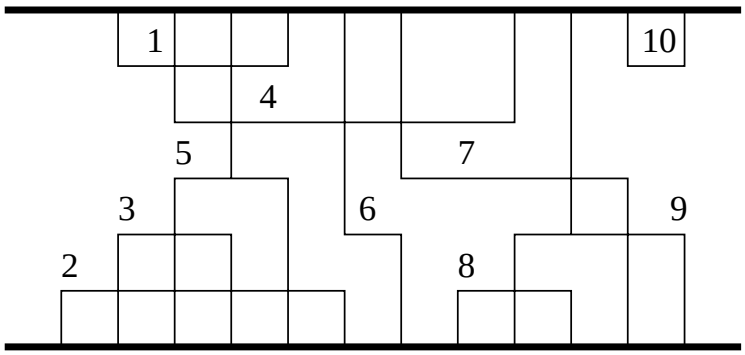


Figure 6.5: Solution of the netlist in 6.4.

vertical and *horizontal* constraints.

Vertical Constraints

Any two nets must not overlap at a vertical column. If it is assumed that there is only one horizontal segment per net [65], then it is clear that the horizontal segment of a net connected to the top terminal at a given column must be placed above the horizontal segment of another net connected to the lower terminal at that column.

This relation can be represented by a directed graph G_v, where each node corresponds to a net and a directed edge from net a to net b means that net a must be placed above net b. Figure 6.6(a) illustrates the Vertical Constraint Graph (VCG) for the netlist in Figure (6.4).

Horizontal Constraints

The horizontal segment of a net is determined by its leftmost and rightmost terminal connections. $S(i)$ represent the set of nets whose horizontal segments intersect column i. Since horizontal segments of distinct nets must not overlap, the horizontal

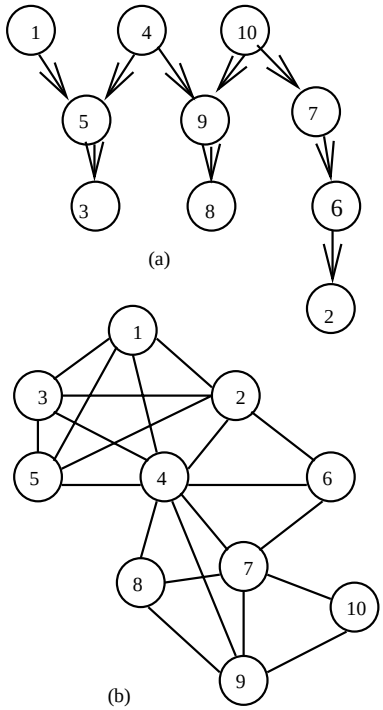


Figure 6.6: (a) VCG (b) HCG for the netlist in Fig 6.4. In HCG, maximal cliques are (1,2,3,4,5), (2,4,6), (4,6,7), (4,7,8,9) and (7,9,10)

segments of any two nets in $S(i)$ must not be placed on the same horizontal track.

This relation is represented by a graph $G(V, E)$, where node $v_i \epsilon V$ represents net n_i and edge $(v_i, v_j) \epsilon E$ if $n_i \cap n_j \neq 0$ (i.e n_i and n_j have horizontal overlap [65, 33]. Figure 6.6(b) illustrates the Horizontal Constraint Graph (HCG) for the netlist in Figure (6.4).

Density

The *local density* at column i is defined as the number of nets whose spans intersects column i. If l_k and r_k are the positions of the left most pin and right most pin of net k, then *channel density*, denoted by d, is the maximum local density over all the columns in the channel, i.e:

$$d = max\{d_1, d2, \cdots, d_q\}$$

where q is the number of columns in the channel.

Lower Bounds on Channel Width

There are two well known lower bounds on a channel width w [56, 67]. The first bound is due to horizontal constraints and is given by the channel density d. The second bound is due to vertical constraints and it is given by the longest path in the vertical constraint graph (VCG).

These bounds can be easily seen. Consider a column with a maximum local density equals the density of the channel d. Clearly, one track is needed to place each net. Therefore, at least d wiring tracks are needed to realize such a netlist. In the VCG, a constraint path $(x_1, x_2, \cdots, x_k)$ implies that net x_1 should be placed above net x_2 which in turn should be placed above net x_3 and so on. Hence, nets

$x_1, x_2, \cdots, x_k$ must be placed on k distinct tracks. Thus, the length p of the longest path in the vertical constraint graph represents another bound on the channel width w. Clearly, the lower bound on the channel width is the maximum of d and p.

If the vertical constraints graph has no edges (i.e, no two terminals of different nets are in the same column), then the routing amounts to finding the best assignment of nets to tracks without having to pay attention to what happens in columns [37]. The track assignment problem (i.e, which track is assigned to every net) is reduced to coloring the nodes of the HCG. Coloring the nodes of the HCG is equivalent to assigning tracks to nets (i.e, each track constitutes a color) and the chromatic number is the channel density. Obviously, in this case, the lower bound on the channel width reduces to the channel density d.

In the next section we shall formulate the channel routing problem as an integer programming problem.

Integer Programming (IP) Formulation

For a routing solution with n horizontal wire segments (nets) and m tracks (m can be a maximum of n tracks), denote the horizontal wire segments by $t_1, t_2, \ldots, t_n$. Assume that the rows and columns are labeled in increasing order with the top-most row being row 1 and the left-most column being column 1. For each horizontal wire segment t_i, there is a variable T_i, the value of which is the track in which t_i is placed in the final solution. Obviously $1 \leq T_i \leq m$ for $1 \leq i \leq n$. Another 0-1 variable w_k, $k = 1, \ldots, m$ is introduced such that:

$$w_k = \begin{cases} 1 & \text{if any } T_i = k \\ 0 & \text{otherwise} \end{cases}$$

A vertical constraint from t_i to t_j can be expressed by the linear constraint

$T_i < T_j$ [23]. To express a horizontal constraint between horizontal wire segment t_i and t_j, the condition "$T_i > T_j$ or $T_i < T_j$" need to be expressed in terms of linear constraints. To do so, a 0-1 integer variable z_{ij} needs to be introduced. The essential linear constraints can then be expressed in terms of z_{ij} as follows[23].

$$T_i - T_j > -mz_{ij}$$

$$T_i - T_j < m(1 - z_{ij})$$

Obviously the difference between T_i and T_j is less than m. If $z_{ij} = 0$, the second constraint is redundant and the first constraint is equivalent to $T_i > T_j$. If $z_{ij} = 1$, the first constraint is redundant and the second constraint is equivalent to $T_i < T_j$. Therefore, the first and the second constraints are equivalent to "$T_i > T_j$ or $T_i < T_j$". The IP formulation can then be given as:

$$\begin{aligned}
&Min \sum_k w_k \\
&s.t.\ T_i < T_j \ (t_i \text{ should be above } t_j) \\
&\quad T_i - T_j > -mz_{ij} \\
&\quad T_i - T_j < m(1 - z_{ij}) \\
&\quad T_i \in \{1, m\} \\
&\quad w_k, z_{ij} \in \{0, 1\}
\end{aligned}$$

Clearly, for large problem instances, the problem size of the IP program may become very large. No polynomial time algorithms exist to solve such large IP problem [61]. Consequently, heuristic techniques are used to find near optimal solutions for the problem. However, the model can be used to find bounds on the optimal solutions for small size problems.

Classification of CRP Algorithms

One method of classifying CRP algorithms would be to classify them based on the approach the algorithms use [56]. Based on this classifying scheme, there are:

- *Left-Edge algorithms* (LEAs): Left-edge based algorithms start with sorting the trunks from left to right and assign the segments to tracks so that no two segments overlap [56, 26].

- *Constraint graph-based algorithms*: The constraint based routing algorithms use a graph theoretic approach to solve CRP [56, 65]. The horizontal and vertical constraints are represented by graphs. The algorithms then apply different techniques on the graphs to generate the routing in the channel.

- *Greedy routing algorithms*: The greedy routing algorithm uses a greedy strategy to route the nets in the channel [56, 50]. It starts with the leftmost column and works towards the right end of the channel by routing the nets one column at a time.

- *Hierarchical routing algorithm*: The hierarchical router generates the routing in the channel by repeatedly bisecting the routing [7] and then routing each net within the smaller routing regions to generate the complete routing.

- *Search heuristics based algorithms*: A search heuristic based algorithm starts from an initial solution, then it moves (by perturbation of the placement of the nets) from one solution to a neighbor solution. The algorithm terminates execution if the maximum number of iterations (specified by the designer) is exceeded or if no significant improvement in the solution quality is obtained compared to the best solution obtained so far [40].

A major difference [40] between a search heuristics based channel router and other channel routers is that in the other routers, solutions are built up incrementally while a search heuristics-based router works with complete solutions, i.e; moving from one complete solution to another. Thus, for a search heuristic based algorithm, one can trade off between solution quality and algorithm efficiency (running time of the algorithm). In other words, the algorithm can be terminated at any time (if running time is crucial). In such a case, the best solution obtained so far is accepted as the optimal (or near optimal) solution found. In fact, this is a very attractive feature since reasonably good solutions can be achieved in reasonable running times.

Performance Criteria

Generally, channel routers are compared based on quality of solutions they produce and how efficient they are in producing such solutions [37]. Quality of solutions is judged by a *performance ratio* defined as the ratio of the the channel width found by the algorithm, to the minimum channel width (optimal width). Such a performance ratio is, therefore, a measure of how the computed channel is deviated from the optimal channel width. The efficiency of the algorithm is measured by the amount of computation time required to solve a CRP.

Another criterion for comparing channel routers is their adaptability to variations in the routing problem [47]. Such variations as terminals not on-grid or channels which are not rectangular are more easily accommodated by some algorithms than others [14]. Search heuristics based algorithms are more easily adaptable to the variations in the routing requirements than the other approaches. This is primarily because such techniques work with complete solutions. Simulated Annealing has been applied to the CRP, Wong *et al.* [40] and optimal and near optimal results

are reported for a set of benchmark problems. The solution space presented in [40] has been adopted in our new hybrid router which is the topic of the next chapter.

In the next section, the adopted solution solution space is described.

Solution Space via Search Heuristics

Let V be the set of nodes in the VCG and let $\pi = \{V_1, V_2, \cdots, V_w\}$ be a partition of the node set V into w groups. Define G_π, the *merged* graph induced by π, to be the graph with node set $\{\hat{V}_1, \hat{V}_2, \cdots, \hat{V}_w\}$ where $\hat{V}_i$ represents the group of nodes V_i in the partition π. If there is a directed edge from some node $x \epsilon V_i$ to some node $y \epsilon V_j$ in VCG, then there exists a directed edge from $\hat{V}_i$ to $\hat{V}_j$ in G_π. If the merged graph G_π is *acyclic* and if there is no horizontal constraint between any nodes in the same group, then partition π is said to be valid.

There exists an equivalence between feasible routing solutions and valid partitions. Given any routing solution that uses w tracks, a partition $\pi = \{V_1, V_2, \cdots, V_w\}$ can be defined where V_i consists of all nets occupying track i. On the other hand every valid partition can be mapped into a routing solution by topologically ordering nodes in G_π and assigning each group V_i of nets to a track using the topological ordering. In general, the number of groups in a partition is equal to the number of tracks in a solution. Thus search heuristics generate routing solutions by generating merged graphs with a minimum number of nodes.

Each valid partition represents a feasible solution and hence a state in the solution space. Transitions between states is made by randomly selecting nets from group V_i and moving them to group V_j in the partition.

Three types of moves are defined to locally modify π, these are:

- Type 1: two nets belonging to two different groups V_i and V_j are swapped.

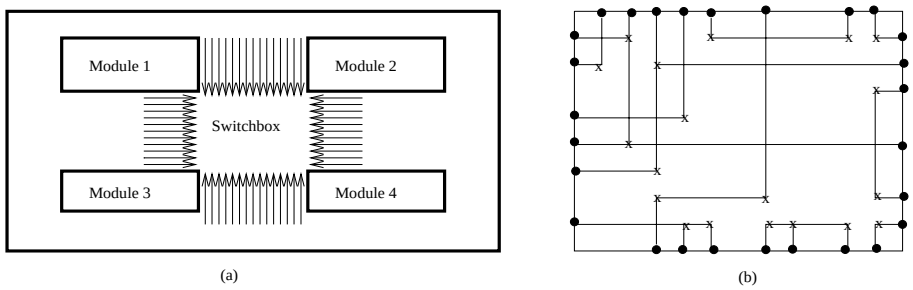


Figure 6.7: Example of a switchbox.

- Type 2: a net is moved from group V_i to group V_j.
- Type 3: a net is removed from group V_i to form a new group.

Two valid partitions π_1 and π_2 are said to be neighbors if one can be obtained from the other via one of the three moves. For a given valid partition π, a neighbor partition $\hat{\pi}$ is generated by repeatedly applying moves (moves can be of the same or different type) on π while observing the vertical and horizontal constraints.

Clearly, a move of type 1 has no effect on the number of tracks w, while a move of type 3 increases w by 1. A move of type 2 will decrease w by 1 if there is only one net in group V_i. Thus, moves of type 2 are responsible for reducing w, while moves of type 3 allow the algorithm to escape local minima. Moves of type 1 allow rearrangements among solutions with the same cost.

6.1.2 SwitchBox Routing Problem (SBRP)

The switchbox routing problem is a generalization of the channel routing problem, where terminals are located on all sides, [46, 56] (see Figure 6.7 (a)). A switchbox is sometimes referred to as a "Four Sided Channel".

A switchbox is formally defined as a rectangular region $R(h \times w)$ where h and w are positive integers. Each pair (i, j) in R is a grid point. The ith column and j^{th} row (track) are the sets of grid points. The 1th and h^{th} columns are the LEFT and RIGHT boundaries respectively. Similarly, the 1th and wth rows are TOP and BOTTOM boundaries respectively.

The problem, like channel routing, can be simply described by labeling the terminal pins by net numbers; pins having the same positive label have to be interconnected. As in the channel routing problem, the terminals on the four boundaries are assumed fixed, and consequently the area in switchbox is fixed. Therefore, the objective of switchbox routing is not to minimize the routing area but to complete the routing within the routing area. In other words, the switchbox routing problem is to decide the existence of a routing solution. An example of a switchbox routing is shown in Figure 6.7 (b). The switchbox routing problem falls in the category of NP-hard problems [7].

The switchbox routing problem is similar to the channel routing problem, except that nets to be routed are on all sides of the rectangle instead of just two sides. Although this problem is considered much more difficult than channel routing (because it is not clear what to do in case of failing to find a routing solution), it is not difficult to extend a channel router to handle switchboxes[5]. For instance, the switchbox routers presented in[57, 42] are extensions of the channel routers presented in [31, 50].

6.2 Summary

In this chapter, the detailed routing problem was described. The problem is solved by routing the channels and the switchboxes. As we indicated at the beginning of

this thesis, our investigation will be limited to the channel routing problem. Accordingly, the channel routing problem was explained in some details. In addition, an IP formulation (model) of the problem was presented. Such a model can be very useful in finding lower bounds on the optimal solutions for small size problems.

In the next chapter, our new hybrid channel router will be presented.

Chapter 7

Utility Function based Channel Router

In this chapter, our new hybrid channel router is described [21]. The router is based on a hybridization of Stochastic Evolution (SE) and Tabu Search (TS) methods. We have already described TS when we presented our iterative improvement technique for cell placement in chapter 4. Thus, in this chapter, besides the details of the hybrid methodology, we will only consider the SE algorithm.

SE is a stochastic method used for combinatorial optimization[66]. SE algorithm is an instance of a more general class of *adaptive heuristics* [51, 66]. It is an extension of the evolution concept suggested by Kling *et al.* in[36]. In [66], SE has been applied to *graph bisection*, *travelling sales-man* and *cell placement* problems. The authors reported obtaining competitive results compared to SA. SE, like SA, diversifies the search such that the search space is effectively explored. However, in SA, it is always possible to accept a move with arbitrarily large negative gain (climb uphill) at any temperature $T > 0$. On the other hand, large changes in the cost, which

tend to slow the optimization process are not allowed in SE [66]. This is because the control parameter p (corresponding to T in SA) is set to a smaller value at the beginning and only incremented if the algorithm gets trapped in a local minima.

In the following section, the SE algorithm is presented in more details.

7.1 Stochastic Evolution (SE)

Stochastic Evolution (like SA) is a stochastic method for combinatorial optimization that exploits an analogy between biological evolution and combinatorial optimization. In biological processes, species become better as they evolve from one generation to the next generation. The evolution process generally eliminates the bad genes and maintains the good genes of the old generation to produce a better new generation. This concept has been exploited in the *iterative improvement* techniques for some combinatorial optimization problems [36, 66]. In this kind of approach, each feasible solution to the problem is considered as a generation. The bad genes of the solution are identified and eliminated to generate a new feasible solution.

As we stated above, SE algorithm belongs to the class of *adaptive heuristics* [51]. An adaptive algorithm uses a set of controlling parameters which are modified either by the user or the algorithm itself. Therefore, the algorithm adapts to the particular resulting solution. In order to use SE, a cost function which measures the quality of the solution must be defined. The general outline of SE is as follows:

SE algorithm:

$S = S_o$; /* initial solution */

```
S_best = S; /* save initial solution */
C_best = cost(S);
p = p_o; /* initialize control parameter */
γ = 0; /* initialize counter */
REPEAT
        C_pre = COST(S);
        S = PERTURB(S, p);
        C_cur = COST(S);
        UPDATE(p, p_o, C_pre, C_cur);
        IF (C_cur < C_best) THEN {
           S_best = S ; /* save best solution */
           C_best = C_cur ;
           γ = γ − R; /* decrement counter by R */
        }
        ELSE
           γ = γ + 1; /* increment counter */
UNTIL γ > R /* R is a user supplied parameter */
RETURN(S_best); /* report best solution */
```

The input to the SE consists of an initial solution S_o, an initial value of the control parameter p_o, and a parameter R which controls the number of iterations. SE retains the solution of the lowest cost among those produced by the function PERTURB. Each time a solution is found which has a lower cost than the currently best solution, SE decrements the counter γ by R, thereby rewarding itself by increasing the number of iterations. The following is a description of the different functions.

The PERTURB function: During each call to PERTURB, the movable elements are scanned according to *a priori* ordering. The details of the move and the choice of the ordering is problem specific. If a move m is applied to the current solution S to generate a neighbor solution $\hat{S}$, the associated gain of m denoted $gain(m) = cost(S) - cost(\hat{S})$, is the reduction in cost after the move is performed. The function PERTURB stochastically decides whether or not to accept move m with the help of a non-negative control parameter p. The value of $gain(m)$ is compared to an integer r randomly generated in the interval $[-p, 0]$. If $gain(m) > r$, then move m is accepted; otherwise, it is rejected. Note that $r \leq 0$ always; hence, moves with positive gains are always accepted. The PERTURB function can be outlined as follows:

```
function PERTURB(S, p)
FOR each move m do
        perform move m to generate Ŝ;
        gain(m) = cost(S) − cost(Ŝ);
        IF (gain(m) > RANDINT(−p, 0)) THEN
           S = Ŝ;
        ENDIF
ENDFOR;
RETURN(S)
```

UPDATE procedure: This procedure is mainly responsible for updating the value of the control parameter p. Initially p is set to a non-negative value close to zero (e.g $p_o = 1$). Such a choice of p implies that only moves with small negative gains are accepted. According to Saab *et al.* [66], moves with large negative gains tend to

upset the optimization process and only increase the running time of the algorithm. They recommended increasing the value of p only when necessary. The value of p is incremented to give the algorithm a chance to escape a local minima via uphill climb. After each call to function PERTURB, if the cost of the new solution C_{cur} is equal to the cost of the previous solution C_{pre}, this may indicate that the algorithm is trapped at a local minima and the value of p is incremented accordingly.

An outline of the procedure UPDATE is as follows:

Procedure UPDATE(p, p_o, C_{pre}, C_{cur})
IF ($C_{pre} = C_{cur}$) THEN
  increment p;
ELSE
  $p = p_o$
ENDIF

Choice of R: The iteration bound R acts as the expected number of iterations the SE algorithm needs until an improvement in the cost takes place. If such an improvement occurs at $\gamma < R$ iterations, then the remaining $R - \gamma$ iterations are added to the next R iterations to be performed. It follows that, if R is chosen too large, the algorithm wastes time during the last set of iterations because it cannot find better solutions. On the other hand, if R is chosen too small, the algorithm might not have enough time to improve the initial solution. In practice, it has been reported that a value between 10 and 20 gives good results [66].

In the following section, the hybrid methodology is presented in details.

7.2 Hybrid Methodology

As we mentioned previously, a hybrid algorithm based on combining SE and TS is chosen as a search engine in our channel router. In this approach, SE is guided by TS such that duplication of solutions is prohibited. That is, during the generation of neighborhood solutions, if a move is chosen and its status is Tabu, another move will be chosen unless the aspiration criteria determines otherwise. *Aspiration* forces SE to backtrack to previous solutions to refine the search in those regions. Furthermore, problem-domain information expressed in the form of utility functions, are also employed to determine the best moves during the search process. Utility is a measure of a decision certainty [6]. In the context of channel routing, the use of a certainty based utility function, over which moves are compared can be constructed to perform calculations indicating the preferability of one move versus another.

Before we present the hybrid algorithm, we first shed some light on utility functions from a theoretical perspective.

7.2.1 Utility Functions

In decision-making theory, multi-attribute utility functions order preferences of different decision outcomes [6]. The ordering of the classification preferences is prescribed by the estimates and assumptions in the decision model. The type of calculation, where a choice is made by maximizing a function is typical of the analytical models of managerial decision in management science. For management decision, a utility function is derived which mimics a manager's preference order for a series of choices. The value of the utility function $u(\mathcal{X})$; i.e, the utility of a situation $\mathcal{X}$, has no meaning in an absolute sense. A value is only useful in comparison to other values acquired from the same utility function. If situation $\mathcal{Y}$

begets utility value $u(\mathcal{Y})$ and $u(\mathcal{Y}) > u(\mathcal{X})$, then the preference model u predicts that situation $\mathcal{Y}$ is preferable to situation $\mathcal{X}$.

An attribute may be any parameter of the decision model which yields preference ordering [6]. A multiattribute utility function can be made either additive (linear) or multiplicative in combining the single attribute utility functions; i.e:

$$u(\mathcal{X}_1, \mathcal{X}_2, \cdots, \mathcal{X}_n) = \sum_{i=1}^{n} \alpha_i u_i(\mathcal{X}_i)$$

$$u(\mathcal{X}_1, \mathcal{X}_2, \cdots, \mathcal{X}_n) = \Pi_{i=1}^{n} u_i(\mathcal{X}_i)$$

Where α_i is a parameter to control the importance of attribute $u_i(\mathcal{X}_i)$.

In the context of channel routing, the decisions involve assigning nets to tracks such that no horizontal and vertical constraints are violated and the number of tracks is minimum. In the previously developed search techniques based routers, the neighborhood solutions are generated by randomly selecting nets and tracks (random moves)[40, 5, 4]. However, random moves do not guarantee convergence in a reasonable time and the algorithm is highly likely to get trapped in a local minima. In this work, a variety of problem-domain information are combined using utility functions such that (during the generation of neighborhood solutions), nets and tracks are selected based on their respective utilities. The net utility function expresses information about the goodness of assigning a net to a track. The track utility function combines information about the goodness of cluster of nets assigned to a track and about the sparsity of the track.

We now present how the channel routing problem-domain information can be mapped into net and track utilities.

7.2.2 Channel Routing Problem-domain Information and Utility Functions

The VCG (Vertical Constraint Graph) provides an insight about the ideal position of a net (ideal track) such that no vertical constraint violations are caused. To be more specific, in a typical netlist, nets that come before net n_i in $p(n_i)$ ($p(n_i)$ is the longest path passing through the vertex corresponding to net n_i in VCG) must be assigned to tracks above n_i track, and nets that follow net n_i in $p(n_i)$ must be assigned to tracks below n_i track.

The certainty of assigning a net n_i to track r is given by the following utility function

$$\pi_i^r = \exp(-0.5z^2)$$

where

$$z = \frac{r - d(n_i)}{p(n_i)}, \qquad r \in 1, \cdots, k$$

If we add two fictitious nodes to the VCG; i.e, a *starting node* s and a *termination node* t, $d(n_i)$ is the longest path from s to the node corresponding to net n_i in VCG and k is the maximum number of tracks. The certainty of a cluster of nets assigned to a track r is determined as follows

$$\mathcal{U}^r = \psi^r \gamma^r$$

$$\psi^r = \Pi_{i=1}^{q} \pi_i^r$$

$$\gamma^r = \exp(-0.5\delta^2)$$

$$\delta = 1 - \frac{\sum_{j=1}^{q} l_j}{\mathcal{L}}$$

Here q is the number of nets placed in track r, δ is a measure of the sparsity of a track, l_j is the span of net n_j and $\mathcal{L}$ is the channel length (number of columns

in the channel). Each of the single-attribute functions $\{\pi_i^r, \gamma^r\}$ is restricted to the interval $[0, 1]$. For a net n_i, π_i^r reflects the certainty or, in other words, the goodness of assigning n_i to track r. π_i^r equals 1, correspond to an ideal situation; i.e, track r is the ideal track for net n_i. For track r, γ^r is a certainty measure of how good and effective track r is utilized. Again, γ^r equals 1, indicates that track r is fully utilized. By multiplication of the values of single-attribute utility functions in the interval $[0, 1]$, the resultant multi-attribute utility functions will also be restricted to the same interval. For instance, utility $\mathcal{U}^r$ equals 1 corresponds to an ideal situation with respect to the information of utilities ψ^r and γ^r. Generally, the values of the utilities are designed to correspond to a confidence in performing a certain move (in other words, in generating neighbor solutions).

7.3 Algorithm Description

Figure 7.1 illustrates an outline of the hybrid channel router (SETS-CR). The following is a detailed description of the different phases of SETS-CR.

7.3.1 Initial Solution

Given a set of nets $\mathcal{S} = \{l_1, l_2, \cdots, l_n\}$ where l_j is the horizontal span of net j, $\mathcal{S}$ is partitioned into three subsets $\mathcal{S}_t$, $\mathcal{S}_m$ and $\mathcal{S}_b$ such that $\mathcal{S}_t$ includes the nets connected to the top boundary of the channel, $\mathcal{S}_m$ includes the nets connected to the top and the bottom boundaries of the channel and $\mathcal{S}_t$ includes the nets connected to the bottom boundary of the channel. The initial solution is generated by assigning the nets in $\mathcal{S}_t$ to the channel starting from the top track, then nets in $\mathcal{S}_m$ followed by nets in $\mathcal{S}_b$. Horizontal constraints between the nets are observed and vertical

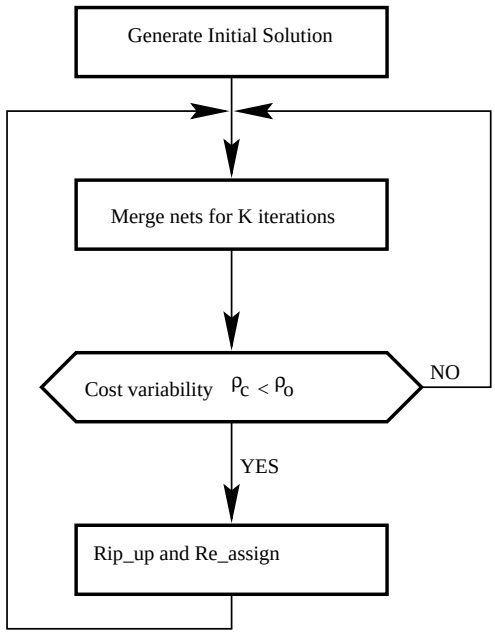


Figure 7.1: Outline of the SETS-CR.

constraints are only observed for the nets in $\mathcal{S}_m$ because nets in $\mathcal{S}_t$ and $\mathcal{S}_b$ cannot overlap vertically.

Using this strategy to assign the nets guarantees distributing the nets such that they are relatively in conform with the VCG and their vertical segments are relatively short compared to selecting and assigning the nets randomly.

7.3.2 Merging Phase

Following the generation of the initial solution, see Figure (7.1), the merging phase is executed. In the merging phase, nets are selected for merging as follows. The certainty (value of utility π_i^r) of assigning net n_i to each track r is determined. The tracks are then sorted in a descending order based on their respective values of π_i^r. Net n_i is attempted for merging with other clusters of nets assigned to other

tracks that exhibit higher certainty compared to the current track (in which n_i is already placed in). The move is accepted if the difference in cost $\Delta\mathcal{C}$ between the newly generated solution S_{new} and the present solution S_{pre} is positive. If $\Delta\mathcal{C}$ is negative, a random number a is generated in the interval $[0, -p]$ and again the move is accepted if $\Delta\mathcal{C} > a$. Once a net is moved to a new track, it will not be allowed to move to any other track for a number of iterations equals the *Tabu List* length $\mathcal{T}$ unless the move satisfies the required *Aspiration Level.* The Aspiration Level used is the cost of the *best solution* obtained so far.

The process of merging the nets continues for K iterations (typically $5 \leq K \leq 10$). At the end of the Kth iteration, the cost variability ρ_c is computed for the K values of the cost function; i.e:

$$\rho_c = \frac{\sigma_c}{\mu_c}$$

where μ_c and σ_c are the average and standard deviation of the K values of the cost. If the value of ρ_c is less than a threshold ρ_o (typically $\rho_o < 0.03$), it is evident that the algorithm might have been trapped in a local minima. In this case, the *Ripup and Reassign* phase is executed to help the algorithm escape the local minima.

The cost function used is given as follows:

$$\mathcal{C} = \alpha_1 w + \alpha_2 n_s$$

where w is the number of tracks and n_s is the number of sparse tracks. α_1 and α_2 are positive weights to control the importance of w and n_s respectively (typically $\alpha_1 = 10, \alpha_2 = 5$). A track is considered sparse if the sparsity of the track δ is less than a threshold δ_0 (typically $\delta_0 = 0.1$).

7.3.3 Ripup and Reassign Phase

In this phase, low certainty tracks are identified and clusters of nets assigned to these tracks are selected for ripping_up and reassign. A certainty $\mathcal{U}^r$ of track r is considered low if it is less than a specified threshold $\mathcal{U}_o$ (typically $\mathcal{U}_o = 0.2$). Tracks that exhibit low certainty values are selected for ripping_up their nets for reassignment to other higher certainty track. A net n_i, placed in track r, is ripped_up and attempted for reassignment (provided that the net Tabu List is 0, or the move satisfies the Aspiration Level) if its assignment certainty π_i^r is less than specified threshold π_o (typically $\pi_o = 0.2$). Moving net n_i to, say track t, is accepted as a valid move if the new computed track certainty $\mathcal{U}^t$ is higher than previous $\mathcal{U}^t$ ($\mathcal{U}^t$ before net n_i is moved to track t).

7.4 Implementation and Results

The algorithm was implemented in C++ on a SUN SparcStation 2. A set of benchmark problems taken from an existing paper [28] and the *difficult problem of Deutsch* [14] are used to evaluate the performance of the algorithm. The optimal routing width of each benchmark is known. The statistics for these benchmarks are shown in Table (7.1). Several scenarios were conducted using these benchmarks. In each scenario, the performance of the algorithm was evaluated based on one of the following criteria (1) convergence behavior; (2) effectiveness of incorporating Tabu Search; (3) effectiveness of using utility functions to select the best moves. Due to the fact that the SE algorithm is stochastic in nature and to ensure the validity of the approach, the algorithm was executed for 20 trials in each scenario.

The first scenario was designed to investigate the convergence of the algorithm.

Benchmark	*No. of nets*	Global optimum
ex1	21	12
ex3a	45	15
ex3b	47	17
ex3c	54	18
Deutsch ex.	72	28

Table 7.1: Statistics for the different benchmarks.

The algorithm was executed with the SETS (SE and TS) hybrid as the search engine. Also, utility functions are used to select best moves during exploration of the solution space. Figure (7.3) illustrates the variation of the *average cost* versus the *generation number* for the Deutsch difficult example. By observing the average cost contour, one can easily see that the algorithm climbs hills and descends valleys of the solution space. This suggests that the algorithm explores the solution space effectively and convergence to the global optimum is very likely. The routing of the difficult Deutsch problem is illustrated in Figure (7.2). Furthermore, as shown in Figure (7.3), the valley representing the optimal routing width was explored at an average number of generation of 172.

The second scenario was designed to ascertain the effectiveness of combining SE and TS as a hybrid search engine (SETS hybrid). In this scenario, two experiments were conducted. In the first experiment, the algorithm employed only SE as a search engine, and in the second experiment the SETS hybrid was employed as a search engine. Moves were randomly selected (i.e, nets and destination tracks are randomly chosen). The algorithm converged to the global optimal routing width in both experiments, except for the ex3a benchmark for which the optimal answer

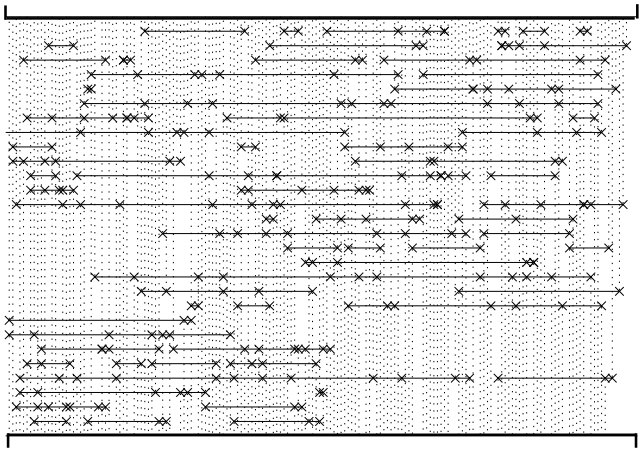


Figure 7.2: The routing of Deutsch's difficult example.

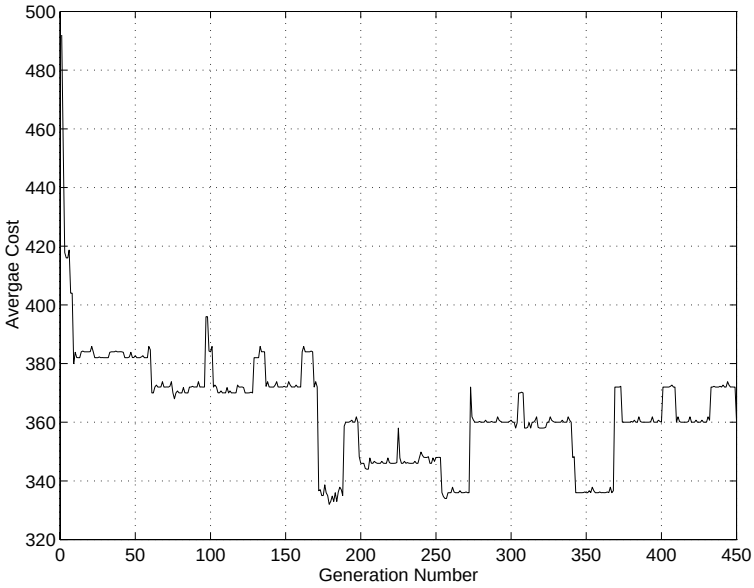


Figure 7.3: The variation of the average cost value as the algorithm progresses over generations for the difficult Deutsch problem.

obtained in the first experiment, is one track beyond the global optimum. For all the benchmarks, the number of generations, *AVG-NUM*, needed to converge to the optimal solution and, *OPT-SOL*, in the second experiment are less compared to the first experiment, see Table (7.2). This observation demonstrates the significance of SETS hybrid as a search engine.

Benchmark	*AVG-GEN*			*OPT-SOL*		
	SE	*SETS*	*SETS-UTFN*	*SE*	*SETS*	*SETS-UTFN*
ex1	4	4	4	12	12	12
ex3a	335	160	310	16	16	15
ex3b	297	227	206	17	17	17
ex3c	67	56	45	18	18	18
Deutsch ex.	341	217	172	28	28	28

Table 7.2: For all the benchmarks, average number of generations (AVG-GEN) required to converge to an optimal solution (OPT-SOL) when the search engine is (i)only SE; (ii)SETS hybrid; (iii)SETS hybrid and utility functions (SETS-UTFN).

In the third scenario, the effectiveness of using utility functions in selecting the best moves is investigated. The algorithm is executed with utility functions used to choose candidate nets and candidate tracks for a move. The results obtained are again reported in Table (7.2). The global optimum for each benchmark is obtained. Also, the AVG-NUM required to obtain the global optimum is smaller compared to the previous scenarios (except for ex3a, in which the global optimum was not found in the previous scenarios). This indicates that using of utility functions to determine best moves (rather than relying on random moves) in generating neighborhood solutions is a quite effective approach.

The major limitation of the SETS channel router is in the choice of the control parameters; i.e, Tabu Length $\mathcal{T}$, stopping parameter $\mathcal{R}$, control parameter p_o and the threshold criteria ρ_o, δ_o, π_o and $\mathcal{U}_o$. Our experience shows that: $\mathcal{T} \in \{4, 5, \cdots, 7\}$; $\mathcal{R} \in \{15, 16, \cdots, 25\}$; $p_o \in \{1, 2\}$; $\rho_o = 0.03$; $\delta_o = 0.1$; $\pi_o = 0.2$ and $\mathcal{U}_o = 0.2$ yield sufficiently good results. Regarding the Aspiration Criterion used, again, our experience shows that such a criterion is quite effective in forcing the SE to backtrack previous solutions and refine those search regions.

The implemented version of the SETS based router may be improved in many ways. For example:

- developing a learning mechanism that allows for data-driven updating of the parameters
- allowing intermediate solutions (infeasible solutions) may help in exploring the search space more effectively. In this case, including an effective mechanism to resolve the violations is also worth investigation.

7.5 Summary

A hybrid channel routing algorithm based on SE and TS is presented. Problem-domain information expressed in the form of utility functions is used to guide the search engine to explore the search space in an effective way compared to previous strategies. The effectiveness of the approach was demonstrated on several benchmark problems. Global optimal solutions are obtained for all the benchmarks. Unlike Simulated Annealing, Stochastic Evolution does not climb so many hills, and at the same time it proves to be very effective in finding global optimal solutions. Combining TS with SE is very simple and it does not need any extra effort. The

use of utility functions to guide the exploration of a problem search space is quite general. To be more specific, utility functions can be used with other heuristic techniques such as Simulated Annealing and Genetic Algorithms to solve a variety of combinatorial problems.

Chapter 8

Conclusions and Future Directions

Given the intractability of the VLSI circuit design, it is appropriate to divide the problem into subproblems and tackle each subproblem independently. However, the division of the problem does not offer an ultimate cure to the immense difficulties associated with the problem. In other words, each subproblem is still NP-hard. Therefore, more research aiming at yielding more effective and efficient techniques is still required to handle the existing difficulties and to cope with the future immense design complexities.

Different mathematical solution methodologies such as numerical optimization, combinatorial optimization and search heuristic techniques exhibit strengths and weaknesses and may not offer the best solution if used individually. Accordingly, it may be necessary to consider combinations of methodologies as means to capitalize on their strengths and avoid the inherited weaknesses in each single methodology.

8.1 Summary and Contributions

In this thesis, we have investigated two subproblems associated with the circuit layout task in the VLSI design cycle, namely, cell placement and channel routing. We start by summarizing our contributions with regard to cell placement.

8.1.1 Cell Placement

Previous analytic techniques for global placement relied on iterative partitioning of the placement area and hard constraints to force cells apart and to attain the required spreading of cells. The partitioning approach suffers from many disadvantages. Among these, the high likelihood of moving a cell to the wrong partition and the infeasibility of correcting this kind of error in the subsequent iterations. Moreover, the inclusion of hard constraints to force the cells to spread within the placement area makes the problem much harder and slows down the convergence of the optimization process.

In this thesis, we proposed a novel approach to global relative placement. The new approach deals with cell overlap in a different manner and produces uniform distribution of the cells within the placement floor. New repulsive and attractive forces have been added to the traditional formulation of wirelength so that overlap among the cells is diminished without repartitioning the placement floor nor using hard constraints. This is achieved as follows. Upon minimization, the repeller model (or cell repeller) guarantees that the geometric locations of two connected cells be spatially shifted apart. But, cell repellers cannot guarantee overlap-free placement as cells with no common nets may still fold on top of each other. The overlap problem can be substantially diminished if excessive repelling forces are employed. By excessive repelling forces, we meant using larger value for the target

distance between connected cells. However, based on what we observed, excessive repelling tends to stretch nets, especially, short nets which represent the overwhelming majority of the nets, and consequently deteriorating the overall wirelength.

To resolve this dilemma, we proposed a simple, yet, effective approach which involves the creation and inclusion of adaptive cell attractors (or dummy fixed cells). The combination of the cell repellers and cell attractors constitute the new model for the global relative placement which we have referred to as the "attractor-repeller" (AR) model. To examine the new placement model, we proposed a new generic placement method based on the AR-model which we referred to as "attractor-repeller placer" (ARP). The new method is iterative in nature. In each iteration, cell attractors are created based on the most recent cell distribution (cell distribution from the previous iteration). The rationale behind this is that, the most recent cell distribution provides insight about the current sparsity and utilization of the placement area. In other words, it provides a guiding information on how cells should be steered to fill the sparse zones. We presented a set of criteria to assign cells to the attractors in the different sparse regions, and we indicated that for the standard cells establishing a connection with the closest attractors produce the best results. Generally, by establishing a connection between a cell attractor and a movable cell, the cell attractor exerts extra force on the movable cell in a direction parallel to the line connecting their geometric locations. The result is a displacement of the movable cell towards the sparse region where the cell attractor is located. Thus, cell spreading is attained in this fashion over the course of several iterations.

In chapter 5, we presented a qualitative analysis of the new placement method and we showed that best results can be accomplished if the combined model (that is, the combination of the attractors and repellers) is used. We further showed that

the cell attractors have a great impact on the smoothness of the objective function and consequently, on the speed of convergence of the global optimization of the objective function. We also demonstrated the competitiveness and effectiveness of the new placement method using a set of 9 industrial benchmarks with different number of nets and cells. Numerical results have been compared to the up-to-date placers. In terms of solution quality and efficiency, the new placer achieves comparable results and, on average, the new placer outperforms the other placers.

A major impact of the AR-model would be on cell placement with no fixed cells (i.e, I/O pads). This is owing to the fact that, in the absence of fixed cells, the traditional formulation of wirelength lacks the capability of forcing cells apart even by a small margin. In such scenario, the partitioning approach would fail too. This is because, fixed cells attract movable cells to the boundaries of the chip, and accordingly play a crucial role in spreading the cells in the absence of any cell-repelling forces. Thus, for the traditional formulations to work, the existence of the I/O pads is extremely important (in fact, it is a must). In the AR-model, however, the existence or absence of the I/O pads is not as important as in case of the traditional formulation. This is because, cell overlap is diminished by the cell repellers and the uniform distribution of the cells within the placement area is a result of the joint efforts of the cell repellers and cell attractors. One thing needs to be stressed in this case though. That is, the absence of the of I/O pads results in an overall positive-semidefinite rather than positive-definite Hessian matrix. However, practically, this problem can be alleviated by fixing one or more cells, or creating dummy fixed cells on the boundaries of the chip. In fact, as we have shown in chapter 3, only one fixed cell is needed to drive the objective function to the convexity region. We only suggested adding or fixing more cells on the boundaries just to create a balance in the displacement of the cells. That is, avoiding the situation

where the cells may be displaced to one side of the placement area.

8.1.2 Channel Routing

For channel routing, we proposed a new channel router that combines two search heuristic techniques as a search engine. As opposed to previous approaches, the problem-domain information is utilized in an effective way to guide the exploration of the search space. A new mapping for the problem-domain information is proposed, namely, the utility theory. That is, a variety of problem-domain information that includes insights on the best track for a net, and the sparsity of the track have been expressed in the form of utility functions. Subsequently, the utility functions are used to decide which track a particular net should be placed in, and whether a track is sparse or not.

The new router has been applied to a set of benchmarks. Qualitative analysis has been presented and a conclusion that the combination of the hybrid engine and the utility function mapping of the problem-domain information produces the best answers in terms of quality and efficiency.

We need to stress that, to the best of our knowledge, this is the first attempt that ever has been made to utilize utility theory in the VLSI design in general, and circuit layout in particular. Based on our experience, we can claim that the utility theory is a very promising tool that can be utilized in solving other problems in the VLSI design cycle, and other combinatorial problems in general.

8.2 Future Directions

The basic techniques proposed in this thesis can be extended to enhance their functionality and performance. The following are several possibilities that can be investigated for extending these basic techniques.

- Investigating an efficient and effective clustering technique to cluster the original netlist. Clustering has the advantage of reducing the size of the problem. Since the computation efforts increase with the number of variables, they are expected to be drastically reduced if the netlist is clustered. It has been reported in [60] that TimberWolf v7.0-hierarchical (which uses clustering) is, on average, 6 times faster than TimberWolf v6.0 which uses flat (original) netlists. This indicates how useful clustering can be.

- Investigating the possibility of integrating the task of legalization with that of computing the global placement. The idea involves adding constraints to force cells to cluster in the row locations. Other constraints are required to avoid violating maximum row capacity while placing the cells. We strongly believe that assigning the cells to the rows while minimizing the wirelength will greatly enhance the overall quality and efficiency of the solution. To be more specific, pushing the cells to the rows while minimizing the wirelength will diminish, if not eliminate, the effect of erroneous assignments of cells to the rows if traditional legalization is used. This will definitely improve the quality of the initial global placement. Furthermore, it will reduce the burden of the placement improvement, and consequently improve the efficiency of the local improvement technique. Thus, the overall quality and efficiency of the solution is expected to improve.

- Investigating accurate delay models to estimate the timing requirements for a circuit. A signal propagating along a net from a driver to several sinks need to arrive quickly enough to guarantee better performance of the circuit. Typically, each cell in the circuit has a required *arrival time* to guarantee the cell provides the correct output. This is especially crucial, given the immense complexity of today's integrated circuits and their associated timing requirements.

- Another important point that is worth emphasizing in this respect is that, the inclusion of constraints to account for timing requirements and the legalization task, implies that it is essential to investigate different solution methodologies. That is, the added constraints may be nonlinear and nonconvex which implies that a solution methodology that can handle general nonlinear nonconvex objective functions may be eventually required. Moreover, the inclusion of nonlinear nonconvex constraints implies that it may be only possible to guarantee a locally optimal solution to the global placement problem. Despite this difficulty, the overall benefit is worth the efforts.

- Examining the possibilities of using the utility theory in tackling other issues in the circuit layout problem, for instance, placement local improvement and global routing. Moreover, the advent of utility theory to the circuit layout problem in this thesis paved the way to investigate other tools used in decision making theory, game theory, etc., and examine their suitability to VLSI design in general.

- Extending the proposed channel router to other channel models and to multilayer channels (channels with more than two routing layers). Furthermore, extending the router to handle switchbox routing. Another interesting di-

rection to explore is examining and including models to estimate problems associated with wires in a channel such as congestion and cross talk.

Bibliography

[1] B. W. Kernighan A. Dunlop. A procedure for placement of standard-cell vlsi placement. *IEEE Trans. on CAD*, 4, no. 4:92–98, 1985.

[2] K. Chaudhary A. Srinivasan and E. Kuh. RITUAL: A Performane-Driven Placement Algorithm. *IEEE Trans. on Circuits and Systems, vol. 39, No. 11*, pages 825–839, November 1992.

[3] S. M. Areibi. *Towards Optimal Circuit Layout Using Advanced Search Techniques*. PhD thesis, University of Waterloo, Ont. Canada, 1995.

[4] L. M. Patnaik B. B. Prahlada and R. C. Hansdah. An extended evolutionary programming algorithm for vlsi channel routing. In *Proc. of the 4th Annual Conf. on Evolutionary Programming*, pages 521–544, 1995.

[5] R. J. Brouwer and P. Banerjee. A parallel simulated annealing algorithm for channel routing on a hypercube multi-processor. In *IEEE Conf. on Comp. Desig.*, pages 4–7, 1988.

[6] D. K. Bunn. *Analysis for Optimal Decisions*. John Wiley and Sons Ltd., 1982.

[7] M. Burstein and R. Pelavin. Hierarchical wire routing. *IEEE Trans. on CAD 2(4)*, pages 223–234, October 1983.

[8] A. Kahng C. J. Alpert T. Chan and I. Markov. Faster minimization of linear wirelength for global placement. *IEEE. Trans. on CAD, vol. 17, no. 1*, pages 3–13, 1998.

[9] et al C. P. Hsu. Apls2: A standard cell layout system for double-layer metal technology. In *in Design Automation Conference, IEEE/ACM*, pages 443–448, 1985.

[10] D. Braun C. Sechen and A. Sangiovanni-Vincentelli. Thunderbird: A complete standard cell layout package. *IEEE journal of Solid State Circuits, 23(2)*, pages 410–420, April 1988.

[11] P. Lu C. Zhu, R. H. Byrd and J. Nocedal. L-bfgs-b: Fortran subroutines for large-scale bound constrained optimization. Technical report, Dept. of Elec. Eng. and Comp. Scien., NorthWestern University, 1996.

[12] C. K. Cheng and E. S. Kuh. Module placement based on resistive network optimization. *IEEE Trans. on Comp. Aided Design, 3 (3)*, pages 218–225, 1984.

[13] J. P. Cohoon and P. L. Heck. Genetic placement. In *Proc. IEEE International Conf. on CAD*, pages 422–425, 1986.

[14] D. Deutsch. A dogleg channel router. In *Proc. 13th DA conf.*, pages 425–433, 1976.

[15] G. Sigl K. Doll and F. Johannes. Analytical placement: A linear or quadratic objective function. In *In proc. 23rd DAC*, pages 57–62, 1991.

[16] Y. Du and A. Vannelli. A Nonlinear Programming and Local Improvement Method for Standard Cell Placement. In *Proc. of IEEE Custom Integrated Circuit Conf.*, 1998.

[17] H. Eisenmann and F. M. Johannes. Generic global placement and floorplanning. In *Proc. of the 35th Design Automation Conference*, 1998.

[18] A. El Gamal et. al. An architecture for electrically configurable gate arrays. *IEEE JSSC*, 24(2):394–398, April 1989.

[19] H. Hseih et. al. A 9000-gate user programmable gate array. In *Proceedings of 1988 CICC*, pages 15.3.1–15.3.7, May 1988.

[20] S. C. Wong et. al. A 5000-gate cmos epld with multiple logic and interconnection arrays. In *Proceedings of 1989 CICC*, pages 5.8.1–5.8.4, May 1989.

[21] H. Etawil and A. Vannelli. Utility Function based Hybrid Algorithm for Channel Routing. In *Proc. of International Symposium on Circuits and Systems (ISCAS), VI 258-261*, 1998.

[22] C. M. Fiduccia and R. M. Mattheyses. A linear-time heuristic for improving network partitions. In *Proc. of the 19th Design Automation Conference*, pages 175–181, 1982.

[23] T. Gao and C. L. Liu. Minimum crosstalk switchbox routing. In *Proc. of Design Automation Conf. (ACM)*, pages 610–615, 1994.

[24] M. R. Gary and D. S. Johnson. *Computers and Intractability.* Freeman, San Francisco CA, 1979.

[25] F. Glover. Tabu search, part i. *ORSA Journal on Computing*, pages 190–206, 1990.

[26] A. Hashimoto and S. Stevens. Wire routing by optimizing channel assignment within large apertures. In *In proc. 8th, Design Automation Workshop*, pages 155–169, 1971.

[27] A. Hertz. Finding a feasible course schedule using tabu search. *Discrete Applied Mathematics 35*, pages 255–270, 1992.

[28] T. C. Hu and E Kuh. Theory and concepts of circuit layout. *VLSI Circuit Layout: Theory and Design*, pages 3–18, 1985.

[29] D. J. H. Huang and A. B. Kahng. Partitioning-based standard-cell global placement with an exact objective. In *Inter. Symp. on Physical Design (ISPD)*, pages 18–25, 1997.

[30] F. Johannes J. M. Kleinhans G. Sigl and K. Antreich. Gordian: Vlsi placement by quadratic programming and slicing optimization. *IEEE. Trans. on CAD, vol. 10, no. 3*, pages 356–365, 1991.

[31] A. Sangiovanni-Vincentalli J. Reed and M. Santamauro. A new symbolic channel router: Yacr2. *IEEE Trans. on CAD 4(3)*, pages 208–219, 1985.

[32] K. Doll F. Johannes and K. Antreich. Iterative placement improvement by network flow methods. *IEEE. Trans. on CAD, vol. 13, no. 10*, pages 1189–1200, 1994.

[33] T. Ohtsuki H. Mori E. Kuh T. Kashiwabara and T. Fujisawa. One-dimensional logic gate assignment and inetrval graphs. *IEEE Trans. Circ. Syst., Vol. CAS-26*, September 1979.

[34] A. Kennings. *Cell Placement Using Constructive and Iterative Methods.* PhD thesis, University of Waterloo, Ont., Canada, 1997.

[35] B. W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *Bell Systems Technical Journal*, 49:291–307, 1970.

[36] R. Kilng and P. Banerjee. Esp: Placement by simulated evolution. *IEEE Trans. on Computer Aided Design, 8(3)*, pages 245–256, 1989.

[37] A. LaPaugh and R. Pinter. Channel routing for integarted circuits. In *Annual Rev. Comput. Sci*, volume 4, pages 307–363, 1990.

[38] U. Lauther. A min-cut placement algorithm for general cell assemblies based on a graph representation. In *proc. of the 16th Design Automation Conference*, pages 1–10, 1979.

[39] F. T. Leighton. *Complexity Issues in VLSI.* MIT Press, Cambridge MA, 1983.

[40] D. F. Wong H. W. Leong and C. L. Liu. *Simulated Annealing for VLSI Design.* Kluwer Academic publishers, 1988.

[41] D. G. Luenberger. *Linear and Nonlinear Programming.* Addison-Wesley Publishing Company, 1984.

[42] W. K. Luk. A greedy switch-box router. *INTEGRATION, the VLSI journal (3)*, pages 129–149, 1985.

[43] C. Alpert T. Chan D. Huang I. Markov and K. Yan. Quadratic placement revisited. In *Proc. of the 34rd DAC*, 1997.

[44] www.cbl.ncsu.edu/benchmarks/layoutsynth92/.

[45] D. S. Mitrinovic'. *Analytic Inequalities.* Springer-Verlag, Berlin. Heidelberg, 1970.

[46] T. Ohtsuki. *Layout Design and Verification.* Elsevier Science publishers B. V., 1986.

[47] B. Preas and M. Lorenzetti. *Physical Design Automation of VLSI Systems.* Benjamin Cummings, 1988.

[48] N. R. Quinn. The placement problem as viewed from the physics of classical mechanics. In *Proc. of the 12th Design Automation Conference*, pages 173–187, 1975.

[49] D. Richards. Complexity of single-layer routitng. *IEEE Trans. on Comp. Aided Design*, pages 286–288, March 1984.

[50] R. L. Rivest and C. M. Fiduccia. A greedy channel router. In *Proc. 19th Design Automation Conf.*, pages 418–424, 1982.

[51] S. Sahni S. Nahar and E. Shragowitz. Simulated annealing and combinatorial optimization. In *Proc. of the 23rd Design Automation Conference*, pages 293–299, 1986.

[52] M. Sarrafzadeh and C. K. Wong. *AN INTRODUCTION TO VLSI PHYSICAL DESIGN.* McGrow-Hill, 1996.

[53] C. Sechen and DD. Chen. An Improved Objective Function for Min-cut Circuit Partitioning. *IEEE Trans. On CAD*, pages 502–505, 1988.

[54] Carl Sechen. *VLSI Placement and Global Routing Using Simulated Annealing.* Kluwer Academic Publishers, 1988.

[55] K. Shahookar and P. Mazumder. A genetic approach to standard cell placement using meta-genetic parameter optimization. *IEEE Trans. on Computers, 9(5)*, pages 500–511, 1990.

[56] N. Sherwani. *Algorithms for VLSI Physical Design Automation.* Kluwer Academic publishers, 1993.

[57] H. Shin and A. Sangiovanni-Vincentelli. A "rip-up and reroute" detailed router. In *Proc. Intel. Conf. on Computer-Aided Design*, pages 2–5, 1986.

[58] L. Y. Song. *A VLSI Design Methodology for Artificial Neural Networks.* PhD thesis, University of Waterloo, Ont., Canada, 1993.

[59] Wern-Jieh Sun and Carl Sechen. Efficient and effective placement for very large circuits. In *in IEEE/ACM ICCAD*, pages 170–177, 1993.

[60] Wern-Jieh Sun and Carl Sechen. Efficient and effective placement for very large circuits. *IEEE Trans. on CAD , vol. 14, No. 3*, pages 349–359, March 1995.

[61] T. G. Szymanski. Dogleg channel routing is np-complete. *IEEE Trans. on Comp. Aided Design, CAD-4*, pages 31–41, January 1985.

[62] J. D. Ullman. *Computational Aspects of VLSI.* Computer Sceince Press Inc, 1984.

[63] H. Vaishnav and M. Pedram. PCUBE: A Performance Driven Placement Algorithm for Low Power Designs. In *Proc. of the EURO-DAC*, pages 72–77, 1993.

[64] J. Vygen. Algorithms for large-scale flat placement. In *Proc. of 34rd DAC*, pages 746–751, 1997.

[65] T. Yoshimura and E. Kuh. Efficient algorithms for channel routing. *IEEE Trans. on Comp. Aided Design, CAD-1*, pages 25–35, January 1982.

[66] S. G. Youssef and V. B. Rao. Combinatorial optimization by stochastic evolution. *IEEE Trans. on Computer Aided Design, 10(4)*, pages 525–535, April 1991.

[67] G. Zobrist. *Routing, Placement, and Partitioning.* Ablex Publishing Co., 1994.