Large Scale Circuit Placement and Partitioning using Nonlinear Analytical Optimization Methods

A thesis submitted in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

by

Sameer Pawanekar
(Roll Number: 126102027)

Under the Supervision of

Dr. Gaurav Trivedi

DEPT. OF ELECTRONICS & ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI

August, 2017

Contents

DECLARATION

It is certified that the work contained in the thesis titled **"Large Scale Circuit Placement and Partitioning using Nonlinear Analytical Optimization Methods"** has been done by me, a student in the Department of Electronics and Electrical Engineering, Indian Institute of Technology Guwahati under the guidance of **Dr. Gaurav Trivedi** for the award of Doctor of Philosophy and that this work has not been submitted elsewhere for a degree.

August, 2017

Sameer Pawanekar
Department of Electronics and Electrical Engineering
Indian Institute of Technology Guwahati

CERTIFICATE

It is certified that the work contained in the thesis titled **"Large Scale Circuit Placement and Partitioning using Nonlinear Analytical Optimization Methods"** by **Sameer Pawanekar**, a student in the Department of Electronics and Electrical Engineering, Indian Institute of Technology Guwahati for the award of the degree of Doctor of Philosophy has been carried out under my supervision and this work has not been submitted elsewhere for a degree. The student worked under joint supervision of myself and Dr. Kalpesh Kapoor till his synopsis seminar. Thereafter, he had been working solely under my supervision.

August, 2017

Dr. Gaurav Trivedi
Assistant Professor
Department of Electronics and Electrical Engineering
Indian Institute of Technology Guwahati

Acknowledgement

Writing this thesis has been a dream come true. There are many people who have contributed their time and energy for helping me to complete this thesis.

I would like to thank Dr. Kalpesh Kapoor and Dr. Gaurav Trivedi for their constant help during the period of writing this thesis.

I want to thank Gaurav Saini, Satyabrata Dash, Dipak Joshi and Dheeraj Sinha for their consistent help in completing the registration formalities every other semester. It was always fun to be with them during my temporary stay at IIT Guwahati.

I want to thank my friends Amol Dhok, Baskar Venugopal, Mukesh Kumar and Kishore UPP at NXP semiconductors for supporting me in consistently working towards my goal.

I cannot thank enough to my wife Geetanjali Pawanekar who encouraged and supported me whenever there was some issue. Thanks to my mother Mandakini Pawanekar , and father Sadashiv Pawanekar, and my kids Swara Pawanekar and Parth Pawanekar.

August, 2017 **(Sameer Pawanekar)**

Abstract

VLSI circuit placement and partitioning are critical steps of the VLSI circuit design flow. Adoption of an optimal placement policy is essential to the optimal performance of the electronic circuit. Due to technology scaling and integration of a large number of transistors on silicon, efficient floorplanning, placement and routing have become of paramount importance. Rapid prototyping of the electronic systems has been linked with placement directly and it has become an important parameter of the process yield. Therefore, partitioning and circuit placement methodologies need to be revisited again for improving yield during the process steps and to optimize post-fabrication performance of the electronic circuits.

In this thesis, an analytical approach is presented which is based on the nonlinear programming to perform VLSI standard cell placement and an indigenous placement tools ***Kapees3*** has been developed incorporating our proposed method. ***Kapees3*** first clusters a netlist to reduce the number of cells and then performs quadratic optimization on the reduced netlist to initialize the placement solution. Finally, it uses Nesterov's method to analyze nonlinear equations for the given problem. ***Kapees3*** is capable of performing placements of large size circuits, for example circuit composed of 12 Million cells, efficiently and its results have been verified by using standard benchmark evaluation methods. The experimental results for PEKO Suite 1, PEKO Suite 2, MMS and Free MMS benchmarks show promising improvements in terms of Half Perimeter Wirelengh (HPWL).

Kapees3 outperforms NTUPlace3, Dragon, Feng Shui, Capo by 46%, 57%, 48% and 25%, respectively, on PEKO Suite 1, while for PEKO Suite 2, it demonstrate better performance as compared to NTUPlace3, Dragon, Feng Shui, Capo and mPL6 by 30%, 47%, 57%, 69% and 2.7%, respectively. On MMS benchmarks, ***Kapees3*** obtains wirelength improvement over Capo by 56.62%, FLOP by 7.84%, FastPlace by 11.55%, ComPLx by 4.58%, POLAR by 23.67%, mPL6 by 9.96%, NTUPlace3-NR by 0.74% and NTUPlace3 by 0.46%. On Free MMS benchmarks, it outperforms NTUPlace3 and ComPLx by 32.52% and 5.73%, respectively, in terms of HPWL.

The proposed methodology is applied to the placement of 3D-ICs. ***Kapees3*** exhibits 16.4% improvement over F3D when wirelength is considered for the optimization and demonstrates an improvement of 52% in the number of Through Silicon Vias (TSVs) when the goal is to minimize usage of TSVs. This method has also been extended for the placement of cells over FPGA where a speedup

of 750% is illustrated by ***Kapees3*** when compared to a well known FPGA placement tool VPR without affecting HPWL greatly.

Hypergraph partitioning is commonly used in solving VLSI placement problem, data mining, sparse matrix multiplication, and parallel computing. For a balanced partitioning, it is imperative to achieve equal size blocks with minimum interconnection among the blocks. In this thesis a novel heuristic is presented for partitioning hypergraphs based on nonlinear programming. In the proposed method, one dimensional physically adjacent bins are considered for the placement of vertices. Since the reduction of min-cuts is equivalent to reducing wirelength of the nets across two bins, the vertices are moved across the bins in such a way that the density of vertices in each bin is balanced and wirelength is optimized as per the constraints imposed for obtaining balance partitions. The proposed method produces better cut-cost as compared to FMpart, which is an open source hypergraph partitioner based on the Fiduccia Mattheyses (FM) heuristic. It is validated for producing 2-way optimal hypergraph partitions and can be extended for k-way hypergraph partitioning.

A nonlinear analytical partitioning tool, ***NAP***, has been developed using proposed methodology and has been tested with various standard benchmarks. For the Walshaw hypergraph partitioning benchmarks, performance of ***NAP*** is consistently comparable with the performance of Chaco. It produces better min-cuts than Chaco on 22 out of 29 testcases. On ISPD98 hypergraph partitioning benchmarks, ***NAP*** outperforms FMpart on 17 out of 18 testcases and obtains an improvement of 111.5% in terms of quality of cut.

Further, partition driven placement approach is also explored and it is further improved by applying simulated annealing method for wirelength reduction. The experiments verify that the half perimeter wire lengths (HPWLs) obtained for the benchmark circuits designs by our proposed methodology are on an average 9% and 5% less as compared to HPWLs obtained using Amoeba and Capo, respectively.

In this thesis, we have investigated placement of large scale VLSI circuits in detail using nonlinear programming and partitioning based approaches and exhibits better performance of the proposed methodology to well-known standard placement and partitioning heuristics.

List of Figures

List of Tables

Chapter 1

Introduction

The advent of integrated circuits and miniaturization of electronic circuits have greatly improved our lives. The need for faster and energy efficient circuits is growing day-by-day due to emerging technologies. Technology scaling has enabled us to integrate billions of transistors on a single silicon die. On the one side, it has aided electronic circuit designers to bring distinct functions together on a single chip in the form of System-on-Chip (SoC) and to miniaturize electronic systems, but on the other side, it has made design steps such as electronic circuit design, layout design, post layout design verification etc., very critical. Therefore, it has become imperative to develop efficient methods (or more precisely, efficient Electronic Design Automation (EDA) tools) for the designing of electronic systems. The aim of EDA tools is to design large electronic circuits efficiently in terms of area, power, speed, yield, and reliability. Floorplanning, placement, and routing are the key steps of large scale electronic circuit design, and their efficiency leads to optimize yield of the circuit fabrication process.

Designing and verification of large circuits have become more challenging due to technology scaling and the limitation imposed by hardware resources. Therefore, circuit designers use state-of-the-art resources such as multi-FPGA based systems, for design and verification of large circuits. For efficient utilization of these resources, it is essential to divide (or partitioning) a large circuit into many optimal size blocks. These blocks are mapped into the design environment and emulate a large size circuit in real space. Thus, to facilitate large scale circuit design process, partitioner must be efficient qualitatively (in the processing of large size circuits as per the constraints) as well as quantitatively (i.e. runtime). After successful designing of electronic circuits, it becomes critical to realize them on the die. For the optimal floorplanning, placement and

routing, partitioner plays an important role. Each die is divided into multiple bins and each bin is populated with fix size cells arising from a standard cell, semi-custom or full custom designs. An efficient partitioner is expected to optimize circuit area on the die as well as interconnect delay and can also be used for the placement of cells over silicon. Thus, to summarize, a partitioner plays an important role in the process of design of digital circuits.

In this thesis, we present a nonlinear analytical method based placement and partitioning methodologies of electronic circuits. Placement of electronic circuits is realized for standard cell based designs and 3D circuits.

1.1 Standard Cell Placement

VLSI standard cell placement is an important and widely studied problem. This is a stage in the VLSI design flow during which the cell locations are identified. It affects timing, routability, power consumption and performance of a chip. Therefore, the chosen placement methodology should be able to produce optimal placement of the circuit over silicon. In the design process, circuit's information is stored in the form of a *netlist*. Objective of the placement is to identify coordinates of all the cells of a given netlist and place it over silicon in such a way that the average wirelength of a given netlist is minimum [5]. The general placement problem is known to be NP-hard [51]. This can be proved by considering a simplified version of the problem known as the optimal linear arrangement that requires optimizing the wirelength by placement of circuit netlist with cells of unit size and net weights (weight of the interconnects) with 2-pins [43].

There are various ways to model wirelength and Half Perimeter Wirelength (HPWL) is a commonly used model employed in a number of placement engines. This is defined as half of the perimeter of the smallest bounding box which encloses all the cells of a net. We consider the sum of half perimeter of such bounding boxes for all the nets to be optimized during placement process. Besides obtaining the coordinates that gives least HPWL, a placement tool's objective is also to legalize the placement. This ensures that all the cells occupy positions on given rows of the layout and no cell is out of the placement region or overlaps with another cell. It should be noted that each row is of the same height which is also the height of standard cells of a given netlist.

A variety of techniques have been developed to solve the placement problem. Four broad methods that have been used to solve the placement problem are

min-cut approach [3,16,127,160], simulated annealing [141], quadratic approach [11, 45, 66, 67, 81, 85, 86, 88, 96, 154, 155] and nonlinear approach [22, 28, 39, 73, 100, 101, 112]. In spite of the availability of academic and commercial tools, there is a scope to improve the placement quality because of the complexity of the problem. An industry standard placement tool should not only be able to solve the placement problem in a reasonable time but also give a solution that is of high quality. It has been observed in a placement contest conducted by International Symposium on Physical Design (ISPD) [109] that none of the placement tools could dominate across the entire set of benchmarks. A comparative study [23] also indicates that the solutions obtained from current state-of-the-art placement tools are far from optimal.

1.2 3D Integrated Circuit (3D IC) Placement

The possibility of heterogeneous integration in 3D integration technology makes Internet of Things (IoT) a potential application of three-dimensional Integrated Circuits (3D ICs). The researchers and circuit designers are working towards the development of 3D IC integration and implementation technology. As a results novel 3D placement methods are being proposed for the efficient implementation of circuits in the 3D stack. For the emerging technologies, electronic circuits should be energy efficient and 3D ICs are no exception to this. Energy optimization in 3D ICs can be linked to the optimization of wirelength, number of TSVs and thermal management (or power density) of 3D ICs. Therefore, optimization of wirelength, the number of Through-Silicon-Vias (TSVs) and power density can be addressed during the placement of 3D ICs.

The advantage of 3D ICs over 2D ICs is that 3D ICs offer higher performance with nearly same density of cells. Thus, it is expected that the computer aided design (CAD) tools used for the design of 3D ICs, focus on the performance of their output. The performance of 3D ICs can be improved by optimized 3D placement that will result in optimal wirelength, number of TSVs and power density. Number of TSVs used in the design affects the density of the 3D IC as TSVs themselves utilize area on the die. Therefore, it is important to optimize the number of TSV utilization during placement. Since the heat dissipation is managed through TSVs, it is imperative to optimize power density (for thermal management) of the 3D ICs. This requires solution provided by 3D placement tool to distribute power density of the cells in a suitable manner to achieve optimal performance of the 3D IC.

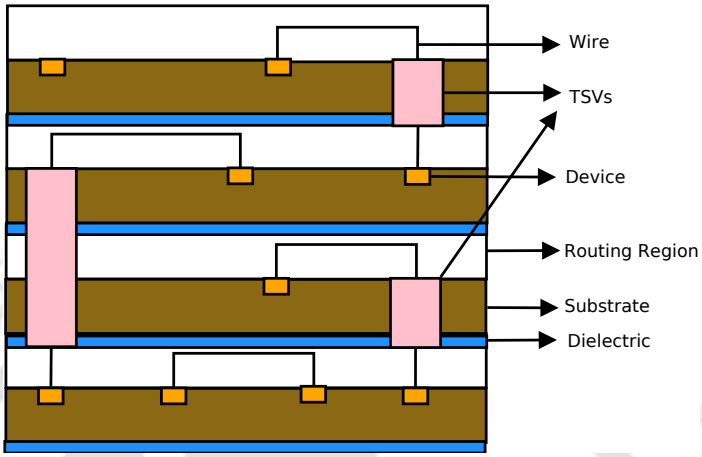

Figure 1.1: Cross section of a 3D IC.

Cross section of 3D IC with TSV is shown in Figure 1.1. TSVs are required to form nets which span across different layers of 3D ICs. TSVs connect routing region of one layer to the routing region of another layer as they pass through the substrate of the target layer. In case, if TSV spans over multiple layers, it has to cross all the regions of the intermediate layers. In general, TSVs are connected to the devices using metal wires in the routing region.

1.3 Partitioning and Partition Driven Placement

Hypergraph partitioning has its applications in many areas such as VLSI physical design [16, 128, 160], VLSI formal verification [110, 137], parallel computing [41, 90, 104, 118], sparse matrix-vector multiplication [10, 18, 165], data mining [173] etc. Hypergraph partitioning is known to be an NP-complete problem [52].

A hypergraph is a generalization of a graph where each edge (referred to as hyperedge in hypergraph) can have more than two end points. Given a hypergraph H, a partitioning of H divides the vertices into two parts, in such a way that each partition consists of a nearly equal number of vertices with minimum crossing edge set between partitions. The ratio of the sizes of two

parts of a partition is known as the imbalance. The number of hyperedges crossing the two parts in a partition is known as cuts.

Some of the existing hypergraph partitioning tools are Chaco [93], hMetis [79], MLPart [7], PaToH [19], Mondriaan [153], Parkway [147] and Zoltan [175] etc. The hypergraph partitioning algorithms can be broadly classified into (1) Constructive methods and (2) Iterative Improvement Partitioning methods. The methods based on spectral partitioning [25, 54] and network flow based partitioning [169] come under the first category. Most of the hypergraph partitioners developed in academia and industry are based on iterative improvement. Many state of the art methods such as hMetis [79] and MLPart [7], PaToH [19] rely on methods such as Kernighan-Lin (KL) [82] and Fiduccia-Mattheyses (FM) [49].

Hypergraph partitioning application to the placement of VLSI circuits is particularly important as it affects the quality of placement. A poor cut obtained during partitioning can lead to an increased wirelength in the resulting layout that is obtained after placement. A partitioning solution which is formed considering the fixed terminals is of particular interest in VLSI circuit placement. Some of the vertices (pads in VLSI circuits having fixed coordinates) are assigned a fixed partition during hypergraph partitioning. Fixed terminal based partitioning has been emphasized in [6]. Hypergraph partitioning based VLSI placement has been performed in the placement tools [3, 16, 128, 160]. It is evident from the recent International Symposium on Physical Design (ISPD) placement contests that partition driven placement tools lag behind analytical placement tools. This may be due to the quality of the partitions produced by the partitioners. There is a scope for improvement of the hypergraph partitioning algorithms in the context of VLSI placement and particularly VLSI Placement with fixed terminals.

In this thesis, we present our study of hypergraph partitioners and partition driven placement along with the placement methods for standard cell and 3D IC placement.

1.4 Organization of the Thesis

This thesis is organized in the following manner:

- In Chapter 2, basic preliminaries, needed for the thesis to be self-contained, are discussed.

- A survey of existing tools and techniques for placement and partitioning of VLSI circuits is presented in Chapter 3.
- In Chapter 4, standard cell placement using nonlinear analytical techniques is described in detail.
- Chapter 5 consists of a description of a nonlinear analytical approach based hypergraph partitioner along with its comparison with existing well-known methods.
- Chapter 6 describes 3D placement technique for TSV based designs.
- In Chapter 7, partition driven placement heuristic and its comparison with standard methods are presented.
- Conclusion and future directions of the proposed work are described in Chapter 8.

Chapter 2

Preliminaries

In this chapter, we describe basics which will be used frequently in order to explain other technical details in the rest of the thesis.

2.1 Hypergraph Partitioning

This section presents formulation and notations of partitioning problem which is addressed in this thesis.

A circuit forms the input of partitioning problem and is represented by a hypergraph, $H(V, E)$, where the set of vertices is denoted by $V = v_i | i = 1, 2, \cdots, n$ which are the representation of modules, cells and instances of the circuit, and the set of hyperedge is denoted by $E = e_j | j = 1, 2, \cdots, m$ which are nets of the input circuit. Each net e_j is a subset of V with cardinality $|e_j| \geq 2$. The modules in e_j are called the *pins* of e_j.

As shown in Figure 2.1, a circuit represented in the form of hypergraph consists of six nets (hyperedges) and nine modules. where nets e_1, e_3 and e_5 are two-pin nets, net e_6 is a three-pin net, and nets e_2 and e_4 are four-pin nets. If the hypergraph consists of only two-pin nets (hyperedges), the representation is equivalent to a graph $G(V, E)$. In this thesis, we assume that all the circuits are represented as hypergraph. The terms netlist, circuit, and hypergraph have been used interchangeably throughout this thesis, and the hypergraphs can be considered as a topological representation of electronic circuits. In the subsequent subsections, we have described partitioning schemes based on Kernighan-Lin (KL) [82] and Fiduccia-Mattheyses (FM) algorithms.

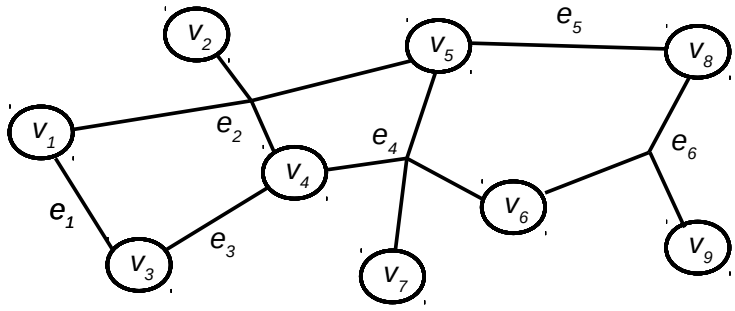

Figure 2.1: Example of a hypergraph.

2.1.1 Kernighan-Lin (KL) Partitioning

Input to the algorithm is in the form of an undirected graph $G = (V, E)$ with vertex set V, edge set E, and numerical weights on the edges in E. The aim of the KL algorithm is to partition V into two subsets A and B which are disjoint and of equal (or nearly equal) size, in such a way that it minimizes the sum T of the weights of the subset of edges that cross from A to B. If the graph is not weighted, then instead of the sum of weights of the crossing edges, the number of crossing edges is minimized which is equivalent to consider unit weight on each edge. The KL algorithm employs a greedy strategy in each pass of the algorithm to improve partitions and maintain it. The algorithm pairs up vertices of A with vertices of B, so that each partition is improved by the movement of paired vertices from one partition to another. After performing matching of vertices, it chooses a subset of pairs to have the best outcome on the solution quality. Each pass of the KL algorithm runs in the time of order $O(n^2 log(n))$, where n is the number of vertices present in the graph.

The whole process of refinement of partitions is explained in the following manner. Let the internal cost of node a be represented by Int_a, which is defined as a sum of the costs of edges between node a and other nodes in A. Let the external cost of node a be represented by Ext_a, which is defined as sum of the costs of edges between node a and other nodes in B. The difference between the external and internal costs of node a be explained as $Diff_a = Ext_a - Int_a$ If a and b are interchanged, then reduction in the cost is $T_{old} - T_{new} = Diff_a + Diff_b - 2c_{a,b}$, where $c_{a,b}$ is the cost of the possible edge

between a and b. The algorithm searches for an optimal series of interchanges between elements of A and B which maximizes $T_{old} - T_{new}$ and then generates an optimal partition of A and B.

2.1.2 Fiduccia-Mattheyses (FM) Partitioning

The algorithm of Fiduccia-Mattheyses (FM) for hypergraph partitioning [49] is an iterative refinement process. FM algorithm starts with an initial random seed solution and changes it iteratively by a sequences of organized vertex movement across partitions. Initially, all the vertices are free to be considered to get moved between partitions and all possible movements are labeled with the change it causes. This change is defined as the gain of the move. Positive gain reduces solution cost, whereas negative gain increases the solution cost. To start the process, a move with the highest gain is selected and the process of vertex movement is executed. After the movement, the moved vertex is locked and not allowed to move in further passes. After the movement of a vertex in a pass, the gains of the adjacent vertices are updated. In each pass, the best gain move is selected for the movement and gains of adjacent vertices are updated after the movement. This process is carried out until all the vertices are locked and no further movement is possible. At this stage, a pass is considered to be complete. During these passes, the best solution is recorded, and is considered as initial solution for the next pass. The algorithm is terminated when a pass fails to improve the quality of the solution. The three main steps of FM algorithm are described below.

1. In the beginning of the pass, initial gain values are computed.
2. Selection of the move with best positive gain.
3. Incremental update of the gain values as the selected best gain vertices are moved across the partition.

For sparse hypergraphs, gain of any movement, g_m is bounded by $(-{d_v}^{max} \leq g_m \leq {d_v}^{max})$ where ${d_v}^{max}$ is the maximal vertex degree in the hypergraph. In case of weighted hypergraph, the bound on the gain is defined as $(-{d_v}^{max} w_e^{max} \leq g_m \leq {d_v}^{max} w_e^{max})$, where w_e^{max} is the maximal hyperedge weight. Due to this constraint, gains of moves can be prioritized which helps in refinement of the partitions and in producing balanced partitions.

2.1.2.1 FM Passes

In FM algorithm, the focus is on improving the passes incrementally [107]. Within each pass, the algorithm searches for a neighborhood of an initial partition and updates the best solution during the process. The passes continue until an improvement is seen in the solution. During each pass, the algorithm chooses the best move, repetitively, and applies it, which is followed by the updating the gains and saving the best solution thus obtained. No vertex can be moved more than once during a pass, thus, beyond a certain point, the number of available moves reduces. It is possible that the best gain move increases the cost of the solution, therefore, the solution obtained in the final pass may not be as good as intermediate passes. So, it is important to store the best seen intermediate solutions throughout the process.

2.1.2.2 Gain Computation and Gain Update

Initialization of FM data structure at the start of each pass is very important. At first, the number of vertices in each partition are counted by traversing through hyperedges. Later gain values of the vertices are decreased which are incident on the hyperedge forming crossing edge set (or cut). This gain is decreased by the weight of the hyperedge or by -1, if hyperedge has no weight. If there is only one vertex in a partition, then gain of that vertex is increased by hyperedge weight (or by 1 when hyperedge has no weight) and gains of all other adjacent vertices are unchanged. FM fetches gain of the best pair from the gain container and continues to do so until a feasible move is found. As FM applies the selected move and performs locking of the vertices, gains of adjacent vertices need to be updated in each iteration. In performing the gain update, the FM algorithm traverses through all hyperedges incident to the moving vertex and for each hyperedge gains are updated for each of the incident vertices. These partial update of gains is immediately applied to the gain container. If the incremental gain for a said move is zero, it is inserted into gain container to resolve the conflict between moves with the same gain.

2.1.3 hMetis

hMetis [79] is the well-known hypergraph partitioner based on multilevel hypergraph partitioning schemes. For partitioning, *hMetis* employs three important phases which are coarsening phase, initial partitioning phase and uncoarsening

phase. In the coarsening phase, a sequence of successively smaller hypergraphs is constructed. After several coarsening phases, a small hypergraph is obtained having very few vertices. This helps in getting a good solution since methods such as Kernighan-Lin are efficient on smaller hypergraphs but are not effective in refining larger hypergraphs. The purpose for reducing the problem size is to obtain a good cut on a smaller problem size in such a way that the solution is not significantly worse than the solution obtained on the original hypergraph. In the initial partitioning phase, a bisection of the coarsened hypergraph is obtained. As the number of vertices are very small, different algorithms can be used without affecting the runtime and quality of the solution. *hMetis* employs multiple random bisections followed by the Fiduccia-Mattheyses (FM) refinement algorithm. In uncoarsening and refinement phase, the partitioning of the coarsest hypergraph is used to obtain partitions for the finer hypergraph. This is done by successively projecting partitions to the next level hypergraph and employing a partitioning refinement algorithm to optimize cut and, thus, improve the quality of partitions. Since the next level finer hypergraph has more degrees of freedom, such refinement algorithms tend to improve the quality.

2.2 Placement

VLSI placement deals with placement of the cells of electronic circuit design on silicon. The circuit designed can either perform logical or arithmetic functions or can be used as a memory. Placement of cells of a logical or arithmetic design is more difficult than the placement of memory elements on a die area. In VLSI placement, standard cells/modules need to be placed in a die area in such a way that their edges are parallel to x or y axis. The modules or cells are logic gates that implement a Boolean function and communicate with each other through nets. In the following subsections, standard cell placement, 3D IC placement and partition driven placement are discussed.

2.2.1 Standard Cell Placement

The placement problem can be defined as follows. Placement is a mapping of x_f and y_f to functions

$$x : C \rightarrow x_\delta N \; and \; y : C \rightarrow y_\delta N$$

i.e., $x(c) = x_f(c)$ and $y(c) = y_f(c)$ for $c \in C_f$ such that the open rectangles in $]x(c), x(c) + w(c)[\times]y(c), y(c) + h(c)[|c \in C$ are pairwise disjoint subsets of r_0 and do not intersect any blockage. Here, $(x(c), y(c))$ is the lower left corner of the rectangle of width $w(c)$ and height $h(c)$ that is covered by c. It should be noted that the areas covered by different cells do not overlap. The elements of C_{row} are called *row cells* or *standard cells* and their height y_δ is called *standard height*. The elements of C_{macro} are called *macros*. We may assume that C_{macro} contains much less elements than C_{row}. In reality, there are several millions of row cells but at most a few thousand macros.

For aligning standard cells to the rows, it is required that y-coordinate of the cells are integer multiples of the height of the standard cell rows y_δ. This is required as the cells are connected to a power supply which are routed within the gaps between rows. The row-based layout helps in the routing of power supply in an easier way, along the length of the standard cell rows. It would be difficult to route the power supply to the cells, if standard cell rows do not have a regular structure. The x-coordinate of the cells must also be an integer multiple of x_δ for enabling routing step. x_δ is normally much smaller than y_δ and also smaller than most of the circuit widths. It should be noted that finding a feasible solution (x, y) for a given design netlist is an NP-complete problem (see Karp [51]).

2.2.2 3D IC Placement

For performing 3D integration, an additional third dimension is added to VLSI design process. It shortens the length of global interconnects and improves device packaging density. The benefits of 3D integration are performance, functionality and power of 3D ICs. Most of the advantages get reflected by optimizing the length of global interconnects as compared to 2D interconnects. It is important to develop design methodologies and CAD algorithms that take advantage of the 3D features. These methods can be employed at floorplanning stage of placement of 3D circuits. Apart from traditional wire length optimization, 3D IC placement objectives include number of vias, thermal management (or power density) as important and significant design performance optimization parameters. Due to the stacked structure of 3D ICs, thermal issues occur frequently. Heat dissipation is channelized by using Through-Silicon-Vias (TSVs) because of its greater thermal conductivity as compared to the dielectric layers. An increase in the number of TSVs during placement of 3D IC over silicon improves thermal conductivity but affects area over silicon and wirelength adversely, whereas, a decrease in the number of TSVs improves the

performance of 3D IC due to optimal selection of wirelength and minimization of area, but thermal conductivity becomes poor. Therefore, while placement of 3D ICs over silicon, it is imperative to optimize design using parameters responsible for the performance of 3D ICs.

2.2.3 Partition Driven Placement

Many placement algorithms employ various approaches for top-down partitioning while placing electronic circuits over silicon. During the partition driven placement, chip area and the netlist are divided recursively, which is followed by assigning the sub-area to a sub-circuit obtained from recursive partitioning. Recursive p partitioning continues until the regions are small enough to perform legalization. The main problem to be analyzed in partition driven approach for placement is the assignment of area to the partitioned sub-circuits. The widely used objective function is the cut-size of the partition, which is the number of nets crossing bisected partitions. Min-cut based partition driven placement algorithms apply heuristics to find a min-cut because the problem of computing a balanced partition of the circuit into two sets is known to be NP-complete [14](Bui and Jones [1992]). The multilevel partitioners, such as MLPart [4] and hMetis [79] etc., are employed in obtaining a min-cut of the circuit. These multilevel partitioners tend to produce a better cut, at the expense of extra runtime. The min-cut partitioners are faster than simulated annealing or genetic programming based algorithms and can produce reasonably good results. Group of cells that are connected strongly with each other is protected from being partitioned and get placed in the same area. Due to this wirelength of the region containing strongly connected cells is smaller, whereas, if the cells are not strongly connected, they may be partitioned into different blocks, and are placed in such a way which optimizes wirelength of the design.

An initial guess, far from optimal, in the beginning of the partitioning, may lead to producing poor partitions during recursive partitioning. This affects the quality of the placement achieved by the said approach. Therefore, in order to make partition driven placement a successful and efficient methodology, choice of good initial seed partitions should be emphasized in the placement process.

2.3 Optimization Methods

The problems addressed in the thesis are optimization problems, in general. Therefore, we present a brief discussion on a few optimization methods in this

section for the thesis to be self-contained.

2.3.1 Simulated Annealing

Simulated Annealing is inspired from nature, the phenomenon of annealing of solids is adapted to optimize a complex problem. Annealing is gradual cooling of a hot solid by which atoms of the solid achieve a globally minimum energy state. Simulated Annealing was developed by Metropolis in 1952 as an algorithm to solve optimization problems. This algorithm simulated a small random change in the position of an atom which may increase of decrease the energy. If this energy change is negative, the new state was accepted, whereas, if the energy change is positive, it may get accepted based on the probability. Let ΔE be the change in energy. According to Boltzmann probability factor, the probability of acceptance of change is $e^{-\Delta E/k_b T}$, where T is the current temperature and k_b is the Boltzmann constant. Two important observations can me inferred from this equation:

1. The probability is proportional to temperature as the solid cools down.

2. The probability is inversely proportional to change in energy because an increase in the change of energy reduces its probability of acceptance.

Application of simulated annealing for the solution of optimization problems enables change of energy to be considered as the objective function. The cost (energy) is high due to high temperature during initialization step. As shown in Algorithm 1 random perturbation is performed on the objective function along with generation of a random number. If the perturbation decreases the cost of the objective function, the change is accepted, whereas, the acceptance of rejection of the perturbation, which increases the cost of objective function, depends on the random number generated. If this random number is smaller than the Boltzmann probability P, the perturbation is accepted, whereas, if the random number is larger than Boltzmann probability P, the perturbation is rejected. This prevents simulated annealing from getting trapped in local minima. As the temperature is reduced gradually, the probability of improving the cost of the objective function increases. Simulated annealing algorithm may not find optimal solution but it can find a near optimal solution of any optimization problem. This algorithm is easy to implement but it can be computationally expensive if the problem size is large.

Algorithm 1 Simulated Annealing

```
while init_temp ≥ final_temp do
    P = e^{δCost/init_temp}
    Perturb
    if δ Cost ≤ 0 then
        Accept the perturbation
    else
        if random_number ≤ P then
            Accept the perturbation
        else
            Revert the perturbation
        end if
        if sufficient_number_of_perturbations then
            Decrease init_temp
        end if
    end if
end while
```

2.3.2 Conjugate Gradient

Conjugate gradient method is typically used to solve a system of linear equations described by $A^TAx = A^Tb$. The nonlinear conjugate gradient method is a general form of the conjugate gradient method for nonlinear optimization. For a quadratic function

$$f(x) = \|Ax - b\|^2$$

The minimum of f can be obtained when its gradient is zero i.e.

$$\nabla_x f = 2A^T(Ax - b) = 0$$

The nonlinear conjugate gradient method is used to find the local minimum of a nonlinear function using its gradient $\nabla_x f$. This method works well when the function is approximately quadratic near minima. Thus, the function should twice differentiable at minima and its second derivative should be non-singular. Given a function $f(x)$ of N variables to minimize, the direction of maximum increase is given by its gradient $\nabla_x f$. In order to obtain minima, solution space is explored in the opposite (steepest descent) direction.

$$\Delta x_0 = -\nabla_x f(x_0)$$

In order to find step length α, a line search needs to be performed and continue to do so until it reaches minimum.

$$\alpha_0 := \arg\min_{\alpha} f(x_0 + \alpha \Delta x_0)$$

$$x_1 = x_0 + \alpha_0 \Delta x_0$$

After the first iteration towards the steepest direction Δx_0, the following steps constitute one iteration of solution search along a subsequent conjugate direction s_n, where $s_0 = \Delta x_0$.

- Calculate the steepest direction:

$$\Delta x_n = -\nabla_x f(x_n)$$

- Compute β_n as per the mathematical expressions given below.
  - Fletcher Reeves:[1]

$$\beta_n^{FR} = \frac{\Delta x_n^T \Delta x_n}{\Delta x_{n-1}^T \Delta x_{n-1}}$$

  - Polak Ribiere:[2]

$$\beta_n^{PR} = \frac{\Delta x_n^T (\Delta x_n - \Delta x_{n-1})}{\Delta x_{n-1}^T \Delta x_{n-1}}$$

  - Hestenes-Stiefel:[3]

$$\beta_n^{HS} = -\frac{\Delta x_n^T (\Delta x_n - \Delta x_{n-1})}{s_{n-1}^T (\Delta x_n - \Delta x_{n-1})}$$

  - Dai Yuan:[4]

$$\beta_n^{DY} = -\frac{\Delta x_n^T \Delta x_n}{s_{n-1}^T (\Delta x_n - \Delta x_{n-1})}$$

  A popular choice among the four above is $\beta = \max\{0, \beta^{PR}\}$, which resets the direction automatically.

- Update the conjugate direction:

$$s_n = \Delta x_n + \beta_n s_{n-1}$$

- Perform line search:

$$optimize\ \alpha_n = \arg\min_{\alpha} f(x_n + \alpha s_n)$$

- Update the position:

$$x_{n+1} = x_n + \alpha_n s_n$$

If the objective function is quadratic, minima can be obtained in N iterations, whereas, an optimizing non-quadratic function is anticipated to take more steps. When no further refinement is possible in reducing the cost of objective function, conjugate gradient algorithm is terminated and solution vector at this step is considered to be the final solution. The pseudo code of Conjugate Gradient is expressed as Algorithm 2.

Algorithm 2 Conjugate Gradient Method

Require: $f(x)$: objective function and x_0: initial solution
Ensure: optimal x^*

1: $h_0 = 0$; $d_0 = 0$
2: **while** $f(x_k) < f(x_{k-1})$ **do**
3: $h_k = \nabla f(x_k)$ ▷ Gradient directions
4: $B_k = \dfrac{h_k^T(h_k - h_{k-1})}{\|h_{k-1}\|^2}$ ▷ Polak-Ribiere parameter B_k
5: $d_k = -h_k + B_k d_{k-1}$ ▷ Conjugate directions
6: $\alpha_k = \arg min_{\alpha} f(x_k + \alpha \Delta x_k)$ ▷ Line search
7: $x_k = x_{k-1} + \alpha_k d_k$ ▷ Solution update
8: **end while**

2.3.3 Nesterov's Method

In general, Nesterov's algorithm is employed for minimizing a convex function and is applied to many fields including machine learning. The objective of our study (placement of electronic circuits) is to minimize $f(x)$, where f is a convex function, smooth or non-smooth, and $x \in R$ is the variable. In 1983, Nesterov proposed an accelerated gradient method, which takes the following form, with the initialization starting with x_0 and $y0 = x0$, following equations can be inductively defined [146]

$$x_k = y_{k-1} - s\nabla f(y_{k-1}) \tag{2.1}$$

$$y_k = x_k + \frac{k-1}{k-2}(x_k - x_{k-1}) \tag{2.2}$$

For any fixed step size $s \leq 1/L$, where L is the Lipschitz constant of ∇f, this method exhibits convergence rate

$$f(x_k) - f^* \leq O(\frac{\|x_0 - x^*\|^2}{sk^2}). \tag{2.3}$$

Above x^* is any minimizer of f and $f^* = f(x^*)$. The main contribution of this method is the introduction of the momentum term $x_k - x_{k-1}$ as well as the coefficient $(k-1)/(k+2) \approx 1-3/k$. The runtime of this method is proportional to $1/k^2$ where k is the number of iteration.

Chapter 3

Literature Survey

In this chapter, methodologies proposed by various researchers are presented for the placement and partitioning of electronic circuits.

3.1 Simulated Annealing Based Methods

It is known that the placement problem is computationally hard. Therefore, for the placement of large electronic circuits, meta-heuristic techniques based on simulated annealing and genetic programming have been explored [141]. A typical approach is to cause a perturbation in the cell coordinates before optimizing the wirelength. This often achieves good placement for small circuits. However, for large circuits the above-mentioned methods imposed heavy penalty on the runtime [162].

3.2 Partitioning Based Methods

The placement approach that employs min-cut technique partitions a circuit into smaller sub-circuits for improving HPWL. The tool Feng Shui [3] performs large scale k-way partitioning during global placement and legalization by iterative deletion to get initial terminal propagation information. The detailed placement is performed by a branch and bound method.

Placement tools Capo [16], Dragon [160] and Kapees [127] also incorporate a top-down hierarchical partitioning approach. Capo makes use of the hypergraph partitioning tool MLPart [4] whereas Dragon employs hMetis [79].

During global placement Dragon performs bin swapping and in detailed placement step it uses low temperature annealing to improve net cuts.

The placement tools that use partition driven paradigm typically employ a terminal propagation scheme to obtain a better cut during partitioning stage. The characteristics of cut nets are gathered by partitioner using a terminal propagation scheme which is further used to derive good wirelength results. In NTUPlace [27] a terminal propagation method is proposed which is a generalization of a method from Theto [142].

3.3 Force Directed Methods

Eisenmann et al. [47] proposed force directed placement and developed a global placement engine, *Kraftwerk*. They introduced forces to reduce overlaps in addition to forces working towards reducing wirelength in their global placement work titled Kraftwerk. To reduce the overlap, the various forces work on each cell in the x and y direction and try to move them from high density region to low density region of the layout. The additional forces of density formulation are determined based on the current placement solution.

In each iteration, individual forces acting on the cells are assumed to be constant and are used to calculate a new placement. This new placement acts as an initial placement solution for the next iteration step. Each step of the algorithm is called a placement transformation. The transformation step can be applied to fully overlapping placements as well as nearly legal placements. It is discussed in [47] that the proposed algorithm can handle large mixed-size placement problems without treating macros and standard cells differently.

Kraftwerk2 [117] is another force directed placement tool which does not perform clustering and works directly on a flat netlist. Its runtime is high as it solves Poisson's equations during the placement flow.

The tool mPL6 [39] employs a generalized force directed approach using partial differential equations to solve the placement problem. Their implementation is multilevel which performs clustering and de-clustering using a V-cycle refinement approach.

3.4 Quadratic optimization Methods

The placement tool FAR [67] uses quadratic optimization and introduces pseudo fixed points. It maintains a force equilibrium among the cells that are not fixed. Another placement tool RQL [155] is also based on quadratic optimization which performs density aware module spreading. FastPlace [154] introduces methods such as iterative local refinement, cell shifting and hybrid net model, when performing quadratic optimization. Kraftwerk2 [117] is another force directed placement tool which does not perform clustering and works directly on a flat netlist. Its runtime requirements are high as it solves Poisson's equations during the placement flow.

Gordian [88] proposed netlist partitioning driven placement using quadratic programming. BonnPlace [11] partitions the netlist using quadratic placement such that the resulting sub regions contain area of the cells proportional to its size. As the level of partitioning grows, cells are spread evenly across the die area. Proud [150] also employs multilevel partitioning of the input circuit using quadratic optimization but it is based on the concept of resistive network optimization [29]. FFTPL [100] based on use of fast Fourier transform for bin density equalization. Another placement tool eplace [101] attempts electrostatic force optimization which works on a flat netlist. SimPL [85], is a flat quadratic global placement tool which maintain upper and lower bounds on placements that converge to a solution. Various placement tools and their techniques are summarized in Table 3.1.

3.5 3D Standard Cell Placement

The first generation of 3D placement tools such as of Cong et al. [38] are based on transformation of 2D placement to 3D placement via folding techniques. Goplen et al. [53] proposes a forced directed approach for the placement of 3D ICs considering thermal issues. In this approach, Finite Element Analysis is used to optimize power density during placement. Cong and Luo et al. [103], [36] and Hsu et al. [64] approach to 3D placement is based on the nonlinear analytical approach and to address issues such as number of TSVs, thermal management (power density) and wirelength optimization. Cong et al., in [37], demonstrates that the minimization of power density is achieved when combined area of TSV cells for each bin is proportional to the power consumption in that bin. In [84], Kim et al., studies the effect of TSVs while implementing force directed 3D

Table 3.1: Various Placement Tools and Techniques

Technique	**Placement Tool**
Simulated Annealing	Dragon
Partitioning	Capo Feng Shui Theto
Quadratic	RQL FDP FAR mFAR FastPlace Kraftwerk PROUD Gordian BonnPlace
Nonlinear	Aplace NTUPlace3 mPL6 FFTPL ePlace ***Kapees3***

placement method for the placement of 3D ICs and proposes an approach for the optimization of TSVs.

Liu et al., in [98], [99], partition circuit to be placed on different stacked dies and later apply quadratic programming to optimize wirelength. Hentschke et al., in [60], apply 3D quadratic optimization method along with techniques such as cell shifting to spread the cells, and iterative refinement techniques to reduce the wirelength further. Gao et al., in [50], implement a novel layer assignment method using minimum cost flow which does not degrade the solution space from optimal during legalization. [168] proposes a 3D placement method based on Discrete Cosine Transformation (DCT). In this proposed methodol-

Table 3.2: Various 3D placement tools and their techniques

3D placement tool	Technique	Sub-Technique
Goplen et al. [53]	Analytical	Force Directed
Cong et al. [103]	Analytical	Nonlinear
Cong et al. [36]	Analytical	Nonlinear
Cong et al. [38]	Transformation	Layout Folding
Hsu et al. [64]	Analytical	Nonlinear
Cong et al. [37]	Analytical	Nonlinear
Kim et al. [84]	Analytical	Force Directed
Liu et al. [98], [99]	Analytical	Quadratic Optimization
Hentschke et al. [60]	Analytical	Cell Shifting and Quadratic
Gao et al. [50]	Analytical	Layer Assignment and Quadratic
Yan et al. [168]	Analytical	DCT transformations
Li et al. [94]	Finite Difference	FD-TVP
Ours	Analytical	Nonlinear

ogy, quadratic optimization is employed for achieving a thermal aware placement solution. In [94], method proposed by Li et al., is based on finite difference thermal mesh model and finite difference thermal via model (FD-TVP) to analyze thermal aware placement problem. Various 3D standard cell placement tools and their techniques are summarized in Table 3.2.

3.6 Partitioning

In this section, we give an overview of the existing methods of hypergraph partitioning.

Implementation of Chaco by Leland and Hendrickson et al. in [93], is an example of multilevel graph partitioning which includes implementation of a variety of algorithms based on spectral graph partitioning and FM to support parallelization . It includes generalization of the KL/FM algorithm to support partitioning of graphs with vertex weights.

Karypis et al. [78, 79] primarily demonstrates FM [49] with multilevel implementation. The proposed method is a state-of-the-art approach, superior in quality of cut cost and runtime as compared to many other methods. In this implementation, first a hypergraph is clustered to reduce its size, and then FM is performed on the reduced size hypergraph. Finally, declustering is

performed to obtain partitioned hypergraph. These steps are performed in a V-cycle refinement until number of cuts are reduced to a satisfactory limit.

Alpert et al. [6, 7] showcases a hypergraph partitioning tool MLPart, which is very popular in academia and industry. MLPart is an open source hypergraph partitioner and is also used by the placement tool Capo [16] for the placement of electronic circuits. It is a faster as compared to hMetis but often produces large cut sizes than hMetis [108]. MLPart is based on MLFM i.e. multi level FM [49] scheme.

A hypergraph partitioning tool, *PaToH*, based on MLFM is developed by Catalyurek et al. [19, 124]. It is successfully applied in the sparse matrix multiplication for optimizing number of fill-ins. The quality of cuts produced by PaToH is comparable to the cuts produced by hMetis and MLPart. Devine et al. [40, 175], exhibits a parallel hypergraph partitioning tool, *Zoltan*, in 2006, which is also based on multilevel framework, and supports wide range of applications such as parallel preconditioners, graph coloring, graph ordering and dynamic load balancing. *Parkway* is a parallel hypergraph partitioner developed by Trifunovic et al. [147] employing multilevel framework. Instead of recursive bisection of a hypergraph, k-way partitioning approach is employed for parallel partitioning of a hypergraph.

Mondriaan, a two-dimensional graph partitioning method for the solution of matrix partitioning problem is proposed by Vastenhauw et al. [153]. This can be used to solve graph partitioning problem by representing graph in adjacency matrix form.

Chris Walshaw et al. [158] have developed a multilevel mesh partitioning algorithm based on KL method. These are part of a comprehensive package of graph partitioning *JOSTLE*. Peter Sanders et al. [139] have developed a fast heuristic *KaPPa* using new local search techniques that are based on max-flow and min-cut strategies.

Trivedi et al. in [148] developed a heuristic to generate initial seed for the FM partitioner using a DC analyzer. This improved the cuts substantially obtained from FM algorithm. Various partitioners and their associative techniques are summarized in Table 3.3.

3.7 Legalization Methods

Legalization is an important step in VLSI placement. Legalization is performed to ensure the following.

Table 3.3: Various Graph Partitioning Tools and Techniques

Technique	**Sub-Technique**	**Partitioning Tool**	Year
Iterative	Multi-level KL	Chaco	1995
	Multi-level FM	hMetis	1995
	LSF/MFFS	Cong et al. [35]	1997
	Edge Frequency	Wichlund et al. [163]	1998
	Multi-level FM	MLPart	1999
	Multi-level FM	PaToH	1999
	Hybrid Bucket	Eem et al. [44]	1999
Parallel	Multi-level FM	Zoltan	2000
Perturbations	GBAW	Cheung et al. [166]	2003
Parallel	Multi-level FM	Parkway	2004
	Matrix Partitioning	Mondriaan	2005
	Multi-level KL	JOSTLE	2007
	Multi-level Diffusion	DIBAP	2008
	Multi-level local search	KaPPa	2011
Analytical	Nonlinear	*NAP* (this paper)	2015

- Each of the standard cell occupies a location on the given row of the layout of the chip.
- There are no overlaps among the standard cells.
- All the cells of the design are confined within the area of the chip.

Typically, legalization is performed after global placement and is followed by detailed placement procedures. A simple method commonly used for standard cell placement legalization is Tetris [61]. At first, x-coordinates are multiplied by a scaling factor (ranges from 0.85 to 0.95) to ensure that all the cells should not get pushed out of the placement region. Later, the cells are sorted based on their x-coordinates. It is followed by founding, from left to right for each cell, the nearest location on any of the rows. The row location which gives least increase in HPWL is selected. During legalization, increase in HPWL may be observed, which is expected because of a significant number of cells may get moved from their original positions.

T. Chen et al. in [28] extend the Tetris [61] method for mixed size designs. In this proposed method, the cells based on *priority*, which is computed by considering coordinates, width and height of standard cells, are sorted initially. Legalization is performed multiple times during global placement phase itself by saving the best results in each of the iteration. This process is known as look-ahead legalization.

T. Ho et al. in [62] present a fast algorithm for legalization of standard cells. The proposed method aims at obtaining minimum displacement of the legalized cells from global placement. The main contributions of this proposed methodology are listed below.

- A fast row selection method by using k-mediod clustering
- An exact linear wirelength model to minimize wirelength

P. Spindler et al. in [145] present a new legalization method which is fast and causes minimum displacement to the cells of globally placed design. The methods begin in a similar manner as Tetris [61], in which cells are legalized one at a time. Starting from left to right, each cell is considered for placement on all the rows and is placed in the row where the displacement is minimum. Dynamic programming is performed for the placement of cells already assigned to rows. The difference between tetris and [145] is that, in tetris, once a cell has been assigned a row, its location is fixed, whereas, in [145], the location of a cell may change (due to dynamic programming) even after assignment of row.

N. Vishwanathan et al. in [156] present density based legalization procedure. This is applied when the macro cells in the globally placed design have fixed their locations. This method begins with creating an auxiliary bin structure over the die area. The height of each bin is equal to the row height and its width is equal to four to five standard cells. This is followed by computation of the utilization of bins and their capacities, and creating a *move map* for each bin. If the *move map* has value equal to 1, movement of cells into this bin structure is allowed, if *move map* value is zero the movement is not allowed. The weight of a move is decided by three factors which are described below.

- The reduction in HPWL with the move
- Utilization function of the source and sink bins
- Weighted difference between the *move map* values of the source and target bins

S. Hur et al. in [69] present their work based upon relaxation based local search, in which, constraints for the placement problem are prepared that can be solved optimally and efficiently. However, the solution consists of overlaps among the cells. Thus, legalization is an important procedure to be followed in this kind of approach. In legalization procedure, we begin with an initial solution P (unrelaxed) and then sequentially move each cell to its relaxed position. If this move results in constraints violation, then a constraint resolution procedure is invoked. After each cell is moved, a new legal placement is obtained. Each new placement is recorded and solution with the least cost is saved in each move. When no further reduction in the cost is found, the last saved solution with the least cost is declared as the final solution.

H. Ren et al. in [132] present a diffusion based legalization method. This method causes minimum perturbation to the existing global placement. This method is a discrete approximation to the closed form solution of diffusion equations. It performs smooth spreading, keeping the neighborhood characteristics preserved.

A. Agnihotri et al. in [3] present legalization method based on dynamic programming. At first cells are sorted by the y-coordinate. A region (row of the die) is selected for the placement and legalization is performed starting from the topmost row. A set of cells is considered for placement in each region. The cells in a given region is selected using dynamic programming. Finally, cells are sorted by their x-coordinates.

Dynamic programming approach by A. B. Kahng in [76] works in the following steps.

- Balancing of number of cells in the given rows of the layout
- Removal of overlaps using dynamic programming method

The objectives in this formulation are minimum increase in wirelength, minimum total displacement and minimum total perturbation.

U. Brenner et al. in [12] perform legalization based on linear programming. In this method, legalization problem is divided into appropriate sub problems that can be analyzed optimally in polynomial time. The solution of the sub problems is combined using an algorithm that legalizes a placement minimizing total movement of the cells.

M. Cho et al. in [31] introduce history based legalization procedure using min-cost network flow optimization. The network flow method is $gate-centric$ and iterative in nature. The gate movements are realized into a network flow

problem. When network flow iterations are performed, history of *likely to fail* moves are avoided and a legally acceptable solution is obtained.

Y. Lee et al. in [92] present a top-down approach for the legalization of global placement called *HiBinLegalizer*. In the first step, the die area is divided into several bins. Then, starting with the most populated unlegalized bin, a procedure is used to merge bins into cross-shaped region or square shaped region, until the cell density in that region is less than a threshold value. This is followed by a legalization algorithm which minimizes the weighted sum of movement and preserves the cell ordering.

Placement migration is an important step, wherein, along with the legalization, other design issues such as timing, routing congestion, signal integrity, heat dissipation etc. are addressed. T. Luo et al. in [106] present a placement migration procedure based on computational geometry. It works in two steps; bin based spreading at coarse level and Delaunay triangulation based spreading at fine grain level. It has advantage over traditional legalization methods and neighborhood characteristics are preserved.

3.8 Detailed Placement Methods

Typically, detailed placement step is carried out once the legalization is complete. The cell locations are pairwise swapped to reduce the wirelength further. Detail placement may achieve 1 to 5% reduction in wirelength. There are many techniques to achieve this and some of the well-known detailed placement methods are described below.

A. Khatkhate et al. in [83] present a window based method for detailed placement of standard cells. In this method, an auxiliary window is created over die area. The cells that are located in this window are moved and swapped among each other to search for a possibility of wirelength reduction. All possible cell locations are considered in this step. In the next step, window size is reduced by half and the same procedure is repeated until there is no further reduction in wirelength.

M. Sarrafzadeh et al. in [140] present a simulated annealing based algorithm for performing detailed placement. They perform low temperature simulated annealing of the cells after legalization.

S. Cauley et al. in [20] proposed an algorithm based on mixed integer programming (MIP) and, branch and bound method. First, branch and bound method is used to eliminate several integer decision variables. In the next step,

MIP is used to find optimal solution from the obtained reduced order problem. This method is suitable for parallelization for faster runtimes.

S. Li et al. in [95] present MIP based efficient solver for detailed placement problem. Two methods are proposed to address this issue. In the first method, a model with fewer integer variables is employed and another method is derived from Dantzig-Wolfe decomposition principle which imposes tighter bounds while finding the solution. Both methods are suitable for application on large problem instances.

A. B. Kahng et al. in [74] describe detailed placement problem of single row and two methodologies to solve this problem. In the first method, a dynamic programming method having time complexity of $O(m^2)$, where m is the number of nets incident on the cells of the given row is proposed. In the second method, a technique with time complexity of $O(mlogm)$, which is based on the convexity of wirelength objective is discussed to address the issue.

T. Chen et al. in [28] propose window based detailed placement algorithm that works on cell matching. In this method, a group of cells is identified from a window placed on the die area and apply bipartite matching algorithm to reduce wirelength among the cells present in the given window. It is accomplished by the matching algorithm by selecting group of cells from the window which are independent i.e. do not have a common net between them.

M. Pan et al. in [120] present a cell swapping technique to address the issue of detailed placement. They propose four important steps; Global swap, Vertical Swap, Local Re-Ordering and Single segment clustering. For a given cell, the algorithm tries to find a suitable pair for swapping such that the given cell will be in the best position, in terms of wirelength, after it is considered for swapping from various cells in the vicinity of the given cell.

S. Hur et al. in [69] describe a method called *optimal interleaving* for detailed placement for cells present in a given row of the layout. This method works in the following manner.

- Select a sequence of cells A from a row
- Select another sequence $B = \bar{A}$ from the same window
- Interleave the cells from A and B, keeping the ordering of cells in A and B fixed

3.9 Routability driven approaches

With placement optimization for traditional **matrices** of placement, it is possible that the placed design is difficult to route. Hence, routability driven placement is used to mitigate this problem. There are various congestion estimation schemes that are employed during placement, such as *pin density*, uniform wire density, smoothened wire density etc. There are various optimization schemes for routability driven placement which are applied at global placement, intermediate placement, detailed placement or as a part of post processing step. It is to be noted that due to its importance in the present VLSI design ere, ISPD conducted a contest [157] in 2011 for routability driven placement.

X. He et al. in [59] present a routability driven placer called *Ripple*. They propose two techniques called net base movement and cell inflation, that are used in global placement stage. In net based movement algorithm, all the cells comprising of the net are moved as a whole, whereas, cell inflation is carried out alternatively in horizontal and vertical directions. The global placement is followed by rough legalization. The detail placement procedure used in this placer consists of two strategies, traditional HPWL based placement and congestion driven optimization.

Z. Jiang et al. in [72] introduce a new method for the optimization of routability during global placement. This method is called net overlapping removal which moves the nets away from the congested areas in the layout. Net bounding box model is generalized to handle routing constraints and to speed up routability optimization. In this paper, a Gaussian smoothing method is proposed to handle macro porosity problem.

Y. Chunag et al. in [34] present a force directed routability driven placement method. They use design hierarchy information to formulate a fence force in the analytical placement framework along with the use of a spreading force to determine the location of standard cells and macros. With the use of these forces, both HPWL and routability optimization is achieved.

P. Spindler et al. in [144] present a novel routing demand estimation method called $RUDY$ which is used in a force directed routability driven placer. The estimation of routability is based on rectangular uniform wire density per net. Unlike other estimation methods, $RUDY$ does not depend on bin structure or routing model for the accurate estimation of routing demand.

M. Hsu et al. in [65] propose a routability driven placer which achieves following objectives during placement.

- Optimal density of pins and their routing directions
- Efficient use of sigmoid function to optimize routing overflow
- A virtual macro porosity expansion technique for consideration of the constrained routing resources consumed by large macros

K. Tsota et al. in [151] introduce efficient technique for the estimation of routed wirelength during global placement. Wire density of the net is used to achieve this objective. To mitigate congestion, this method identifies congested regions of the layout and then uses routed wirelength model into the objective function.

M. Kim et al. in [85] demonstrate a force directed placement tool which maintains upper and lower bounds of wirelength for optimization. For density optimization, auxiliary bins are created over placement region. The bins that are overfilled, rough legalization is performed by clustering cells of the bins using breadth first search.

U. Brenner et al. in [13] present a routability driven placement method based on partitioning approach. They perform single run of the placement, however, re-partitioning of the netlist is performed to recalculate the cuts after inflating cells of the design.

J. A. Roy et al. in [135] present a simultaneous placement and routing tool. Standard cells are inflated and spread iteratively during routing in the congested regions. They could achieve reduction in die area for commercial designs using this method.

P. N. Parakh et al. in [123] demonstrate an extension to quadratic placement for routability. To compute the routing cost, the proposed algorithm uses an A^* router and line probe heuristics. The interaction between routing estimation and quadratic placement using growth matrix helps in global optimization of the wire congestion. Based on the relaxation of π-constraints, further reduction in optimization of wire congestion is observed.

M. Pan et al. in [119] propose an integrated place and route approach to solve routability issue. In certain methods, the congestion estimation at placement stage may not be accurate and is different with respect to the actual congestion measured during routing. To solve this problem, an integrated approach is employed and is mentioned below.

- Local refinement technique based on steiner-tree wirelength calculation

- Routability driven refinement method to optimize routability during global placement
- Routability driven global cell swapping method
- Routability driven local swapping method for improving routability

D. Jariwala et al. in [71] present various models of routing congestion in their experimental setup. Their experiment was to measure the reliability of existing models. Motivated by unreliable estimates obtained from the experiments, a single combinatorial framework is developed in which "exact" routing structures are captured and optimized.

J. A. Roy et al. in [136] present a placement tool which is based upon optimization of steiner wirelength instead of HPWL. The motivation behind this approach is the close resemblance of StWL (steiner wirelength) with rWL (routed wirelength). A routability driven algorithm is developed which performs white space allocation to the congested regions and a novel cell shifting technique is employed to reduce congestion.

Y. Zhang et al. in [174] propose a routability driven placement technique which can be integrated with any other placement tool easily. It performs post processing of the legal placement solution by shifting cells without disturbing original placement. This method employs a congestion driven detailed placement technique along with congestion driven module shifting technique. The cell shifting technique is performed in such a way that allocation of routing resources is optimized. For this, a longest path computation formulation is proposed, instead of using computationally expensive LP solver. By using congestion coefficient, it is aimed to achieve congestion driven placement.

3.10 Timing Driven Placement

In [164] path based timing optimization scheme is presented. Timing of K critical paths are considered during a multi-objective simulated annealing (SA) algorithm. The SA cost function $COST$ comprises of two terms. The first term is the wire length WL and the second term is timing penalty function represented by P_T. The SA cost function $COST$ is defined as $COST = WL + \lambda P_T$.

In [70] authors impose physical constraint on the placement problem. First an initial placement is defined and then cells are moved to achieve optimization. During the movement, cells are forced to move in direction imposed by

constraints, which in turn increases the monotonic behavior of the placement of timing critical paths.

In [171], path based optimization is considered which is guided by hierarchy of the design. In the proposed scheme, a novel slack assignment function $f(x) = 1 - e^{-cx}$ is presented, where c is a constant determined by criticality of the net and x is the slack assigned to the net. Delay budgeting in a top-down placement framework is employed in this algorithm. Methodology proposed in [8] employs a smooth timing analysis approach which is implemented along with a nonlinear optimization standard cell placer. Timing cost functions in this proposed method are constructed as a smooth function of cell placement and are called in the placement engine. In [75], a new net weighing technique is proposed which is also based on the net weighing scheme of [9], and incorporate net weighing in the placement algorithm. In [161] linear programming (LP) based methodology is proposed for improving timing driven placement. All the topological paths are captured in a linear sized LP in such a way that net weights and net budgets are not necessary. The flow is applied to an existing cell placement and is improved iteratively. These proposed methods have been validated with FPGA based designs.

H. Chang et al. in [24] propose computation of criticality metrics (predictive models) for VLSI design. It is shown that the timing of the design can be improved by making use of these criticality matrices. For achieving better results in timing, these matrices need to be applied to the design before making a layout and are used for driving the layout system. There metrics are computed as ratios of net physical characteristics to the net delay bounds. There metrics are applied to any placement tool that applies weights to the nets while performing its optimization algorithm.

A.E. Dunlop et al. in [42] is one of the commonly use methodology of net weighing as we know today. In this method, a procedure is described to apply weights on the nets during placement optimization. These weights are computed by doing static timing analysis of the design. The weights are computed for critical nets on the clock and for data-paths.

T. Kong in [89] developed a new net weighing scheme based on the concept of path-counting. This algorithm is theoretically accurate all-path counting algorithm. This algorithm has been verified using timing driven FPGA placement tool.

B. Halpin et al. in [57] depicts that the net length reduction of nets in a critical path may affect timing of that path. It is observed that similar net length reduction in the nets of a path may not impact delay of the path in the

same way. Thus, sensitivity of a net corresponding to a path delay is calculated for optimal placement. Accurate net models are employed to extract sensitivity information and apply it to the placement tool for further analysis.

It is quite popular to assign net weight based on the slack of a net. There is another figure of merit (FOM), which is defined as the total slack difference compared to a certain slack threshold for all timing end points. H. Ren et al. in [133] perform comprehensive analysis of slack and figure of merit sensitivities to the net weight, and propose a new net weighing scheme, which is based on the slack and FOM sensitivities.

Z. Xiu et al. in [167] present a grid-warping placement strategy. Instead of shifting gates of the design, the 2D chip surface upon which the gates are coarsely placed, is elastically deformed to improve the timing. In the proposed approach, a novel grid-warping formulation is presented that incorporates slack-sensitivity based net weighing.

M. Burstein et al. in [15] is introduced dynamic net weighting approach in timing driven placement. A new approach is proposed to layout design, in which after performing static timing analysis, net weights are dynamically varied during the placement procedure. Since routing is performed at each level of partitioning, lengths of interconnects among the partitions are estimated to be used to update timing information associated with the design.

B. M. Riess et al. in [134] develop a new method for accurate net delay estimation. In this scheme, delay between source pin and sink pin of a net is estimated. The net weights obtained in this way are used to dynamically update nets for each successive placement steps.

B. Halpin et al. in [55] introduce a new timing driven placement algorithm, in which physical net constraints are explicitly met. This method is applied during recursive bisection based placement method. At each level of the recursive placement method, linear programming is used to ensure that all the physical net constraints are being satisfied on timing critical nets.

K. Rajagopal et al. in [130] describe timing drive force directed placement algorithm. In this proposed method, physical net constraints are applied during quadratic optimization. For achieving optimal placement, a new net model is proposed that changes contribution of the nets during quadratic programming.

R. Tsay et al. in [149] propose a placement approach based on analytic net weighing controls for nonlinear performance constraints. The proposed heuristic is justified by showing that the net weighing can be derived from Kuhn-Tucker condition of placement optimization.

B. Halpin et al. in [56] focus on a flow based on net length constraints

(NLC). They assign net length constraints during detailed placement. In the first detailed placement procedure, NLC operates with grid-based placement and assigns cell instances to grids. In the second detailed placement procedure, simulated annealing is performed for optimization of NLC based placement.

It is observed that during engineering change orders (ECOs), locations of the cells are changed and wires are resized. This causes increase in wirelength and requires timing optimization to be re-employed. To avoid this, H. Ren et al. in [131] propose a novel "do-no-harm" algorithm which is performed at detailed placement level. In this algorithm, pin based timing and electrical constraints (such as maximum output load constraints and maximum input slew constraints) are employed in their detailed placement scheme to reduce wirelength and to prevent degradation of the timing.

Besides net weighing approaches, there have been methods that consider timing paths during performance driven standard cell placement. V. Natesan et al. in [111] present performance driven quadratic placement algorithm. First, the performance bounds for a given circuit design using a zero-slack algorithm is estimated. Later, these performance bounds are used to derive a timing driven standard cell placement of the design. If the bounds are satisfied, it guarantees correct performance of the placed design.

Another example of path based optimization is proposed by W. Swartz et al. in [164]. In this proposed method, simulated annealing based algorithm is employed to compute the cost associated with the worst negative slack of critical paths of the design. Later, this cost is used in a simulated annealing framework for optimization.

T. Hamada et al. in [58] present a path based timing driven placement approach. At first timing constraints are transformed into a Lagrange problem. After that, primal-dual approach is used to optimize locations of the cells. The primal problem is solved by linear resistive network method, whereas, the dual is used to update the Lagrange multiplier.

A. Chowdhary et al. in [32] propose a new method for accurately measuring timing information of a design known as differential timing analysis. The differential timing analysis accurately captures timing information from perturbations of the cells in the placement procedures. Thus, a linear programming based incremental placement algorithm is proposed based on the feedback obtained from differential timing analysis.

T. Luo et al. in [105] present net-based as well path-based placement method. In this proposed method, a linear programming model is employed for optimal placement. This method focuses on the optimal placement of both

the critical path cells and the cells that are adjacent to these critical path cells. Timing aware spreading procedure is also employed during placement that preserves timing during legalization of high performance designs.

3.11 Power Techniques in Placement

In modern times, voltage islands are employed during placement to reduce dynamic power of the chip. This is achieved by switching certain regions (island) in on and putting other regions (islands) in off state. Thus, assigning voltage to regions or creating islands which have different operating voltages is different from traditional placement strategy. J. Hu et al. in [68] present an algorithm for voltage island allocation to a given placement netlist. Simultaneous partitioning of voltage islands is performed in the proposed algorithm to achieve the desired goal.

B. Liu et al. in [97] propose a standard cell placement method, which supports for dual supply voltages and aims to reduce the total power. This method performs coarse placement initially considering timing and power. This step is followed by generation of voltage islands, which involves a few iterations between voltage assignment and placement refinement. This method employs techniques such as net weighing based on voltage and power.

C. Yeh in [172] introduce block layout style to support the voltage scaling using traditional standard cell libraries. In their method, a layout block is generated using a placement method based on simulated annealing algorithm. A new cell layout style is also proposed, which consists of multiple supply rails. This type of cell layout can be easily used in a traditional standard cell placement flow.

ASIC design offers best power efficiency compared to FPGA due to the use of multiple voltage islands and multiple threshold voltages. In [129], Puri R. et al., explore trade-off between multiple supply voltages and multiple threshold voltages. Several schemes for the use of level shifter between different power domains are proposed. Various techniques are also showcased for perturbing physical layout to feed correct voltages in different power domains. Discussion on the scheduling the clock skew for power minimization is also presented.

Y. Cheon et.al, in [30] present techniques to reduce power by doing efficient placement. An activity base register clustering scheme is proposed, in which registers are placed in the same leaf cluster of the clock trees resulting in area minimization. Net weighing is determined depending on the switching activ-

ity of the net. Nets with more switching activity or more critical timing are assigned higher weights. This method is applied on the designs with multiple clocks and gated clocks.

B. Obermeier et al. in [116] present placement method to minimize the total power and achieves a smooth temperature profile across the chip. The proposed approach is based on quadratic placement formulation in which weighted wirelength is minimized. In the proposed approach, higher weights are assigned to nets with more power dissipation. Cells are spread all over the die such that uniform temperature profile is obtained.

S. M. Sait et al. in [138] implement tabu search for low power drive VLSI placement. A multiobjective optimization problem targeted towards the optimization of power dissipation, timing performance and wirelength reduction is formulated. Since nature of the objective is not precise, fuzzy logic is used in the design of function. This technique is tested on ISCAS benchmarks and shows promising results.

H. Vaishnav et al. in [152] present a performance driven placement algorithm for minimizing power consumption. The problem is considered as constrained programming problem for which solution is obtained in two phases; global optimization and slot assignment. Nets with the weights as switching activity and path delays are employed to formulate constrains.

Y. Chuang et al. in [33] describe a method for power reduction for designs having pulsed latches. They consider interplay between placement and clock-network synthesis to reduce power in the clock network and to optimize timing simultaneously. The proposed work is based on the progressive network forces to guide placement algorithm during iterative improvements. The clock-network synthesizer optimizes power and timing by exploiting updated latch locations.

Traditionally clock network synthesis for reduced dynamic power consumption is independent of standard cell placement. D. Lee et al. in [91] describe an algorithm which integrates clock network synthesis within global placement framework by optimizing register locations. Their contributions are obstacle aware virtual clock-tree synthesis, clock-net contraction force with virtual node insertion for handling multiple clock domains and gated clocks.

D. Papa et al. in [122] present physical synthesis optimization for latch placement called *RUMBLE* that uses a linear timing model to optimize timing by replacing multiple gates simultaneously. Incremental runs in conjunction with static timing analyzer are employed to improve timing of the design.

Chapter 4

Partitioning Based Placement

4.1 Introduction

In this chapter, we present a new method, *Kapees*, for the cell placement. *Kapees* employs (a) partitioning in global placement phase, and (b) low temperature annealing in detailed placement phase, followed by greedy cell swapping. The algorithmic flow of this method is shown in Figure 4.1. The performance of *Kapees* in terms of HPWL is compared with commercially available tool Cadence Encounter's Amoeba and publically available placement tool Capo [17]. The rest of the chapter is organized as follows. In Section 4.2, the flow of our proposed algorithm and global placement strategy is presented. Section 4.3 presents low temperature annealing methodology. Finally, experimental results and comparison with placement tools, and summary of the proposed work are given in Section 4.4 and Section 4.5, respectively.

4.2 Global Placement Phase

The proposed partitioning based placement tool works in two phases, *Global Placement Phase* and *Detailed Placement Phase*. The global placement phase involves partitioning of hypergraph using graph partitioner hMetis [79]. It recursively bi-partitions hypergraph and performs the arrangement. An arrangement is a step in which coordinates are assigned to bins in all possible ways. Partitioning is performed until an optimum level is reached. We have chosen to bi-partition hypergraph recursively because it achieves better partitioning (less number of cuts) and wirelength as compared to k-way partitioning.

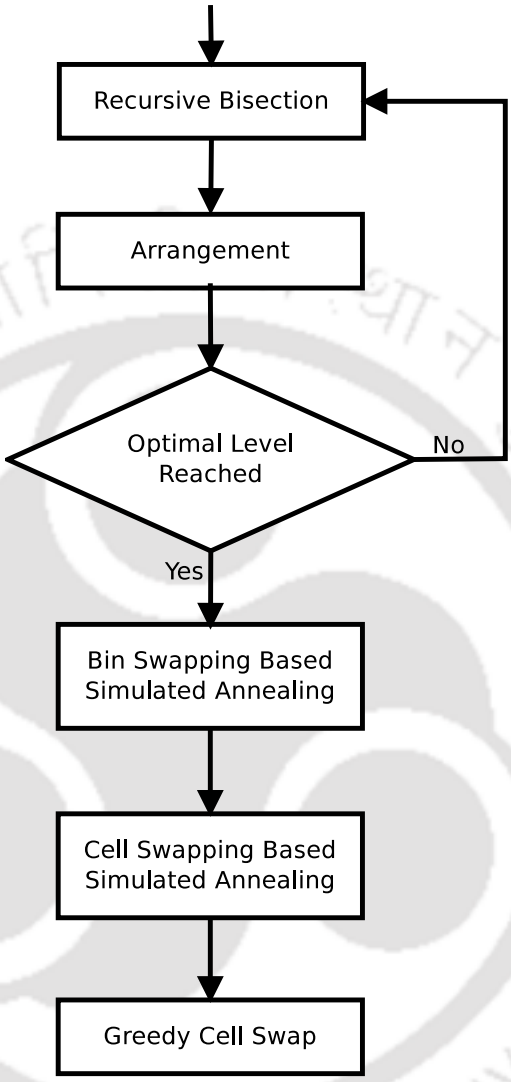

Figure 4.1: Flow of Kapees

In global placement phase, the given circuit is recursively bi-partitioned. First, circuit is partitioned horizontally followed by vertically partitioning the two sub-circuits that are obtained in the previous step. In the second step, these four partitioned circuits are further partitioned into two halves. This process continues until an optimum level is reached, which is defined by $\log(number\ of\ rows)$. At the end of partitioning phase, we are left with $4^{\log(number\ of\ rows)}$ sub-

circuits, which can be referred as bins. Each bin typically consists of five to ten cells.

During global phase, all the cells in a bin have the same coordinates. At every level, an arrangement of the bins in all possible locations is performed. For the first level, entire circuit forms a bin and is placed at the center. In the second level, total bins are four which can be arranged in 24 possible ways in four locations. In the third level, each bin out of total $4 \times 4 = 16$ bins can be arranged in 24 ways within the region of its parent bin.

The existing placement tools such as [160] and [142] typically carry out terminal propagation. In the terminal propagation, when the cells of circuit is partitioned into two bins, dummy cells are introduced into each bins. These dummy cells are included in external (cut) nets. When the two bins are possible candidates for further partitioning, external nets are ignored to get cut because of their connectivity to the bins externally. This results in obtaining a better wirelength. However, in the experiments with terminal propagation, noticeable improvement in the wirelength resulting from terminal propagation scheme is not found. Therefore, instead of performing terminal propagation, swapping and moving cells within the bins are preferred. This is equivalent to improve upon the cuts obtained in the global placement phase. Cell swapping is discussed in detail in the section 4.3.2.

4.3 Detailed Placement Phase

The detailed placement phase is required for further improving wirelength after the global placement phase. This phase deals with the placement of leaf level sub-circuits obtained from the global placement phase. In the detailed placement phase, simulated annealing algorithm are employed with multiple objectives. Bin swapping based simulated annealing and cell swapping based simulated annealing schemes are performed, followed by greedy cell swapping. These schemes are described in the following sub-sections.

4.3.1 Bin Swapping based Simulated Annealing

In bin swapping phase, a simulated annealing algorithm (see Algorithm 3) is employed with the objective to reduce wirelength and to balance row width. For implementing multi-objective simulated annealing, the objective function, C is defined by $C = \alpha A + \beta B$, where, A and B are the two objectives functions

contributing in the formulation of C. It is difficult to determine the ratio α and β as these values may vary for different designs. We overcome this problem by selecting the moves in such a way that any move that causes an imbalance is not accepted. We swap coordinates of the bins to achieve HPWL reduction without causing any row imbalance. The minimization function, $F(x)$, is the x-direction HPWL estimate and is given by Equation (4.1) from [80], where n_H and e_H are the node and edge sets of the hypergraph H, respectively.

$$F(x) = \sum_{e_k \in e_H} \max_{\forall i,j \in n_H} |x_i - x_j| \tag{4.1}$$

Algorithm 3 Multi Objective Simulated Annealing

```
while init_temp ≥ final_temp do
    P = e^{δWL/init_temp}
    Perturb                          ▷ Will cause swapping of bin coordinates
    if δ WL ≤ 0 and row_imbalance ≤ 0 then
        Accept the perturbation
    else
        if random_number ≤ P and row_imbalance ≤ 0 then
            Accept the perturbation
        else
            Revert the perturbation
        end if
        if sufficient_number_of_perturbations then
            Decrease init_temp
        end if
    end if
end while
```

4.3.2 Cell Swapping based Simulated Annealing

Due to the marginal impact of terminal propagation in the wirelength optimization, it becomes important to improve cuts obtained during global placement phase. This is achieved by performing cell swapping between the bins. In this step, all the cells within a bin are placed on the top of each other, therefore, these overlapping cells have the same coordinates. After completion of cell swapping between the bins, all the cells are spread out to remove overlap.

In the spreading step, cells of the bins are placed adjacent to each other in such a way that there is no overlap between them. It is important that the wirelength obtained after spreading out all the cells must correlate with the wirelength obtained during cell swapping phase. Cell swapping and cell movement between the bins is studied by [170] and is implemented in Dragon. They attempt to achieve a bin width balance by swapping of cells, which establish a correlation between wirelength of the spread out cells and wirelength during cell swapping. In our proposed methodology, two different approaches for doing annealing based on cell swapping have been attempted. These methods are explained in Algorithm 4 and Algorithm 5.

Algorithm 4 Perturbation Method 1

Randomly Select Two Bins in sufficiently close vicinity
Select one Cell from Each Bin
Calculate Incremental Cost = Cost1
Swap the Cell coordinates
Calculate Incremental Cost = Cost2
Return $\delta WL = Cost1 - Cost2$

Algorithm 5 Perturbation Method 2

Randomly Select Two Bins in sufficiently close vicinity
Select one Cell from Each Bin
Calculate Incremental Cost = Cost1
Swap the Cell coordinates and arrange all the Bins to the right of the selected Bins separated by a distance equal to their widths
Calculate Incremental Cost = Cost2
Return $\delta WL = Cost1 - Cost2$

In the first approach (refer Algorithm 4), the bin location are not moved i.e., all the bins have fixed coordinates. Here, the bin coordinates are not changed and there exists even spacing between the bins. We randomly select bins, which are within a distance of $numberofrows/3$ bins. In our proposed methodology, this approach does not correlate well with the spread out wirelength.

In the second approach, (refer Algorithm 5) as described in Figure 5, bins are placed at a distance equal to their widths. The bins are shifted towards right by the difference in cell widths. Let cell A belongs to bin A and cell B belongs to bin B, then after swapping the cells, all the bins to the right of bin

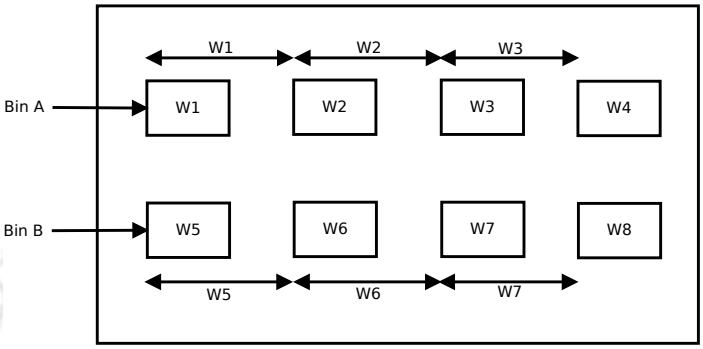

Figure 4.2: Method 2 for Perturbation

A are moved by $(Width(cell\ B) - width(cell\ A))$, where as all the bins to the right of bin B are displaced to right by $(Width(cell\ A) - width(cell\ B))$. If cells are spread out using the first method, wirelength becomes worse. This is because of the incorrect estimation of bin width separation. In our proposed approach, focus is to obtain balance for the row width instead of bin widths. In the second method, the wirelength during cell swapping stage correlates well with the spread out cost, which leads to find optimized placement solution.

4.3.3 Greedy Cell Swapping

In this stage, initially, all the cells for the bins are spreaded out. After that, two cells are randomly picked in sufficiently close vicinity and are checked whether swapping their coordinates with each other can reduce the wirelength. The swapping is performed in such a way that there should not be any overlap between the adjacent cells. All the cells to the right of the selected cells are spreaded out at the end of this step.

4.4 Experimental Results

Kapees, for addressing large scale standard cell placement issue, is implemented in *C* language. The experiments are performed on a machine having 1.5 GHz processor with 02 GB RAM, and Linux as an operating system. *Kapees* is

validated with IBM version 2.0 benchmarks as shown in Table 4.1. These benchmarks have cell counts ranging from 12028 to 68046. Most of the current state of the art placement tools have compared their methodologies using these benchmarks. For the experiments and HPWL comparison, the tools Cadence Encounter's Ameoba (version 9.1) and Capo (version 8.8) [17] are employed.

Table 4.1: Characteristics of IBM version 2 benchmarks

Circuits	Cell Count	Net Count	Rows	core utilization
ibm01_easy	12028	11753	132	85.12%
ibm01_hard	12028	11753	130	88.00%
ibm02_easy	19062	18688	153	90.42%
ibm02_hard	19062	18688	149	95.28%
ibm07_easy	44811	44681	233	89.95%
ibm07_hard	44811	44681	226	95.30%
ibm08_easy	50672	48230	243	90.03%
ibm08_hard	50672	48230	236	95.16%
ibm09_easy	51382	50678	246	90.24%
ibm09_hard	51382	50678	240	95.12%
ibm10_easy	66762	64971	321	90.22%
ibm10_hard	66762	64971	313	95.08%
ibm11_easy	68046	67422	281	90.11%
ibm11_hard	68046	67422	273	95.33%

It can be observed from Table 4.2, the runtime of *Kapees* is 10 times more as compared to Capo for the designs with cells less than 2000. For the design ibm01_easy, runtime of Capo is 21 seconds, whereas, runtime of *Kapees* is 428 seconds, which illustrates *Kapees* to be 22 times slower than Capo for this design. For the design ibm12_hard, runtime of Capo is 205 seconds, whereas, runtime of *Kapees* is 9084 seconds, which showcases *Kapees* to be is 44 times slower than Capo for this design. The runtime of *Kapees* increases with the increase in number of cells due to the very nature of simulated annealing method. As reported in Table 4.2, experiments show that the half perimeter wirelength obtained for the benchmark designs by *Kapees* are on an average 9% and 5% less than that obtained from Amoeba and Capo, respectively.

4.5 Summary

Partition driven placement is one of the earliest methods for performing place-

Table 4.2: HPWL comparison for IBM version 2 benchmarks

Circuits	Amoeba (A)	Capo (C)	Kapees (K)		
	$\times 10^6$	$\times 10^6$	$\times 10^6$	K/A	K/C
ibm01_easy	58.0278	55.8873	51.7759	0.89	0.92
ibm01_hard	57.6793	55.1354	51.456	0.89	0.93
ibm02_easy	165.906	158.743	146.253	0.88	0.92
ibm02_hard	164.541	156.048	145.224	0.88	0.93
ibm07_easy	373.388	368.7	354.121	0.94	0.96
ibm07_hard	359.586	355.686	353.964	0.98	0.99
ibm08_easy	406.834	387.502	352.928	0.86	0.91
ibm08_hard	395.116	379.512	363.081	0.91	0.95
ibm09_easy	341.532	317.126	311.554	0.91	0.98
ibm09_hard	338.188	321.029	321.057	0.94	1.00
ibm10_easy	605.746	636.689	601.824	0.99	0.94
ibm10_hard	642.065	629.543	624.721	0.97	0.99
ibm11_easy	521.413	481.637	478.191	0.91	0.99
ibm11_hard	514.758	476.332	472.878	0.91	0.99
			Average	0.91	0.95

ment of VLSI circuits. The popular partitioners used by various placement tools (Capo, Dragon, Fengshui, Theto) are hMetis and MLPart. We employ hMetis in our proposed placement tool, *Kapees*, for partitioning input netlist. The proposed partitioning driven approach is further improved by applying simulated annealing method for wirelength reduction. The performance of *Kapees* is 5%-10% better over other simulated annealing techniques as well as commnercially and publicly available placement tools. This is due to the better correlation of wirelength obtained during cell swapping stage with the spread out cost. We propose to improve upon runtime performance of *Kapees* in future.

Chapter 5

Nonlinear Analytical Optimization

Nonlinear programming is a method of finding optimal solution of an optimization problem having nonlinear constraints and objective function over a set of unknown variables. Placement of electronic circuit components (gates, macros etc.) over silicon optimally is an essential part of the VLSI design process. There are various known methods, as described in Chapter 3, being employed to address this issue. The existing methods that use quadratic programming have limitations and may not perform well, if the design has no pads or less number of pads. This is because of quadratic optimization requiring a pivot around periphery of the circuit to derive a solution [154]. We have made an attempt to overcome this problem, wherein, cells can be spread efficiently over the die area without pad information. In this chapter, we present an analytical approach based on nonlinear programming to solve placement problem of VLSI circuits. In particular, the contributions made in the proposed methodology are as follows:

1) A novel and high quality framework for VLSI placement. An efficient placement algorithm, which works efficiently for designs without pads.

2) A variant of Nesterov's Method (VNM) for minimization of nonlinear objective density penalty function.

3) An experimental comparison of VNM with conjugate gradient method.

4) A comparative study of the proposed model for calculating objective function parameter (defined in section 5.3). Improvements in wirelength are

observed due to this parameter value.

5) A new approach in which objective function parameter needs to be estimated only once in the entire flow.

6) A novel termination criterion from the iterations of conjugate gradient or Nesterov's method. Two parameters *Exit_Iteration* and *Accuracy* are defined to improve the solution quality.

7) A greedy approach for detailed placement of standard cells.

5.1 The Model for Nonlinear Analytical Placement Method

The model for nonlinear analytical placement method has been proposed in [112], and is implemented by [73], [28] and [26]. This approach regards placement as a constrained nonlinear optimization problem.

Consider a hypergraph $H = (V, E)$, where the sets $V = \{v_1, \ldots, v_n\}$ and $E = \{e_1, \ldots, e_m\}$ represent the cells and hyperedges, respectively, with each $e_i \subseteq V$. The objective of the placement problem is to find coordinates for all the cells in V such that they are placed on the given rows of a layout within the placement region without any overlap between any two cells minimizing total Half Perimeter Wire Length (HPWL).

Let x_i and y_i be the x and y coordinates of a movable cell v_i. HPWL is defined by Equation (5.1) [80].

$$H(x, y) = \sum_{e_k \in E} \left(\max_{\forall i,j \in e_k} |x_i - x_j| + \max_{\forall i,j \in e_k} |y_i - y_j| \right) \tag{5.1}$$

The placement area is divided into uniform grids and total HPWL is targeted to be minimized under the constraint that the total module area in every grid is equal. The problem is expressed by a set of equations mentioned below.

$$\min H(x, y) \quad \text{such that} \quad B_g \leq A_g \tag{5.2}$$

where A_g is the maximum area of movable cells in a grid-cell g, $H(x, y)$ is the wirelength and $B_g(x, y)$ is the potential which is the total area of movable cells in a grid-cell g.

It is difficult to minimize $H(x, y)$ directly as it is neither differentiable nor smooth. In order to solve this problem, several smooth wirelength approximations have been proposed in the literature, such as quadratic wirelength [46], L_p norm wirelength [39] and log-sum-exp wirelength [112]. Among these the log-sum-exp model achieves the best results as it is a closer approximation of HPWL as compared to the other models [22].

$$\begin{aligned} H(x,y) = \delta \sum_{e \in E} (\log \sum_{v_k \in e} exp(x_k/\delta) + \log \sum_{v_k \in e} exp(-x_k/\delta) \\ + \log \sum_{y_k \in e} exp(y_k/\delta) + \log \sum_{y_k \in e} exp(-y_k/\delta)) \end{aligned} \tag{5.3}$$

For density penalty minimization using nonlinear method, it is required that the density function should be differentiable or smooth. However, as the density function $B_g(x, y)$ is not smooth, it cannot be minimized directly. In [39], a smoothing function is proposed, which is based on inverse Laplace Transform. In [100], a smoothing function based on electrostatic force has been proposed. In the proposed methodology, we use bell shaped density function originally proposed in Aplace [73], which is differentiable at all points.

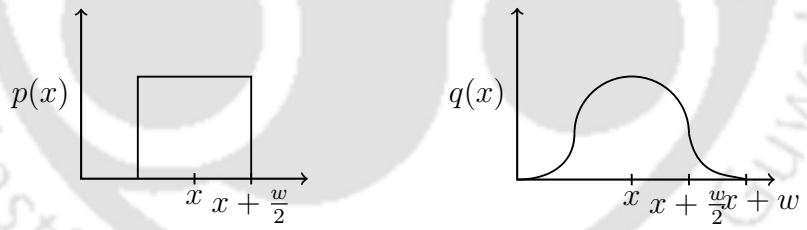

Figure 5.1: Potential function $p(x)$ is non-differentiable whereas $q(x)$ is differentiable.

$$B_g(x,y) = \sum_{v \in V} Q_x(g,v) Q_y(g,v) \tag{5.4}$$

where Q_x and Q_y are the overlap functions of grid-cell g and cell v along the x- and y-directions, respectively. To smoothen Q_x, the bell-shaped potential is

defined by Equation (5.5).

$$q_x(g,v) = \begin{cases} 1 - 2\frac{d_x^2}{w_g^2}, & 0 \leq d_x \leq \frac{w_g}{2} \\ \frac{2}{w_g^2}(d_x - w_g)^2, & w_g/2 \leq d_x \leq w_g \\ 0, & w_g \leq d_x \end{cases} \tag{5.5}$$

It should be noted that for mixed size benchmarks, Q_x is defined by Equation (5.6).

$$q_x(g,v) = \begin{cases} 1 - ad_x^2, & 0 \leq d_x \leq \frac{w_i}{2} + w_g \\ b(d_x - \frac{w_i}{2} - 2w_g)^2, & \frac{w_i}{2} + w_g \leq d_x \leq \frac{w_i}{2} + 2w_g \\ 0, & \frac{w_i}{2} + 2w_g \leq d_x \end{cases} \tag{5.6}$$

where

$$a = \frac{4}{(w_i + 2w_g)(w_i + 4w_g)}, b = \frac{2}{w_g(w_i + 4w_g)} \tag{5.7}$$

where d_x is the distance between the center coordinates of the cell v and the grid-cell g in x-direction and w_g is the grid-cell width. The original and smoothed overlap functions are shown in Figure 5.1. A list of various variables used in the proposed methodology is shown in Table 5.1. In the next section, minimization of $H(x, y)$ is discussed in detail.

Table 5.1: Notation and Variables

x_i, y_i	Centre coordinates for the cell v_i
w_i, h_i	Width and height of the cell v_i
w_g, w_h	Width and height of the grid-cell g
Q_g	Potential of grid-cell g_i
B_g	Density of grid-cell g_i
A_g	Maximum density of grid-cell g_i
δ	Wirelength parameter
μ	Objective function parameter
$H(x, y)$	Half Perimeter Wirelength (HPWL)
α	Step size

5.2 Implementation Details of Proposed Methodology *Kapees3*

In this section, implementation details of the proposed methodology, which uses analytical techniques and solves nonlinear equations, are presented. In the first step, first choice clustering is performed on the input netlist to reduce the problem size. Later, Nesterov's method based optimization is applied to all the hierarchical levels that are obtained from the clustering. The legalization is performed during nonlinear optimization step itself, which is referred to as look-ahead legalization. This is followed by cell order polishing and greedy cell swapping. The flow of the proposed algorithm is illustrated in Figure 5.2 and is resulted into an indigenous placement tool *Kapees3* .

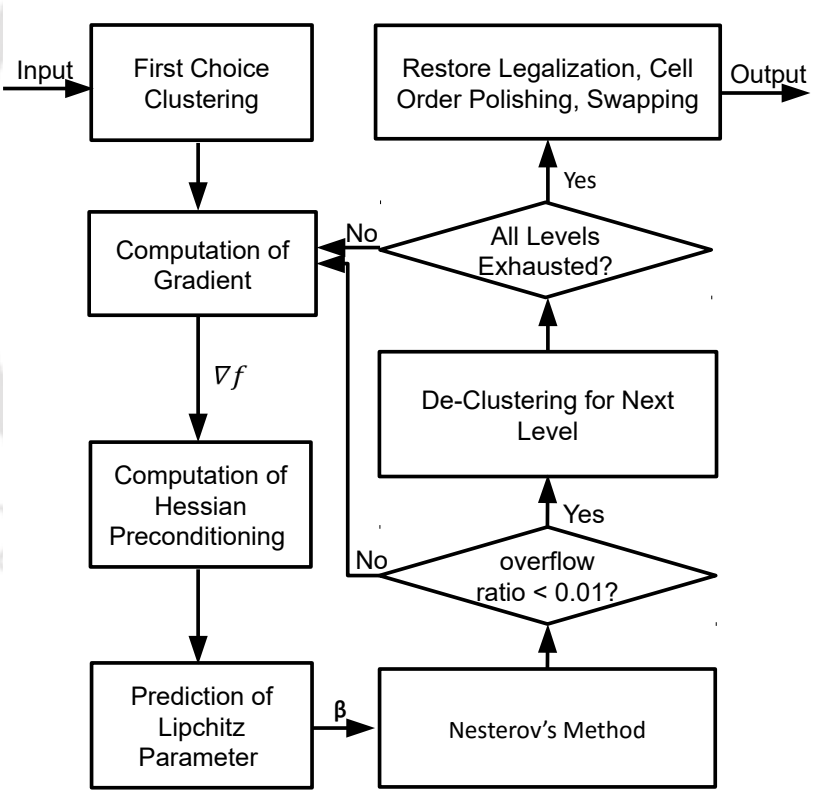

Figure 5.2: Flow diagram of *Kapees3*.

Unlike previously reported methods, [28] and [26], the proposed method

does not work in phases. The overall flow of this approach is encapsulated in Algorithm 6.

Algorithm 6 *Kapees3*: Multilevel Algorithm

Require: Hypergraph H_0: standard cell circuit
and n_{max} Number of cells at the coarsest level
Ensure: (x^*, y^*) optimal cell positions

First choice clustering till the number of cells is $> n_{max}$
Let the levels of FCC be LEV to 0 ▷ LEV is coarsest
if Pads connected to movable standard cells **then**
    Initialize cell positions by solving QP
else
    Initialize cell positions by randomization
end if
Initialze the value of μ_0 by solving Eq (11)
for $level = LEV$ to $level = 0$ **do**
    Estimate grid-cells, $NBx = NBy = \sqrt{n_{level}}$
    Solve Equation (7.4) and (7.5) for each grid-cell
    Initialze $\mu_{level} = \mu_{level-1}/2$, $m = 0$, $\mu_m = \mu_l$
    while *overflow_ratio* > 0.01 **do**
        Solve min $H(x, y) + \mu_m \sum (B_g - A_g)^2$
        $m = m + 1$
        $\mu_m = 2 * \mu_{m-1}$
        Estimate *overflow_ratio*
        if *overflow_ratio* < 0.1 and $level = 0$ **then**
            Legalize and save best result
        end if
    end while
end for

5.2.1 First Choice Clustering

Clustering is commonly employed in analytical placement tools to reduce the problem size. Clustering may not be required from input netlists of smaller size (≤ 6000) as it may not improve the runtime significantly. When it is performed carefully, it reduces execution time with slight degradation in the quality of a

solution of the placement. There are two types of clustering performed in analytical placement tools. In *first choice clustering*, the grouping of the cells is based on the netlist information. In the second type, which is called *best choice clustering*, information of the placed cells is considered during grouping of the cells.

Clustering can be performed, by doing k-way partitioning, using state-of-the-art partitioning tools such as hMetis [79] and MLPart [4], when the number of clusters required are less (roughly 100). If the number of clusters required are in the range of $100K+$, the k-way partitioning approach may not be suitable.

First choice clustering of the input netlist is performed as described in [22] in the initial step of the proposed placement method. The clustering proceeds until total number of cells in the cluster are less than a given value n_{max} (see Algorithm 6). First, all the cells are arranged in the decreasing order of their degrees. Every cell is then traversed sequentially and affinities are calculated for the cells with which it has a shared hyperedge. The affinity, r_{mn}, between cells m and n is defined as

$$r_{mn} = \sum_{\{m,n\} \subseteq e \in E} \frac{w(e)}{(|e| - 1)\ area(e)} \tag{5.8}$$

This is used to join a cell to its neighbor with which it has maximal affinity. Such a group of joined cells is called a cluster at the coarser level. **The hyperedges that were connecting the joined cells are lost and every existing hyperedge at a finer level becomes a hyperedge at the coarser level resulting in a smaller netlist at a coarser level.** We recursively apply first choice clustering until number of cells is less than n_{max} which is a user specified value. A large value of n_{max} increases the runtime of placement algorithm. In the experiments, we have performed, the value of n_{max} is fixed at 3000.

5.2.2 Quadratic Optimization

The quadratic optimization is performed for placement initialization in line 1 of the placement Algorithm 6 for a design that has pads connected with the standard cells. This approach has also been used in other tools, see for example, [85, 87, 154] and [117].

Consider a graph $G = (V, E)$ where V and E are the sets of vertices and edges, respectively. Further, let $w_{ij} > 0$ be the edge weight of an edge $e_{ij} \in E$.

The *quadratic objective* function, ϕ_G, is defined as

$$\phi_G(x,y) = \sum_{e_{ij} \in E} w_{ij}[(x_i - x_j)^2 + (y_i - y_j)^2]. \tag{5.9}$$

Using matrix representation for x and y components,

$$\phi_G(x,y) = \frac{1}{2} x^T M_x x + c_x^T x + constant. \tag{5.10}$$

Let m_{ij} be the entry in i^{th} row and j^{th} column of the matrix M_x. Further, let $\frac{1}{2}w_{ij}(x_i^2 + x_j^2 - 2x_i x_j)$ be the cost in the x direction between two movable cells x_i and x_j. The coefficients in matrix M_x are estimated using following rules.

(i) Add w_{ij} to m_{ii} and m_{jj}.

(ii) Subtract w_{ij} from m_{ij} and m_{ji}.

Let $\frac{1}{2}w_{if}(x_i^2 + x_f^2 - 2x_i x_f)$ be the cost in the x direction between a movable cell x_i and a fixed cell x_f.

(iii) Add w_{if} to m_{ii}.

(iv) Subtract $w_{if}x_f$ from the vector c_x at row i.

The objective function defined in Equation (5.9) can be minimized by solving Equation (5.11). As the matrix M_x is positive definite, the minimum of $\phi_G(x,y)$ can be obtained by solving the equation,

$$M_x x + c_x = 0. \tag{5.11}$$

5.2.3 Initialization using Random Placement

Random placement of the cells is performed when the pads do not have connection to the other movable standard cells (see line 6 of Algorithm 6). This is required for the computation of the initial values of μ (defined in the subsequent sections). A random placement helps in calculating non-zero values for the initial wirelength and the density function. Since a small initial value of μ is selected, the randomly placed circuit gets optimized in terms of the wirelength.

5.2.4 Determination of Potential and its Gradient

After completion of clustering and the initial placement as determined by quadratic optimization, the next step is to compute grids and their potentials, density and gradient of the density of grids. These will be required to analyze nonlinear equations for finding optimal placement. At line 10 in Algorithm 6, a rectangular grid is created with the dimension as the square root of the number of cells in the clustered hypergraph. Each rectangular grid formed is called a grid-cell. Let the number of grid-cells in the x and y direction be NBx and NBy, then $NBx = NBy = \sqrt{n_{level}}$, where n_{level} is the number of cells in the hypergraph at that level. This grid will be recomputed for the next level. Later, (at line 11 in Algorithm 6), the four vectors namely potential of grid, gradient of potential of grids, density of grids and gradient of density of grids are calculated using Equations (5.4) and (5.5).

5.2.5 Computation of Objective Function Parameter to Solve Nonlinear Equation

A sequence of unconstrained minimization problem specified by the constraint (5.12) is solved in line 6 of Algorithm 6,

$$\min(H(x,y) + \mu \sum_{\text{grid-cells}} (B_g(x,y) - A_g)^2) \tag{5.12}$$

The solution of this equation is of fundamental importance in the entire placement flow. This equation constitutes of two parts, one is related with wirelength and another with the density of bins. The minimization of wirelength opposes the spreading of cells. However, achieving optimal density for the grid-cells increases wirelength. Thus, a balance between the two objectives is required to be established to obtain a good placement. This is achieved by the factor μ and its value is computed at line 8 in the algorithm. The estimation of the value of μ is critical to placement quality and runtime of the placement tool. This value is computed for the coarsest level (denoted by LEV in the algorithm) as

$$\mu = \frac{2 \sum_{x_i,y_i} (|\frac{\partial H(x,y)}{\partial x} + \frac{\partial H(x,y)}{\partial y}|)}{\sum_{x_i,y_i} \sum_{\text{grid-cells}} |B_g - A_g| \times (|\frac{\partial B_g}{\partial x}| + |\frac{\partial B_g}{\partial y}|)} \tag{5.13}$$

For the subsequent levels, the value of μ is computed as $\mu_{level-1} = \mu_{level}/2$ and within each level, value of μ is incremented by a factor of 2.

Analytical placement which is based on the quadratic optimization have overlaps in the intermediate solution. Our approach is based on the nonlinear analytical method where cells are spread across the die which reduces overlap. As shown in Figure 5.3, cells are spreaded as the value of μ increases. The increase in the value of μ also shows gradual spread of cells in order to occupy adjacent auxiliary bins. The spreading of cells across the die increases as we increase the value of μ. To measure evenness of the spreading of cells, a parameter $overflow\_ratio$ term has been described in [28]. We use this parameter to measure spreading of cells in the proposed placement framework. The $overflow\_ratio$ is defined by Equation (5.14).

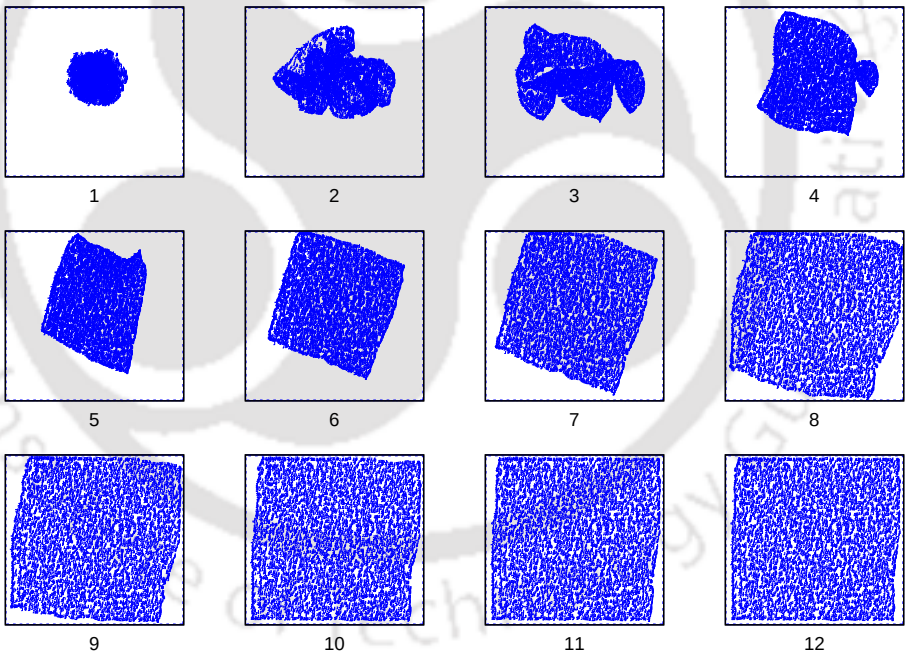

Figure 5.3: This figure shows the progression of placement solution, starting from initial placement (marked by number 1), till the final global placement (marked by number 12). The spreading of cells for design Peko01 is illustrated. The overlap among the cells gradually decreases as the value of μ is incremented by a factor of 2.

$$overflow\_ratio = \frac{\sum \max(B_g(x,y) - A_g, 0)}{\sum Total\ Area} \tag{5.14}$$

The value of δ in Equation (5.3) determines the approximate value of HPWL and is an important parameter. A smaller value of δ gives good approximation of HPWL. However, for a very small value, there may be an issue with arithmetic precision during computations. Thus, we choose a reasonably small value which is 1% of the width of die area.

5.2.6 Nesterov's Method to Solve Nonlinear Equation

Yuri Nesterov in [114] has proposed a method for optimizing convex minimization problems with a convergence rate of $(1/k^2)$, where k is the number of iterations. This requires evaluation of β_k which is equal to the inverse of Lipchitz constant [114]. In [101], [102], β_k is approximated as Equation 5.15.

$$\beta_k = \frac{\|\nabla f(x_k) - \nabla f(x_{k-1})\|}{\|x_k - x_{k-1}\|} \tag{5.15}$$

As this is an expensive computation and causes steplength over-estimation, back tracking has been used as an alternative in [101].

We introduce a new prediction parameter using an approximation of β_k, which is simpler to implement as compared to backtracking based β_k prediction in Equation 5.15. Our approach is novel and close approximation to the optimal step size calculation for density penalty based optimization. Similar approximation has been used in [28], however, the difference is that, in [28], conjugate directions have been used (see Algorithm 8, step size computation), whereas, we use euclidean of the pre-conditioned gradient. We compute β_k by Equation (5.16), where S is a constant. The choice of this parameter is critical for the quality of solution because a large value of the step size can increase HPWL, whereas, a much smaller value will affect runtime of the placement tool.

$$\beta_k = S * \frac{w_g}{\|g_k\|_2} \tag{5.16}$$

We use Nesterov's method described in Algorithm 7 to solve Equation (5.12). We first compute the gradient directions, which is then followed by pre-conditioning of the gradients. We preform pre-conditioning similar to [101]

by taking the ratio of the gradient and the degree of a node. The next step is to compute b_k and update the values of x_k and u_k as specified in [114].

Algorithm 7 Variant of Nesterov's Method

Require: $f(x)$: objective function and x_0: initial solution
Ensure: optimal x^*

$h_0 = 0$; $d_0 = 0$; $N_k = Degreeof Node$;
while $f(x_k) < f(x_{k-1})$ **do**
  Estimate Gradient Directions $h_k = \nabla f(x_k)$;
  Compute $g_k = \frac{h_k}{N_k}$;
  Compute $\beta_k = 2\frac{w_g}{\|g_k\|^2}$;
  Compute $b_k = 1 + \sqrt{1 + 4b_{k-1}^2}$;
  Compute $u_k = x_k - \beta_k g_k$;
  Update $x_k = u_k + \frac{(b_{k-1}-1)(u_k - u_{k-1})}{b_k}$;
end while

The criterion in *while* loop in Algorithm 7 is important as it affects placement solution and spreading of the cells. We define two parameters, *Accuracy* and *Exit_Iteration*, that are used in the evaluation of exit criterion from *while* loop. If *Exit_Iteration* is the number of iterations of the Nesterov's method, then we set this condition as Equation (5.17).

$$f_k(x) < Accuracy * f_{k\text{-}Exit\_Iteration}(x) \tag{5.17}$$

5.2.7 Conjugate Gradient Method to Solve Nonlinear Equations

In another experiment, Conjugate Gradient (CG) method is employed to solve Equation (5.12). This is performed to compare the proposed VNM based optimization with CG method. The CG method is explained briefly in Algorithm 8.

5.2.8 Look Ahead Legalization

After the cells spread across the die area, there may still exist an overlap among them. In addition to the requirement of non-overlapping cells, all of

Algorithm 8 Conjugate Gradient Method

Require: $f(x)$: objective function and x_0: initial solution
Ensure: optimal x^*

$h_0 = 0$; $d_0 = 0$;
while $f(x_k) < f(x_{k-1})$ **do**
  Estimate Gradient Directions $h_k = \nabla f(x_k)$;
  Estimate the Polak-Ribiere parameter B_k
  $B_k = \frac{h_k^T(h_k - h_{k-1})}{\|h_{k-1}\|^2}$;
  Estimate the Conjugate Directions $d_k = -h_k + B_k d_{k-1}$;
  Estimate the Step Size $\alpha_k = s/\|d_k\|_2$;
  Update the solution $x_k = x_{k-1} + \alpha_k d_k$;
end while

them must occupy the rows, that is, y-coordinate of all the cells must be a row coordinate. In order to achieve this goal, legalization is performed when the overflow ratio is less that 10%. Our legalization scheme is based on [61]. First, the x-coordinates are multiplied by a scaling factor (ranges from 0.85 to 0.95 for the mentioned benchmarks). It is to ensure, all the cells do not get pushed out of the placement region. Later, the cells are sorted based on their x-coordinates. Then starting from left to right, for each cell, the nearest location on any of the rows is found. The row location which gives the least increase in HPWL is selected. During legalization, some increase in HPWL is observed, which is expected because a significant number of cells may get moved from their original positions.

5.2.9 Cell Order Polishing

The cell order polishing has been performed in other placement engines as well, for example, [28] and [117]. After legalization the cell ordering may not be aligned with the minimum wirelength. In order to solve this issue, a group of five cells of a row is selected, and all permutations of cell positions are considered to select the best possible position of cells that reduces wirelength. This method reduces wirelength by 1% without affecting runtime significantly.

Algorithm 9 *Kapees3*: Greedy Cell Swapping Algorithm

```
1: TotalCells = Number of cells in sliding window
2: while TotalCells ≥ iteration do
3:     temp1 = Location of a randomly selected cell
4:     temp2 = Location of a randomly selected cell
5:     iteration = iteration + 1
6:     wl1 = incWlCostBeforeSwapping(temp1, temp2)
7:     swapCoordinates(temp1, temp2)
8:     wl2 = incWlCostAfterSwapping(temp1, temp2)
9:     diff = wl2 - wl1
10:    if diff ≤ 0 then
11:        cost = cost + diff
12:    else
13:        swapCoordinates(temp1, temp2)
14:    end if
15: end while
```

5.2.10 Greedy Cell Swapping

We perform greedy cell swapping when the cells are overlap free and ordered by wirelength using cell order polishing. The steps are shown in Algorithm 9. In this method, a window is selected which slides over the die area. Any two cells that are within the window and have the same widths are randomly selected and are considered for a swap of coordinates. If the swap results in a lower incremental wirelength the swap is accepted. We have observed an improvement of 1% in wirelength with negligible change in overall runtime after executing this step.

5.3 Experiments and Results

We have implemented *Kapees3* in C-Language. We select `icc` and `icpc` compilers to achieve better runtimes. Experiments are performed on a 2.6 GHz, 64-bit machine having Ubuntu (a variant of GNU Linux) as operating system.

5.3.0.1 Benchmarks

We have experimented with the PEKO Suite 1 and PEKO Suite 2 benchmarks to test the proposed methodology as these are commonly used by many placement methods, for example, [3,26,73,160]. The benchmark statistics are shown in Table 5.2 and 5.3. The circuits that are classified as easy have 85% density and 15% white space, whereas, those classified as hard have 90% density and 10% white space. The number of the nets in all the designs are approximately equal to the number of cells.

Table 5.2: Characteristics of Suite1 of PEKO Benchmarks

Circuits	**Cell Count**	**Net Count**	**Rows**	**% Utilization**
Peko01	12506	13865	113	85
Peko02	19342	19325	140	85
Peko03	22853	27118	152	85
Peko04	27220	31683	166	85
Peko05	28146	27777	169	85
Peko06	32332	34660	181	85
Peko07	45639	47830	215	85
Peko08	51023	50227	227	85
Peko09	53110	60617	231	85
Peko10	68685	74452	263	85
Peko11	70152	81048	266	85
Peko12	70439	76603	266	85
Peko13	83709	99176	290	85
Peko14	147088	152255	385	85
Peko15	161187	186225	402	85
Peko16	182980	189544	429	85
Peko17	184752	188838	431	85
Peko18	210341	201648	460	85

PEKO benchmarks are synthetic benchmarks designed to study optimal evaluation of placement algorithms [23]. The designs are such that all the cells of a design have the same width, unlike the real circuits where width of standard cells may not be identical. All the designs in PEKO benchmarks have the same core utilization. For PEKO Suite 1 benchmarks, the cell count varies in the range from 12506 to 210341, whereas, PEKO Suite 2 benchmarks have bigger designs and the cell count varies in the range from 126602 to 2109724. For experiments with mixed sized benchmarks, we choose MMS benchmarks

Table 5.3: Characteristics of Suite2 of PEKO Benchmarks

Circuits	**Cell Count**	**Net Count**	**Rows**	**% Utilization**
Peko01	126602	138650	355	85
Peko02	195336	193250	441	85
Peko03	230614	271180	479	85
Peko04	274472	316830	523	85
Peko05	283772	277770	532	85
Peko06	325798	346600	570	85
Peko07	459334	478300	677	85
Peko08	513340	502270	715	85
Peko09	534274	606170	730	85
Peko10	690460	744520	830	85
Peko11	705168	810480	839	85
Peko12	708046	766030	840	85
Peko13	841074	991760	916	85
Peko14	1476160	1522550	1214	85
Peko15	1617398	1862250	1271	85
Peko16	1835690	1895440	1354	85
Peko17	1853438	1888380	1360	85
Peko18	2109724	2016480	1451	85

as shown in Table 5.4. These are more advanced benchmarks in terms of the level of difficulty as compared to ISPD5 and ISPD6. For MMS benchmarks, we use FastPlace3.0_Linux64_DP [121] for legalization and detailed placement.

5.3.0.2 Value of Various Parameters

The value of various parameters during the experiments with PEKO Suite 1 benchmarks are mentioned in Table 5.5. These parameter values are critical for the quality of results and a small change in these values affects runtime of the *Kapees3* . Although near optimal HPWL is obtained after nonlinear optimization, worsening is observed during legalization (refer Table 5.6). An average increase of 32% in HPWL is observed after performing legalization. This is due to the fact that coordinates of almost all the cells changed during legalization.

Table 5.4: Statistics of the MMS benchmark suite

Circuits	# Objects	# Cells	# Macros	# I/Os	# Nets	# Pins
AD1	211447	210904	63	480	221142	944053
AD2	255023	254457	127	439	266009	1069482
AD3	451650	450927	58	665	466758	1875039
AD4	496054	494716	69	1260	515951	1912420
AD5	843128	842482	76	570	867798	3493147
BB1	278164	277604	32	528	284479	1144691
BB2	557866	534782	959	22125	577235	2122282
BB3	1096812	1093034	2549	1229	1123170	3833218
BB4	2177353	2169183	199	7970	2229886	8900078
NB1	330474	330073	64	337	338901	1244342
NB2	441516	436516	3748	1252	465219	1773855
NB3	494011	482833	51	11127	552199	1929892
NB4	646139	642717	81	3341	637051	2499178
NB5	1233058	1228177	91	4790	1284251	4957843
NB6	1255039	1248150	74	6815	1288443	5307594
NB7	2507954	2481372	161	26421	2636820	10104920

Table 5.5: Value of Various Parameters used in the Placement Flow

Parameter	Value
δ	1% chip width
μ_{level}	$\mu_{level-1}/2$
$overflow\_ratio_{min}$	0.1
Step_Size S	1
Accuracy	10^{-4}
Scaling_Factor	0.9
Exit_Iteration	50
n_{max}	3000

5.3.0.3 Comparison of VNM with Conjugate Gradient Method

The proposed optimization method, VNM, is compared with conjugate gradient method. The best possible results in terms of the wirelength are obtained for CG method by varying parameters listed in Table 5.5. The results of comparison of VNM and conjugate gradient are listed in Table 5.7. It can be concluded that VNM outperforms conjugate gradient method on both HPWL and runtime parameters. An improvement of 6.97× in runtime and 8.02% in

Table 5.6: Increase in HPWL Due to Legalization

Circuits	**Increase in HPWL due to Legalization**					
	PEKO Suite 1			**PEKO Suite 2**		
	Before	**After**	**% Increase**	**Before**	**After**	**% Increase**
Peko01	0.73	1.05	30.4	9.3	12.07	22.9
Peko02	1.21	1.62	25.3	12.3	16.38	24.9
Peko03	1.44	1.94	25.7	14.5	19.78	26.6
Peko04	1.96	2.55	23.1	16.6	23.03	27.9
Peko05	1.89	2.44	22.5	19.2	24.83	22.6
Peko06	1.97	2.66	25.9	20.0	27.13	26.2
Peko07	2.75	3.70	25.6	27.7	37.82	26.7
Peko08	3.03	4.57	33.6	30.1	40.93	26.4
Peko09	4.17	5.29	21.1	51.0	60.02	15.0
Peko10	4.74	6.11	22.4	52.0	67.48	22.9
Peko11	4.48	6.12	26.7	45.8	62.28	26.4
Peko12	5.07	6.46	21.5	50.6	66.06	23.4
Peko13	7.51	9.18	18.1	58.1	77.65	25.1
Peko14	10.38	13.38	22.4	112	139.90	19.9
Peko15	11.49	15.05	23.6	152.9	186.74	18.1
Peko16	12.39	16.24	23.7	178.1	209.17	14.8
Peko17	13.65	17.48	21.9	192.6	220.95	12.8
Peko18	12.59	17.05	26.1	210.2	234.39	10.3
Average			24.4			21.8

terms of HPWL is observed over conjugate gradient method.

5.3.0.4 Comparison of Objective Function Parameters

As stated earlier, the estimation of objective function parameter is critical as it affects both runtime and quality of the solution. In the proposed methodology, objective function parameter is called μ which is calculated using Equation 5.13, whereas, in [28] this objective function parameter is called λ which is calculated using Equation (5.18).

$$\lambda = \frac{2\sum(|\frac{\partial H(x,y)}{\partial x} + \frac{\partial H(x,y)}{\partial y}|}{\sum\limits_{x_i,y_i}\sum\limits_{grid\text{-}cells}|B_g|} \tag{5.18}$$

It is found that the value of μ calculated using Equation (5.18) is high.

Table 5.7: HPWL and Runtime Comparison of VNM and CG Methods

Circuits	$Kapees3_{CG}$		$Kapees3_{VNM}$	
	HPWL	Runtime min sec	HPWL	Runtime min sec
Peko01	1.10338e+06	3m29.517s	1.05263e+06	0m19.529s
Peko02	1.89462e+06	4m18.172s	1.62964e+06	0m23.305s
Peko03	2.01576e+06	5m54.735s	1.94654e+06	0m27.590s
Peko04	2.69084e+06	3m58.433s	2.55041e+06	0m38.253s
Peko05	2.77262e+06	3m45.511s	2.44237e+06	0m27.376s
Peko06	2.79434e+06	4m14.681s	2.66424e+06	0m30.742s
Peko07	4.14588e+06	4m27.557s	3.70466e+06	0m46.677s
Peko08	4.39697e+06	5m12.096s	4.57102e+06	1m07.432s
Peko09	4.87444e+06	4m37.212s	5.29045e+06	1m15.053s
Peko10	6.95601e+06	9m10.074s	6.11867e+06	1m13.719s
Peko11	6.38287e+06	6m32.440s	6.12627e+06	1m8.234s
Peko12	7.22223e+06	9m1.444s	6.467e+06	1m20.312s
Peko13	8.4532e+06	8m27.485s	9.18612e+06	1m32.347s
Peko14	1.38641e+07	15m48.727s	1.33844e+07	3m3.602s
Peko15	1.90398e+07	24m3.945s	1.50578e+07	3m17.815s
Peko16	2.05599e+07	23m28.732s	1.6244e+07	3m51.226s
Peko17	1.93902e+07	19m43.307s	1.74894e+07	4m7.184s
Peko18	1.89798e+07	25m15.447s	1.70589e+07	4m50.757s
Average	1.082	6.972	1.00	1.00

For comparing this expression with that of the proposed method, experiment are performed on various benchmark designs. The results of this experiment is shown in Table 5.8. It can be seen that the HPWL and runtime obtained by mathematical expression stated in Equation (5.18) is 11.44× and 28.75× more than that obtained when Equation (5.13) is employed, respectively.

5.3.0.5 Comparison with the Other Placement Tools

Kapees3 is compared with widely available and acceptable placement tools, Capo, Dragon, Feng Shui, NTUPlace3, VPR. The placement tools Capo (version 8.8), Feng Shui (version 5.1) and NTUPlace3 are available on [17], [48] and [115], respectively. The placement tools Capo, Dragon and Feng Shui are based on partition driven paradigm, whereas, NTUPlace3 is based on nonlinear analytical optimization, which is similar to *Kapees3* . The placement tool VPR

Table 5.8: HPWL and Runtime Comparison of Objective Function Parameters

Circuits	$Kapees3_{\lambda}$		$Kapees3_{\mu}$	
	HPWL	Runtime min sec	HPWL	Runtime min sec
Peko01	1.11958e+07	1m0.809s	1.05263e+06	0m19.529s
Peko02	1.65201e+07	2m7.495s	1.62964e+06	0m23.305s
Peko03	2.25126e+07	2m20.458s	1.94654e+06	0m27.590s
Peko04	2.57454e+07	3m21.643s	2.55041e+06	0m38.253s
Peko05	2.72275e+07	3m48.800s	2.44237e+06	0m27.376s
Peko06	2.87981e+07	5m38.121s	2.66424e+06	0m30.742s
Peko07	3.46407e+07	15m27.742s	3.70466e+06	0m46.677s
Peko08	3.7846e+07	22m11.856s	4.57102e+06	1m07.432s
Peko09	3.97386e+07	31m43.996s	5.29045e+06	1m15.053s
Peko10	5.50299e+07	61m29.325s	6.11867e+06	1m13.719s
Peko11	1.79781e+08	4m30.703s	6.12627e+06	1m8.234s
Peko12	5.66697e+07	60m36.377s	6.467e+06	1m20.312s
Peko13	8.00843e+07	59m52.935s	9.18612e+06	1m32.347s
Peko14	1.29908e+08	177m21.863s	1.33844e+07	3m3.602s
Peko15	1.75961e+08	187m26.131s	1.50578e+07	3m17.815s
Peko16	2.05745e+08	207m47.158s	1.6244e+07	3m51.226s
Peko17	2.16966e+08	253m18.402s	1.74894e+07	4m7.184s
Peko18	2.40978e+08	206m23.558s	1.70589e+07	4m50.757s
Average	11.44	28.57	1.00	1.00

is based on the simulated annealing algorithm.

5.3.0.6 Results of Comparison

Kapees3 is compared with other leading placement tools on synthetic benchmarks (PEKO Suite 1 and PEKO Suite 2) and mixed size benchmarks (MMS). For mixed size benchmarks (MMS), we have cited results from their respective publications. Results of FLOP and NTUPLace3-Unified on MMS benchmarks have been cited from [63] and results for MMS benchmarks of all other the placement tools (Capo10.5, FastPlace, ComPLx, POLAR and mPL6) have been cited from [101]. Note that we have not made comparison of runtime for the cited results as these runtimes are reported in the past and it may not be justified to compare them with the current CPU times.

As shown in Table 5.9, for PEKO Suite 1, we achieve better HPWL as

compared to NTUPlace3 by 46%, Dragon by 57%, Feng Shui by 48% and Capo10.5 by 25%. As shown in Table 5.10, on benchmark PEKO Suite 2, *Kapees3* outperforms NTUPlace3 by 30%, Dragon by 47%, Feng Shui by 57%, Capo10.5 by 69% and mPL6 by 2.7%. As shown in Table 5.11, the HPWL generated for these benchmarks is 25.4% and 29.8% far from optimal value.

As shown in Table 5.12, on MMS benchmarks, *Kapees3* obtains wirelength improvement over Capo10.5 by 56.62%, FLOP by 7.84%, FastPlace by 11.55%, ComPLx by 4.58%, POLAR by 23.67%, mPL6 by 9.96%, and NTUPlace3-Unified by 2.96%. However, *Kapees3* is 5.64% worse than eplace in terms of HPWL comparison on MMS Circuits. For FPGA benchmarks, as shown in Table 5.13, *Kapees3* performs better as compared to state-of-the-art FPGA placement tool VPR. It obtains 750% speed up over VPR, whereas, the wirelength of the designs placed by our tool are marginally (8.5%) longer when compared with VPR.

Table 5.9: HPWL comparison for PEKO Suite1 benchmarks

Circuits	**mPL6**	**NTUPlace3**	**Dragon**	**Feng Shui**	**Capo10.5**	**Kapees3**
	HPWL $\times 10^6$	HPWL $\times 10^6$	HPWL $\times 10^6$	HPWL $\times 10^6$	HPWL $\times 10^6$	HPWL $\times 10^6$
Peko01	1.09542	1.8143	1.58346	1.47699	1.27	1.05263
Peko02	1.5768	2.9137	2.43811	2.20006	2.01	1.62964
Peko03	1.86976	3.37261	3.16922	3.1177	2.49	1.94654
Peko04	2.17264	3.52787	3.84755	3.12234	2.83	2.55041
Peko05	2.40986	3.94547	4.16906	3.39114	2.98	2.44237
Peko06	2.56237	3.77107	4.35974	4.03386	3.47	2.66424
Peko07	3.84451	5.5912	6.5193	5.51619	4.54	3.70466
Peko08	3.88346	6.65594	6.22125	6.45594	4.92	4.57102
Peko09	4.51619	6.78346	7.77408	7.38026	5.98	5.29045
Peko10	5.87328	8.63622	8.62384	9.60282	8.81	6.11867
Peko11	6.19216	8.8345	9.24365	9.9063	8.43	6.12627
Peko12	6.21344	9.01085	9.16346	10.8519	8.45	6.467
Peko13	7.31475	10.6745	12.3809	12.2663	10.05	9.18612
Peko14	11.1294	17.7127	17.8304	17.0264	16.30	13.3844
Peko15	14.6259	23.028	24.9829	24.3874	19.97	15.0578
Peko16	18.3354	22.2778	27.8824	25.8121	21.83	16.244
Peko17	20.0313	21.9446	31.3012	28.3241	21.26	17.4894
Peko18	19.5966	25.8896	34.1602	26.4426	25.25	17.0589
Average	0.97	1.4617	1.57	1.48	1.2563	1.00

Table 5.10: HPWL comparison for PEKO Suite2 benchmarks

Circuits	**mPL6**	**NTUPlace3**	**Dragon**	**Feng Shui**	**Capo10.5**	**Kapees3**
	HPWL $\times 10^6$	HPWL $\times 10^6$	HPWL $\times 10^6$	HPWL $\times 10^6$	HPWL $\times 10^6$	HPWL $\times 10^6$
PEKO01	10.1271	15.076	20.248	16.4146	13.5326	12.0781
PEKO02	15.6529	24.0175	31.2822	27.163	21.0727	16.3846
PEKO03	22.4259	28.3669	36.1874	31.7795	33.7368	19.7872
PEKO04	23.6169	32.9212	35.8333	36.2248	41.4622	23.0348
PEKO05	23.8461	35.3598	37.2307	39.8124	45.1628	24.8374
PEKO06	25.7724	38.2358	48.1904	44.5226	38.6956	27.1373
PEKO07	35.7566	53.4979	82.4951	59.9856	67.2508	37.8252
PEKO08	47.1814	57.5143	82.0223	64.5356	80.6975	40.9355
PEKO09	55.8006	62.6516	91.3205	76.6115	92.7892	60.0286
PEKO10	66.178	95.5689	110.9	105.448	132.421	67.4856
PEKO11	74.2934	85.7527	114.442	102.637	120.501	62.2829
PEKO12	82.5655	97.7335	118.453	113.135	123.109	66.0617
PEKO13	86.865	118.926	172.172	134.853	142.334	77.6594
PEKO14	156.285	160.181	NA	213.205	NA	139.908
PEKO15	185.418	202.77	NA	OOM	NA	186.745
PEKO16	207.394	222.271	NA	OOM	NA	209.179
PEKO17	217.015	238.421	NA	OOM	NA	220.95
PEKO18	227.443	234.794	NA	OOM	NA	234.397
Average	1.027	1.302	1.478	1.5757	1.695	1.00

Table 5.11: HPWL comparison with optimal solution

Circuits	**PEKO Suite 1**		**PEKO Suite 2**	
	Optimal	**Kapees3**	**Optimal**	**Kapees3**
	HPWL $\times 10^6$	HPWL $\times 10^6$	HPWL $\times 10^6$	HPWL $\times 10^6$
Peko01	0.814	1.05263	0.81	12.07
Peko02	1.26	1.62964	1.26	16.38
Peko03	1.50	1.94654	1.50	19.78
Peko04	1.75	2.55041	1.75	23.03
Peko05	1.91	2.44237	1.91	24.83
Peko06	2.06	2.66424	2.06	27.13
Peko07	2.88	3.70466	2.88	37.82
Peko08	3.14	4.57102	3.14	40.93
Peko09	3.64	5.29045	3.64	60.02
Peko10	4.73	6.11867	4.73	67.48
Peko11	4.71	6.12627	4.71	62.28
Peko12	5.00	6.467	5.00	66.06
Peko13	5.98	9.18612	5.98	77.65
Peko14	9.01	13.3844	9.01	139.90
Peko15	11.5	15.0578	11.5	186.74
Peko16	12.5	16.244	12.5	209.17
Peko17	13.4	17.4894	13.4	220.95
Peko18	13.2	17.0589	13.2	234.39
Average	0.746	1.0	0.702	1.00

Table 5.12: HPWL comparison for MMS Circuits

Circuits	**Capo10.5**	**FLOP**	**FastPlace**	**ComPLx**	**POLAR**	**mPL6**	**NP3U**	**eplace**	**Kapees3**
AD1	84.77	76.83	82.39	79.05	92.17	77.84	79.05	67.15	73.94
AD2	92.61	84.14	88.53	99.11	149.43	88.40	84.26	77.37	80.00
AD3	202.37	175.99	187.98	175.78	197.48	180.64	168.11	164.50	175.98
AD4	202.38	161.68	187.50	156.75	175.19	162.02	156.33	148.39	164.02
AD5	565.88	357.83	338.77	338.67	380.45	376.30	306.12	315.76	283.64
BB1	112.58	94.92	104.91	96.18	99.12	99.36	99.78	86.82	97.06
BB2	149.54	153.02	145.89	147.19	157.72	144.37	149.96	130.18	148.39
BB3	583.37	346.24	400.40	344.63	420.28	319.63	392.72	302.29	357.28
BB4	915.37	777.84	775.43	772.53	814.07	804.00	809.41	657.92	795.05
NB1	110.54	67.97	73.91	65.26	70.68	66.93	61.75	62.56	66.88
NB2	303.25	187.40	197.15	187.87	197.65	179.18	170.80	166.59	181.42
NB3	1282.19	345.99	325.72	269.47	601.17	415.86	223.38	304.24	294.34
NB4	300.69	256.54	270.70	256.97	277.60	277.69	253.39	229.95	200.99
NB5	570.32	510.83	500.09	453.05	450.69	515.49	450.07	393.21	377.92
NB6	609.16	493.64	512.19	452.83	475.78	482.44	485.73	410.04	441.47
NB7	1481.45	1078.18	1016.10	1010.00	1107.59	1038.66	1125.84	897.81	1089.58
Avg WL	1.5662	1.0784	1.1155	1.0458	1.2367	1.0996	1.0296	0.9436	1.00

Table 5.13: Runtime and HPWL comparison for FPGA benchmarks

Circuits	**VPR**		**Ours**	
	HPWL	sec	HPWL	sec
alu4	11999	11.828	13549	1.628
apex2	18477	20.015	19678	2.173
apex4	13274	10.405	14243	1.435
des	13523	15.147	14077	1.827
diffeq	39333	53.479	42917	11.937
elliptic	13753	8.051	15022	0.999
ex5p	37650	52.416	41603	7.619
frisc	13801	12.781	14385	1.392
misex3	9509	14.831	10851	2.457
seq	18558	16.272	21194	1.936
Average	0.915	7.50	1.00	1.00

5.3.0.7 Runtime Analysis and Results for *Kapees3*

From Table 5.14 it can be observed that, in terms of runtime, *Kapees3* is 3.42× faster than NTUPlace3, 16.64× faster than Dragon, 4.03× faster than Feng Shui, 5.73× faster than Capo10.5 and 3.38× faster than mPL6 on PEKO Suite 1 benchmarks. Similarly, from Table 5.15, it is observed that, on PEKO Suite 2, which consists of bigger designs than PEKO Suite1, *Kapees3* is 2.13× faster than NTUPlace3, 14.26× faster than Dragon, 3.74× faster than Feng Shui, 14.1× faster than Capo10.5 and 4.16× faster than mPL6. As shown in Table 5.16, runtime of *Kapees3* on MMS benchmarks is comparable to that of NTUPlace3-Unified, and *Kapees3*is 1.88× faster than FLOP.

Almost 50% of the runtime (see Figure 5.4) is taken in computing density and its gradient as we consider density estimation for a large number of grid-cells. When *Kapees3* is restricted to estimate density of fewer grid-cells, the runtime improves by 20% but with a degradation of 5% in wirelength. Runtime comparison of all the placement tools is shown in Figure 5.5.

The final placement of cells for the design Peko01 is shown in Figure 5.6, and final placement for MMS circuits is show in Figure 5.7–5.18

Table 5.14: Runtime comparison for PEKO Suite1 benchmarks

Circuits	**mPL6**	**NTUPlace3**	**Dragon**	**Feng Shui**	**Capo10.5**	**Kapees3**
	Runtime min sec	Runtime min sec	Runtime min sec	Runtime min sec	Runtime min sec	Runtime min sec
Peko01	0m50s	0m36s	1m57s	0m52s	1m5s	0m19s
Peko02	1m24s	1m15s	3m53s	1m27s	1m43s	0m23s
Peko03	1m46s	1m51s	4m0s	1m38s	2m0s	0m27s
Peko04	2m1s	2m24s	7m48s	1m53s	2m28s	0m38s
Peko05	2m17s	2m24s	11m46s	2m8s	2m46s	0m27s
Peko06	2m27s	1m11s	10m12s	2m23s	3m2s	0m30s
Peko07	3m21s	1m56s	7m56s	3m24s	4m17s	0m46s
Peko08	4m24s	2m30s	19m8s	3m50s	5m7s	5m49s
Peko09	4m9s	3m4s	17m13s	4m6s	5m40s	1m15s
Peko10	6m6s	12m24s	24m6s	5m44s	7m44s	1m13s
Peko11	5m28s	3m17s	19m19s	5m30s	7m21s	1m8s
Peko12	5m58s	3m21s	26m1s	6m0s	8m0s	1m20s
Peko13	7m6s	4m31s	25m38s	7m3s	9m12s	1m32s
Peko14	13m17s	12m45s	34m8s	11m54s	16m0s	3m3s
Peko15	15m22s	10m36s	47m43s	14m52s	21m6s	3m17s
Peko16	17m7s	11m57s	53m19s	16m25s	23m54s	3m51s
Peko17	18m27s	9m16s	107m35s	17m37s	25m29s	4m7s
Peko18	18m23s	13m20s	94m29s	18m8s	63m22s	4m50s
Average	4.16	3.426	15.646	4.039	5.736	1.00

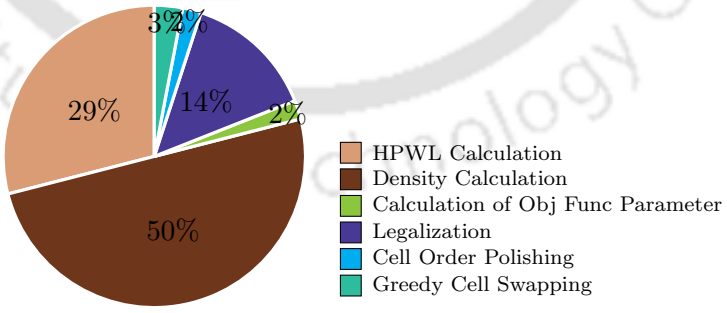

Figure 5.4: Runtime distribution of *Kapees3* for Peko01 benchmark.

Table 5.15: Runtime comparison for PEKO Suite2 benchmarks

Circuits	**mPL6**	**NTUPlace3**	**Dragon**	**Feng Shui**	**Capo10.5**	**Kapees3**
	Runtime min sec	Runtime min sec	Runtime min sec	Runtime min sec	Runtime min sec	Runtime min sec
PEKO01	11m46s	8m33s	51m54s	11m2s	13m30s	2m33s
PEKO02	17m9s	10m19s	79m7s	19m34s	25m53s	4m30s
PEKO03	21m31s	17m48s	79m9s	21m46s	73m58s	4m55s
PEKO04	24m7s	18m52s	82m45s	25m50s	89m43s	6m25s
PEKO05	25m29s	18m32s	122m17s	28m27s	91m11s	6m57s
PEKO06	28m46s	18m16s	104m30s	32m32s	110m2s	8m6s
PEKO07	46m5s	32m27s	209m15s	45m59s	219m3s	12m11s
PEKO08	56m58s	36m38s	299m1s	54m4s	236m8s	15m20s
PEKO09	57m53s	24m29s	252m56s	57m57s	254m16s	20m12s
PEKO10	74m25s	47m32s	199m36s	78m34s	362m17s	22m58s
PEKO11	69m28s	38m44s	161m15s	77m25s	362m10s	21m2s
PEKO12	82m11s	56m59s	219m46s	83m33s	398m43s	22m24s
PEKO13	89m31s	72m1s	346m47s	97m36s	530m44s	26m30s
PEKO14	179m0s	83m45s	>12hrs	178m30s	>12hrs	69m16s
PEKO15	213m16s	110m30s	>12hrs	NA	>12hrs	74m12s
PEKO16	230m3s	92m46s	>12hrs	NA	>12hrs	91m46s
PEKO17	240m52s	95m11s	>12hrs	NA	>12hrs	88m28s
PEKO18	256m3s	106m50s	>12hrs	NA	>12hrs	97m29s
Average	3.38	2.134	14.26	3.746	14.10	1.00

Table 5.16: Runtime comparison for MMS Circuits

Circuits	**FLOP**	**NP3U**	**Kapees3**
	sec	sec	sec
AD1	824	438	518
AD2	1192	651	375
AD3	2116	688	795
AD4	2427	536	1707
AD5	3293	3208	2017
BB1	2007	727	762
BB2	3110	1192	3424
BB3	5437	3505	4386
BB4	19671	8730	11600
NB1	1012	660	779
NB2	2414	822	1248
NB3	2757	1898	1022
NB4	2455	1786	2293
NB5	9163	4075	3694
NB6	9563	3066	5052
NB7	25104	8230	14235
Avg	1.881	0.914	1.0

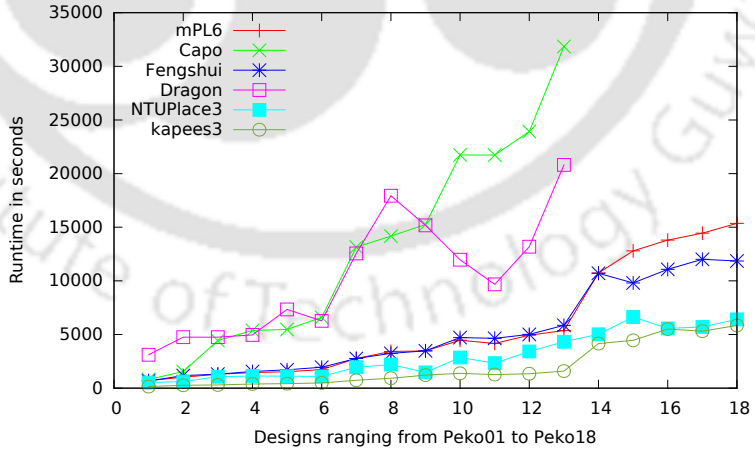

Figure 5.5: Plot of runtimes of the various placement tools for PEKO Suite 2 benchmarks.

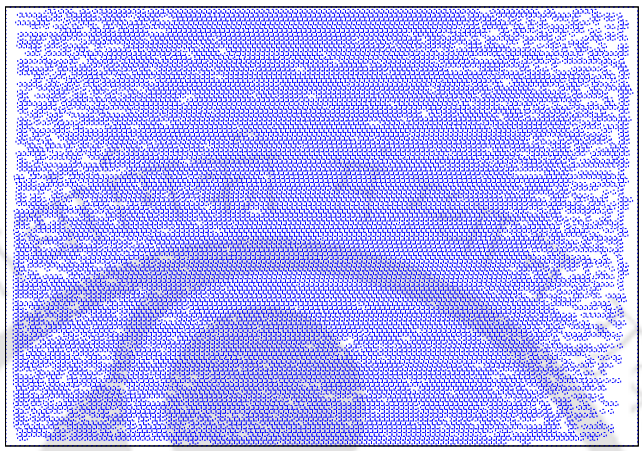

Figure 5.6: The final placement of cells for Peko01 design.

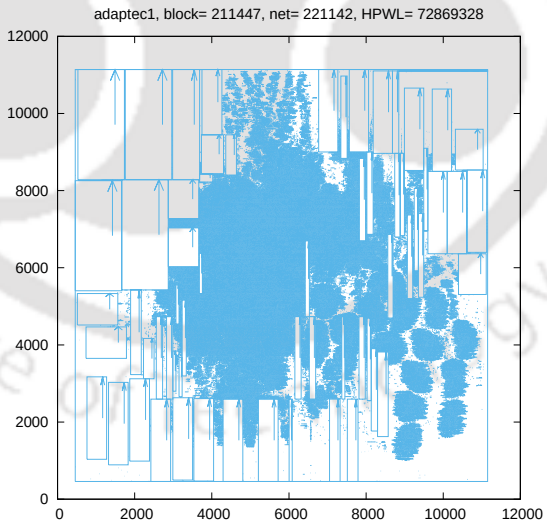

Figure 5.7: The final placement of cells for adaptec1 design.

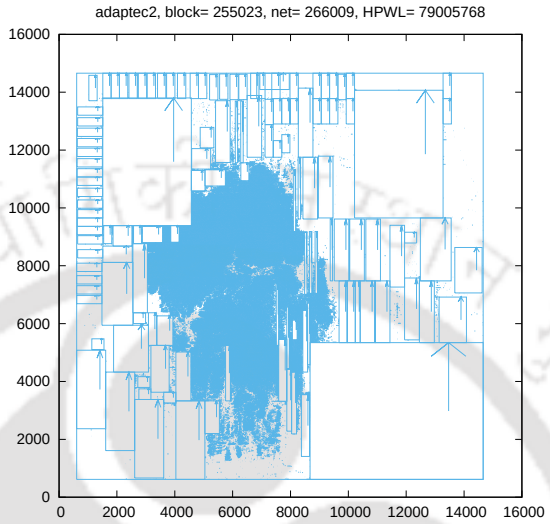

Figure 5.8: The final placement of cells for adaptec2 design.

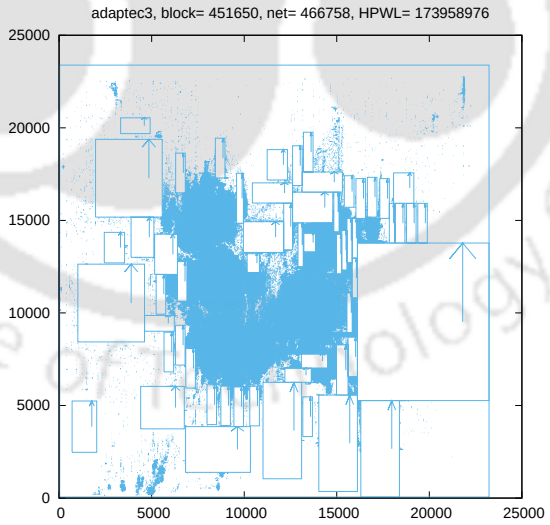

Figure 5.9: The final placement of cells for adaptec3 design.

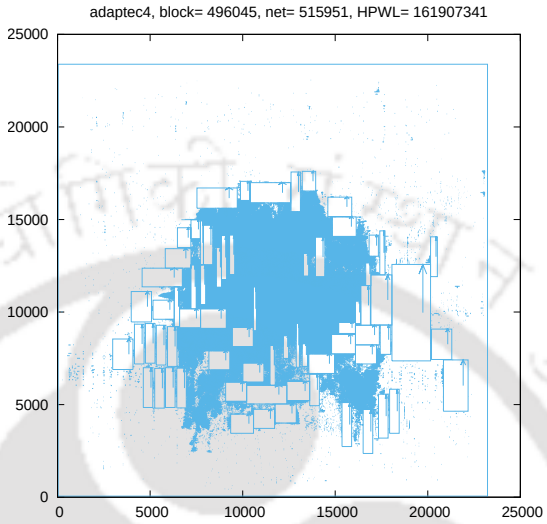

Figure 5.10: The final placement of cells for adaptec4 design.

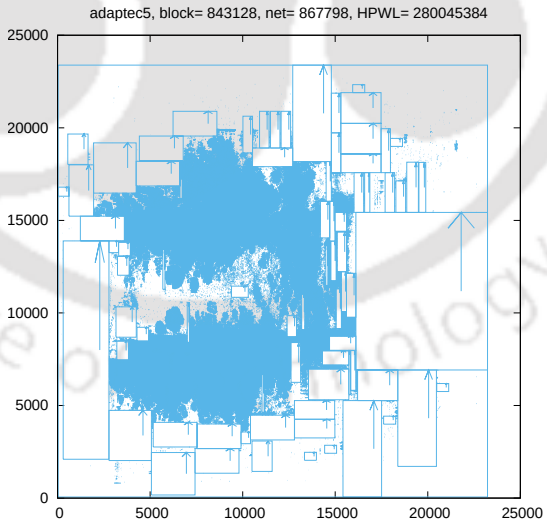

Figure 5.11: The final placement of cells for adaptec5 design.

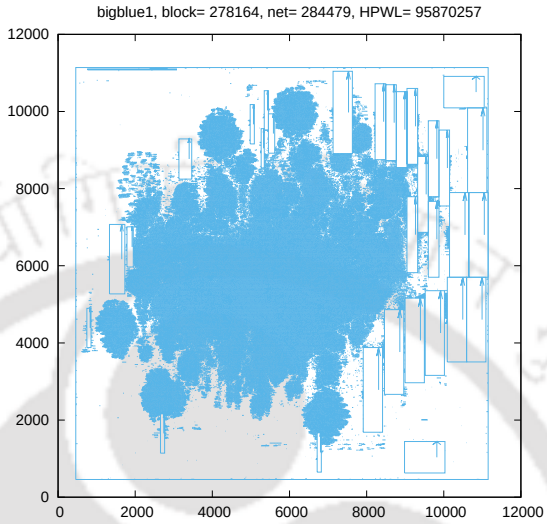

Figure 5.12: The final placement of cells for bigblue1 design.

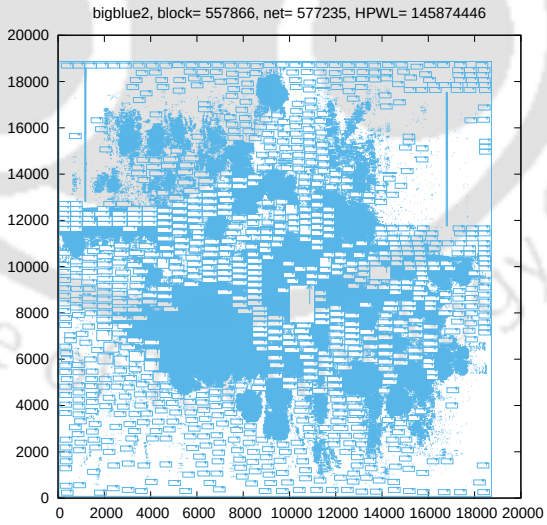

Figure 5.13: The final placement of cells for bigblue2 design.

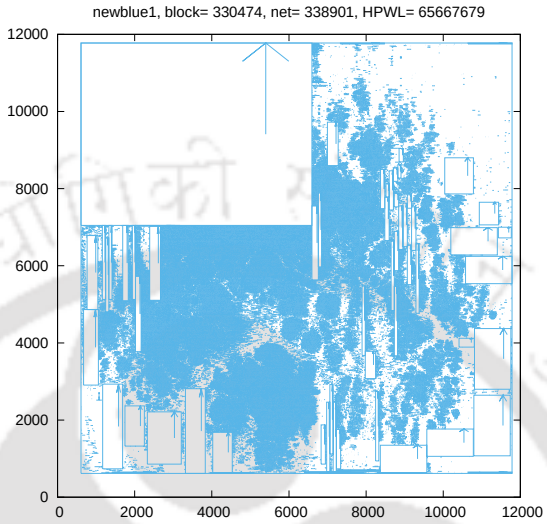

Figure 5.14: The final placement of cells for newblue1 design.

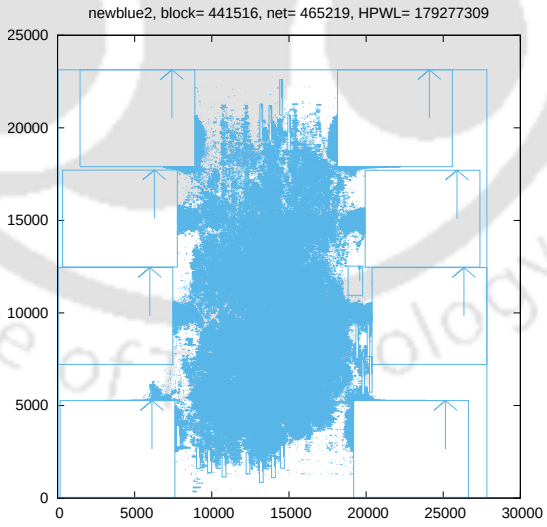

Figure 5.15: The final placement of cells for newblue2 design.

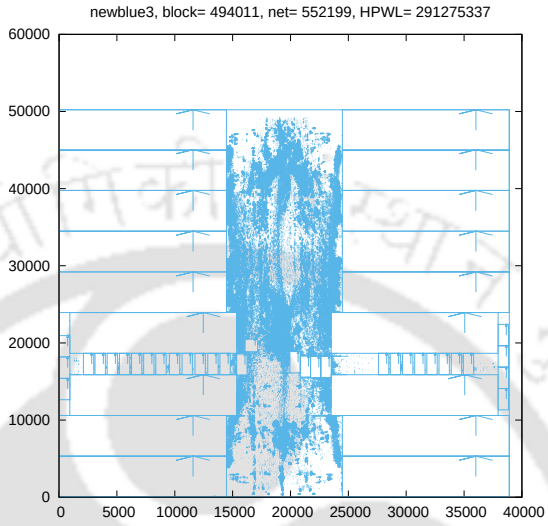

Figure 5.16: The final placement of cells for newblue3 design.

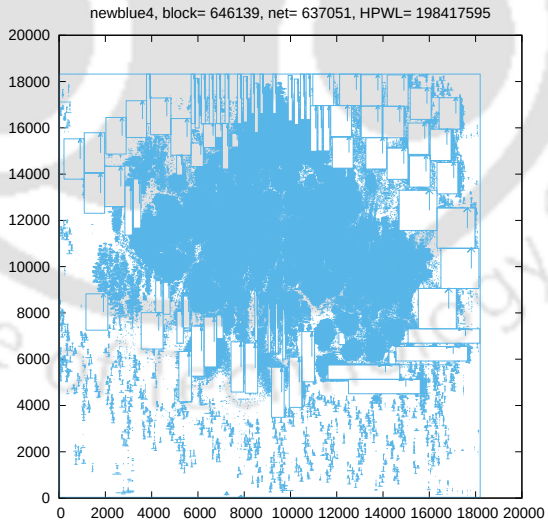

Figure 5.17: The final placement of cells for newblue4 design.

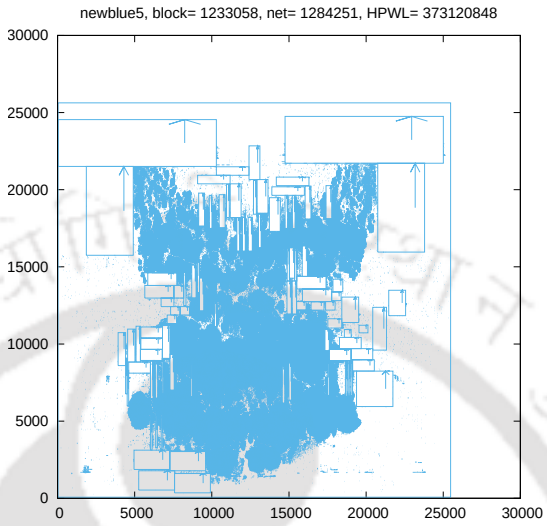

Figure 5.18: The final placement of cells for newblue5 design.

5.3.0.8 Modifications to MMS benchmarks (Free MMS circuits)

The terminals fixed to the layout in MMS circuits do not have any purpose, in terms of design, as they are of zero width and height. These terminals help the placement tools based on quadratic optimization for efficient placement. In the proposed circuits, called Free MMS benchmarks, these terminals are freed from their fixed locations, thus, taking mixed size placement circuits to a higher level of difficulty. Specifically, we introduce the following modification to MMS circuits, to convert them into Free MMS circuits.

- Select circuits of cell count greater than one million
- Set number of nodes to zero
- Remove the terminal keyword from nodes file
- Remove Fixed keywork from *pl* file. As a result of these modifications, the terminals will remain dangling in the circuits, but will have no impact on the design or optimization algorithms

5.4 Scalability analysis of *Kapees3*

In this section, we present scalability analysis of placement tools that we have experimented with. In order to study and compare scalability of placement tools, we introduce synthetic benchmarks. These benchmarks vary from two million to 100 million cell count. We develop a perl script which replicates a two million cell count design (Peko18). The suffix to the name of the benchmark represents the number of million cells present in the design. Example: *synth.2* consists of two million cells and *synth.12* consists of 12 million cells.

The experiments have been performed on a standalone machine with 32GB RAM, clock speed of 1.2 GHz and having on Centos 6.4 operating system. Experiments have been performed using placement tools such as Capo [16], feng shui [3], Dragon [160], mPL6 [22] , NTUPlace3 [28] and *Kapees3* . The results for HPWL and runtime are tabulated in Table 5.17 and Table 5.18 respectively. From the Table 5.17, it can be observed that *Kapees3* produces solution with superior quality compared to other placement tools. Table 5.18 shows the runtime of various placement tools. *Kapees3* consumes the least runtime for the placement of synthetic benchmarks.

Dragon [160], and capo [16] execute for 17 hours and 25.78 hours, respectively, for the first benchmark composed of two million cells. We do not run them for designs with cell count more than two million because these tools take enormous runtime and the quality of the solution is consistently worse than other placement tools. Feng shui [3] aborts for the first design composed of two million cells. The solution provided by mPL6 [22], is better than that produced by Dragon and Capo, however, mPL6 is aborted for designs with more than two million cells. NTUPlace3 executes for all of the large benchmarks which affirms efficient memory management of NTUPlace3. However, runtime of NTUPlace3 is slower than *Kapees3* and its solution has non-optimal wirelength as compared to *Kapees3* .

From the Table 5.17, it can be concluded that, *Kapees3* produces the shortest HPWL for all the designs. Runtime analysis is depicted in Table 5.18, where it shows, *Kapees3* performance is better in terms of runtime as compared to other placement tools. However, *Kapees3* fails to run on the designs having more than 12 million cells. This is due to the fact that, it requires more memory to store the intermediate circuit information during clustering phase, therefore, there is a scope for the improvement in memory management of *Kapees3* . The reasons for better performance of *Kapees3* are discussed in the following subsection.

Table 5.17: HPWL Comparison for Synthetic Benchmarks

Benchmarks	**Dragon** $\times 10^6$	**Capo** $\times 10^6$	**mPL6** $\times 10^6$	**NTUPlace3** $\times 10^6$	*Kapees3* $\times 10^6$
synth.2	396.59	356.54	226.47	226.38	174.55
synth.4	Not Run	Not run	Abort	552.02	417.7
synth.8	Not run	Not run	Abort	1246.08	836.85
synth.12	Not run	Not run	Abort	1655.57	1282.06
synth.16	Not run	Not run	Abort	2823.33	Abort
synth.20	Not run	Not run	Abort	2726.62	Abort
synth.24	Not run	Not run	Abort	Abort	Abort

Table 5.18: Runtime Comparison for Synthetic Benchmarks

Benchmarks	**Dragon** Hrs	**Capo** Hrs	**mPL6** Hrs	**NTUPlace3** Hrs	*Kapees3* Hrs
synth.2	17	25.78	3.68	1.51	1.31
synth.4	Not Run	Not run	Abort	6.59	3.66
synth.8	Not run	Not run	Abort	16.84	7.46
synth.12	Not run	Not run	Abort	14.01	12.01
synth.16	Not run	Not run	Abort	18.41	Abort
synth.20	Not run	Not run	Abort	27.21	Abort
synth.24	Not run	Not run	Abort	Abort	Abort

5.5 Theoretical analysis of *Kapees3*

VLSI Global placement is known to be an NP-complete problem [51]. In the past, most of the proposed algorithms have been developed to address the placement problem. Modern methods based on convex programming are challenged by the non-convexity of the placement problem. This section discusses briefly about Conjugate Gradient (CG), which is a popular method to solve placement problem. Later we illustrate the proposed modification in Nesterov's method to solve the nonlinear placement problem. In the end, we also discuss preconditioning technique which helps us to improve the quality of the solution.

5.5.0.1 Conjugate Gradient Method and Line Search

In Algorithm 10, the details of CG method are illustrated. At first, Polak-Ribiere parameter B_k is computed from the gradient directions h_k at line 4. This is followed by computation of conjugate directions d_k at line 5. Typically line search, at line 6, is to determine the step-length, which is the best solution along the search path d_k and within the search interval ${\alpha_k}^{max}$. A new solution is formulated using previous solution and new step-size as indicated in line 7. This method requires multiple iterations to converge to an optimal solution. As CG is generally used to target locally quadratic functions, it converges faster, if the function f is closer to a quadratic form, else, conjugacy is lost due to B_k becoming zero.

Algorithm 10 Conjugate Gradient Method

Require: $f(x)$: objective function and x_0: initial solution
Ensure: optimal x^*

1: $h_0 = 0$; $d_0 = 0$
2: **while** $f(x_k) < f(x_{k-1})$ **do**
3: $h_k = \nabla f(x_k)$ ▷ Gradient directions
4: $B_k = \frac{h_k^T(h_k - h_{k-1})}{\|h_{k-1}\|^2}$ ▷ Polak-Ribiere parameter B_k
5: $d_k = -h_k + B_k d_{k-1}$ ▷ Conjugate directions
6: $\alpha_k = \arg min_\alpha f(x_k + \alpha \Delta x_k)$ ▷ Line search
7: $x_k = x_{k-1} + \alpha_k d_k$ ▷ Solution update
8: **end while**

From [143], it is observed that the upper bound of the error rate of CG is $\left\|e_{(k)}\right\| \leq 2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)\left\|e_{(0)}\right\|$, where, $\left\|e_{(k)}\right\|$ is the error at kth iteration. κ is the condition number of the H and H is Hessian of the objective function. Also, from [113], it can be derived that the convergence rate by CG method does not exceed $O(1/k)$ [113]. CG has been used in the optimization in various placement tools, for example [73] and [28], however, there are a few shortcomings to this method.

- At line 6 of the CG Algorithm 10, step size is computed, which is the bottleneck during runtime. This can be verified by comparing runtime of Aplace [73] to that of NTUPlace3 [28] (runtime of Aplace is 10× to

that of NTUPlace3). Most of the runtime of placement tools, such as Aplace [73], is consumed during line search. Thus, line search is not efficient when CG is used to solve nonlinear placement formulation.

- During CG iterations, line search finds point at which local gradient is zero. The zero gradient of the objective function may be lying beyond the range of interval used for line search. This causes the solution to be less accurate.

The implementation of CG in NTUPlace3 [28] is faster compared to Aplace [73] because of employing a different formulation for step size calculation. Instead of performing a line search, the step size is estimated as $\alpha_k = \frac{sw_b}{\|d_k\|_2}$, where w_b is the bin dimension and $\|d_k\|_2$ is the Euclidean of the direction vector. s is a constant factor whose value is between 0.2 and 0.3. The value of s establishes a trade-off between quality and runtime.

5.5.0.2 Variant of Nesterov's Method

In *Kapees3* , Nesterov's method is employed to perform nonlinear optimization. In this method, calculation of first order gradient of the objective function and an additional memory (equivalent to an array) are the similar to CG. Our implementation of Nesterov's method, as shown in Algorithm 11, is similar to the one used in eplace [101] with one major difference, the step size computation.

In line 3, h_k is computed, which is the first order gradient of the objective function. In line 4, pre-conditioning (discussed later) of the gradient direction is performed. This is followed by step size calculation in line 5. In ePlace [101], the step size is computed as shown in equation 5.19.

$$\beta_k = \frac{\|v_k - v_{k-1}\|}{\|\nabla f(v_k) - \nabla f(v_{k-1}\|} \tag{5.19}$$

In our implementation of *Kapees3* , step size is computing using $\beta_k = S\frac{w_g}{\|g_k\|}$. This step size calculation mechanism handles issues related with precision and overflow in a better manner. In the approach based on density penalty method, the denominator does not change greatly with every iteration. As a results, step size is very high, and is not efficient for the convergence of objective function, whereas, the rate of convergence of Nesterov's method as shown in Algorithm 11 is $O(1/k^2)$ [114]. It is the first algorithm reported in literature to achieve $O(1/k^2)$ rate of convergence. It is an efficient method under the following condition shown in Equation 5.20.

Algorithm 11 Variant of Nesterov's Method

Require: $f(x)$: objective function and x_0: initial solution
Ensure: optimal x^*

1: $h_0 = 0$; $d_0 = 0$; $N_k = Degree\ of\ Node$
2: **while** $f(x_k) < f(x_{k-1})$ **do**
3: $h_k = \nabla f(x_k)$ ▷ Gradient directions
4: $g_k = \dfrac{h_k}{N_k}$
5: $\beta_k = S\dfrac{w_g}{\|g_k\|_2}$
6: $b_k = 1 + \sqrt{1 + 4b_{k-1}^2}$
7: $u_k = x_{k-1} - \beta_k g_k$
8: $x_k = u_k + \dfrac{(b_{k-1} - 1)(u_k - u_{k-1})}{b_k}$
9: **end while**

$$f(v_k) - f(v_k - \alpha_k \nabla f(v_k)) \geq 0.5\alpha_k \|\nabla f(v_k)\|^2 \tag{5.20}$$

We derive the proposed step size computation from equation 5.20 as follows.

$$\implies f(v_k) - f(v_k - \alpha_k \nabla f(v_k)) \geq 0.5\alpha_k \|\nabla f(v_k)\|^2$$

$$\implies \alpha_k \leq 2\frac{f(v_k) - f(v_k - \alpha_k \nabla f(v_k))}{\|\nabla f(v_k)\|^2}$$

substituting values of v_k and u_k,

$$\implies \alpha_k \leq 2\frac{f(v_k) - f(u_k)}{\|\nabla f(v_k)\|^2}$$

Approximating $f(v_k) - f(u_k) = w_g$, where w_g is width of the grid cell. As we know, in density penalty method based nonlinear placement optimization, the solution changes incrementally in each iteration of VNM.

$$\implies \alpha_k \leq 2\frac{v_k - u_k}{\|\nabla f(v_k)\|^2}$$

$\implies \alpha_k \leq 2\frac{w_g}{\|\nabla f(v_k)\|^2}$

after pre-conditioning,

$\implies \alpha_k \leq 2\frac{w_g}{\|h_k\|^2}$

As discussed in Section 5.3, VNM shows improvement over CG in terms of wirelength as well as runtime. VNM obtains 8.2% shorter wirelength and is 6.972× faster as compared to CG..

5.5.0.3 Runtime complexity

Assuming that there are n cells and e nets in the input netlist for the placement and let the number of auxiliary bins be m. Since we perform clustering of the input netlist, the runtime complexity is estimated for the major computations involved in the proposed methodology. The various complexities are computed as follows.

- Wirelength Computation: Requires traversal of the entire hypergraph in each iteration; Complexity = $O(n + e)$
- Gradient of Wirelength Computation: Similar to wirelength calculation, estimation of gradient of wirelength requires traversal of the entire hypergraph in each iteration; Complexity = $O(n + e)$
- Potential, density, gradient of potential and density: For each cell, it is required to visit neighboring bins to evaluate potential functions associated with the bins for every cell in the netlist. Since we need to consider all the bins in the vicinity of the current bin for a given cell, Complexity = $O(n * m^2)$, which is same as $O(n^2)$, since $m = sqrt(n)$.

Thus, the overall complexity of the proposed approach is $O(n^2 + e)$.

5.5.0.4 Preconditioning

Precondition is an essential step to improve the convergence rate and to reduce condition number of an objective function. In order to perform pre-conditioning of the objective function f, inverse of Hessian H, is computed. Preconditioning has been implemented in various quadratic placement tools such as Fastplace

[154], Fastplace3 [156], Simpl [85], CompLx [86] and POLAR [96], but it is not applied in nonlinear placement tools such as mPL5 [22], NTUPlace3 [28], Aplace [77]. This is due the nonconvex nature of the nonlinear programming problems.

As we know, numerical optimization of objective function can be smoothed by performing pre-conditioning. Due to technology scaling, problem size need to be handled by nonlinear placement methods is very large. This makes traditional pre-conditioning i.e. computation of Hessian, very compute intensive. Hence, we choose Jacobi pre-conditioner, which has only diagonal elements to improve convergence rate of the objective function during our proposed nonlinear placement method. Our pre-conditioning approach is similar to that reported in [101].

$$H_{f_{x,x}} = \begin{pmatrix} \frac{\partial^2 f}{\partial {x_1}^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial {x_2}^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \ddots & \vdots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial {x_n}^2} \end{pmatrix} = \widetilde{H}_{f_{x,x}} = \begin{pmatrix} \frac{\partial^2 f}{\partial {x_1}^2} & 0 & \cdots & 0 \\ 0 & \frac{\partial^2 f}{\partial {x_2}^2} & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & \frac{\partial^2 f}{\partial {x_n}^2} \end{pmatrix} \tag{5.21}$$

5.6 Summary

In recent times, there has been renewed interest in the placement problem because of its inherent difficulty and the current placement tools to be far from optimal. We have investigated this problem and devised an alternative approach based on the analytical framework to address this issue. Our indigenous placement tool, *Kapees3* , first performs clustering of the input netlist followed by quadratic optimization. Later, it solves nonlinear equations to minimize the objective function. We have introduced a new variant of Nesterov's method and compared it with conjugate gradient method. We have also compared our proposed objective function parameter, which is vital to rate quality of the solution of *Kapees3* , with other parameters reported in the literature.

We have compared the proposed placement tool *Kapees3* with the other cutting edge placement tools from the academia and the industry. Our experimental results show that *Kapees3* produces high quality output and is also scalable with the problem size. On benchmarks PEKO Suite 1, the proposed placement tool outperforms NTUPlace3, Dragon, Feng Shui and Capo10.5 by 46%,

57%, 48% and 25%, respectively, in terms of wirelength. In terms of runtime, *Kapees3* is 3.46×, 15.64×, 4.03×, 5.73× faster than NTUPlace3, dragon, Feng Shui and Capo10.5, respectively. On benchmarks PEKO Suite 2, *Kapees3* outperforms NTUPlace3, Dragon, Feng Shui, Capo mPL6 by 30%, 47%, 57%, 69% and 2.7%, respectively, in terms of wirelength. In terms of runtime, *Kapees3* is 2.13×, 14.26×, 3.74×, 14.1× and 4.16× faster than NTUPlace3, dragon, Feng Shui, Capo10.5 and mPL6, respectively. On MMS benchmarks, *Kapees3* demonstrates wirelength improvement over Capo10.5 by 56.62%, FLOP by 7.84%, FastPlace by 11.55%, ComPLx by 4.58%, POLAR by 23.67%, mPL6 by 9.96%, and NTUPlace3-Unified by 2.96%. On MMS benchmarks, we have not compared runtime of *Kapees3* with the published data because it may not be justified, given the progress in computing, whereas, the reported runtimes are for the placers which have been run on our machines. On Free MMS benchmarks, *Kapees3* outperforms NTUPlace3 and ComPLx by 32.52% and 5.73%, respectively, in terms of HPWL. For FPGA benchmarks, *Kapees3* performs better as compared to state-of-the-art FPGA placement tool VPR. It demonstrates 750% speed up over VPR, whereas, the wirelength of the designs placed by *Kapees3* is marginally (average 8.5%) longer as compared to VPR.

Chapter 6

Partitioning

In this chapter, we present a novel approach based on nonlinear programming for partitioning hypergraphs. To the best of our knowledge, nonlinear programming technique has not been employed for partitioning hypergraphs in earlier reported methods. We consider fixed vertices in our implementation, so that the solution is applicable to VLSI placement problems. The proposed nonlinear technique is based on the placement of vertices in two one-dimensional grids (bins) adjacent to each other, and then reducing wirelength associated with hypergraph under density penalty constraints. In the first step, coordinates of vertices are assigned in a random manner within one bin. In order to reduce hyperedge cut and to obtain desired vertex count for each bin, we reduce wirelength associated with the hypergraph and also minimize quadratic penalty associated with each bin. As reported in Chapter 5, the wirelength is modeled by Log-Sum-Exponent approximation of half perimeter wirelength. The gradient of wirelength and gradient of the density associated with each of the bins are calculated using Conjugate Gradient method to obtain an optimal solution.

6.1 Analytical Hypergraph Partitioning Model

In our proposed approach, the hypergraph partitioning problem is addressed by employing analytical technique (or by solving nonlinear equations) i.e. nonlinear optimization using Conjugate Gradient method. Analytical placement was proposed in [112], and is implemented by various placement tools [26, 28, 73]. The hypergraph partitioning problem can be defined as placement of a hyper-

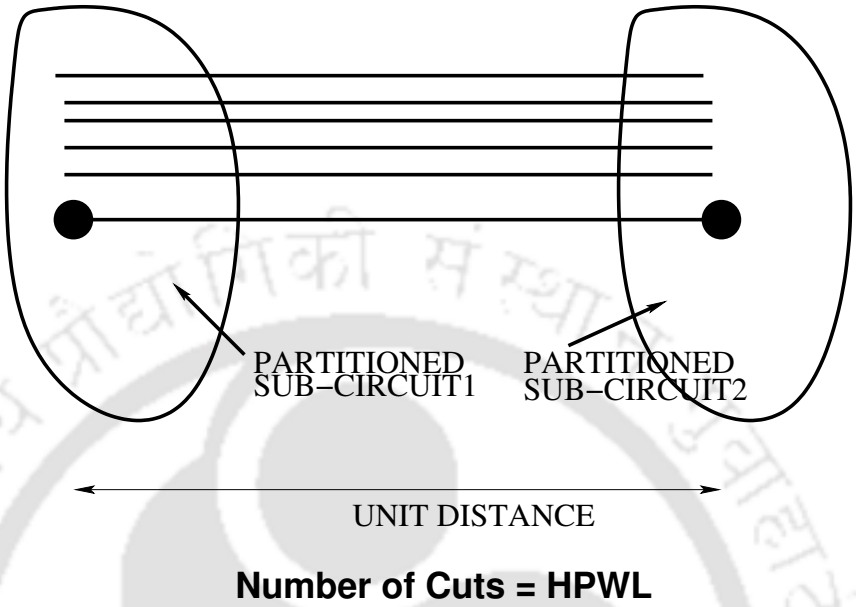

Figure 6.1: HPWL of bisected hypergraph placed in one dimensional bin is equal the the number of Cuts

graph $H = (V, E)$ in two one-dimensional bins placed adjacent to each other, where $V = \{v_1, v_2, \ldots, v_n\}$ and $E = \{e_1, e_2, \ldots, e_n\}$ represent vertices and hyperedges in H, respectively. Let x_i be the x coordinate of movable vertex v_i, the objective of hypergraph partitioning problem is to minimize total Half Perimeter Wire Length (HPWL) or linear distance by finding optimum coordinates for all the vertices in V and density of vertices in both the bins is equal to the desired distribution. HPWL can be defined by Equation (6.1) [80] shown below.

$$W(x, y) = \sum_{e_k \in e_H} (\max_{\forall i,j \in n_H} |x_i - x_j|) \tag{6.1}$$

From the figure 6.1, it can be observed that hypergraph is placed such that one part of the bisected hypergraph is occupying the center of one bin and while the other bisected part is occupying the center of other bin. If the center of the bins are unit distance apart, it is easy to conclude that HPWL of the hypergraph placement after bisection is equal to the number of external hyperedges of the two hyper-subgraphs after bisection. This fact is the motivation behind our approach to introducing analytical hypergraph partitioning. By reducing HPWL associated with placing one dimensional hypergraph circuit, we aim at

reducing the cut size.

The hypergraph placement area is divided into two uniform bins and total HPWL is minimized in such a way that the total module area in both the bins becomes equal. The problem is expressed using the equations shown below.

$$minW(x,y) \quad s.t.\ B_b \leq A_b \tag{6.2}$$

where, A_b is the maximum area of movable vertices in bin-cell b, $W(x)$ is the wirelength, $B_b(x)$ is the potential which is the total area of movable vertices in the bin-cell b. Since $W(x)$ is neither differentiable nor smooth, it cannot be minimized directly. Many smooth wirelength approximations have been proposed, such as quadratic wirelength [46], L_p norm wirelength [22] and log-sum-exp wirelength [112]. Among the above mentioned approximations, log-sum-exp model achieves the best results.

$$W(x,y) = \gamma \sum_{e \in E} (log \sum_{v_k \in e} exp(x_k/\gamma) + log \sum_{v_k \in e} exp(-x_k/\gamma)) \tag{6.3}$$

For density penalty minimization using nonlinear method, it is desired that the density function should be either differentiable or smooth. Since the density function $B_b(x)$ is not smooth it cannot be minimized directly. An inverse Laplace Transform based smoothing function is proposed in [22]. A bell shaped density function which is differentiable at all the points is proposed in Aplace [73]. The above mentioned density function from [73] is incorporated into our partitioning framework.

$$B_b(x,y) = \sum_{v \in V} Q_x(g,v) \tag{6.4}$$

where, Q_x is the overlap functions of bin-cell b and vertex v along the x directions. Equation (6.5) states bell-shaped potential for smoothening Q_x.

$$q_x(g,v) = \begin{cases} 1 - 2\frac{d_x^2}{w_b^2}, & 0 \leq d_x \leq \frac{w_b}{2} \\ \frac{2}{w_b^2}(d_x - w_b)^2, & w_b/2 \leq d_x \leq w_b \\ 0, & w_b \leq d_x \end{cases} \tag{6.5}$$

where, d_x is the center-to-center distance between the vertex v and the bin-cell b in the x-direction and w_b is the bin-cell width. The original and smoothed overlap functions are shown in Figure 5.1. The notations used for different variables in this chapter are described in Table 6.1.

Table 6.1: Variables Used in this Paper

x_i	Centre Coordinate for the Vertex v_i
w_i	Width of the Vertex v_i
w_b	Width of the Grid-cell b
Q_b	Potential of Grid-cell b_i
B_b	Density of Grid-cell b_i
A_b	Maximum Density of Grid-cell b_i
γ	Wirelength Parameter
λ	Objective Function Parameter
α	Step Size

6.2 Discussion on types of graphs and analysis of proposed partitioner

In this section, we present a detailed analysis on type of graphs and quality of solution of the proposed algorithm. We discuss the results of our proposed hypergraph partitioner in the subsequent section on the chosen benchmarks (Walshaw graph benchmarks [159]) as given in Table 6.2. The fourth column in Table 6.2 represents the density of the graph which is calculated by Equation (6.6).

$$\text{Density of Graph} = \frac{|E|}{|V| * (|V| - 1)} \tag{6.6}$$

For the graphs having density more than 50×10^{-5}, our proposed methodology performs better as compared to Chaco when the number of edges are more in the graph (dense). In our proposed method, an iterative scheme, Variations to Nesterov's Method (VNM), having convergence rate $(1/k^2)$, is employed. Since the quality of partitioned output depends on the initial random seed, it is difficult to estimate its runtime complexity. Our proposed approach performs inefficiently for sparse graphs. This is due to the fact that objective function to be minimized has less terms and in spite of moving vertices across the bins, wirelength reduction is not substantial. The dense) and sparse graphs are application dependent. In the chosen benchmarks, we have nearly equal number of dense and sparse graphs. Our proposed algorithm exhibits better results with dense graphs (having density greater than 50×10^{-5}) but produces large cuts for sparse graphs as compared to Chaco.

Table 6.2: Walshaw Benchmark Statistics

Graph	Number of Vertices	Number of Edges	Density ($\times 10^{-5}$)
3elt	4720	13722	123.21
4elt	15606	45878	37.67
598a	110971	741934	12.04
add20	2395	7462	260.28
add32	4960	9462	76.93
bcsstk29	13992	302748	309.30
bcsstk30	28924	1007284	240.81
bcsstk31	35588	572914	90.47
bcsstk32	44609	985046	99.00
bcsstk33	8738	291583	763.86
brack2	62631	366559	18.68
crack	10240	30380	57.95
cs4	22499	43858	17.32
cti	16840	48232	34.01
data	2851	15093	371.50
fe_4elt2	11143	32818	52.86
fe_ocean	143437	409593	3.98
fe_pwt	36519	144794	21.71
fe_rotor	99617	662431	13.35
fe_sphere	16386	49152	36.61
fe_tooth	78136	452591	14.82
finan512	74752	261120	9.34
m14b	214765	1679018	7.28
t60k	60005	89440	4.96
uk	4824	6837	58.77
wave	156317	1059331	8.67
whitaker3	9800	28989	60.37
wing	62032	121544	6.31
wing_nodal	10937	75488	126.22

6.3 Implementation Details of *Nonlinear Analytical Partitioner*

The Algorithm 12 forms backbone of the proposed Nonlinear Analytical Partitioner (NAP) for partitioning hypergraphs. Its implementation details are

described below.

Algorithm 12 *NAP*: Multilevel Algorithm

Require: Hypergraph H_0: standard cell circuit
Ensure: x^* optimal cell positions
1: First Choice Clustering
2: Initialize Cell Positions by Randomization
3: Estimate the base potential of each grid-cell
4: Initialze the value of λ_0, $m = 0$
5: **while** $overflow\_ratio > 0.1$ **do**
6: Solve min $W(x) + \lambda_m \sum (B_g - A_g)^2$
7: $m = m + 1$
8: $\lambda_m = 2 * \lambda_{m-1}$
9: Estimate $overflow\_ratio$
10: **end while**

6.3.1 First Choice Clustering

For reducing problem size in placement and graph partitioning methods, clustering is commonly used e.g. [125], [22] and [28]. By clustering, not only problem size is reduced but the runtime of partitioning algorithm is also abated. Clustering is of two types; *first choice clustering*, where vertices are grouped solely on the basis of graph information (degree, adjacency list) and *best choice clustering*, where, coordinate information is used while grouping the cells. We use first choice clustering of the input graph as described in [22] and [125] in the first step of our graph partitioning algorithm. Each cell is scanned, starting from the vertex with the highest degree to the lowest degree, and affinities are calculated for the vertices which are adjacent to it (sharing a hyperedge).

Let the vertices be m and n, then, the affinity between these vertices is defined as Equation (6.7).

$$r_{mn} = \sum_{\{m,n\} \subseteq e \in E} \frac{w(e)}{(|e| - 1)\ area(e)} \tag{6.7}$$

6.3.2 Initial Placement of Vertices

In our implementation terminal vertices are not movable, whereas, coordinates of the non-terminal vertices are updated during the partitioning flow. As shown

in Figure 6.2, terminals are assigned center coordinate of the bins, whereas, non-terminal vertices are placed randomly and around the center coordinate of one of the bins. Randomization is required to give the optimizer a nonzero value of the wirelength gradient. If initial placement is not randomized, the calculation of initial value of λ would be incorrect.

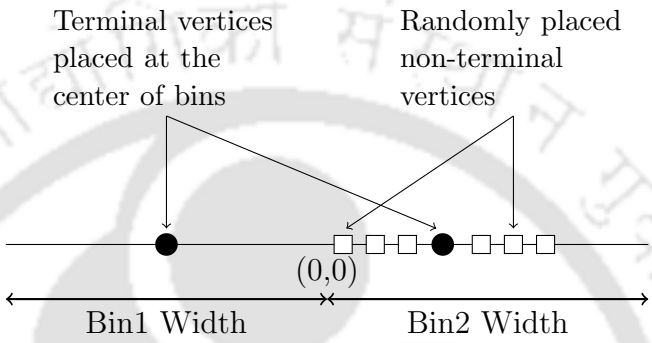

Figure 6.2: Hypergraph solution model in *NAP*.

6.3.3 Calculation of Potential

After random initialization of coordinates of all the vertices, it is required to calculate potential, density and gradient of density of the bins. These values are required to analyze nonlinear equations for performing placement.

6.3.4 Solution of Nonlinear Equation

The solution to the hypergraph placement in one dimension is described by Equation (6.8).

$$min(W(x) + \lambda \sum_{b} (B_b(x) - A_b)^2) \tag{6.8}$$

The Equation (6.8) consists of sum of two terms; approximation of linear distance and density penalty. Both the above objectives oppose each other during the algorithm execution. Reduction in linear distance leads to imbalance of the partitions and, spreading of cells across partition increases wirelength and cut-cost. For optimizing both of these terms in Equation (6.8), a factor leading to the balanced solution is determined. This balancing factor is defined

as λ in Equation (6.9). Calculation of λ is critical to the quality of solution being generated by the partitioner. A smaller value of λ causes imbalance in the vertex area of the partitions, whereas, a larger value causes linear distance to increase. Therefore, an appropriate value of λ is essential to be calculated. The value of λ is estimated using Equation (6.9).

$$\lambda = 2\frac{\sum(|\frac{\partial W(x)}{\partial x}|}{\sum_{x_i}\sum_{bin-cells}|\hat{B_b} - A_b|.(|\frac{\partial \hat{B_b}}{\partial x}|)} \tag{6.9}$$

For measuring evenness of vertex distribution across the bins, a term called $overflow\_ratio$ [28] is employed. The same definition to measure the balance of vertex areas in the two bins of our partitioning framework is applied.

$$Overflow\ Ratio\ = \frac{\sum \max(B_b(x) - A_b, 0)}{\sum Total\ Area} \tag{6.10}$$

To solve Equation (6.8), Conjugate Gradient Method as described in Algorithm 13 is adopted.

Algorithm 13 Conjugate Gradient Method

Require: $f(x)$: objective function and x_0: initial solution
Ensure: optimal x^*

1: $h_0 = 0$; $d_0 = 0$;
2: **while** $f(x_k) < f(x_{k-1})$ **do**
3: Estimate Gradient Directions $h_k = \nabla f(x_k)$;
4: Estimate the Polak-Ribiere parameter B_k
5: $B_k = \frac{h_k^T(h_k - h_{k-1})}{\|h_{k-1}\|^2}$;
6: Estimate the Conjugate Directions $d_k = -h_k + B_k d_{k-1}$;
7: Estimate the Step Size $\alpha_k = s/\|d_k\|_2$;
8: Update the solution $x_k = x_{k-1} + \alpha_k d_k$;
9: **end while**

The condition to exit *while* loop in Algorithm 13 is very important as early exit causes solution to be of poor quality and late exit increases runtime of the partitioning tool unnecessarily. To estimate precise value of threshold set for exiting from the *while* loop, two parameters called $Precision$ and $Exit\_iteration$

are defined. The values of these parameters are calculated empirically. The *while* loop exits when the exit criteria, defined by Equation (6.11), is fulfilled.

$$f_k(x) < Precision * f_{k-Exit\_iteration}(x) \tag{6.11}$$

In Equation (6.4), the parameter γ is an important factor in determining approximation of linear distance For linear distance function to be smooth and differentiable, it is required that γ should not be too small. Very small value of γ may lead to impact numerical precision, which may result into the failure of execution of the partitioner. On the other hand, γ should be sufficiently small such that log-sum-exp function described in Equation (6.4) becomes closer approximation of linear distance We choose γ to be 1% of the width of the bins.

In Conjugate Gradient Method, first, the gradient directions are computed, followed by preconditioning of the gradients by dividing them by the degree of nodes. For determining, optimal step size β_k, back tracking has been performed in [101]. In the proposed method, β_k is calculated using Equation (6.12), where S is the step size. The step size S affects quality of the solution and runtime of the partitioner. For a smaller value of S ($S < 0.01$), better convergence to the near optimal cut is achieved at the cost of runtime. For larger value of S ($S > 1$), partitioner finishes iterations quickly by obtaining poor cut.

$$\beta_k = S\frac{w_b}{\|g_k\|_2} \tag{6.12}$$

6.3.4.1 Run time complexity

Assuming that there are n vertices and e edges/hyperedges in the input graph, and the number of auxiliary bins are m. Here, $m = 2$, since we consider two auxiliary bins placed adjacent to one another. The runtime complexity of the major computations involved in the proposed partitioner are described below.

- Wirelength Computation: Requires traversal of the entire hypergraph in each iteration. Complexity = $O(n + e)$

- Gradient of Wirelength Computation: Similar to wirelength calculation, estimation of gradient of wirelength requires traversal of the entire hypergraph in each iteration. Complexity = $O(n + e)$

- Potential, density and gradient of density: For each cell, it is required to visit neighboring bins to evaluate the potential functions associated with the bins for every vertex in the hypergraph. Since we need to consider only two bins in the vicinity of the current bin of each cell, Complexity $= O(n * m^2)$, which is same as $O(n)$, since $m = 2$.

Thus, the overall complexity of the proposed approach is $O(n + e)$.

6.4 Experiments and results

The proposed Nonlinear Analytical Partitioner *NAP* is implemented in *C* and the experiments are performed on a 2.6 GHz, 64 bit machine operating on Linux. We used *icpc* and *icc* compilers to generate efficient executable suitable for Intel $x86$ architectures.

6.4.1 Benchmarks

Walshaw graph partitioning benchmarks [159] are employed in performing experiments. Details of the available 29 benchmarks are shown in Table 6.2. Apart from Walshaw benchmarks, ISPD98 partitioning benchmarks [2], as shown in Table 6.3, are also used for the comparison with FM based partitioner.

6.4.2 Comparison with Chaco

The quality of the results of our proposed hypergraph partitioner *NAP* is compared with hypergraph partitioner Chaco [21]because it is considered one of the best partitioning tools in terms of quality and runtime. Unlike our approach, Chaco is based on an iterative refinement of hypergraph partitions and is a multilevel implementation of KL algorithm. Each 20 runs of *NAP* and Chaco are compared to obtain best hypergraph partitions with 10% tolerance and to estimate runtimes. The comparison of both the partitioners are summarized in Table 6.4. It can be seen that *NAP* produces better cut-cost as compared with Chaco, while Chaco has better runtime efficiency as compared to *NAP* .

Table 6.3: Properties of ISPD98 partitioning benchmarks

Ckt	# Cells	# Pads	# Modules	# Nets	# Pins
ibm01	12506	246	12752	14111	50566
ibm02	19342	259	19601	19584	81199
ibm03	22853	283	23136	27401	93573
ibm04	27220	287	27507	31970	105859
ibm05	28146	1201	29347	28446	126308
ibm06	32332	166	32498	34826	128182
ibm07	45639	287	45926	48117	175639
ibm08	51023	286	51309	50513	204890
ibm09	53110	285	55395	60902	222088
ibm10	68685	744	69429	75196	297567
ibm11	70152	406	70558	81454	280786
ibm12	70439	637	71706	77240	317760
ibm13	83709	490	84199	99666	357075
ibm14	147088	517	147605	152772	546816
ibm15	161187	383	161570	186608	715823
ibm16	182980	504	183484	190048	778823
ibm17	184752	743	185495	189581	860036
ibm18	201341	272	210613	201920	819697

6.4.3 Comparison with FM based Graph Partitioner

The quality of the results of our proposed hypergraph partitioner *NAP* is also compared with FM based hypergraph partitioner because its wide global acceptability and is being employed by many state-of-the-art partitioning tools. For comparison with *NAP* and performing experiments, the implementation of FM has been taken from [1]. Each 20 runs of *NAP* and FM are compared to obtain best hypergraph partitions with 10% tolerance and to estimate runtimes. The comparison of both the partitioners are summarized in Table 6.5. It can be seen that *NAP* produces better cut-cost as compared with FM, while FM has better runtime efficiency as compared to *NAP* .

6.4.4 Value of parameters

Various parameter values in our experiments with Walshaw graph partitioning benchmarks are mentioned in Table 6.6. These parameters are critical to resulting optimized output of *NAP* . A small change in these parameters causes

Table 6.4: Comparison of Partitioner NAP with existing partitioning tools Chaco on Walshaw benchmarks with 10% tolerance

Benchmark	**Chaco Cut**	**Runtime (in seconds)**	**NAP Cut**	**Runtime (in seconds)**
3elt	90	0.412342	**89**	1.060
4elt	**142**	4.85288	143	8.417
598a	2448	6.80977	**2339**	291
add20	**222**	0.311424	1404	0.616
add32	10	0.707444	**9**	0.928
bcsstk29	3862	4.19367	**2794**	20.833
bcsstk30	6358	3.78961	**6258**	99.137
bcsstk31	2994	6.10933	**2207**	55.693
bcsstk32	6269	12.113	**4427**	144.426
bcsstk33	9911	0.230523	**9911**	7.337
brack2	761	4.81679	**632**	77.467
crack	209	1.75891	**193**	3.482
cs4	388	0.731292	**370**	7.613
cti	318	1.15136	**277**	7.444
data	208	0.344674	**190**	0.848
fe_4elt2	225	1.03731	**130**	4.731
fe_ocean	1855	15.2038	**311**	220.545
fe_pwt	360	12.2771	**347**	26.917
fe_rotor	2143	26.6679	**1881**	114.467
fe_sphere	**386**	0.805466	414	8.073
fe_tooth	4233	15.1409	**3797**	66.729
finan512	162	35.7973	162	84.166
m14b	**3947**	36.4966	26754	38.978
t60k	162	11.1999	**95**	33.025
uk	37	0.340016	**26**	0.709
wave	**9611**	40.338	18415	22.357
whitaker3	133	1.30205	**129**	3.361
wing	1092	7.28288	**1029**	58.296
wing_nodal	2075	0.249361	**1654**	4.566
Average	2090.34		2978.862	

NAP to slow down and to generate a sub-optimal solution. The variation of various functions implemented in our partitioner are shown in Figure 6.3-6.6 for graph *3elt*. A reduction in objective function and linear distance (refer Figure

Table 6.5: Comparison of Cuts obtained by our partitioner with existing partitioning tools FM on ISPD98 benchmarks with 10% tolerance

Benchmark	FM	Runtime	Proposed Tool	Runtime
	Cut	(in seconds)	Cut	(in seconds)
ibm01	630	0.151976	180	5.62
ibm02	478	0.19297	262	6.02
ibm03	2176	0.455931	1097	5
ibm04	1270	0.270959	703	6.99
ibm05	3292	0.542917	2869	5.98
ibm06	1494	0.421936	1183	10.4
ibm07	2349	0.703893	1241	23.4
ibm08	3434	1.38879	1284	22.67
ibm09	2839	1.27481	631	28.6
ibm10	3123	1.9947	2090	35.84
ibm11	2992	2.71659	1141	21.25
ibm12	2707	2.03569	4861	16.73
ibm13	2711	1.9937	1202	29.17
ibm14	11806	7.01593	3767	71.49
ibm15	10265	3.99839	8138	85.63
ibm16	7887	5.62014	3470	65.7
ibm17	6513	13.251	3934	67.92
ibm18	4510	7.79082	2093	108.06
Average	2.115	0.074	1.00	1.00

6.3 and 6.5, respectively) with increasing the number of iterations of Conjugate Gradient Method can be observed. The ratio of area of bins (refer Figure 6.4) increases as the number of iterations progress. This is due to the movement of vertices across the bins. A reduction in number of cuts (refer Figure 6.6) is observed with the number of iterations along with the reduction of wirelength.

Table 6.6: Various Parameters and their values used in the Partitioning Flow

Parameter	**Value**
γ	1% chip width
$Exit\_iteration$	25
$overflow\_ratio_{min}$	0.1
$Precision$	10^{-4}

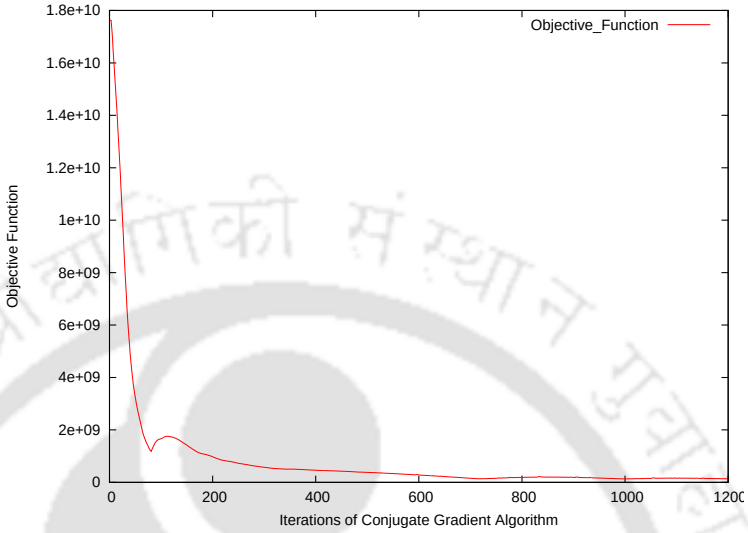

Figure 6.3: Graphs showing Objective Function optimization for the design 3elt

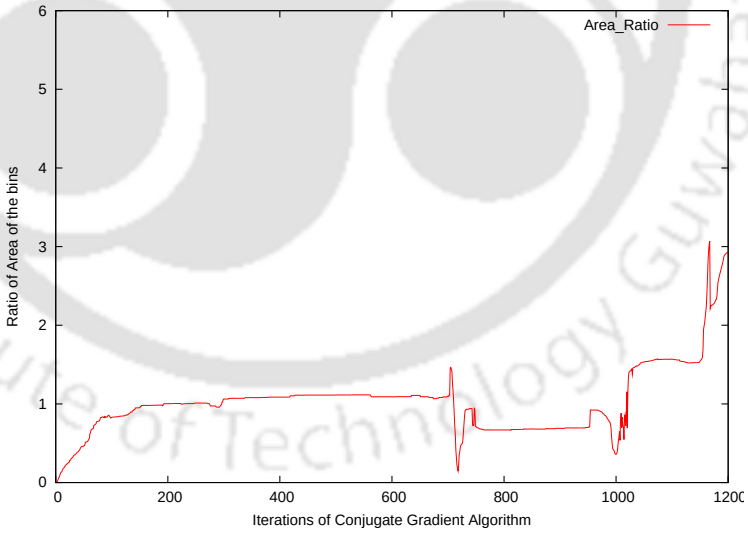

Figure 6.4: Graphs showing Area ration of the partitions for the design 3elt

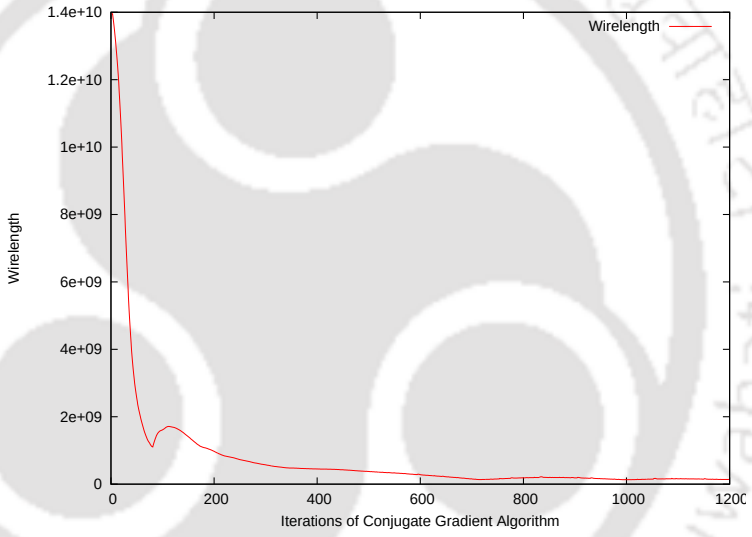

Figure 6.5: Graphs showing linear distance function for the design 3elt

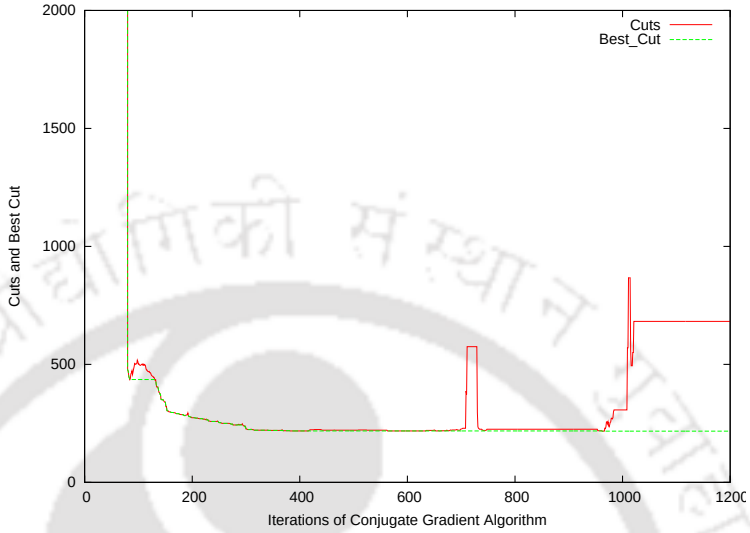

Figure 6.6: Graphs showing Number of Cuts improvement for the design 3elt

6.4.5 Results of Comparison

Tables 6.4 and 6.5 list comparison of best cut obtained after 20 trial runs for each partitioner and runtime for tolerance of 10% of the Chaco and *NAP* . The cuts obtained by *NAP* are better than the cuts obtained by Chaco. Specifically, *NAP* produces better cuts than Chaco on 22 occasions, whereas, Chaco produces better cuts than *NAP* on five occasions while on one instance, both have produced same cuts. *NAP* outperforms FM in 17 benchmarks out of 18, and an average improvement of 111.5% is reported in the quality the of cuts. However, we have observed that *NAP* is three times more runtime inefficiently as compared to Chaco. This is due to the fact that coordinates of all the nodes are modified in every iteration of the Conjugate Gradient method. As a results, incremental cut computation is not possible as compared to iterative heuristics. We perform cut computations on the entire netlist in each iteration of the Conjugate Gradient method.

6.5 Summary

Graph partitioning is a well studied problem. There are several algorithms and heuristics to analyze this NP-Hard problem. We have presented a novel approach for partitioning hypergraphs and it is implemented as NAP . NAP is compared with Chaco and FM based hypergraph partitioner and produces good quality results on the expense of runtime. In future, the proposed approach is extended for k-way partitioning by recursive bisection of a graph.

Chapter 7

3D IC Placement

Due to technology scaling and a need for integrating different functionalities on a single die, interconnect delay, power and area have become of prime importance. For the optimal performance of an integrated circuit on silicon, optimization of delay, power and area of a circuit has become an important step in the design process. This has augmented standard 2D circuit design process and has introduced 3D circuit design process. Floorplanning, placement and routing have an important role in 2D and 3D integrated circuit design process, and have post layout impact on the circuit performance. There are various methods devised to generate optimal floorplanning, placement and routing solutions. Our approach to the 3D placement problem is based on nonlinear analytical methods, which is similar to [64] and [36], and it results in producing an optimal wirelength. The proposed methodology is extended employing partitioning the input 3D integrated circuit and then applying nonlinear optimization to all the planes of 3D layers. In this chapter, our contributions to 3D IC placement are given below.

1) A novel and high quality framework for standard cell placement of 3D ICs

2) A novel approach for performing Through-Silicon-Vias (TSV) aware placement by initial partitioning of the circuit netlist

3) An efficient legalization method considering multiple layers of 3D ICs which is an extension of 2D legalization algorithm

7.1 Preliminaries of 3D-Analytical Placement

In our proposed approach, nonlinear equations are analyzed to address the issue of 3D Standard Cell Placement. Unlike [28], [126], in which optimization is performed on a clustered netlist, We employ nonlinear optimization on a flat netlist. Our 3D analytical placement model is similar to any 2D analytical placement model with a few changes. We add a third dimension, z, to 2D placement model. This third dimension accounts for the layers of 3D ICs along z direction. In this section, the extension of 2D placement model to 3D placement model is described [126].

The approach for standard cell analytical placement presented in this chapter has been previously implemented by [112], [73], [28] and [26]. However, as stated above, we add a third dimension in order to achieve placement for 3D ICs.

Given a hypergraph $H = (V, E)$, where $V = \{v_1, v_2, \ldots, v_n\}$ are the vertices representing cells and $E = \{e_1, e_2, \ldots, e_n\}$ are the edges representing nets of a given netlist, placement is to identify x and y coordinates of all the vertices for optimal the Half Perimeter Wire Length (HPWL) of the circuit. The placement has also an additional constraint that all the cells occupy positions on the rows of a given layout without any overlap. HPWL is defined by Equation 7.1 [126].

$$L(x, y) = \sum_{e_k \in e_H} (\max_{\forall i,j \in n_H} |x_i - x_j| + \max_{\forall i,j \in n_H} |y_i - y_j|) \tag{7.1}$$

The die area is divided into uniform bins and the number of bin cells is roughly equal to the number of cells. Later, HPWL minimization is performed under a constraint that total cell area occupied by bin cells is nearly equal to the area of the bin. For achieving this, quadratic penalty method is used which penalizes bin cells violating the constraints. This can be represented by Equation 7.2.

$$min L(x, y) s.t.\ B_b \ \leq \ A_b \tag{7.2}$$

where, $L(x, y)$ represents wirelength, $B_b(x, y)$ represents potential, which is sum of areas of movable cells in the bin b, and A_b represents the maximum area of movable cells in bin b. $L(x, y)$ is not differentiable, therefore, it is difficult to find minima of this function. In order to smooth $L(x, y)$, approximate HPWL models are proposed such as quadratic [46], L_p norm and log-sum-exp wirelength [112]. The log-sum-exp wirelength is interest to us as it can be com-

puted faster and is found to be a better approximation than L_p. It is expressed in Equation 7.3 [126].

$$\begin{aligned} L(x,y) = \alpha \sum_{e \in E} (log \sum_{v_k \in e} exp(x_k/\alpha) + log \sum_{v_k \in e} exp(-x_k/\alpha) \\ + log \sum_{y_k \in e} exp(y_k/\alpha) + log \sum_{y_k \in e} exp(-y_k/\alpha)) \end{aligned} \tag{7.3}$$

Since our aim is to reduce a non-convex function using differentiation, we must have a density penalty function that is differentiable at all the points. In order to smooth $B_b(x,y)$, mPL [22] uses inverse Laplace transformation to smooth density, whereas APlace [73] uses a bell-shaped function for each cell to cell the density which is smooth at all the points. Bell-shaped function, $B_b(x,y)$ has been used in this work (from [73]) which is expressed as

$$B_b(x,y) = \sum_{v \in V} O_x(b,v) O_y(b,v) O_z(b,v) \tag{7.4}$$

where O_x, O_y and O_z are the overlap functions of bin b and cell v along the x, y and z-directions. To smooth O_x, the bell-shaped potential is defined by the following equation 7.5 [126].

$$o_x(g,v) = \begin{cases} 1 - 2\frac{d_x^2}{w_b^2}, & 0 \le d_x \le \frac{w_b}{2} \\ \frac{2}{w_b^2}(d_x - w_b)^2, & \mathrm{w}_b/2 \le d_x \le w_b \\ 0, & \mathrm{w}_b \le d_x \end{cases} \tag{7.5}$$

where, d_x is centre-to-centre distance between the cell v and the bin b in the x-direction and w_b is the bin width. The original and smoothed overlap functions are shown in Figure 5.1. The notations for different variables used in this chapter are described in Table 7.1.

7.2 Algorithmic Details of the Proposed 3D Placement Tool

In this section, implementation details of the proposed 3D placement tool are presented. Unlike [126], [28] and [26], the proposed tool does not perform clustering to reduce the problem size. Instead, it works on a flat netlist for

Table 7.1: Notations Used and their meanings

x_i, y_i, z_i	Centre Coordinates for the Cell v_i
w_i, l_i, h_i	Width, Length and Height of the Cell v_i
w_b, l_b, h_b	Width, Length and Height of the bin b
O_b	Potential of bin b_i
B_b	Density of bin b_i
A_b	Maximum Density of bin b_i
α	Wirelength Parameter
λ	Objective Function Parameter
α	Step Size

achieving better optimization at the cost of runtime. The flow diagram of proposed 3D placement tools is illustrated in Figure 7.1. Our approach to solve 3D placement problem is described by Algorithm 14. It is a modified version of the algorithm applied for the 2D cell placements described in [126]. Here, the proposed algorithm is demonstrated after incorporating modifications for the 3D cell placement.

7.2.1 Initialization using Random Placement

As shown in line 1 of Algorithm 14, random placement of the cells is performed. This is required for the calculation of initial values of λ. A random placement aids in estimating non-zero values for the initial wirelength and density function. Since, algorithm executes with a small initial value of λ, the randomly placed circuit gets optimized in terms of wirelength. In case the design has pads connected with internal cells, quadratic optimization is performed. Quadratic placement approach has been followed in [126], [117]. The same method is adopted for the quadratic optimization.

The placement area is divided into bins, which is a uniform rectangular grid of cells. We take square root of the number of cells in the netlist. Each cell of the rectangular grid formed is called a bin. Dimensions of the array of grids is given by $NBx = NBy = \sqrt{N/l}$, where NBx and NBy are the number of bins along x and y direction, N is the total number of cells in the input netlist, and l is the number of layers in 3D stacked IC. In order to solve nonlinear Equation 7.2, it is required to have information about potential of the bins. The potential, density, gradient of density and gradient of potential are calculated by solving Equation 7.4 and 7.5.

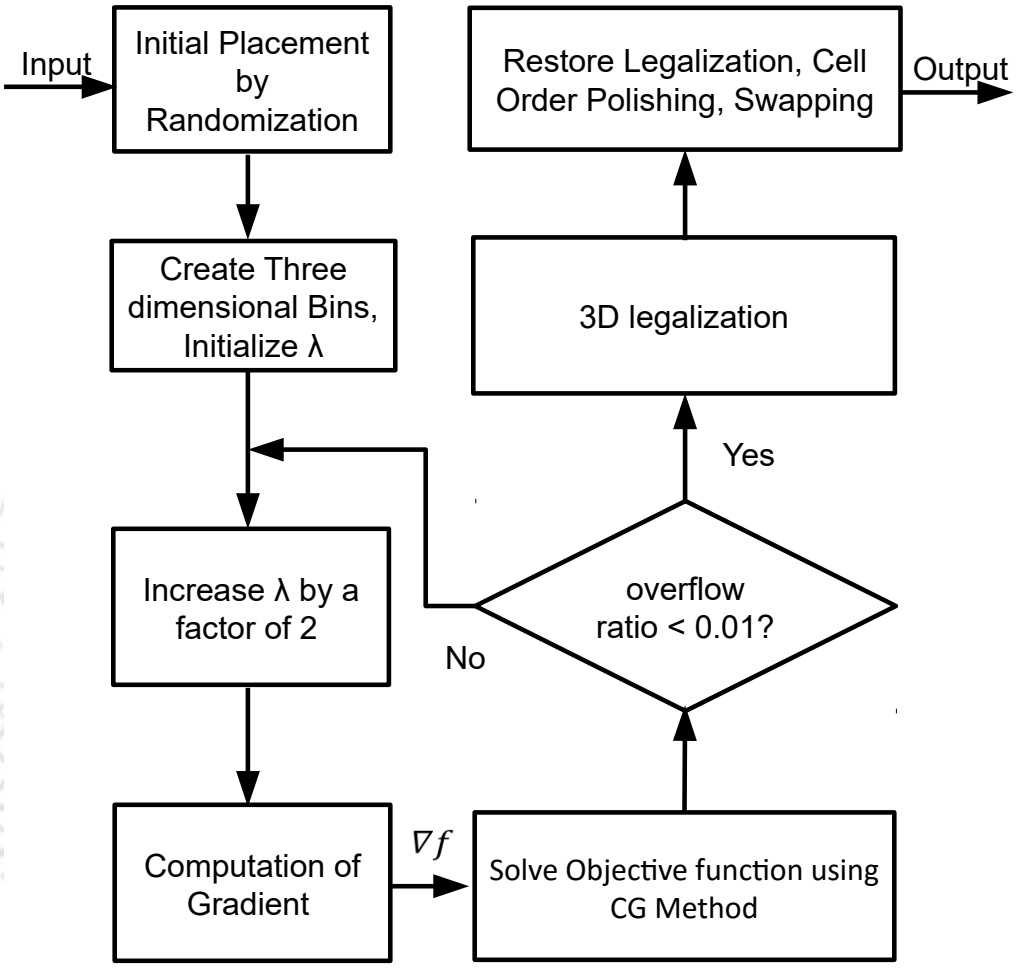

Figure 7.1: Flow diagram of our 3D placement tool

7.2.2 Conjugate Gradient algorithm to solve Nonlinear equation

In line 6 of the proposed Algorithm 14, Equation 7.6 is analyzed.

$$min(L(x,y) + \lambda \sum_{b} (B_b(x,y,z) - A_b)^2) \tag{7.6}$$

The solution of Equation 7.6 forms core of our placement algorithm. This

Algorithm 14 3D Standard Cell Placement Algorithm

Require: Hypergraph H_0: standard cell circuit
and n_{max} Number of cells at coarsest level
Ensure: (x^*, y^*) optimal cell positions

```
Initialize Cell Positions by Random Placement
Calculate number of bins
Calculate base potential of each grid-cell
Initialze value of λ_0, m = 0
while overflow_ratio > 0.01 do
    Solve min L(x, y) + λ_m Σ(B_b − A_b)^2
    m = m + 1
    λ_m = 2 * λ_{m−1}
    Estimate overflow_ratio
    if thenoverflow_ratio < 0.1 and level == 0
        Legalize and save best result
    end if
end while
```

equation is composed of two parts, wirelength and density. When trying to optimize wirelength, cells get clustered, which opposes spreading objective in part *b*. When spreading out the cells to reduce density penalty, a sharp increase in wirelength is observed. Thus, the two objectives oppose each other and a critical balance between the two needs to be established for the optimal placement of 3D IC. The balancing parameter λ is computed by Equation 7.7 in the proposed placement algorithm.

$$\lambda = 2\frac{\sum(|\frac{\partial L(x,y)}{\partial x} + \frac{\partial L(x,y)}{\partial y}|}{\sum_{x_i,y_i}\sum_{bins}|\hat{B_b} - A_b|.(|\frac{\partial \hat{B_b}}{\partial x}| + |\frac{\partial \hat{B_b}}{\partial y}| + |\frac{\partial \hat{B_b}}{\partial z}|)} \tag{7.7}$$

When the solution of Equation 7.6 is obtained by using conjugate gradient method, the value of λ is increased further by a factor of 2, until the cells spread out evenly across the die. In order to determine even spreading of the cells, a parameter called $overflow\_ratio$ is used, which was proposed in [28]. $overflow\_ratio$ is shown in Equation 7.8 [126].

$$Overflow\_ratio \ = \frac{\sum \max(B_b(x,y,z) - A_b, 0)}{\sum Total\ Area} \tag{7.8}$$

For finding solution of equation 7.6, Conjugate Gradient Algorithm is employed as described in Algorithm 15.

Algorithm 15 Conjugate Gradient Algorithm

Require: $f(x)$: objective function and x_0: initial solution
Ensure: optimal x^*

$h_0 = 0$; $d_0 = 0$;
while $f(x_k) < f(x_{k-1})$ **do**
  Estimate Gradient Directions $h_k = \nabla f(x_k)$;
  Estimate the Polak-Ribiere parameter B_k
  $B_k = \frac{h_k^T(h_k - h_{k-1})}{\|h_{k-1}\|^2}$;
  Estimate the Conjugate Directions $d_k = -h_k + B_k d_{k-1}$;
  Estimate the Step Size $\alpha_k = s/\|d_k\|_2$;
  Update the solution $x_k = x_{k-1} + \alpha_k d_k$;
end while

It is important to exit *while* loop of Algorithm 15, when the desired optimization is achieved. Exiting early or late from the *while* loop results in the non-optimal spreading or an increased wirelength. The exit condition of *while* loop is defined by Equation 7.9. The loop must exit when the calculated term of $f_k(x)$ in the current iteration is less than a constant times the calculated term of $f_k(x)$ in the 100^{th} iterations prior to the current iteration. This value is found to be appropriate for the benchmarks considered in the proposed study.

$$f_k(x) < Precision * f_{k-100}(x) \tag{7.9}$$

Closer Approximation of HPWL in Equation 7.3 depends on the value of α. For larger value of α, a smoother approximation is achieved but its value may be far from the actual HPWL. For very small values, there may be issued related with the numerical precision issues, and the wirelength may be obtained beyond bounds. Thus, the α is decided empirically to be reasonably small and is fixed at 1% of the die width.

When applying CG algorithm, it is important to select step size correctly, as it is critical to the quality of solution. Larger step sizes may increase the wirelength, whereas, smaller step sizes may affect placement methods to take a longer time to achieve the optimum. Golder line search algorithm used in [73]

consumes a large portion of runtime. Dynamic step size used in [28] offers a balance between runtime and solution quality. In the proposed methodology the dynamic step size is computed by using Equation 7.10.

$$\beta_k = \frac{sw_b}{\|d_k\|_2} \tag{7.10}$$

7.2.3 Experiment with Initial Partitioning of the netlist for TSV aware implementation

In order to obtain Through-Silicon-Vias (TSV) aware implementation of 3D ICs, we employ k-way partitioning using hMetis [79]. Since partitioner is employed for the partitioning of 3D ICs, it is ensured that the different planes (tiers) of 3D IC should be broken into multiple planes. The planes of partition are maintained during entire flow, which means that z coordinates are not allowed to vary. Thus, in the experiment performed, a planar implementation and objective function does not incorporate z. In this approach, Equation 7.11 is optimized using conjugate gradient.

$$min(L(x,y) + \lambda \sum_b (B_b(x,y) - A_b)^2) \tag{7.11}$$

In Equation 7.11, it is aimed to find the minimum HPWL of the design netlist along with the minimum quadratic penalty of the densities of bins. Densities and their gradients are computed for each layer separately without considering the z coordinate. During the solution of this equation, the overlap between the cells can be ignored as we already have initial partitioning information of the cells, which helps in separating cells belonging to different layers (tiers).

7.2.4 Legalization for 3D placement

The proposed 3D legalization algorithm is presented in Algorithm 16. First, we perform layer assignment, which is an important step in 3D legalization procedure. All the layers should have uniform distribution of cells everywhere around the die. If uniformity is not achieved, it would result into overfilling of cells present in the rows, which causes wirelength to degrade after legalization. As shown in Algorithm 16, first, a two dimensional grid structure B_{2D} is created over the die area. Densities arrays for every layer are created and are filled

Algorithm 16 3D Legalization Algorithm

```
Create two dimensional grid structure of bins B_2D
Create Density array for all layers D_1 ,D_2, D_3, D_4.
Initialize Density array of All Layers to zero
for each cell of the Netlist do
    bin = bin of the cell from B_2D
    d1 = D_1[bin];
    d2 = D_2[bin];
    d3 = D_3[bin];
    d4 = D_4[bin];
    Select the layer which has minimum value of density among d1,d2,d3,d4
    and assign the cell to that layer
    incrementally compute the density for all the bins.
end for
Sort all the cells in increasing value of their x-coordinate
Perform 2D layer wise legalization
```

incrementally because cells are assigned to the layers in the next step. This method achieves uniform distribution of cells across the die area for each layer.

Following this step, layer by layer legalization procedure is followed, which is based on [61]. For each layer of the design, there exists overlap among the cells, which needs to be addressed in a proper manner. Besides this non-overlapping condition, all the cells must occupy different positions on the rows for efficient placement.

To achieve desired 2D legalization of the cells in each layer, tetris [61] based legalization scheme is employed. In this method, first, cells are sorted in the increasing order of their x-coordinate. Later, scaling is performed, which involves multiplication of the sorted x-coordinate with a $scaling\_factor$. The $scaling\_factor$ is chosen to be 0.9 in the experiments performed. This is required to avoid cells from shifting out of the placement region. Following this, every cell is analyzed in the sorted order, and a suitable position on one of the rows of the layout is determined, which gives the least cost in terms of wirelength. It is to be noted that the increase in overall wirelength is observed after legalization due to the movement of large number of cells from their initial locations.

7.2.4.1 Runtime complexity

Assuming, there are n cells and e nets in the input netlist, and the number of auxiliary bins is m. Similar to the estimation of runtime complexity in 2D placement, the runtime complexity for the major computations involved in the proposed methodology is computed as mentioned below.

- Wirelength Computation: Requires traversal of the entire hypergraph in each iteration. Complexity = $O(n + e)$
- Gradient of Wirelength Computation: Similar to wirelength calculation, estimation of gradient of wirelength requires traversal of the entire hypergraph in each iteration. Complexity = $O(n + e)$
- Potential, density, gradient of potential and density: For each cell, it is required to visit neighboring bins to evaluate the potential functions associated with the bins for every cell in the netlist. Since we need to consider all the bins in the vicinity of the current bin for a given cell, Complexity = $O(n * m^2)$, which is same as $O(n^2)$, since $m = sqrt(n)$.

Thus, the overall complexity of the proposed approach is $O(n^2 + e)$.

7.3 Experiments

The proposed 3D placement tool is implemented in *C* Language using the compilers *icc* and *icpc* for optimal executable over Intel x86 architectures. For performing experiments, a 2.6 GHz, 64-bit, multicore GPU based machine with Linux operating system is employed.

7.3.0.1 Benchmarks

IBM version 1 placement benchmarks are selected for the experiments. The statistics of IBM version 1 benchmarks are shown in Table 7.2. The number of cells in these benchmarks vary from 12752 to 53395. These benchmarks are also adapted by ntuplace3d [64] and F3D [36] to showcase effectiveness of their methodologies.

Table 7.2: Characteristics of benchmarks used for experiments (IBM version 1)

Circuits	Cell Count	Net Count	Rows	% Utilization
ibm01	12752	14111	64	10
ibm03	23136	27401	64	10
ibm04	27507	31970	64	10
ibm06	32498	34826	64	10
ibm07	45926	48117	64	10
ibm08	51309	50513	64	10
ibm09	53395	60902	64	10

7.3.0.2 Comparison with the Other Placement Tools

We compared results of our proposed 3D placement tool with the state-of-the-art 3D placement tool F3D [36]. Although F3D [36] is based on analytical optimization, it incorporates different approach compared to us. For more details about F3D please refer [36] and [103]. The comparison between the placement tools are shown in Table 7.3 and Table 7.4. It can be seen that the proposed 3D placement tools outperforms F3D in optimizing HPWL ignoring TSVs . Further, it also produces better TSVs count as compared to F3D while ignoring HPWL.

In the Table 7.3, we have presented results for the proposed 3D placement framework experiment. Results for F3D are taken from [36]. Runtimes of 3D placement tools cannot be compared because they run on different machines. We present results for our TSV oblivious approach and TSV aware approach in columns *Comparison 1* and *Comparison 2*. In *Comparison 1*, we have TSV oblivious approach, where we obtain 16.4% improvement in terms of wirelength over F3D. In this case the TSV count in our approach is more by 8× compared to the TSVs obtained from F3D. For TSV aware approach, *Comparison 2*, TSV counts produced by our proposed 3D placement tool are less than that of F3D by 52% but HPWL has increased by 36% compared to F3D. Final coordinates and 3D view of the netlist in IBM01 circuit is shown in Figure 7.2.

7.3.0.3 Value of parameters

Various parameters employed in the proposed placement tool are listed in Table 7.5. These parameters are used while placement of the benchmarks circuits. We

Table 7.3: 3D Placement results comparison with Cong and Luo [36] on IBM version 1 benchmarks

Circuits	**Comparison 1**					
	F3D [36]			**Proposed Placement Tool (TSV Oblivious)**		
	HPWL $\times 10^7$	#TSVs $\times 10^3$	runtime min	HPWL $\times 10^7$	#TSVs $\times 10^3$	runtime min
ibm01	0.37	0.87	7.19	0.28	9.8	18.7
ibm03	0.84	2.92	12.31	0.70	18.1	55.5
ibm04	1.11	3.36	24.21	0.98	22.1	39.45
ibm06	1.45	3.40	27.07	1.25	28.4	60.2
ibm07	2.27	4.46	36.00	1.92	38.0	68.1
ibm08	2.36	4.43	30.81	2.09	40.9	120.2
ibm09	2.08	3.37	30.14	2.03	42.9	116
Avg	1.164	0.1135	0.38	1.0	1.0	1.0

Table 7.4: 3D Placement results comparison with Cong and Luo [36] on IBM version 1 benchmarks

Circuits	**Comparison 2**					
	F3D [36]			**Proposed Placement Tool(TSV aware)**		
	HPWL $\times 10^7$	#TSVs $\times 10^3$	runtime min	HPWL $\times 10^7$	#TSVs $\times 10^3$	runtime min
ibm01	0.37	0.87	7.19	0.45	0.463	3.5
ibm03	0.84	2.92	12.31	1.09	2.108	7.5
ibm04	1.11	3.36	24.21	1.53	2.263	7.7
ibm06	1.45	3.40	27.07	2.28	2.653	11.54
ibm07	2.27	4.46	36.00	4.18	3.385	18.8
ibm08	2.36	4.43	30.81	4.47	2.850	23.95
ibm09	2.08	3.37	30.14	4.36	1.912	21.52
Avg	0.64	1.52	1.96	1.0	1.0	1.0

observe that slight change in the value of these parameters affects the quality of solution of our proposed placement tool.

7.4 Summary

In this chapter, we present framework of the proposed 3D placement tool with two approaches. In TSV oblivious implementation, we solve nonlinear

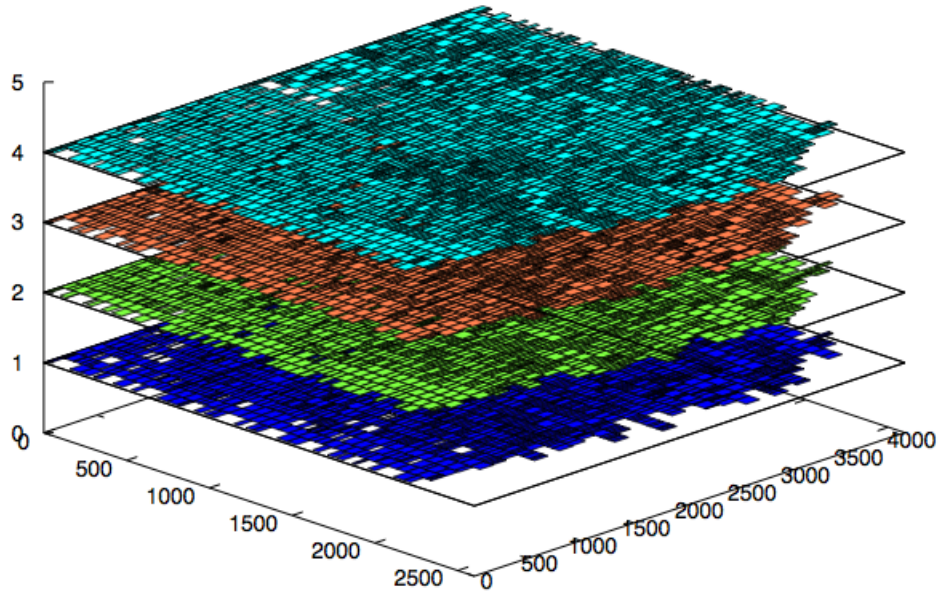

Figure 7.2: Three dimensional view of design IBM01, placed by our 3D placement tool

Table 7.5: Various Parameters and their Values used in the Placement Flow

Parameter	**Value**
α	1% chip width
$overflow\_ratio_{min}$	0.1
Step_Size	0.2
Precision	10^{-4}
Scaling_Factor	0.85

equations to optimize HPWL, where we obtain an improvement of 16.4% over F3D [36]. In TSV aware implementation, nonlinear equations are analyzed for planar topology, where an improvement of 52% is achieved in terms of number of TSVs utilized, when compared with F3D [36]. In future, we plan to improve the runtimes of our 3D placement tool by implementing clustering of the input netlist along with simultaneous optimization of HPWL and the number of TSVs.

Chapter 8

Conclusion and Future Work

In recent times, there has been renewed interest in the placement problem because of its inherent difficulty and the current tools to be far from optimal. In this thesis, this problem is studied and an alternative approach based on a nonlinear analytical framework is proposed. We have developed an indigenous placement tool *Kapees3* using our proposed methodology and it is applied extensively to address issues related to placement. It has been efficiently demonstrated on circuits placement and has been applied effectively for the placement of 3D ICs and partitioning hypergraphs.

Kapees3, first performs clustering of the input netlist followed by quadratic optimization. Later, it solves nonlinear equations to minimize the objective function. We have presented a new variant of Nesterov's method and have compared it with conjugate gradient method. Extensive study of objective function parameters with other parameters reported in the literature is conducted. This is vital to the quality of solution of our placement tool *Kapees3* for the efficient placement of VLSI circuits. We have compared *Kapees3* with the other cutting edge placement tools from the academia and the industry. Our experimental results show that *Kapees3* produces high-quality output and is also scalable in nature. On benchmarks PEKO Suite 1, *Kapees3* outperforms NTUPlace3, Dragon, Feng Shui and Capo10.5 by 46%, 57%, 48% and 25%, respectively in terms of wirelength. In terms of runtime, it is 3.46×, 15.64×, 4.03× and 5.73× faster than NTUPlace3, dragon, Feng Shui and Capo10.5, respectively. On benchmarks PEKO Suite 2, *Kapees3* outperforms NTUPlace3, Dragon, Feng Shui and Capo10.5 by 30%, 47%, 57% and 69%, respectively in terms of wirelength. In terms of runtime, *Kapees3* is 2.13×, 14.26×, 3.74× and 14.1× faster than NTUPlace3, dragon, Feng Shui and Capo10.5, respec-

tively. On MMS benchmarks, *Kapees3* demonstrates wirelength improvement over Capo10.5 by 56.62%, FLOP by 7.84%, FastPlace by 11.55%, ComPLx by 4.58%, POLAR by 23.67%, mPL6 by 9.96%, and NTUPlace3-Unified by 2.96%. On Free MMS benchmarks, *Kapees3* outperforms NTUPlace3 and ComPLx by 32.52% and 5.73%, respectively, in terms of HPWL.

In this thesis, we present framework of the proposed 3D placement tool with two approaches, TSV aware and TSV oblivious. In TSV oblivious approach, nonlinear equations are analyzed to optimize HPWL, where an improvement of 16.4% over F3D [36] is reported. In TSV aware implementation, our proposed 3D placement tool demonstrates an improvement of 52% in terms of number of TSVs utilized as compared to F3D [36]. In future, we plan to improve runtimes of our proposed 3D placement tool by implementing clustering of the input netlist. We also propose to extend this approach for the multi-objective optimization related to 3D IC placement.

Graph partitioning is a well-studied problem. Several methods have been reported to address this NP-complete problem. In this thesis, a novel method of graph partitioning based on the nonlinear analytical optimization is introduced. Using the proposed approach, we have demonstrated partitioning of hypergraphs. In this approach, hypergraph partitioning is modeled as the placement of vertices in two one-dimensional bins. First, the vertices are placed randomly in one of the two one dimensional bins and, then density penalty approach is used to optimize area between these two bins and to reduce HPWL. The proposed method is fast and produces good quality results. In future, there will be an endeavor to extend the proposed method for k-way partitioning of hypergraphs.

The nonlinear analytical optimization method is successfully applied for the placement of VLSI circuits on FPGA and is further implemented to prove its effectiveness in the partitioning driven placement approach. The results reported in this thesis validates our claim. In future, we propose to investigate the possibility of extending nonlinear analytical optimization method for routability driven placement schemes. The concept of the nonlinear analytical optimization method for placement tools implemented in the thesis may also be extended for thermal aware placement and timing driven placement.

Chapter 9

Publications

Patents Filed

1) Sameer Pawanekar, Kalpesh Kapoor and Gaurav Trivedi, "A Nonlinear Analytical Method and System for Hypergraph Partitioning", *Indian Patent App. No. 347/KOL/2015*, 27 March 2015.

Journal Publications

1) Sameer Pawanekar, Kalpesh Kapoor, Gaurav Trivedi, "NAP: A Nonlinear Hypergraph Partitioning Method", *IETE Journal of Research (Taylor & Francis), http://dx.doi.org/10.1080/03772063.2016.1242381* .

2) Sameer Pawanekar, Kalpesh Kapoor, Gaurav Trivedi, "Kapees3: A High-Quality VLSI Standard Cell Placement Tool using Modified Nesterov's Method for Density Penalty" *Journal of Circuits, Systems and Computers (World Scientific)***Under review**

3) Sameer Pawanekar and Gaurav Trivedi, "Kapees4: A High-Quality VLSI Standard Cell Placement Tool using Electrostatic Analogy" *IEEE Transactions on Computer Aided Design,* **To be Submitted**

Conference Publications

1) Sameer Pawanekar andGaurav Trivedi, "Fast FPGA Placement using Analytical Optimization", *Proceedings of VLSI Design and Test (VDAT) 2017.*

2) Sameer Pawanekar andGaurav Trivedi, "Analytical Partitioning:Improvement over FM", *Proceedings of VLSI Design and Test (VDAT) 2017.*

3) Sameer Pawanekar and Gaurav Trivedi, "TSV Aware Standard Cell Placement Tool for 3D ICs", *Proceedings of VLSI Design and Test (VDAT) 2015,https://doi.org/10.1109/ISVDAT.2015.7208113.*

4) Sameer Pawanekar and Gaurav Trivedi, "Net Weighing Based Timing Driven Standard Cell Placer",*Proceedings of VLSI Design and Test (VDAT) 2015, https://doi.org/10.1109/ISVDAT.2015.7208114.*

5) Sameer Pawanekar, Kalpesh Kapoor and Gaurav Trivedi, "A Nonlinear Analytical Optimization Method for Standard Cell Placement of VLSI Circuits", *28th International Conference on VLSI Design, pp. 423-428, January 2015, https://doi.org/10.1109/VLSID.2015.77.*

6) Sameer Pawanekar, Kalpesh Kapoor and Gaurav Trivedi, "Kapees: A New Tool for Standard Cell Placement", *Proceedings of VLSI Design and Test (VDAT) 2013, Springer-Verlag, pp. 66-73, July 2013, DOI: 10.1007/978-3-642-42024-5_9.*

References

[1] Fm hypergraph partitioning code. `http://vlsicad.ucsd.edu/UCLAWeb/cheese/codes/code.tar.gz`. [Online; accessed 17 Jan 2017].

[2] Ispd98 hyergraph partitioning benchmarks. `http://vlsicad.ucsd.edu/UCLAWeb/cheese/ispd98.html`. [Online; accessed 17 Jan 2017].

[3] A. Agnihotri, M. C. Yildiz, A. Khatkhate, A. Mathur, S. Ono, and P. H. Madden. Fractional cut: improved recursive bisection placement. In *Proceedings of International Conference on Computer Aided Design (ICCAD)*, pages 307–310, Nov 2003.

[4] C. J. Alpert, A. E. Caldwell, A. B. Kahng, and I. L. Markov. Hypergraph partitioning with fixed vertices [VLSI CAD]. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 19(2):267–272, Feb 2000.

[5] Charles J. Alpert, Dinesh P. Mehta, and Sachin S. Sapatnekar. *Handbook of Algorithms for Physical Design Automation*. Auerbach Publications, Boston, MA, USA, 1st edition, 2008.

[6] C.J. Alpert, A.E. Caldwell, A.B. Kahng, and I.L. Markov. Hypergraph partitioning with fixed vertices [vlsi cad]. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 19(2):267–272, Feb 2000.

[7] C.J. Alpert, Jen-Hsin Huang, and A.B. Kahng. Multilevel circuit partitioning. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 17(8):655–667, Aug 1998.

[8] A. Ayupov and L. Kraginskiy. A novel timing-driven placement algorithm using smooth timing analysis. In *Design Test Symposium (EWDTS), 2008 East-West*, pages 137–140, Oct 2008.

[9] Vaughn Betz and Jonathan Rose. Vpr: A new packing, placement and routing tool for fpga research, 1997.

[10] E.G. Boman and M.M. Wolf. A nested dissection partitioning method for parallel sparse matrix-vector multiplication. In *High Performance Extreme Computing Conference (HPEC), 2013 IEEE*, pages 1–6, Sept 2013.

[11] U. Brenner and M. Struzyna. Faster and better global placement by a new transportation algorithm. In *Design Automation Conference, 2005. Proceedings. 42nd*, pages 591–596, June 2005.

[12] U. Brenner and J. Vygen. Legalizing a placement with minimum total movement. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 23(12):1597–1613, Dec 2004.

[13] Ulrich Brenner and André Rohe. An effective congestion driven placement framework. In *Proceedings of the 2002 International Symposium on Physical Design*, ISPD '02, pages 6–11, New York, NY, USA, 2002. ACM.

[14] Thang Nguyen Bui and Curt Jones. Finding good approximate vertex and edge partitions is np-hard. *Inf. Process. Lett.*, 42(3):153–159, May 1992.

[15] M. Burstein and M.N. Youssef. Timing influenced layout design. In *Design Automation, 1985. 22nd Conference on*, pages 124–130, June 1985.

[16] A. E. Caldwell, A. B. Kahng, and I. L. Markov. Can recursive bisection alone produce routable, placements? In *Proceedings of Design Automation Conference*, pages 477–482, 2000.

[17] Capo. Tool. `http://vlsicad.eecs.umich.edu/BK/PDtools/tar.gz/Placement-bin/`. [Online; accessed 17 Jan 2013].

[18] U.V. Catalyurek and C. Aykanat. Hypergraph-partitioning-based decomposition for parallel sparse-matrix vector multiplication. *Parallel and Distributed Systems, IEEE Transactions on*, 10(7):673–693, Jul 1999.

[19] U.V. Catalyurek and C. Aykanat. Hypergraph-partitioning-based decomposition for parallel sparse-matrix vector multiplication. *Parallel and Distributed Systems, IEEE Transactions on*, 10(7):673–693, Jul 1999.

[20] Stephen Cauley, Venkataramanan Balakrishnan, Y. Charlie Hu, and Cheng-Kok Koh. A parallel branch-and-cut approach for detailed placement. *ACM Trans. Des. Autom. Electron. Syst.*, 16(2):18:1–18:19, April 2011.

[21] Chaco. Hypergraph partitioning tool. `http://www3.cs.stonybrook.edu/~algorith/implement/chaco/implement.shtml`.

[22] Tony Chan, Jason Cong, and Kenton Sze. Multilevel generalized force-directed method for circuit placement. In *Proceedings of the 2005 international symposium on Physical design*, ISPD '05, pages 185–192, New York, NY, USA, 2005. ACM.

[23] Chin-Chih Chang, J. Cong, M. Romesis, and Min Xie. Optimality and scalability study of existing placement algorithms. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 23(4):537–549, Apr 2004.

[24] H. Chang, E. Shragowitz, J. Liu, H. Youssef, B. Lu, and S. Sutanthavibul. Net criticality revisited: An effective method to improve timing in physical design. In *Proceedings of the 2002 International Symposium on Physical Design*, ISPD '02, pages 155–160, New York, NY, USA, 2002. ACM.

[25] So-Zen Yao Charles J. Alpert. Spectral partitioning: The more eigenvectors, the better. In *Design Automation, 1995. DAC '95. 32nd Conference on*, pages 195–200, 1995.

[26] Jianli Chen and Wenxing Zhu. An analytical placer for vlsi standard cell placement. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 31(8):1208–1221, 2012.

[27] Tung-Chieh Chen, Tien-Chang Hsu, Zhe-Wei Jiang, and Yao-Wen Chang. Ntuplace: a ratio partitioning based placement algorithm for large-scale mixed-size designs. In *Proceedings of the 2005 international symposium on Physical design*, ISPD '05, pages 236–238, New York, NY, USA, 2005. ACM.

[28] Tung-Chieh Chen, Zhe-Wei Jiang, Tien-Chang Hsu, Hsin-Chen Chen, and Yao-Wen Chang. Ntuplace3: An analytical placer for large-scale mixed-size designs with preplaced blocks and density constraints. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 27(7):1228–1240, 2008.

[29] Chung-Kuan Cheng and E.S. Kuh. Module placement based on resistive network optimization. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 3(3):218–225, July 1984.

[30] Yongseok Cheon, Pei-Hsin Ho, Andrew B. Kahng, Sherief Reda, and Qinke Wang. Power-aware placement. In *Proceedings of the 42Nd Annual Design Automation Conference*, DAC '05, pages 795–800, New York, NY, USA, 2005. ACM.

[31] Minsik Cho, Haoxing Ren, Hua Xiang, and R. Puri. History-based vlsi legalization using network flow. In *Design Automation Conference (DAC), 2010 47th ACM/IEEE*, pages 286–291, June 2010.

[32] A. Chowdhary, K. Rajagopal, S. Venkatesan, Tung Cao, V. Tiourin, Y. Parasuram, and B. Halpin. How accurately can we model timing in a placement engine? In *Design Automation Conference, 2005. Proceedings. 42nd*, pages 801–806, June 2005.

[33] Yi-Lin Chuang, Hong-Ting Lin, Tsung-Yi Ho, Yao-Wen Chang, and D. Marculescu. Price: Power reduction by placement and clock-network co-synthesis for pulsed-latch designs. In *Computer-Aided Design (ICCAD), 2011 IEEE/ACM International Conference on*, pages 85–90, Nov 2011.

[34] Yi-Lin Chuang, Gi-Joon Nam, C.J. Alpert, Yao-Wen Chang, J. Roy, and N. Viswanathan. Design-hierarchy aware mixed-size placement for routability optimization. In *Computer-Aided Design (ICCAD), 2010 IEEE/ACM International Conference on*, pages 663–668, Nov 2010.

[35] J. Cong, H.P. Li, Sung Kyu Lim, T. Shibuya, and Dongmin Xu. Large scale circuit partitioning with loose/stable net removal and signal flow based clustering. In *Computer-Aided Design, 1997. Digest of Technical Papers., 1997 IEEE/ACM International Conference on*, pages 441–446, Nov 1997.

[36] J. Cong and Guojie Luo. A multilevel analytical placement for 3d ics. In *Design Automation Conference, 2009. ASP-DAC 2009. Asia and South Pacific*, pages 361–366, Jan 2009.

[37] J. Cong, Guojie Luo, and Yiyu Shi. Thermal-aware cell and through-silicon-via co-placement for 3d ics. In *Design Automation Conference (DAC), 2011 48th ACM/EDAC/IEEE*, pages 670–675, June 2011.

[38] J. Cong, Guojie Luo, Jie Wei, and Yan Zhang. Thermal-aware 3d ic placement via transformation. In *Design Automation Conference, 2007. ASP-DAC '07. Asia and South Pacific*, pages 780–785, Jan 2007.

[39] J. Cong and Min Xie. A robust detailed placement for mixed-size ic designs. In *Proceedings of Asia and South Pacific Conference on Design Automation*, pages 188–194, Jan 2006.

[40] Karen Devine, Bruce Hendrickson, Erik Boman, Matthew St. John, and Courtenay Vaughan. Design of dynamic load-balancing tools for parallel applications. In *Proc. Intl. Conf. on Supercomputing*, pages 110–118, Santa Fe, New Mexico, 2000.

[41] K.D. Devine, E.G. Boman, R.T. Heaphy, R.H. Bisseling, and U.V. Catalyurek. Parallel hypergraph partitioning for scientific computing. In *Parallel and Distributed Processing Symposium, 2006. IPDPS 2006. 20th International*, pages 10 pp.–, April 2006.

[42] A.E. Dunlop, V.D. Agrawal, D.N. Deutsch, M.F. Jukl, P. Kozak, and M. Wiesel. Chip layout optimization using critical path weighting. In *Design Automation, 1984. 21st Conference on*, pages 133–136, June 1984.

[43] T. Easton and R.G. Parker. On critical spanning trees and the linear-arrangement problem. *Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on*, 49(12):1839–1843, Dec 2002.

[44] C.K. Eem and J.W. Chong. An efficient iterative improvement technique for vlsi circuit partitioning using hybrid bucket structures. In *Design Automation Conference, 1999. Proceedings of the ASP-DAC '99. Asia and South Pacific*, pages 73–76 vol.1, Jan 1999.

[45] Hans Eisenmann and F.M. Johannes. Generic global placement and floorplanning. In *Design Automation Conference, 1998. Proceedings*, pages 269–274, June 1998.

[46] Hans Eisenmann and F.M. Johannes. Generic global placement and floorplanning. *Design Automation Conference, 1998. Proceedings*, pages 269–274, 1998.

[47] Hans Eisenmann and Frank M. Johannes. Generic global placement and floorplanning. In *Proceedings of the 35th Annual Design Automation Conference*, DAC '98, pages 269–274, New York, NY, USA, 1998. ACM.

[48] feng shui. Tool. `http://vlsicad.cs.binghamton.edu/pubs/doc/`. [Online; accessed 1 July 2013].

[49] C. M. Fiduccia and R. M. Mattheyses. A linear-time heuristic for improving network partitions. In *Proceedings of the 19th Design Automation Conference*, DAC '82, pages 175–181, Piscataway, NJ, USA, 1982. IEEE Press.

[50] Wenchao Gao, Qiang Zhou, Xu Qian, Yici Cai, and Sifei Wang. Chip layer assignment method for analytical placement of 3d ics. In *Communications, Circuits and Systems (ICCCAS), 2013 International Conference on*, volume 1, pages 403–407, Nov 2013.

[51] Michael R. Garey and David S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1990.

[52] Michael R. Garey and David S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1990.

[53] B. Goplen and S. Sapatnekar. Efficient thermal placement of standard cells in 3d ics using a force directed approach. In *Computer Aided Design, 2003. ICCAD-2003. International Conference on*, pages 86–89, Nov 2003.

[54] L. Hagen and A.B. Kahng. A new approach to effective circuit clustering. In *Computer-Aided Design, 1992. ICCAD-92. Digest of Technical Papers., 1992 IEEE/ACM International Conference on*, pages 422–427, Nov 1992.

[55] B. Halpin, C.Y.R. Chen, and N. Sehgal. Timing driven placement using physical net constraints. In *Design Automation Conference, 2001. Proceedings*, pages 780–783, 2001.

[56] B. Halpin, N. Sehgal, and C.Y.R. Chen. Detailed placement with net length constraints. In *System-on-Chip for Real-Time Applications, 2003. Proceedings. The 3rd IEEE International Workshop on*, pages 22–27, June 2003.

[57] Bill Halpin, C. Y. Roger Chen, and Naresh Sehgal. A sensitivity based placer for standard cells. In *Proceedings of the 10th Great Lakes Symposium on VLSI*, GLSVLSI '00, pages 193–196, New York, NY, USA, 2000. ACM.

[58] T. Hamada, Chung-Kuan Cheng, and P.M. Chau. Prime: A timing-driven placement tool using a piecewise linear resistive network approach. In *Design Automation, 1993. 30th Conference on*, pages 531–536, June 1993.

[59] Xu He, Tao Huang, Linfu Xiao, Haitong Tian, Guxin Cui, and E.F.Y. Young. Ripple: An effective routability-driven placer by iterative cell movement. In *Computer-Aided Design (ICCAD), 2011 IEEE/ACM International Conference on*, pages 74–79, Nov 2011.

[60] R. Hentschke, G. Flach, F. Pinto, and R. Reis. 3d-vias aware quadratic placement for 3d vlsi circuits. In *VLSI, 2007. ISVLSI '07. IEEE Computer Society Annual Symposium on*, pages 67–72, March 2007.

[61] D. Hill. Method and system for high speed detailed placement of cells within an integrated circuit design, April 9 2002. US Patent 6,370,673.

[62] Tsung-Yi Ho and Sheng-Hung Liu. Fast legalization for standard cell placement with simultaneous wirelength and displacement minimization. In *VLSI System on Chip Conference (VLSI-SoC), 2010 18th IEEE/IFIP*, pages 369–374, Sept 2010.

[63] M. K. Hsu and Y. W. Chang. Unified analytical global placement for large-scale mixed-size circuit designs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 31(9):1366–1378, Sept 2012.

[64] Meng-Kai Hsu, V. Balabanov, and Yao-Wen Chang. Tsv-aware analytical placement for 3-d ic designs based on a novel weighted-average wirelength model. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 32(4):497–509, April 2013.

[65] Meng-Kai Hsu, Sheng Chou, Tzu-Hen Lin, and Yao-Wen Chang. Routability-driven analytical placement for mixed-size circuit designs. In *Computer-Aided Design (ICCAD), 2011 IEEE/ACM International Conference on*, pages 80–84, Nov 2011.

[66] Bo Hu and M. Marek-Sadowska. Multilevel fixed-point-addition-based vlsi placement. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 24(8):1188–1203, Aug 2005.

[67] Bo Hu and Malgorzata Marek-Sadowska. Far: Fixed-points addition & relaxation based placement. In *Proceedings of the 2002 International Symposium on Physical Design*, ISPD '02, pages 161–166, New York, NY, USA, 2002. ACM.

[68] Jingcao Hu, Youngsoo Shin, N. Dhanwada, and R. Marculescu. Architecting voltage islands in core-based system-on-a-chip designs. In *Low Power Electronics and Design, 2004. ISLPED '04. Proceedings of the 2004 International Symposium on*, pages 180–185, Aug 2004.

[69] Sung-Woo Hur and J. Lillis. Mongrel: hybrid techniques for standard cell placement. In *Computer Aided Design, 2000. ICCAD-2000. IEEE/ACM International Conference on*, pages 165–170, Nov 2000.

[70] Chanseok Hwang and M. Pedram. Timing-driven placement based on monotone cell ordering constraints. *Design Automation, 2006. Asia and South Pacific Conference on*, pages 6 pp.–, 2006.

[71] D. Jariwala and J. Lillis. Rbi: Simultaneous placement and routing optimization technique. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 26(1):127–141, Jan 2007.

[72] Zhe-Wei Jiang, Bor-Yiing Su, and Yao-Wen Chang. Routability-driven analytical placement by net overlapping removal for large-scale mixed-size designs. In *Design Automation Conference, 2008. DAC 2008. 45th ACM/IEEE*, pages 167–172, June 2008.

[73] A. B. Kahng and Qinke Wang. Implementation and extensibility of an analytic placer. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 24(5):734–747, May 2005.

[74] A.B. Kahng, P. Tucker, and A. Zelikovsky. Optimization of linear placements for wirelength minimization with free sites. In *Design Automation Conference, 1999. Proceedings of the ASP-DAC '99. Asia and South Pacific*, pages 241–244 vol.1, Jan 1999.

[75] A.B. Kahng and Qinke Wang. An analytic placer for mixed-size placement and timing-driven placement. *Computer Aided Design, 2004. ICCAD-2004. IEEE/ACM International Conference on*, pages 565–572, 2004.

[76] Andrew B. Kahng, Igor L. Markov, and Sherief Reda. On legalization of row-based placements. In *Proceedings of the 14th ACM Great Lakes Symposium on VLSI*, GLSVLSI '04, pages 214–219, New York, NY, USA, 2004. ACM.

[77] Andrew B. Kahng and Qinke Wang. A faster implementation of aplace. In *Proceedings of the 2006 International Symposium on Physical Design*, ISPD '06, pages 218–220, New York, NY, USA, 2006. ACM.

[78] G. Karypis, R. Aggarwal, V. Kumar, and Shashi Shekhar. Multilevel hypergraph partitioning: Application in vlsi domain. In *Design Automation Conference, 1997. Proceedings of the 34th*, pages 526–529, June 1997.

[79] G. Karypis and V. Kumar. Multilevel k-way hypergraph partitioning. In *Proceedings of 36th Design Automation Conference*, pages 343–348, 1999.

[80] A Kennings and I Markov. Analytical minimization of half-perimeter wirelength. In *Design Automation Conference, 2000. Proceedings of the ASP-DAC 2000. Asia and South Pacific*, pages 179–184, June 2000.

[81] A Kennings and K.P. Vorwerk. Force-directed methods for generic placement. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 25(10):2076–2087, Oct 2006.

[82] B.W. Kernighan and S. Lin. An Efficient Heuristic Procedure for Partitioning Graphs. *The Bell Systems Technical Journal*, 49(2), 1970.

[83] Ateen Khatkhate, Chen Li, Ameya R. Agnihotri, Mehmet C. Yildiz, Satoshi Ono, Cheng-Kok Koh, and Patrick H. Madden. Recursive bisection based mixed block placement. In *Proceedings of the 2004 International Symposium on Physical Design*, ISPD '04, pages 84–89, New York, NY, USA, 2004. ACM.

[84] Dae Hyun Kim, K. Athikulwongse, and Sung Kyu Lim. A study of through-silicon-via impact on the 3d stacked ic layout. In *Computer-Aided Design - Digest of Technical Papers, 2009. ICCAD 2009. IEEE/ACM International Conference on*, pages 674–680, Nov 2009.

[85] Myung-Chul Kim, Dong-Jin Lee, and Igor L. Markov. Simpl: an effective placement algorithm. ICCAD '10, pages 649–656, Piscataway, NJ, USA, 2010. IEEE Press.

[86] Myung-Chul Kim and I.L. Markov. Complx: A competitive primal-dual lagrange optimization for global placement. In *Design Automation Conference (DAC), 2012 49th ACM/EDAC/IEEE*, pages 747–755, June 2012.

[87] Myung-Chul Kim, Natarajan Viswanathan, Charles J. Alpert, Igor L. Markov, and Shyam Ramji. Maple: multilevel adaptive placement for mixed-size designs. ISPD '12, pages 193–200, New York, NY, USA, 2012. ACM.

[88] J.M. Kleinhans, G. Sigl, F.M. Johannes, and K.J. Antreich. Gordian: Vlsi placement by quadratic programming and slicing optimization. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 10(3):356–365, Mar 1991.

[89] Tim Kong. A novel net weighting algorithm for timing-driven placement. In *Computer Aided Design, 2002. ICCAD 2002. IEEE/ACM International Conference on*, pages 172–176, Nov 2002.

[90] C.H. Lee, M. Kim, and C.I. Park. An efficient k-way graph partitioning algorithm for task allocation in parallel computing systems. In *Systems Integration, 1990. Systems Integration '90., Proceedings of the First International Conference on*, pages 748–751, Apr 1990.

[91] Dong-Jin Lee and I.L. Markov. Obstacle-aware clock-tree shaping during placement. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 31(2):205–216, Feb 2012.

[92] Yu-Min Lee, Tsung-You Wu, and Po-Yi Chiang. A hierarchical bin-based legalizer for standard-cell designs with minimal disturbance. In *Design Automation Conference (ASP-DAC), 2010 15th Asia and South Pacific*, pages 568–573, Jan 2010.

[93] R. Leland and B. Hendrickson. An empirical study of static load balancing algorithms. In *Scalable High-Performance Computing Conference, 1994., Proceedings of the*, pages 682–685, May 1994.

[94] Jing Li and H. Miyashita. Efficient thermal via planning for placement of 3d integrated circuits. In *Circuits and Systems, 2007. ISCAS 2007. IEEE International Symposium on*, pages 145–148, May 2007.

[95] Shuai Li and Cheng-Kok Koh. Mixed integer programming models for detailed placement. In *Proceedings of the 2012 ACM International Symposium on International Symposium on Physical Design*, ISPD '12, pages 87–94, New York, NY, USA, 2012. ACM.

[96] Tao Lin, C. Chu, J.R. Shinnerl, I. Bustany, and I. Nedelchev. Polar: Placement based on novel rough legalization and refinement. In *Computer-Aided Design (ICCAD), 2013 IEEE/ACM International Conference on*, pages 357–362, Nov 2013.

[97] Bin Liu, Yici Cai, Qiang Zhou, and Xianlong Hong. Power driven placement with layout aware supply voltage assignment for voltage island generation in dual-vdd designs. In *Design Automation, 2006. Asia and South Pacific Conference on*, pages 6 pp.–, Jan 2006.

[98] Guilin Liu, Zhuoyuan Li, Qiang Zhou, and Xianlong Hong. 3d placement algorithm with vertical via count constraints. In *Communications, Circuits and Systems, 2005. Proceedings. 2005 International Conference on*, volume 2, pages –1234, May 2005.

[99] Guilin Liu, Zhuoyuan Li, Qiang Zhou, Xianlong Hong, and Hannah Honghua Yang. 3d placement algorithm considering vertical channels and guided by 2d placement solution. In *ASIC, 2005. ASICON 2005. 6th International Conference On*, volume 2, pages 787–791, Oct 2005.

[100] Jingwei Lu, Pengwen Chen, Chin-Chih Chang, Lu Sha, D.J.-H. Huang, Chin-Chi Teng, and Chung-Kuan Cheng. Fftpl: An analytic placement

algorithm using fast fourier transform for density equalization. In *ASIC (ASICON), 2013 IEEE 10th International Conference on*, pages 1–4, Oct 2013.

[101] Jingwei Lu, Pengwen Chen, Chin-Chih Chang, Lu Sha, D.J.-H. Huang, Chin-Chi Teng, and Chung-Kuan Cheng. eplace: Electrostatics based placement using nesterov's method. In *Design Automation Conference (DAC), 2014 51st ACM/EDAC/IEEE*, pages 1–6, June 2014.

[102] Jingwei Lu, Hao Zhuang, Pengwen Chen, Hongliang Chang, Chin-Chih Chang, Yiu-Chung Wong, Lu Sha, D. Huang, Yufeng Luo, Chin-Chi Teng, and Chung-Kuan Cheng. eplace-ms: Electrostatics-based placement for mixed-size circuits. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 34(5):685–698, May 2015.

[103] Guojie Luo, Yiyu Shi, and Jason Cong. An analytical placement framework for 3-d ics and its extension on thermal awareness. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 32(4):510–523, 2013.

[104] Shengmei Luo, Lixia Liu, Hongxu Wang, Bin Wu, and Yang Liu. Implementation of a parallel graph partition algorithm to speed up bsp computing. In *Fuzzy Systems and Knowledge Discovery (FSKD), 2014 11th International Conference on*, pages 740–744, Aug 2014.

[105] Tao Luo, D. Newmark, and D.Z. Pan. A new lp based incremental timing driven placement for high performance designs. In *Design Automation Conference, 2006 43rd ACM/IEEE*, pages 1115–1120, 2006.

[106] Tao Luo, Haoxing Ren, C.J. Alpert, and D.Z. Pan. Computational geometry based placement migration. In *Computer-Aided Design, 2005. ICCAD-2005. IEEE/ACM International Conference on*, pages 41–47, Nov 2005.

[107] MLPart. Hypergraph partitioning survey. `http://web.eecs.umich.edu/~imarkov/pubs/book/part_survey.pdf`.

[108] MLPart. Hypergraph partitioning survey. `http://web.eecs.umich.edu/~imarkov/pubs/book/part_survey.pdf`.

[109] Gi-Joon Nam. ISPD 2006 placement contest: Benchmark suite and results. In *International Symposium on Physical Design*, pages 167–167, New York, NY, USA, 2006. ACM.

[110] Amit Narayan. *Bdd Partitioning for Formal Verification and Synthesis of Digital Systems*. PhD thesis, University of California, Berkeley, 1998. AAI9902174.

[111] V. Natesan and D. Bhatia. Performance driven placement for cell-based designs. In *ASIC Conference and Exhibit, 1995., Proceedings of the Eighth Annual IEEE International*, pages 237–240, Sep 1995.

[112] W. C. Naylor, R. Donelly, and L. Sha. Non-linear optimization system and method for wire length and delay optimization for an automatic electric circuit placer . *U.S. Patent 6 301 693*, U.S. Patent 6 301 693(12), 2001.

[113] AS Nemirovsky and DB Yudin. Problem complexity and method efficiency in optimization. 1983.

[114] Yurii Nesterov. *Introductory lectures on convex optimization: a basic course*. Applied optimization. Kluwer Academic Publishers, Boston, Dordrecht, London, 2004.

[115] NTUPlace3. Tool. `http://eda.ee.ntu.edu.tw/w04/download/ntuplace.php`. [Online; accessed 1 July 2013].

[116] Bernd Obermeier and Frank M. Johannes. Temperature-aware global placement. In *Proceedings of the 2004 Asia and South Pacific Design Automation Conference*, ASP-DAC '04, pages 143–148, Piscataway, NJ, USA, 2004. IEEE Press.

[117] Bernd Obermeier, Hans Ranke, and Frank M. Johannes. Kraftwerk: a versatile placement approach. ISPD '05, pages 242–244, New York, NY, USA, 2005. ACM.

[118] Chao-Wei Ou and S. Ranka. Parallel incremental graph partitioning. *Parallel and Distributed Systems, IEEE Transactions on*, 8(8):884–896, Aug 1997.

[119] Min Pan and C. Chu. Ipr: An integrated placement and routing algorithm. In *Design Automation Conference, 2007. DAC '07. 44th ACM/IEEE*, pages 59–62, June 2007.

[120] Min Pan, N. Viswanathan, and C. Chu. An efficient and effective detailed placement algorithm. In *Computer-Aided Design, 2005. ICCAD-2005. IEEE/ACM International Conference on*, pages 48–55, Nov 2005.

[121] Min Pan, N. Viswanathan, and C. Chu. An efficient and effective detailed placement algorithm. In *Computer-Aided Design, 2005. ICCAD-2005. IEEE/ACM International Conference on*, pages 48–55, Nov 2005.

[122] D.A. Papa, Tao Luo, M.D. Moffitt, C.N. Sze, Zhuo Li, Gi-Joon Nam, C.J. Alpert, and I.L. Markov. Rumble: An incremental timing-driven physical-synthesis optimization algorithm. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 27(12):2156–2168, Dec 2008.

[123] P.N. Parakh, R.B. Brown, and K.A. Sakallah. Congestion driven quadratic placement. In *Design Automation Conference, 1998. Proceedings*, pages 275–278, June 1998.

[124] PaToH. Hypergraph partitioning tool. `http://bmi.osu.edu/umit/software.html#patoh`.

[125] S. Pawanekar, G. Trivedi, and K. Kapoor. A nonlinear analytical optimization method for standard cell placement of vlsi circuits. In *VLSI Design (VLSID), 2015 28th International Conference on*, pages 423–428, Jan 2015.

[126] S. Pawanekar, G. Trivedi, and K. Kapoor. A nonlinear analytical optimization method for standard cell placement of vlsi circuits. In *VLSI Design, 2015. Held jointly with 6th International Conference on Embedded Systems., 20th International Conference on*, pages 863–868, Jan 2015.

[127] Sameer Pawanekar, Kalpesh Kapoor, and Gaurav Trivedi. Kapees: A new tool for standard cell placement. In *VDAT*, volume 382 of *Communications in Computer and Information Science*, pages 66–73. Springer, 2013.

[128] Sameer Pawanekar, Kalpesh Kapoor, and Gaurav Trivedi. Kapees: A new tool for standard cell placement. In *VDAT*, volume 382 of *Communications in Computer and Information Science*, pages 66–73. Springer, 2013.

[129] Ruchir Puri, Leon Stok, John Cohn, David Kung, David Pan, Dennis Sylvester, Ashish Srivastava, and Sarvesh Kulkarni. Pushing asic performance in a power envelope. In *Proceedings of the 40th Annual Design Automation Conference*, DAC '03, pages 788–793, New York, NY, USA, 2003. ACM.

[130] Karthik Rajagopal, Tal Shaked, Yegna Parasuram, Tung Cao, Amit Chowdhary, and Bill Halpin. Timing driven force directed placement with physical net constraints. In *Proceedings of the 2003 International Symposium on Physical Design*, ISPD '03, pages 60–66, New York, NY, USA, 2003. ACM.

[131] Haoxing Ren, D.Z. Pan, C.J. Alpert, Gi-Joon Nam, and P. Villarrubia. Hippocrates: First-do-no-harm detailed placement. In *Design Automation Conference, 2007. ASP-DAC '07. Asia and South Pacific*, pages 141–146, Jan 2007.

[132] Haoxing Ren, D.Z. Pan, C.J. Alpert, P.G. Villarrubia, and Gi-Joon Nam. Diffusion-based placement migration with application on legalization. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 26(12):2158–2172, Dec 2007.

[133] Haoxing Ren, D.Z. Pan, and D.S. Kung. Sensitivity guided net weighting for placement-driven synthesis. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 24(5):711–721, May 2005.

[134] B.M. Riess and G.G. Ettelt. Speed: fast and efficient timing driven placement. In *Circuits and Systems, 1995. ISCAS '95., 1995 IEEE International Symposium on*, volume 1, pages 377–380 vol.1, Apr 1995.

[135] J.A. Roy, N. Viswanathan, Gi-Joon Nam, C.J. Alpert, and I.L. Markov. Crisp: Congestion reduction by iterated spreading during placement. In *Computer-Aided Design - Digest of Technical Papers, 2009. ICCAD 2009. IEEE/ACM International Conference on*, pages 357–362, Nov 2009.

[136] Jarrod A. Roy, James F. Lu, and Igor L. Markov. Seeing the forest and the trees: Steiner wirelength optimization in placemen. In *Proceedings of the 2006 International Symposium on Physical Design*, ISPD '06, pages 78–85, New York, NY, USA, 2006. ACM.

[137] Debashis Sahoo, Subramanian Iyer, Jawahar Jain, Christian Stangier, Amit Narayan, David L. Dill, and E. Allen Emerson. A partitioning methodology for bdd-based verification. In *In Formal Methods in Computer-Aided Design, volume 3312 of Lecture Notes in Computer Science*, pages 399–413. Springer, 2004.

[138] S.M. Sait, M.R. Minhas, and J.A. Khan. Performance and low power driven vlsi standard cell placement using tabu search. In *Evolutionary Computation, 2002. CEC '02. Proceedings of the 2002 Congress on*, volume 1, pages 372–377, May 2002.

[139] Peter Sanders and Christian Schulz. Engineering multilevel graph partitioning algorithms. *CoRR*, abs/1012.0006, 2010.

[140] M. Sarrafzadeh and Maogang wang. Nrg: global and detailed placement. In *Computer-Aided Design, 1997. Digest of Technical Papers., 1997 IEEE/ACM International Conference on*, pages 532–537, Nov 1997.

[141] C. Sechen and A. Sangiovanni-Vincentelli. The timberwolf placement and routing package. *IEEE Journal of Solid-State Circuits*, 20(2):510–522, Apr 1985.

[142] Navaratnasothie Selvakkumaran and George Karypis. Theto - a fast and high-quality partitioning driven global placer. Technical report, in university of minnesota - computer science and engineering technical reports, 2003.

[143] Jonathan Richard Shewchuk. An introduction to the conjugate gradient method without the agonizing pain. `https://www.cs.cmu.edu/~quake-papers/painless-conjugate-gradient.pdf`.

[144] P. Spindler and F.M. Johannes. Fast and accurate routing demand estimation for efficient routability-driven placement. In *Design, Automation Test in Europe Conference Exhibition, 2007. DATE '07*, pages 1–6, April 2007.

[145] Peter Spindler, Ulf Schlichtmann, and Frank M. Johannes. Abacus: Fast legalization of standard cell circuits with minimal movement. In *Proceedings of the 2008 International Symposium on Physical Design*, ISPD '08, pages 47–53, New York, NY, USA, 2008. ACM.

[146] Weijie Su, Stephen Boyd, and Emmanuel Candes. A differential equation for modeling nesterovs accelerated gradient method: Theory and insights. In *Advances in Neural Information Processing Systems*, pages 2510–2518, 2014.

[147] Ar Trifunovic and William J. Knottenbelt. Parkway 2.0: A parallel multilevel hypergraph partitioning tool. In *in Proc. 19th International Symposium on Computer and Information Sciences*, pages 789–800. Springer, 2004.

[148] G. Trivedi and H. Narayanan. Application of fast dc analysis to partitioning hypergraphs. In *Circuits and Systems, 2007. ISCAS 2007. IEEE International Symposium on*, pages 3407–3410, May 2007.

[149] Ren-Song Tsay and J. Koehl. An analytic net weighting approach for performance optimization in circuit placement. In *Design Automation Conference, 1991. 28th ACM/IEEE*, pages 620–625, June 1991.

[150] Ren-Song Tsay, E.S. Kuh, and Chi-Ping Hsu. Proud: a sea-of-gates placement algorithm. *Design Test of Computers, IEEE*, 5(6):44–56, Dec 1988.

[151] Kalliopi Tsota, Cheng-Kok Koh, and Venkataramanan Balakrishnan. Guiding global placement with wire density. In *Proceedings of the 2008 IEEE/ACM International Conference on Computer-Aided Design*, ICCAD '08, pages 212–217, Piscataway, NJ, USA, 2008. IEEE Press.

[152] H. Vaishnav and M. Pedram. Pcube: A performance driven placement algorithm for low power designs. In *Design Automation Conference, 1993, with EURO-VHDL '93. Proceedings EURO-DAC '93., European*, pages 72–77, Sep 1993.

[153] B. Vastenhouw and R. Bisseling. A two-dimensional data distribution method for parallel sparse matrix-vector multiplication. *SIAM Review*, 47(1):67–95, 2005.

[154] N. Viswanathan and C.C.N. Chu. Fastplace: efficient analytical placement using cell shifting, iterative local refinement,and a hybrid net model. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 24(5):722–733, May 2005.

[155] N. Viswanathan, Gi-Joon Nam, C.J. Alpert, P. Villarrubia, Haoxing Ren, and C. Chu. Rql: Global placement via relaxed quadratic spreading and linearization. In *Design Automation Conference, 2007. DAC '07. 44th ACM/IEEE*, pages 453–458, June 2007.

[156] N. Viswanathan, Min Pan, and C. Chu. Fastplace 3.0: A fast multilevel quadratic placement algorithm with placement congestion control. ASP-DAC '07, pages 135–140, Washington, DC, USA, 2007. IEEE Computer Society.

[157] Natarajan Viswanathan, Charles J. Alpert, Cliff Sze, Zhuo Li, Gi-Joon Nam, and Jarrod A. Roy. The ispd-2011 routability-driven placement contest and benchmark suite. In *Proceedings of the 2011 International Symposium on Physical Design*, ISPD '11, pages 141–146, New York, NY, USA, 2011. ACM.

[158] C. Walshaw and M. Cross. Mesh partitioning: A multilevel balancing and refinement algorithm. *SIAM J. Sci. Comput.*, 22(1):63–80, January 2000.

[159] Chris Walshaw. Walshaw graph partitioning benchmarks. `http://staffweb.cms.gre.ac.uk/~wc06/partition/#graphs`.

[160] Maogang Wang, Xiaojian Yang, and M. Sarrafzadeh. Dragon2000: standard-cell placement tool for large industry circuits. In *Proceedings of IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, pages 260–263, 2000.

[161] Qingzhou Ben Wang, J. Lillis, and S. Sanyal. An lp-based methodology for improved timing-driven placement. In *Design Automation Conference, 2005. Proceedings of the ASP-DAC 2005. Asia and South Pacific*, volume 2, pages 1139–1143 Vol. 2, Jan 2005.

[162] Yao wen Chang, Zhe wei Jiang, and Tung-Chieh Chen. Essential issues in analytical placement algorithms, 2007.

[163] S. Wichlund and E.J. Aas. On multilevel circuit partitioning. In *Computer-Aided Design, 1998. ICCAD 98. Digest of Technical Papers. 1998 IEEE/ACM International Conference on*, pages 505–511, Nov 1998.

[164] Carl Sechen William Swartz. Timing driven placement for large standard cell circuits. *Design Automation, 1995. DAC '95. 32nd Conference on*, pages 211–215, 1995.

[165] Michael M. Wolf, Erik G. Boman, and Bruce A. Hendrickson. Optimizing parallel sparse matrix-vector multiplication by corner partitioning. In *in PARA08*, 2008.

[166] Yu-Liang Wu, Chak-Chung Cheung, David Ihsin Cheng, and Hongbing Fan. Further improve circuit partitioning using gbaw logic perturbation techniques. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 11(3):451–460, June 2003.

[167] Zhong Xiu and Rob A. Rutenbar. Timing-driven placement by grid-warping. In *Proceedings of the 42Nd Annual Design Automation Conference*, DAC '05, pages 585–591, New York, NY, USA, 2005. ACM.

[168] Haixia Yan, Qiang Zhou, and Xianlong Hong. Efficient thermal aware placement approach integrated with 3d dct placement algorithm. In *Quality Electronic Design, 2008. ISQED 2008. 9th International Symposium on*, pages 289–292, March 2008.

[169] Honghua Yang and D.F. Wong. Efficient network flow based min-cut balanced partitioning. In *Computer-Aided Design, 1994., IEEE/ACM International Conference on*, pages 50–55, Nov 1994.

[170] Xiaojian Yang, B.-K. Choi, and M. Sarrafzadeh. A standard-cell placement tool for designs with high row utilization. In *Computer Design: VLSI in Computers and Processors, 2002. Proceedings. 2002 IEEE International Conference on*, pages 45–47, 2002.

[171] Xiaojian Yang, B.-K. Choi, and M. Sarrafzadeh. Timing-driven placement using design hierarchy guided constraint generation. In *Computer Aided Design, 2002. ICCAD 2002. IEEE/ACM International Conference on*, pages 177–180, 2002.

[172] Chingwei Yeh, Yin-Shuin Kang, Shan-Jih Shieh, and Jinn-Shyan Wang. Layout techniques supporting the use of dual supply voltages for cell-based designs. In *Design Automation Conference, 1999. Proceedings. 36th*, pages 62–67, 1999.

[173] Hongyuan Zha, Xiaofeng He, Chris Ding, Horst Simon, and Ming Gu. Bipartite graph partitioning and data clustering. In *Proceedings of the Tenth International Conference on Information and Knowledge Management*, CIKM '01, pages 25–32, New York, NY, USA, 2001. ACM.

[174] Yanheng Zhang and C. Chu. Crop: Fast and effective congestion refinement of placement. In *Computer-Aided Design - Digest of Technical Papers, 2009. ICCAD 2009. IEEE/ACM International Conference on*, pages 344–350, Nov 2009.

[175] Zoltan. Hypergraph partitioning tool. `http://www.cs.sandia.gov/zoltan/`.