CHAPTER 2

VLSI FLOORPLANNING AND GLOBAL OPTIMIZATION TECHNIQUES

2.1 VLSI FLOORPLANNING

Floorplanning is becoming more and more important in VLSI physical design, as circuit size grows rapidly and now hierarchical design with IP modules is widely used to reduce the design complexity. Many floorplanning problems are NP-complete, so most floorplanning algorithms adopt either analytical force directed methods or perturbations with random searches and heuristics. Because the efficiency and effectiveness of these operations rely on the expression of the geometrical relation between circuit blocks, floorplanning representation becomes a fundamental issue. The redundancy of the representations and the complexity of the transformation between a representation and its corresponding floorplan can determine the execution time and the quality of the results.

The classical floorplanning formulation determines the layout of a given set of modules, such that no modules overlap and the enclosing layout region has minimum area while satisfying aspect ratio constraints. Each module can be rectangular or irregular with various aspect ratios i.e. the ratio of the height to the width of a module. Modules which have fixed aspect ratio and pin locations are called hard modules. On the other hand, soft modules have fixed area with continuously variable aspect ratio in a given bound.
2.1.1 Floorplanning Model

A VLSI circuit consists of a collection of variable-dimension modules interconnected by nets, each module with its own prescribed fixed area. The floorplanning problem is to determine the positions and dimensions of the circuit modules on a chip to optimize the circuit performance such that all the modules are enveloped in the rectangular floorplan. The floorplanning problem is described below:

The inputs for a floorplanning problem are given as follows:

1. a set $B$ of $n$ rectangular modules $B = \{1, 2, \ldots, i, \ldots, n\}$.
2. an interconnection matrix $C_{n \times n} = [c_{ij}]$, $1 \leq i, j \leq n$, where $c_{ij}$ indicates the connectivity between modules $i$ and $j$.
3. a list of $n$ tuples $(h_1, w_1), \ldots, (h_i, w_i), \ldots, (h_n, w_n)$, where $h_i$ and $w_i$ are the height and width of the module $i$ respectively.

The aspect ratio of module $i$ is given by $h_i / w_i$.

The required outputs are as follows:

1. there is no overlap between the modules.
2. $w_i \times h_i = A_i$, $1 \leq i \leq n$.
3. $x_i \geq w_i$ and $y_i \geq h_i$, $1 \leq i \leq n$, where $x_i$ and $y_i$ are the dimensions of basic rectangle $i$.
4. all the modules are enveloped in the floorplan and the total area and wirelength are optimized.

In the following paragraphs some commonly used techniques for the estimation of wirelength required by a given placement are presented.
2.1.2 Wirelength Estimation

The techniques of wirelength estimation can be classified into:

- **Half-perimeter method**: This is an efficient and the most widely used method to estimate the wirelength of a net. This method is used to find the smallest bounding rectangle that encloses all the pins of the net to be connected. The estimated wirelength of the interconnects is half the perimeter of this bounding rectangle.

- **Complete graph method**: For an n pin net, the complete graph has \( \frac{n(n-1)}{2} \) edges. Since a tree has \( (n-1) \) edges which is \( \frac{2}{n} \) times the number of edges in the complete graph, the estimated tree length \( (L) \) using this method is,

\[
L = \frac{2}{n} \times \sum_{\text{pair} \in \text{net}} \text{(pair separation)}
\]

- **Minimum chain method**: The method is to start from one vertex and connect to the closest one, and then to the next closest, in a certain sequence, until all the vertices are included. The estimation technique is simple but takes more interconnects.

- **Source to sink connection method**: One of the modules is assumed to be the source and the rest to be the sinks. All the sinks are connected to the source by separate wires. The total wirelength of this method is high and therefore it is suitable for heavily congested placement and floorplanning.
• **Spanning tree estimation method**: A spanning tree of a connected graph is a subgraph which is a tree and contains all the vertices of the graph. In an n-pin net, the algorithm searches the distances between all potential pairs of pins and connects the smallest (n-1) edges without cycles to form the spanning tree.

• **Steiner tree estimation**: A Steiner tree is a tree in a distance graph which spans a given subset of vertices with the minimal total distance on its edges. The Steiner tree problem is to find a minimum Steiner tree, i.e., a Steiner tree of minimum length. This problem is known as the Steiner tree problem and finding the minimum Steiner tree is known to be NP-complete. A Steiner tree is the shortest route for connecting a set of pins in modules. In an n-pin net, the net is able to start from any pin along its length to connect to other pins of the net to form a Steiner tree.

### 2.2 LITERATURE SURVEY

Several approaches have been reported to tackle the floorplanning problem. The reported approaches belong to three general classes: (1) constructive, (2) iterative, and (3) knowledge-based (Sait and Youssef 1995). Constructive algorithms attempt to build a feasible solution by starting from a seed module; then other modules are selected one (or a group) at a time and added to the partial floorplan. This process continues until all modules have been selected. The approaches that fall into this class are cluster growth, partitioning and slicing, connectivity clustering, mathematical programming, and rectangular dualization. Iterative methods start from an initial floorplan. Then this floorplan undergoes a series of perturbations until a feasible
floorplan is obtained or no more improvements can be achieved. Iterative techniques which have been applied to floorplanning are simulated annealing, force directed, and genetic algorithm. In knowledge-based approach, a knowledge expert system is implemented which consists of three basic elements: (a) a knowledge base that contains data describing the floorplan problem and its current state, (b) rules stating how to manipulate the data in the knowledge base in order to progress toward a solution, and (c) an interference engine controlling the application of the rules to the knowledge base.

2.2.1 Cluster Growth

In this approach, the floorplan is constructed in a greedy fashion, one module at a time until each module is assigned to a location of the floorplan. A seed module is selected and placed into a corner of the floorplan (lower left corner). Then, the remaining modules are selected one at a time and added to the partial floorplan, while trying to grow evenly on upper, diagonal and right sides simultaneously and maintaining aspect ratio constraint on the chip. To determine the order in which modules should be selected, the modules are initially organized into a linear order. Linear ordering is one of the most widely used techniques for constructively building an initial placement configuration (Kang 1983).

2.2.2 Partitioning and Slicing

One of the early approaches to floorplanning is recursive bipartition to construct a floorplan tree. The min-cut algorithms such as KL algorithm (Kernighan and Lin 1970) and FM algorithm (Fiduccia-Mattheyses 1982) are applicable. Kernighan and Lin (1970) proposed a two-way graph partitioning algorithm. Fiduccia and Mattheyses (1982) obtained a faster implementation
of KL with the help of a data structure, called the bucket data structure. The idea is to minimize the number of nets crossing two halves of the bipartition. Four way partitioning can also be used instead of bipartition. Min cut algorithms of Lauther (1979) and Dunlop and Kernighan (1985) provide good heuristics for minimizing wiring density and have therefore been frequently used for placement problems.

2.2.3 Connectivity Clustering

In Dai et al (1989), a hierarchical floorplanner for arbitrary size rectangular blocks using the clustering approach has been proposed. The clustering algorithm proceeds bottom-up using a combined criterion of connecting and geometry. Each node in the cluster tree represents a cluster of atmost five cells. Wang and Wong (1992) presented an optimal algorithm for a special class of floorplans called hierarchical floorplans of order 5. This algorithm takes a set of implementations for each block as input and identifies the best implementation for each block so that the resulting floorplan has minimum area.

2.2.4 Mathematical Programming

Yamada and Liu (1988) described an analytical method to determine the optimal module orientation in terms of estimated total wirelength. They solved a set of linear equations to obtain a close approximation to the optimal solution. Sutanthavibul et al (1991) provided a mixed integer programming based method for general floorplan design and optimization. Their formulation can describe arbitrary combinations of rigid and flexible modules. Constraints on interconnection length and routability are also considered in this paper.

Onodera et al (1991) presented a branch-and-bound placement technique for building block layout that effectively searches for an optimal placement in the whole solution space. Constraints on critical nets and the shape of a resulting chip can be taken into account in the search process. However, the solution obtained is only near-optimal.

Rosenberg (1989), Moh et al (1996), Chen and Fan (1998) showed that the floorplan area minimization problem can be formulated as a convex programming problem. Convex programming problem enjoys the property that any local solution is also global. Chen and Fan (1998) reduced the number of variables and constraints from the convex formulation provided by Moh et al (1996). Although these methods find solution in global optimal, however, the placement topology considered in these papers are limited to the slicing structure. Young et al (2000) used the Lagrangian relaxation to solve the area minimization problem. Wang et al (2003) proposed Lagrangian relaxation for soft module floorplanning based on sequence pair (SP) algorithm.
2.2.5 Rectangular Dualization

Ohtsuki et al (1979) described polar graphs for floorplans. A rectangular floorplan consists of basic rectangles that can be represented as a rectangular dissection. Kozminski and Kinnen (1985) presented an algorithm for finding a rectangular dual of a planar triangulated graph. Usually, the graph is processed only if a rectangular dual for the graph exists. Bhaskar and Sahni (1988) extended to present a linear time algorithm for finding a rectangular dual of the planar triangulated graph.

Wimer et al (1988), Dong et al (1989) reported an iterative approach to floorplan design which makes use of the vertical and horizontal constraint graphs to determine the dimensions and positions of all blocks.

Vijayan and Tsay (1991) proposed the topological overlap removal method, that removes a redundant edge from the two critical paths and that is repeated continuously until it makes significant improvement on the layout area. However, the drawback to this approach was random choice, since if there were two or more edges which qualified for removing; it might not predict which one could lead to a better placement. Asato and Ali (1996) improved this method by removing all redundant edges from one of the constraint graphs after the iteration.

2.2.6 Force Directed Methods

In Quinn and Breuer (1979), and Goto (1981) presented force-directed placement algorithms which simulates the mechanics problem in which particles are attached to springs and their movement obeys Hooke’s Law. Youseef et al (1995) combined force directed approach with constraint graph approach. A floorplan solution is produced in two steps. First a timing
and connectivity driven topological arrangement is obtained using a force directed approach. In the second step, the topological arrangement is transformed into a legal floorplan to minimize the overall area of the floorplan.

2.2.7 Knowledge-Based Methods


2.2.8 Slicing Floorplan Representations

The vertical-horizontal (v-h) tree was one of the earliest representations (Otten 1982), for a floorplan with a slicing structure. For slicing floorplans, there is an efficient optimal algorithm to solve the floorplan area optimization problem in polynomial time (Stockmeyer 1983). However, in the same paper for non-slicing floorplans this problem has been proven to be NP-complete.

Wong and Liu (1986) proposed a normalized Polish expression to represent a slicing structure, enabling the speed up of the search procedure. In Young and Wong (1997), it has been shown theoretically that slicing floorplans can pack modules tightly. Lin et al (2004) developed Genetic Algorithm (GA) to efficiently handle the fixed-die floorplanning based on the normalized polish expression. Slicing floorplan has advantages but typical floorplan is non-slicing. First, slicing floorplans significantly reduces the search space and this leads to fast runtime. Second, the shape flexibility of the
soft modules can be fully exploited to pack modules tightly using an efficient shape curve computation technique (Otten 1983).

Cong et al (2004) defined a simple slicing algorithm that guarantees a zero-dead space floorplan while attempting to minimize the maximum block aspect ratio. The analysis shows that the block aspect ratios obtained are uniformly bounded in terms of the area variation of the given blocks. The main limitation of this work is that it does not consider wirelength.

### 2.2.9 Non-slicing Floorplan Representations

For a non-slicing floorplan, in the mid 1990’s, two efficient representations namely Sequence Pair (SP) (Murata et al 1995), and the Bounded Sliceline Grid (BSG) (Nakatake et al 1996) were proposed. These two approaches are different representations but both are based on constraint graph to manipulate the transformation between the representation and their placement. Murata et al (1995) provided a P-admissible solution space, in which each code is a pair of module name sequences. By searching this space, it is possible to pack hundreds of modules efficiently. Nakatake et al (1996) presented BSG structure which utilizes a set of horizontal and vertical bounded-length lines to cut the plane into rooms and represents a placement by these lines and rooms but it incurs redundancies.

Guo et al (1999) proposed an ordered tree (O-tree) representation structure to represent non-slicing floorplans. An O-tree represents partial topological information, which together with the dimensions of all the blocks describes an exact floorplan. Chang et al (2000) presented an efficient, flexible, and effective data structure, B*-trees, for non-slicing floorplans. B*-trees are based on ordered binary trees and are very easy for implementation.
They developed a B*-tree based simulated annealing scheme for floorplan design.

Hong et al (2000) proposed a Corner Block List (CBL) representation for non-slicing floorplan. CBL defines the floorplan independent of the block sizes. Thus the structure is better suited for floorplan optimization with various size configurations of each block.

Adya and Markov (2001) studied the fixed-outline floorplan formulation that is more relevant to hierarchical design style and is justified for very large ASICs and SoCs. Based on sequence pair they suggested new objective functions to drive Simulated Annealing (SA) and new types of moves that better guide the local search for fixed-outline floorplanning.

Adya and Markov (2002) discusses the flow which relies on an arbitrary black-box standard cell placer to obtain an initial placement and then removes possible overlaps using a fixed-outline floorplanner. This result in valid placements for macros, which are considered fixed. Adya and Markov (2003) addressed the wirelength improvements and optimization of aspect ratios of soft blocks on slack computation.

Lin and Chang (2004) proposed a Transitive Closure Graph (TCG) based representation for non-slicing floorplans. TCG is a very flexible representation and constructs constraint graphs to evaluate their packing cost. Lin and Chang (2005), proposed TCG-S representations for non-slicing floorplans. TCG-S is a hybrid of TCG and sequence-pair, which contains a constraint graph from TCG and a sequence from sequence pair.

Chen and Chang (2005, 2006) studied fixed-outline floorplanning and developed B*-tree floorplan representation based on fast three-stage
simulated annealing scheme. Their work included wirelength optimization for fixed-outline floorplanning constraints. They have shown that their results show significant speedups over the classical SA scheme and better solution quality.

Lee et al (2007) presented a multilevel floorplanning framework based on the B*-tree representation, called MB*-tree, to handle the floorplanning and packing for large-scale building modules. The MB*-tree adopts a two-stage technique, clustering followed by declustering. The clustering stage iteratively groups a set of modules based on a cost metric guided by area utilization and module connectivity, and at the same time establishes the geometric relations for the newly clustered modules by constructing a corresponding B*-tree. The declustering stages iteratively ungroup a set of previously clustered modules and then refine the floorplanning solutions by using a simulated annealing scheme.

Luo et al (2007) presented a non-linear optimization based methodology for fixed-outline floorplanning. They proposed a two-stage non-linear optimization based methodology specifically designed to perform fixed-outline floorplanning by minimizing wirelength while simultaneously enforcing aspect ratios constraints on soft modules and handling a zero deadspace situation.

### 2.3 STOCHASTIC OPTIMIZATION TECHNIQUES

Optimization algorithms are search methods, where the goal is to find a solution to an optimization problem, such that a given quantity is optimized, possibly subject to a set of constraints. Each optimization problem consists of an objective function, a set of variables and a set of constraints. An optimization algorithm searches for an optimum solution by iteratively
transforming a current candidate solution into a new, hopefully better solution. Optimization methods can be divided into two main classes, based on the type of solution that is located. Local search algorithms use only local information of the search space surrounding the current solution to produce a new solution. Since only local information is used, local search algorithms locate local optima. Popular hill-climbing techniques belong to this class. Such algorithms consistently replace the current solution with the best of its neighbors if it is better than the current. A global search algorithm uses more information about the search space to locate a global optimum. Optimization algorithms are further classified into deterministic and stochastic methods. Stochastic methods use random elements to transform one candidate solution into a new solution. Deterministic methods, on the other hand do not make use of random elements.

An important aspect that determines the efficiency and accuracy of an optimization algorithm is the exploration-exploitation trade-off. Exploration is the ability of a search algorithm to explore different regions of the search space in order to locate a good optimum. Exploitation, on the other hand, is the ability to concentrate the search around a promising area in order to refine a candidate solution. A good optimization algorithm optimally balances these contradictory objectives.

In the following sections stochastic algorithms such as Simulated Annealing (SA), Genetic Algorithm (GA), Particle Swarm Optimization (PSO), and Differential Evolution (DE) have been discussed.

2.3.1 Simulated Annealing (SA)

Simulated annealing algorithm (Kirkpatrick et al 1983) has been one of the most popular optimization algorithms used in the VLSI CAD field
for the past three decades. It has been applied to almost every step of the VLSI design process, including scheduling, resource allocation, logic synthesis, partitioning, floorplanning and placement. SA resembles the cooling process of molten metal through annealing. At high temperature, the atoms in the molten metal can move freely with respect to each another, but as the temperature is reduced, the movement of the atoms gets restricted. The atoms start to get ordered and finally form crystals having the minimum possible energy. However, the formation of the crystal mostly depends on the cooling rate. If the temperature is reduced at a very fast rate, the crystalline state may not be achieved at all instead, the system may end up in a polycrystalline state, which may have a higher energy state than the crystalline state. Therefore in order to achieve the absolute minimum energy state, the temperature needs to be reduced at a slow rate. The process of slow cooling is known as annealing in metallurgical parlance.

The floorplanning algorithm using simulated annealing is easy to implement and has been successfully used in many floorplanning problems. (Wong and Liu 1986, Murata et al 1998, Chang et al 2000, Tang et al 2001, Lin and Chang 2004, Lin and Chang 2005, Chan et al 2005, Lee et al 2003). The major drawback of simulated annealing is that it requires substantial computation resources. Fine tuning the parameters requires many trial runs and is a time consuming process. Often, no single set of parameters is suitable for all the inputs and it may require repeated adjustments to obtain a good solution. Another problem is that in the simulated annealing algorithm it is possible to be close to an optimal solution but it is not fixed. If the parameters are changed slightly, a completely different solution may be found. One of the attractive features of simulated annealing is that the solution quality has room for improvement as Central Processing Unit (CPU) speed increases. The parameters can be set for a slower annealing rate to obtain a better solution
within the same time. Such advantages cannot be obtained with non statistical approaches without major changes in their software programs.

However, the efficiency of simulated annealing depends much on the energy landscape. If the global minimum solution has a small basin of attraction and is well separated by many local minima with high energy barriers, simulated annealing tends to be trapped in a local minimum solution and is not able to explore the solution space efficiently. The reason is that the temperature cooling schedule used in practice in the simulated annealing process is usually faster than that required by the theory (exponential cooling is used in practice as opposed to logarithmic cooling required by the theory). The solution quality of the simulated annealing based optimization tends to get worse as the design complexity increases, as faster cooling schedules have to be used to assure a reasonable runtime.

### 2.3.2 Genetic Algorithm (GA)

Evolutionary algorithms (EA) are stochastic search methods that mimic the metaphor of natural biological evolution and the social behavior of species. The first evolutionary-based technique introduced in the literature was the genetic algorithms (GA) (Holland 1975). GA was developed based on the Darwinian principle of the ‘survival of the fittest’ and the natural process of evolution through reproduction. GAs work with a random population of solutions (chromosomes). The fitness of each chromosome is determined by evaluating it against an objective function. To simulate the natural survival of the fittest process, best chromosomes exchange information (through crossover or mutation) to produce offspring chromosomes. The offspring solutions are then evaluated and used to evolve the population if they provide better solutions than weak population members. Usually, the process is
continued for a large number of generations to obtain a best fit (near-optimum) solution (Goldberg 1989).


2.3.3 Particle Swarm Optimization (PSO)

PSO is a population-based search approach, modeled on the choreographic behavior of a bird flock (Kennedy and Eberhart 1995). Each individual is referred to as a particle and presents a candidate solution to the optimization problem. The search process can be described as particles being “flown” through a hyper dimensional search space, by means of adjustment of their positions in the search space. Each particle is adjusted to move closer to the best particle in a predefined neighborhood, as well as towards its own best
position found thus far (referred to as the personal best solution of that particle). The global best (gbest) PSO is an implementation where the neighborhood is the entire swarm while local best (lbest) PSO refers to implementations where a smaller neighborhood size is used (Shi and Eberhart 1999). A smaller neighborhood size usually leads to slower convergence, but increases diversity. That is, a larger part of the search space is covered for smaller neighborhoods.

Yamada et al (2003) applied PSO to the optimization of the layout of manufacturing cells to reconfigurable manufacturing systems, while also optimally positioning transport robots to minimize the distance traveled by robots transporting materials.

Venu Gudise and Ganesh Venayagamoorthy (2004) presented particle swarm optimization (PSO) for FPGA placement and routing. They have presented the results for the implementation of an arithmetic logic unit on a Xilinx FPGA to minimize the interconnection lengths between the Configurable Logic Blocks (CLBs) using PSO.


Angeline (1998) provided the first approach to combine GA concepts with PSO showing that PSO performance can be improved for certain classes of problems by adding a selection process similar to that which occurs in evolutionary algorithms. A varying number of mutation operators
have been applied to PSO, mainly to improve exploration abilities. For continuous-valued search spaces, mutation involves adding a random value, sampled from some distribution, to the velocity or position vectors. For discrete valued spaces, elements of the position vector is swapped or replaced by a value, selected randomly from the finite domain of the problem variables (Kennedy et al 2001, Engelbrecht 2005).

2.3.4 Differential Evolution (DE)

Differential Evolution (DE) is a simple but powerful, evolutionary inspired search technique for global optimization. The DE algorithm was introduced by Storn and Price (Storn and Price 1995). It resembles the structure of a GA, but differs from GA, in its generation of new candidate solutions and by its use of a greedy selection scheme.

DE has been applied to several engineering problems in different areas (Storn and Price 1997, Storn 1999). In Nurhan Karaboga (2005), DE is applied to the design of digital IIR filters and its performance is compared to that of a genetic algorithm. In Nasimul Noman and Hitoshi (2005), DE algorithm has been hybridized with crossover based local search strategies in an attempt to accelerate the convergence velocity of DE so that better solutions can be obtained with higher speed and increased robustness.