A Heuristic Approach for VLSI Floorplanning

Rajalakshmi.P,
PG Student,
Department of Electronics and Communication Engineering,
Sri Krishna College of Technology, Coimbatore,
Tamil Nadu, India

Senoj Joseph,
Assistant Professor,
Department of Electronics and Communication Engineering,
Sri Krishna College of Technology, Coimbatore,
Tamil Nadu, India

ABSTRACT
Floorplanning is an essential step in VLSI chip design automation. The main objective of the floorplanning is to find a floorplan such that the cost is minimized. This is achieved by minimizing the chip area and interconnection cost. It determines the performance, size, yield and reliability of VLSI chips. We propose a Memetic Algorithm (MA) for non-slicing and hard module VLSI floorplanning problem. This MA is a hybrid genetic algorithm that uses effective genetic search method to explore the search space and an efficient local search method to exploit information in the search region. The exploration and exploitation are balanced by threshold bias search strategy. MA works better than the existing algorithms and is efficient, faster and cost effective algorithm. A better floorplan with minimal chip area and interconnection cost will be obtained using the MA for non-slicing and hard module VLSI floorplanning problem.MA is mainly used to produce optimal or near optimal solution. The experimental results are analyzed to check the performance of MA.

Keywords
Floorplanning, Genetic Algorithm (GA), Local search, Memetic algorithm (MA), Very Large Scale Integrated circuit (VLSI), Ordered Tree (O-tree) Representation, Depth First Search (DFS) Algorithm.

1. INTRODUCTION
Floorplanning is a mapping between the logical description (netlist) and the physical description (floorplan).It plays an important role in both VLSI and Application Specific Integrated Circuits (ASIC) design. The process of arranging the blocks of the netlist on the chip is called as Floorplanning.

The VLSI floorplanning goal is to find a floorplan for a given set of modules at minimum cost such that no module overlaps with another and the interconnections between the modules are minimized[10].It is easy to deal with layout when structural detail at lowest abstraction is available, one knows the exact number of transistors in the circuit and the way they are interconnected.

When this type of structural information is not available, one can estimate the area to be occupied by various sub blocks and together with a precise or estimated interconnection pattern, try to allocate distinct regions of the integrated circuit to the specific sub blocks. This process is called as Floorplanning. It is an important step in VLSI design automation as it determines the performance, size, yield, and reliability of VLSI chips. The floorplan representation determines the size of the floorplan and the complexity of transformation between a representation and its corresponding floorplan[1].Slicing representation and Non-slicing representation are the two categories of existing floorplan representations [7]. In general, the non-slicing representations can contribute to better results than slicing representations. The Ordered tree (O-tree) representation proposed by Guo *et al*. is *o*ne of the most efficient non-slicing representations[2].The representation not only covers all optimal floorplans but also has a smaller search space. When designing genetic operators of evolutionary algorithms, geometrical relations among modules are required. This representation is useful in identifying the blocks Therefore, the O-tree representation is made use of in this paper.

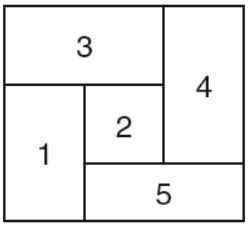


Fig 1: Non-Slicing Floorplan

Local search and Global search are the two existing search methods for the VLSI floorplanning problem. Though the local search methods are efficient, they may not be able to produce an optimal solution sometimes as their search may be trapped in minimal points of the local region. A widely used global search method for VLSI floorplanning problems is genetic algorithm (GA). GA has been successfully applied to solve slicing VLSI floorplanning problems.GA can also be applied for non-slicing VLSI floorplanning, but the performance of the GA is not satisfactory [1].

In this paper, a Memetic Algorithm (MA) is proposed[6] for a non-slicing and hard module VLSI floorplanning problem. MAs are population-based metaheuristic search methods[7].

The MA is a hybrid GA that uses an effective genetic search method to explore the search space and an efficient local search method to exploit information in the search region[1].

2. PROBLEM DESCRIPTION
Given a set of blocks $B = \{b_1, b_2, \ldots, b_n\}$. Each block B_i is rectangular and has fixed width and height. A module m_i is a rectangular block with fixed height h_i and width w_i,

$M = \{m_1, m_2, \ldots, m_n\}$ is a set of modules, and N is a net list specifying interconnections between the modules in M.

The objectives of floorplan optimization problem are to minimize the area of B and reduce wire lengths of interconnects provided no pair of blocks overlaps. Each block Bi is associated with a two tuple (h_i, w_i), where h_i *and* w_i denote the width and height of B_i, respectively. The area A_i of the block B_i is given by $h_i * w_i$, We consider only the placement of hard modules. A hard module is not flexible in its shape, but free to rotate.

A floorplan F is an assignment of M onto a plane such that no module overlaps with another. A floorplan has an area cost, i.e., *Area*(F), which is measured by the area of the smallest rectangle enclosing all the modules and an interconnection cost, i.e., *Wirelength*(F), which is the total length of the wires that are used for the interconnections specified by N. To minimize the costs, a module can be rotated 90◦. The cost of a floorplan F is defined as follows:

$$\text{Cost (F)} = w_1 \times \frac{Area(F)}{Area*} + w_2 \times \frac{Wirelength(F)}{Wirelength*}$$

In the above equation, $Area^*$ and *Wirelength** represent the minimal area and the interconnection costs, respectively[1]. The interconnection cost is the total wire length of all the nets, and the wire length of a net is calculated by the half perimeter of the minimal rectangle enclosing the centers of the modules that have a terminal of the net on it.

Given M and N, the objective of the floorplanning problem is to find a floorplan F such that *cost*(F) is minimized. A floorplan is "admissible" if no module can be shifted left or bottom without moving other modules in the floorplan. That is, it is a LB-compact placement. Given a feasible floorplan (a floorplan on which no module overlaps with another),we can derive an admissible floorplan by compacting the modules to the left and bottom boundaries of the floorplan. The cost of the admissible floorplan is equal to or less than that of the original floorplan. Therefore, the search space of the VLSI floorplanning problem is limited to admissible floorplans [1].

3. LITERATURE SURVEY

Pei-Ning Guo, Toshihiko Takahashi, Chung-Kuan Cheng proposed an Ordered tree (O tree) structure to represent non-slicing floorplans. The O tree uses only $n(2 + \lceil \lg n \rceil)$bits for a floorplan of rectangular blocks. This paper states the floorplanning problem, the descriptions of admissible placement and constraint graph, defines the properties and operations for O tree, presents a deterministic algorithm based on O tree. This paper result in better wire length while their chip area are comparable and the CPU time is much less [2].

Maurizio Rebaudengo, Matteo Sonza Reorda describes a Genetic Algorithm for the Floorplan Area Optimization problem. The algorithm is based on suitable techniques for solution encoding, effective cross-over and mutation operators, and heuristic operators which further improve the method's effectiveness. Experimental results how that the GA is competitive as far as the CPU time requirements and the result accuracy are considered [4].

Christine L. Valenzuela and Pearl Y. Wang presented a GA which uses a normalized postfix encoding scheme to solve the VLSI floorplanning problem. This paper describes a GA which uses a novel encoding to breed normalized postfix expressions for macro cell placement and area optimization in VLSI floorplan design [3].

Hisao Ishibuchi, Tadashi Yoshida and Tadahiko Murata proposed a paper that shows how the performance of EMO algorithms can be improved by hybridization with local search. The main advantage of the hybridization is that the convergence speed is improved and the main drawback is that the increase in the computation time per generation. Thus, the number of generations is decreased when the available computation time is limited. As a result, the global search ability of EMO algorithms is not fully utilized [6].

J.Cohoon, S.Hedge, W.Martin and D.Richards proposed a method to solve the floorplan design problem using distributed genetic algorithms. The genetic algorithm is modified slightly to make it distributed. A number of instances of the genetic algorithm are spawned and run independently an in parallel for a number of generations [5].

4. RELATED WORK

4.1 O-Tree Representation

An O tree can be encoded in a tuple (T, π), where T is a $2n$-bit string specifying the structure of the O-tree, and π is a permutation of the nodes. For each admissible placement, we can find an O-tree representation. Given an O- tree, it takes only linear time to construct the placement and its constraint graph.Generally, a tree contains a finite set of one or more nodes. There is one specially designated node called the root of the tree. The root has more branches that are directed edges pointed from the root to its children. An O tree is a rooted directed tree in which the order of the subtrees is important.

Two types of representations for an O-tree are Horizontal and Vertical O-tree Representations[8].The Horizontal O-tree Representation is considered in this paper.

4.2 Horizontal O-Tree Representation

A floorplan with n rectangular modules can be represented in a horizontal O tree of (n + 1) nodes, of which n nodes correspond to n modules $m_1, m_2, \ldots, m_n$ and one node corresponds to the left boundary of the floorplan. This is called the source node. The left boundary is a dummy module with zero width placed at $x = 0$.

In a horizontal O-tree, there exists a directed edge from module m_i to module m_j and only if $x_j = x_i + w_i$, where x_i is the x coordinate of the left bottom position of m_i, x_j is the x coordinate of the left-bottom position of m_j, and w_i is the width of m_i[1].

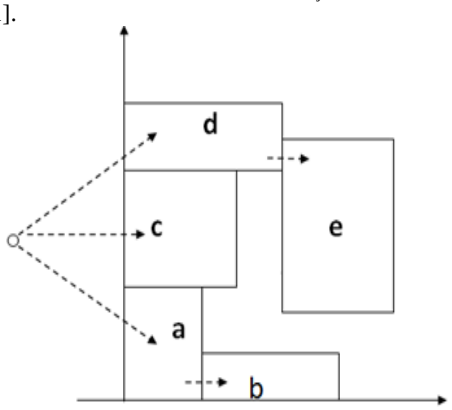


Fig 2:O-Tree Representation

For a horizontal O-tree, the tuple is obtained by depth-first traversing the nodes and edges of the O-tree. When visiting a node other than the root, we append the node to π. When visiting an edge in descending direction, we append a 0

to *T*, and when visiting an edge in ascending direction, we append a 1 to *T*. The root of the O tree represents the left boundary of the chip. Thus, we set its coordinate and its width. The children are on the right side of their parent with zero separation distance in coordinate. For each block let be the set of block with its order lower than in permutation and interval overlaps interval by a nonzero length. From an horizontal O-tree, we can find a placement by visiting the tree in DFS order. The permutation is the label sequence when we traverse the tree in DFS order. Fig 2 shows the Horizontal O-tree Representation.

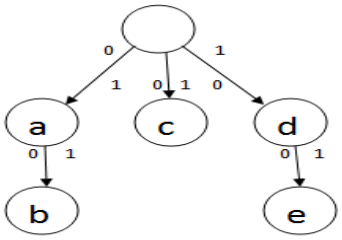


Fig 3:Encoding

For example, Fig 3shows the horizontal O-tree is encoded as (0011010011, *abcde*).The length of the bit string is 10.

4.3 Algorithm for Feasible Tree

Given the generated random number and the number of nodes as input, this algorithm will generate a specified bits of binary equivalent for the random number and detects whether the tree is feasible or not using Horizontal O-tree representation logic. The number of bits(b) generated is twice the number of nodes(n) specified as the input. For example, if n=2, then a 4 bit equivalent of generated random number is obtained. This 4 bit equivalent will have 2^4=16 possible combinations, out of which only few combinations are feasible. The number of nodes in tree is equal to the number of blocks to be placed in the Floorplan.

The examples for Feasible and Infeasible tree are shown below. Let the generated random number be 'r' and the number of nodes be 'n'. If r=702, n=5, then 1010111110, Infeasible tree is obtained. Suppose, if r=211,n=7,then 0011010011,Feasible tree is obtained.

4.4 DFS Algorithm Implementation

The major goal of the DFS Algorithm is to visit all the nodes of the tree only once [7]. In DFS, we follow the path as deeply as we go. When there is no adjacent vertex present, we traverse back and search for unvisited vertex. We should maintain a visited array to mark all the visited vertices. We developed a program to find the path traversal for more number of nodes. Hence, the DFS algorithm for the feasible tree is implemented successfully.

4.5 Maximum Area Calculation

Specifying the position, width and height of the blocks, the developed a program will find the total area required for placing the blocks. It should be placed based on the O-tree Representation and DFS algorithm implementation. It is implemented using concepts of DataStructures [8]. The total width is calculated by adding 'x axis' ('y axis') position and width (height) of individual block. For each individual block, the total width is calculated and the maximum value is the maximum width obtained after placing these blocks.

Maximum Area = Maximum Width x Maximum Height

5. PROPOSED WORK

The proposed method is a MA for a non-slicing and hard-module VLSI floorplanning problem. MAs are population-based metaheuristic search methods.

The MA is a hybrid GA that uses an effective genetic search method to explore the search space and an efficient local search method to exploit information in the search region. The exploration and exploitation are balanced by a novel bias search strategy[1].

MA employs a local search method for the VLSI floorplanning problem. The local search method is based on a deterministic algorithm. Optimization algorithm can be deterministic or probabilistic. Deterministic optimization algorithms are most often used if a clear relation between the characteristics of the possible solution and their utility for a given problem exists. This includes state space search, branch and bound, algebraic geometry. Probabilistic methods, on the other hand, may only consider those elements of the search space in further computations that have been selected by the heuristic.

Steps:

1. Start: Randomly generate a population of N chromosomes.
2. Fitness: Calculate the fitness of all chromosomes.
3. Create a new population:
    a. Selection: According to the selection method, select 2 chromosomes from the population.
    b. Local search: Search for the best chromosomes
    c. Crossover: Perform crossover on the 2 chromosomes selected.
    d. Local search: Search for the best chromosomes
    e. Mutation: Perform mutation on the chromosomes obtained with small probability.

5.1 Genetic Algorithm

GA is a search heuristic that mimics the process of natural evolution. This heuristic is routinely used to generate useful solutions to optimization and search problems. Genetic algorithms belong to the larger class of EAs, which generate solutions to optimization problems using techniques inspired by natural evolution, such as inheritance, mutation, selection, and crossover.

Genetic Algorithm is started with a **set of solutions** (represented by **chromosomes**) called **population**. Solutions from one population are taken and used to form a new population. This is motivated by a hope, that the new population will be better than the old one. Solutions which are selected to form new solutions (**offspring**) are selected according to their fitness - the more suitable they are the more chances they have to reproduce. This is repeated until some best solution is obtained. GA is used to perform global exploration among a population while heuristic methods are used to perform local exploitation around the chromosomes. The behaviors of GA are characterized by the balance between exploitation and exploration in the search space. The balance is affected by the strategy parameters such as population size, maximum generation, crossover probability and mutation probability.

Steps:

1. Start: Randomly generate a population of N chromosomes.
2. Fitness: Calculate the fitness of all chromosomes.
3. Create a new population:

a. Selection: According to the selection method implemented, select 2 chromosomes from the population.
b. Crossover: Perform crossover on the 2 chromosomes selected.
c. Mutation: Perform mutation on the chromosomes obtained.

4. Replace: Replace the current population with the new population.
5. Test: Test whether the termination condition is satisfied. If so, stop. If not, return the best solution in current population and go to Step 2.

A. Fitness Function: The Genetic Representation in MA considers each individual in the population is an admissible floorplan represented by an O-tree and encoded in a tuple (T, π), where T is a $2n$-bit string identifying the structure of the O-tree, and π is a permutation of the nodes[1].

The VLSI floorplanning is a minimization problem, and the objective is to minimize the cost of floorplan F, i.e., ***cost(F)***. Thus, the fitness of an individual (T, π) in the population is defined as follows:

$$f((T,\pi)) = \frac{1}{cost(F(T,\pi))}$$

where $F(T,\pi)$ is the corresponding floorplan of (T, π), and $cost(F(T,\pi))$ is the cost of F defined earlier.

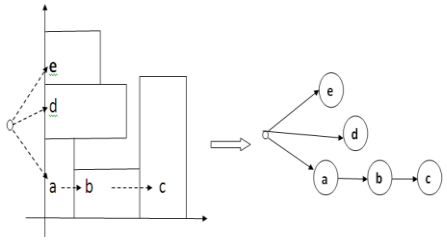
An individual in the initial population is an O-tree (T, π) representing an admissible VLSI floorplan F. A constructive algorithm is designed to construct an admissible O-tree. The algorithm starts with randomly generating a sequence of modules π. Then, it inserts the modules into an initially empty O-tree T in the randomly generated order. When inserting a module into T, it checks all external insertion positions for the module and inserts the module at the position that gives the best fitness.

B. Genetic Operators

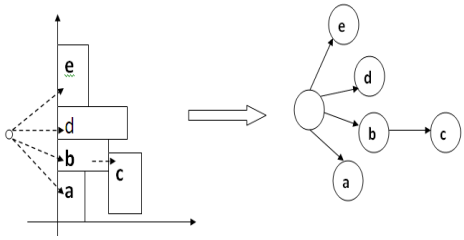
1) Role of the Genetic Operators in the MA: The role that the genetic operators play in our MA is different from that in GAs. In GAs, crossover is used for both exploration and exploitation, and mutation is used for exploration. In our MA, however, the crossover and mutation operators are only used for exploration, or discovering new promising search regions. The crossover and mutation operators discover new promising search regions by evolving memes. Memes can mutate through, for example, misunderstanding, and two memes can recombine to produce a new meme involving elements of each parent meme. It is observed that a subtree of the O-tree represents a compact placement of a cluster of modules. Hence, subtrees are used as memes in our MA. The memes are transmitted and evolved through one crossover operator and two mutation operators, which will be discussed in the following.

2) Crossover: Given two parents, both of which are admissible floorplans represented by an O-tree, the crossover operator transmits the significant structural information from two parents to a child. By recombining some significant structural information from two parents, it is hoped that better structural information can be created in the child.

To create a child $c1$ from two parents $p1$ and $p2$, the crossover randomly selects some top-level subtrees from $p1$, duplicates them, and puts them in $c1$. Then, the crossover operator takes a copy of $p2$ and removes those nodes that have been already present in $c1$ and then adds the remaining structural components to $c1$. In this way, the generated child carries the significant structural information from both $p1$ and $p2$. Fig. 4 indicates the basic idea behind the crossover operator. Fig. 4(a) and (b) are two parents, i.e., $p1$ and $p2$, and Fig. 4(c) is the child produced by the crossover operator.

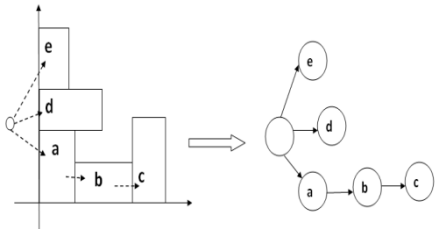


(a)



(b)

Fig 4(a), (b) : Crossover Operator



(c)

Fig 4(c): Crossover Operator

3) Mutation: The basic idea behind the mutation operators is to discover a new search region by mutating the structure of an individual.

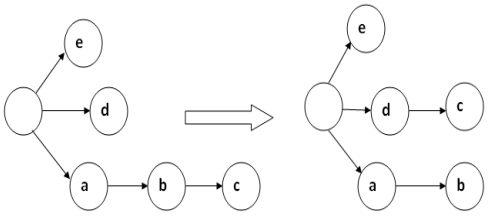


Fig 5: Mutation Operator

A mutation operator used by MA randomly selects a subtree at any level, removes it, and then inserts it back to the O-tree. Sometimes the mutated floorplan may not be admissible. Fig. 5 illustrates the basic idea behind the mutator. In the figure, it shows the initial O-tree and the mutated O-tree in which the subtree c is being moved.

The mutation operators are randomly selected and used by our MA to discover new search regions that have different fitness.

5.2 Local Search Method

The local search method is based on a deterministic algorithm proposed by Guo *et al.* [1].Given an initial floorplan encoded in an O-tree (T, π), the local search method finds a local optimal solution through systematically examining those O-trees that can be obtained by removing a module from, and then putting it back to the O-tree. When a module is added, its orientation may be changed if it leads to a smaller cost [1]. The algorithm is shown below.

1) For each node *mi* in (T, π):

a) delete *mi* from (T, π);

b) insert *mi* in the position where we can get the smallest most value among all possible insertion positions in (T, π) as an external node of the tree;

c) perform (a) and (b) on its orthogonal O-tree.

2) Output (T, π).

Strategy to Bias the Search

In our MA, we use a novel strategy to bias the search since the search space is huge. Instead of exploiting all the search points generated by the genetic operators, our MA only exploits those search points (admissible floorplans) whose fitness value is equal to or greater than a threshold *v* and ignores those search points whose fitness value is less than *v*. This basic idea behind the bias search strategy is illustrated in Fig. 5.

The threshold *v* is very important as determines the balance between the exploration and exploitation of our MA, which affects the computation time and the optimality of solutions. It should not be too big or too small value since the efficiency of our MA is dependent on this threshold value. Experimental results have shown that the threshold strategy is significant when compared with a random search strategy and the cost obtained by the MA using the threshold strategy is less than or equal to the cost obtained by the MA using the random picking up strategy.

6. EXPERIMENTAL RESULTS

The O-tree Representation, DFS Algorithm Implementation and Maximum Area Calculation is carried out in C programming language.

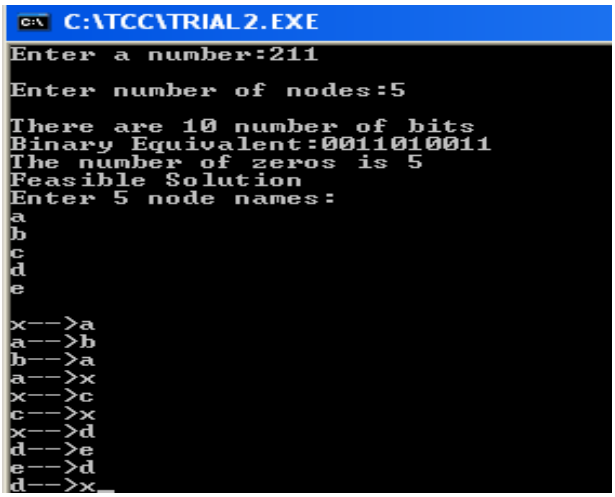


Fig 6: DFS algorithm implementation

Given a random number and number of nodes 'n' as inputs, 2n bits are generated and it is checked for feasibility. Finally, node traversal path is obtained in the output. Fig. 6 depicts the DFS algorithm implementation.

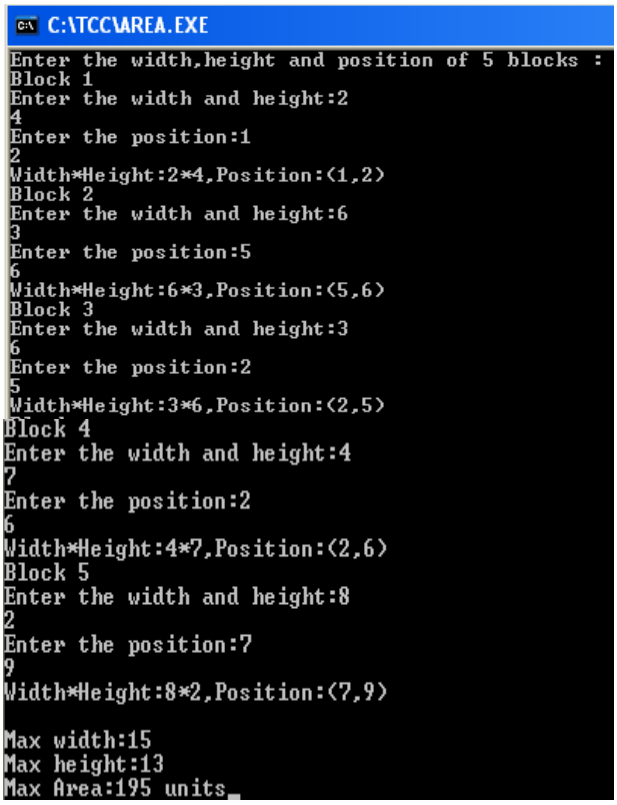


Fig 7: Maximum Area Calculation

The width, height and position of the blocks are specified as input and the maximum area occupied by the blocks is obtained as output. Fig 7 Depicts maximum area calculation.

7. CONCLUSION AND FUTURE WORKS

Based on the Horizontal O-tree Representation and DFS Algorithm, a program has been developed to check the feasibility of the tree and the path traversal of the nodes. The main advantage is its flexibility. It can be used for more number of nodes. Further, a program has been developed to find the maximum area occupied by the blocks.

The future work includes in extending this program to eliminate the block overlap. Finally, MA will be implemented to obtain a better floorplan with minimal chip area and interconnection cost.

8. REFERENCES

[1] Maolin Tang and Xin Yao, "A Memetic Algorithm for VLSI Floorplanning", IEEE transactions on systems, man, and cybernetics—part b: cybernetics, VOL. 37, NO. 1, 2007, pp. 62-69.

[2] Pei-Ning Guo, Toshihiko Takahashi, Chung-Kuan Cheng, "Floorplanning Using a Tree Representation", IEEE transactions on computer-aided design of integrated circuits and systems, VOL. 20, NO. 2, 2001, pp. 281-289.

[3] Christine L.Valenzuela and Pearl Y.Wang, "A Genetic Algorithm for VLSI Floorplanning".

[4] M. Rebaudengo and M. Reorda, "GALLO: A genetic algorithm for floorplan area optimization," IEEE Trans.

Comput.-Aided Design Integr.*Circuits Syst.*, vol. 15, no. 8, pp. 943–951, Aug. 1996

[5] J. Cohoon, S. Hegde, W. Martin, and D. Richards, "Distributed genetic algorithms for the floorplan design problem," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 10, no. 4, pp. 483–492, Apr. 1991.

[6] H. Ishibuchi, T. Yoshida, and T. Murata, 2003 "Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling,"

[7] Sabih H.Gerez, "Algorithms for VLSI Design Automation", John Wiley & Sons Inc., U.K, Second Edition.

[8] Naveed Sherwani, "Algorithms for VLSI Design Automation", John Wiley & Sons Inc., U.K, Second Edition.

[9] Anuradha A.Puntambekar, "Data Structures", Technical Publications, Pune.

[10] Michael John Sebastian Smith, "Application Specific Integrated Circuits", Pearson Edition.

[11] Maolin Tang, Raymond Y. K. Lau," A Parallel Genetic Algorithm for Floorplan Area Optimization", Seventh International Conference on Intelligent Systems Design and Applications,IEEE, 2007, pp. 801-806

[12] Guolong Chen, Wenzhong Guo, Yuzhong Chen, "A PSO-based intelligent decision algorithm for VLSI Floorplanning", Soft Computing, VOL. 14, NO. 12, Springer, 2009, pp. 1329-1337.

[13] Jianli Chen, Wenxing Zhu, and M. M. Ali, "A Hybrid Simulated Annealing Algorithm for Nonslicing VLSI Floorplanning", IEEE transactions on systems, man and cybernetics—part c: applications and reviews, VOL. 41, NO. 4, 2011, pp. 544-553.

[14] Guolong Chen, Wenzhong Guo, Yuzhong Chen, "A PSO-based intelligent decision algorithm for VLSI floorplanning", Springer, Soft Computing, 2010, pp. 1329–1337.

[15] C. Valenzuela and P.Wang, "VLSI placement and area optimization using a genetic algorithm to breed normalized postfix expressions," *IEEE Trans.Evol. Comput.*, vol. 6, no. 4, pp. 390–401, Aug. 2002.