International Journal of Electrical, Electronic and Communication Sciences
ISSN: 2517-9438
Vol:5, No:3, 2011

A Survey of Various Algorithms for Vlsi Physical Design

Rajine Swetha R, B. Shekar Babu, Sumithra Devi K.A

Abstract**—**Electronic Systems are the core of everyday lives. They form an integral part in financial networks, mass transit, telephone systems, power plants and personal computers. Electronic systems are increasingly based on complex VLSI (Very Large Scale Integration) integrated circuits. Initial electronic design automation is concerned with the design and production of VLSI systems. The next important step in creating a VLSI circuit is Physical Design. The input to the physical design is a logical representation of the system under design. The output of this step is the layout of a physical package that optimally or near optimally realizes the logical representation. Physical design problems are combinatorial in nature and of large problem sizes. Darwin observed that, as variations are introduced into a population with each new generation, the less-fit individuals tend to extinct in the competition of basic necessities. This survival of fittest principle leads to evolution in species. The objective of the Genetic Algorithms (GA) is to find an optimal solution to a problem .Since GA's are heuristic procedures that can function as optimizers, they are not guaranteed to find the optimum, but are able to find acceptable solutions for a wide range of problems. This survey paper aims at a study on Efficient Algorithms for VLSI Physical design and observes the common traits of the superior contributions.

Keywords**—**Genetic Algorithms, Physical Design, VLSI.

I. Introduction

A VLSI chip contains more than 100 million transistors. Electronic Design Automation (EDA) systems are able to simplify the extremely complex design process of VLSI chips by not exposing the low level circuit theory and device physics to the designer. This encapsulation allows the designer to concentrate on the functionality of the circuit and ways to optimize it. The typical figures of merit for VLSI systems are concerned with maximizing reliability and circuit speed while minimizing the size of the physical package, power consumption, etc.

Rajine Swetha R, Research Scholar, R.V.C.E, ECE Research Centre, Faculty, K.S Institute of Technology, Bangalore, Karnataka 560062, INDIA (phone: +91-9739402893, email: rajineswethar@gmail.com)
B. Shekar Babu, Research Scholar, R.V.C.E, ECE Research Centre, Bangalore, Karnataka, INDIA (email: digitalshekar@gmail.com)
Dr.Sumithra devi K.A, Director,Dept of MCA, R.V.College of Engineering ,Bangalore, Karnataka 560059, INDIA (email: sumithraka@gmail.com)

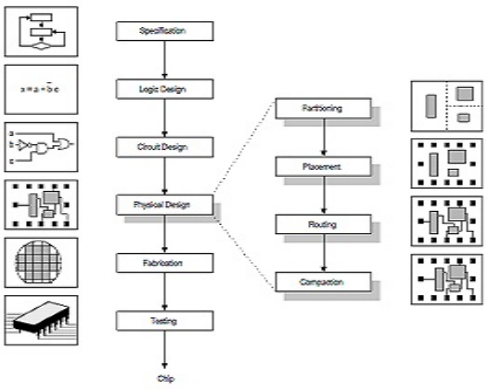

Fig. 1 Stages of Chip Design

In creating a VLSI system, a designer travels through six major steps as shown in figure 1.

Specification: It produces a functional specification of the system under design.

Logic Design: This transforms the functional specification into into a logical representation, typically via Boolean expressions.

Circuit Design: Represents the logic representation as a circuit using components from an available library of modules.

Physical Design: Translates the circuit into a physical package representation. This representation is specified through a set of of mask descriptions, which define how the individual layers of the integrated circuit are to be produced.

Fabrication: Uses the physical package representation to fabricate an actual integrated circuit.

Testing: Identifies the presence of manufacturing errors that prevent the integrated circuit from implementing the functional functional specification.[12] Genetic Algorithms are evolutionary computational models based on Charles Darwin's Darwin's theory of natural evolution based on the concept of the survival of the fittest [1]. The concept of natural selection is used to explain how species have been able to adapt to changing environments and how, consequently, species with similar adaptivity may have evolved. All genetic algorithms work on a population or a collection of several alternative solutions to the given problem. Each individual in the population is called a string or chromosome, in analogy to chromosomes in natural systems. The population size determines the amount of information stored by the GA. The GA population is evolved over a number of generations. All information required for the creation of phenotype and behavior patterns of a living organism is contained in its chromosomes.

GAs adopt two basic processes from evolution: inheritance, or the passing of features from one generation to the next, and competition, or survival of the fittest, which results in weeding

weeding out the bad features from individuals in the population.As the objective of the GA is to find an optimal solution to a problem .Since GA's are heuristic procedures, modeled as function optimizers, they are not guaranteed to find the most suitable solutions, but are able to find very good acceptable solutions for a wide range of problems [2].

II. Basic terminology

A. Hybrid Genetic Algorithm

A Hybrid Genetic Algorithm utilizes heuristics for improvement of offspring produced by crossover. Initial population is randomly generated. The offspring is obtained by crossover between two parents selected randomly. The layout improvement heuristics RemoveSharp and LocalOpt [13] are used to bring the offspring to a local maximum. If fitness of the layout of the offspring thus obtained is greater than the fitness of the layout of any one of the parents then the parent with lower fitness is removed from the population and the offspring is added to the population. If the fitness of the layout of the offspring is lesser than that of both of its parent then it is discarded. For mutation a random number is generated within one and if it is less than the specified probability of the mutation operator a layout is randomly selected and removed from the population. Its layout is randomized and then added to the population.

B. Genotype Representation

The phenotypic representation for the placement problems is basically the pattern that describes the position of the blocks. Binary slicing trees are well suited to represent placement patterns and have already been used in genetic algorithms [3],[12]. During recombination, partial arrangements of blocks are transmitted from parents to offspring. The corresponding operation is the inheritance of sub trees from the parents. Encoding the tree in a string complicates this operation, since the string needs to be decoded into the slicing tree to execute the recombination, then recoded into an offspring chromosome afterwards. There is no reason for using a string encoding except for the analogy to the natural evolution process, where the genetic information is encoded in a DNA string.

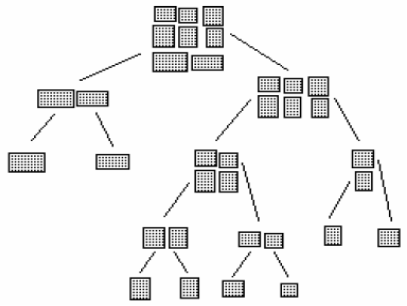

Fig. 2 Genotype

When directly using the slicing tree as the genotype representation, further decoding or encoding the tree when applying genetic operators is avoided. The genotype is encoded as a binary slicing tree, which defines the relative placement of the cells (fig.2). It is composed in a bottom-up fashion. In each inner node two *blocks* (in the lowest level these are single cells) are joined to a *meta-block* (partial placement). In each meta-block the orientations of the combined blocks are fixed (fig.3). Therefore every tree describes several possible shapes for the corresponding layout, which enormously improves the performance of the GA. Blocks or sub-patterns in a tree defining a layout, is always stacked vertically upon each other. The pattern characterized by the right successor of an inner tree node is always positioned on top of the pattern characterized by its left successor when combining both parts into a pattern or meta-block.

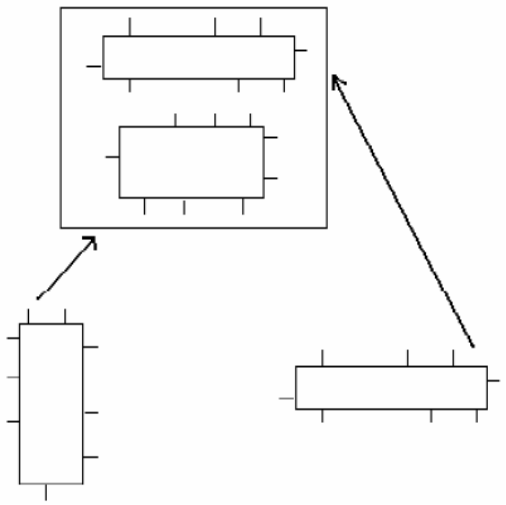

Fig. 3 The composition of a meta-block

C. Genetic Operators

During the optimization process the placement of the blocks has to be changed. The genetic operators directly work on the tree-structure by combining subtrees of parents (crossover) and modifying the tree of an individual (mutation). The crossover operator takes two individuals (parent) out of which one offspring is composed by combining two subtrees, one from each parents [12]. Unfortunately, these parts usually do not add up to a complete layout. After the combination of the two subtrees the redundant blocks are deleted and the missing blocks have to be added at random positions to the tree to ensure that the offspring finally represents a correct layout. Mutation operator modifies either by exchanging simple blocks or a block (leaf) with a meta-block (subtree) or by exchanging two meta-blocks. These cases represent the exchange of two cells, a cell with a partial layout, and the exchange of two partial layouts on the layout surface.

III. Genetic Partitioning Algorithms

Problem of VLSI circuit partitioning is non polynomial, hard and cannot be effectively solved by deterministic

algorithms. Genetic algorithms belong to probabilistic and iterative class of algorithm and are stochastic in nature. Therefore they can be effectively, efficiently applied for VLSI circuit partitioning. The problem involves dividing the circuit net list into two subsets and some of the connections (edges) are also cut. The number of edges belonging to two different partitions is the cost of a partition. The objective function captures the interconnection information and partitioning solution is optimized with respect to interconnection between the partitions with the constraint of forming balanced partitions. The mathematical representation [4] of the objective function is given as minimize cost function as shown in (1) below:

$$C = \sum_{i=1}^{k} \sum_{j=1}^{k} I_{ij} (i \neq j) \quad (1)$$

Where i, j are the vertices of an edge
C = cost of cut
I_{ij} = cost of an edge.
As the problem involves bi-partitioning of a circuit so equality condition must be satisfied as

$$\sum_{i=0}^{k} m_i = \sum_{j=0}^{k} n_j \quad (2)$$

Where m_i and n_j are nodes in the two partitions.
The algorithm has the following steps –
Net list processing Circuit information is accepted in the form of circuit net list. Netlist processing is done so as to convert the circuit netlist in the form of chromosome.

A. BFS Algorithm

The information of interconnection between the components in the netlist is converted in form of adjacency matrix. This Adjacency matrix [14] information is then used to traverse the circuit in BFS algorithm so that the connected components remain clustered together as far as possible.

B. Initial population:

Once the BFS order of components is obtained it is processed to form the initial solution for GA by converting it into 32-bit chromosome. The 32-bit chromosome [15] contains integer values, with each integer value corresponding to each element of chromosome encoded to represent the partition number assigned and number of elements clustered to form single chromosome element. Value of *jth* cell of chromosome is n1n2, where, n1 indicates the partition number assigned and n2 indicates the number of components clustered.

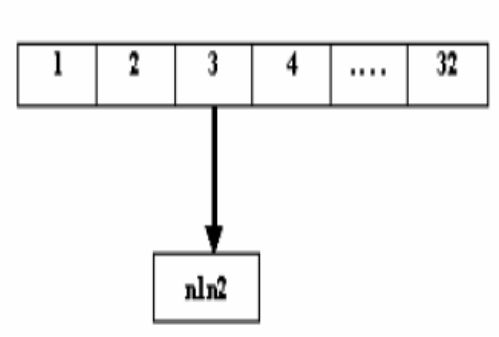

Fig. 4 32-bit Chromosome

Though other sorting data structure algorithm can be used such as depth first search algorithm, spanning tree algorithm etc, breadth first search algorithm has been found to capture circuit information more effectively [5]. Using the initial solution, random population is generated of the population size specified by the user. For each individual of the population, cost is computed. Objective function captures the cost of number of interconnections cut between the partitions.

C. Fitness Evaluation

Using the cost computed, each individual is evaluated for its fitness function. Based on fitness values individuals are randomly selected using roulette wheel selection for crossover operation.

D. Crossover

Each individual is considered for selection as parent for crossover, with probability of selection proportional to its fitness value [4]. Flexibility is incorporated in crossover operation with the user specifying the value for multipoint crossover. Offsprings generated from crossover replace the lowest fit individuals of the population if their fitness value is higher else, no replacement is made in the original population. By the algorithm[13], new offsprings replace the equivalent number of worst solutions from previous population which helps in survival of better solutions over several generations.

E. Mutation

After population replacement, mutation is performed on the bits randomly with small probability of mutation. Probability of mutation is very important, because the number of bits to be mutated depends on this probability. Mutation of bits is not similar to the traditional binary mutation operator, which is simple inversion of any random bits (depending on Probability of mutation), in the population[4].Mutation changes the partition assigned to random number of components, where number of components depends on the probability of mutation. Even the partition assigned is generated randomly. Generally low values of probability of mutation are preferred so that population is not changed drastically which is critical. The population with mutated bits is then evaluated for fitness and again whole cycle of selection, crossover, replacement and mutation is followed and repeats for number of iterations of GA specified by the user. No stopping criteria is specified in

the algorithm itself because one of the advantages of evolutionary approach to partitioning is, availability of ready solution at any stage, which if not globally optimal at least guarantees a good solution. But if no improvement is seen in the fitness and mincut results for consecutive 100 runs on a small scale circuit, GA is terminated [4]. The algorithm is the flowchart in Fig. 5.

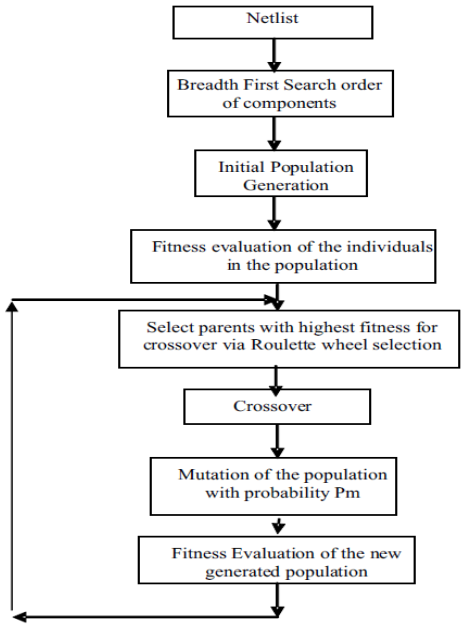

Fig. 5 Flowchart of the GA based partitioning algorithm

IV. Genetic Floorplan Algorithm

The Genetic Floorplan Algorithm (GFA) [10] adopts slicing structure to represent a floorplan and uses evolutionary scheme to explore solutions. Initially a set of slicing floorplans are generated to construct the *population* Π. Next, two chromosomes, *P*1 and *P*2, are randomly selected from Π to perform the crossover operation. The developed *crossover* is to produce new *child* that as far as possible inherits the sub-floorplans that are area-efficient according to a *threshold value Ts*. If the area of the generated child is better than that of its parents, the new string will be selected to directly replace the worst parent. Afterward, a set of mutation operations are performed to avoid local optimum and create diversity during the evolutionary process. Finally, the threshold value *Ts* will be increased with a little value Δ when the best solution is not improved after a *period* of time. The algorithm stops when the stop criterion is satisfied. GFA can rapidly generate floorplans of small area because it inherits area-efficient sub-floorplans from ancestors to form another area-efficient ones based on a simple and well-defined fitness during crossover.

A. Multi-Objective Genetic Floorplan Algorithm(MOGFA)

In order to simultaneously reduce total area and wirelength of a placement, an effective replacement scheme is needed. Hence, we extend the GFA with minor modification. The major difference between GFA and MOGFA is that MOGFA utilizes the concept of *dominance relation* [11][18] to select the survivors among the generated children and original populations. Here, each floorplan is given a score, according to its area and total wirelength. Let p and q be two floorplans whose total area and wirelength correspond to $area(p)$, $wirelength(p)$, $area(q)$, and $wirelength(q)$, respectively. Then p is said to dominate q, denoted as $p \angle q$, if and only if 1) $area(p) \le area(q)$ and $wirelength(p) \le wirelength(q)$, and 2) $area(p) < area(q)$ or $wirelength(p) < wirelength(q)$. That is, suppose p and q are two solutions along with k objectives, denoted as p_i and q_i, p is said to dominate q, if and only if 1) $\forall i \in \{1, 2, ..., k\}: p_i \le q_i$, and 2) $\exists j \in \{1, 2, ..., k\}: p_j < q_j$. Let f, g, and $c \in R_k$, and assume $f \angle c$, and $c \angle g$. We count the number of solutions that dominate c, denoted by $\sum |f \angle c|$. We also count the number of solutions that dominated by c, denoted by $\sum |c \angle g|$. Then the score for a solution, c, is defined as follows, $score(c) = \sum c \angle g - \sum f \angle c + popsize$.

Fig. 6 illustrates the calculation for score of floorplan c, where objectives are total area and total wirelength. Let original *population* = {○}, *popsize* = 10, and $CP1 \cup CP2 = \{\square\}$. Then $\sum |c \angle g| = 3$ and $\sum |f \angle c| = 2$. Hence, *Score* $c = 3 - 2 + 10 = 11$.

The floorplans containing higher scores are strongly favored to survive in the evolutionary search process. The selection scheme chooses the floorplan that has the highest score and marks the chosen floorplan. The selection continues choosing the floorplan of highest score from the unmarked floorplans until the number of the chosen floorplans is equal to the number of original population size.

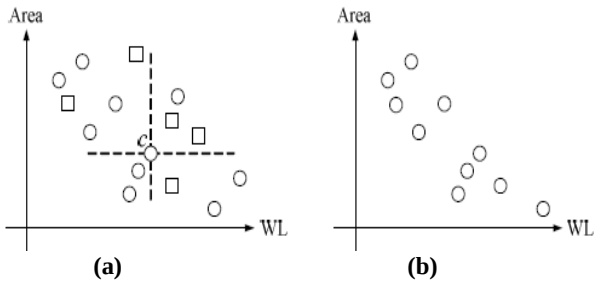

Fig. 6 (a) Illustration for calculating score c,
(b) The new population

IV. Genetic Placement Algorithm

Multilevel Genetic Placement Algorithm; called MGPA, is proposed to solve large-scale mixed-size placement problem. MGPA first adopts the well-known multilevel hypergraph partitioning algorithm [7] ,[8] to effectively reduce the great complexity. Afterward, two key techniques, consisted of GFA [10], MOGFA are individually applied to handle different sets of partition. At the top level, an earlier version of our fixed-outline floorplanner FOGFA [16] is applied to address fixed die-size placement. Finally, the *Capo* standard-cell placer [6]

is used to place standard cells. This new methodology is efficient to simultaneously handle building modules and standard cells. Unlike the methods [17] this approach has eliminated the need to remove overlaps between the movable objects which imply shorter runtime.

A. Procedure Of The Multilevel Genetic Placement Algorithm

Input: A set of building modules and a large number of standard cells

Output: A fixed-outline placement with area and wirelength minimization

1. Recursive bipartition until the size of each partition is smaller enough that floorplanner or placer can handle.
2. Regard each partition that contains only standard cells as a soft module, denoted as *Ps*.
3. Apply GFA to the set of partitions that contain mixed-size modules, and the total area of modules is much larger than the area of standard cells.
4. Apply MOGFA [18] to the set of partitions that contain mixed-size modules, and the total area of standard cells is much larger than the area of modules.
5. Apply FOGFA at the top level.
6. Apply Capo to *Ps*.

B. Multilevel Genetic Placement Algorithm (MGPA)

The multilevel genetic placement algorithm MGPA uses the multilevel partitioning algorithm to effectively reduce the great complexity [17]. The recursive bipartition continues until the size of each partition is smaller enough that floorplanner or placer can handle.

There are three different sets of partition during the multilevel partitioning:

1. The partition is regarded as a *soft* module, denoted as *Ps*, where all objects inside the partition are standard cells.
2. The second kind of partition consists of either mixed-size objects or only building modules, where the total area of building modules is much larger than the area of standard cells.
3. The third kind of partition contains mixed-size modules, where the total area of standard cells is much larger the area of building modules.The number of second and third kind of partition is small that floorplanning techniques can handle. Thereafter, two floorplanning techniques, GFA and MOGFA[18], are respectively applied to determine geometric locations of second and third partitions with area and wirelength minimization. GFA is applied to handle the second kind of partitions to effectively shorten runtime. MOGFA is applied to tackle the third kind of partitions to simultaneously reduce total area and wirelength. At the top-level partition, an earlier version of our floorplan tool FOGFA [16] is applied to efficiently generate fixed-outline placement. Finally, the *Capo* standard-cell placer [6] is used to place standard cells of *Ps.*

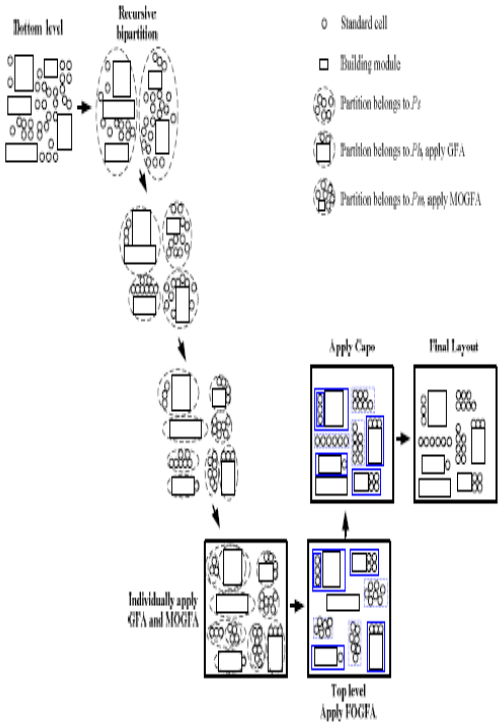

Fig. 7 Flow of multilevel genetic placement algorithm

V. Conclusion

In this survey paper we have surved all the available optimization algorithm which addresses the large-scale partitioning, floorplanning and mixed-size placement problem for fixed die-size SOC/IP designs. These methods adopt the multilevel hyper graph partitioning to effectively reduce complexity. The two key techniques are individually applied to handle different sets of partition. We have observed that MGPA has no need to remove overlaps between the movable objects. Our research aims at experimenting efficient techniques for large-scale partitioning, floorplan, mixed-size & time-driven placement benchmarks in improving the runtime and wirelength during the placement process.

References

[1] Palesi Maurizio and Tony Givargis, "Multi-Objective Design Space Exploration Using Genetic Algorithms", Proceedings of the 10thInternational Symposium on Hardware/software Codesign, ACM Press,Estes Park, Colorado, pp 67-72, 2002.

[2] W. Tan, et al., "An efficient multi-way algorithm for balance partitioning of VLSI Circuits", IEEE International Conference on Computer Design, pp 608-613, 1997.

[3] Schnecke V., Vornberger O (1997) Hybrid Genetic Algorithms for Constrained Placement Problems. IEEE Transactions on Evolutionary Computation. Vol. I. No.4. :266-277.

[4] Sandeep Singh Gill, Rajeevan Chandel, and Ashwani Chandel , " Comparative study of Ant Colony and Genetic Algorithms for VLSI circuit partitioning" , International Journal of Electrical and Computer Engineering 4:6 2009

[5] P. Mazumder, E.M. Rudnik, "Genetic Algorithms for VLSI Design, Layout and Test Automation", Pearson Education, 2003.

[6] A. E. Caldwell, A. B. Kahng and I. L. Markov, "Can Recursive Bisection Alone Produce Routable Placements?", In Proc. Design Automation Conf, pp. 477- 482, 2000.
[7] G. Karypis and V. Kumar, "Multilevel k-way hypergraph
[8] partitioning," In Proc. Design Automation Conf, pp. 343–348, 1998.
[9] G. Karypis, R. Aggarwal, V. Kumar and S. Shekhar, "Multilevel Hypergraph Partitioning: Applications in VLSI Domain," In Proc. Design Automation Conf, pp. 526–529, 1997.
[10] Chang-Tzu Lin, De-Sheng Chen and Yi-Wen Wang, "An Efficient Genetic Algorithm for Slicing Floorplan Area Optimization," In Proc. Int.Symp. on Circuits and Systems, pp.879 -882, 2002.
[11] Chang-Tzu Lin, De-Sheng Chen and Yi-Wen Wang, "An Efficient Genetic Algorithm for Slicing Floorplan Area Optimization," In Proc. Int.Symp. on Circuits and Systems, pp.II-879 -II-882, 2002.
[12] Jens Lienig, James P.Cohoon, "Lecture notes in computer science ",volume-1141,Parallel Problem Solving from Nature—PPSN IV , In pp. 839-848,1996.
[13] Sathyamoorthy S, Andaljayalakshmi G.,"Hybrid Genetic algorithm For VLSI Macro Cell layout",In 6th online World conference on soft computing in industrial applications, September 10-24,2001.
[14] Sandeep sign Gill, Dr.Rajeevan Chandel, Dr.Ashwani Chandel,"Genetic Algorithm Based Approach To circuit Partioning",International Journal of computer and electrical engineering ,Vol. 2,No. 2, April,2010.
[15] D.E. Goldberg, "Genetic Algorithms in Search, Optimization and Machine learning", Pearson Education, 2004.
[16] http://www.gigascale.org/bookshelf
[17] Chang-Tzu Lin, De-Sheng Chen and Yi-Wen Wang, "Robust Fixed-outline Floorplanning Through Evolutionary Search," In Proc. Asia and South Pacific on Design Automation Conf, pp. 42-44, 2004.
[18] Saurabh N. Adya and Igor L. Markov, "Consistent Placement of Macro-Blocks Using Floorplanning and Standard-Cell Placement," In Proc. Int. Symp. on Physical Design, pp. 12-17, 2002
[19] S.-Y. Ho, and X.-I. Chang, "An efficient generalized multiobjective evolutionary algorithm," In Proc. Genetic and Evolutionary Computation Conference, pp. 871-878, 1999

Rajine Swetha R, born in Madurai, Tamilnadu on 11.7. 1981 has done her Bachelor's in engineering from Avinashilingam university for women Coimbatore, Tamilnadu, in 2002 with the major field of study as Biomedical instrumentation. She did her Masters in engineering from Shanmugha arts science technology research centre (SASTRA), Tamilnadu in 2004 with the major field of study as Biomedical Signal Processing and instrumentation.
She has worked as RESEARCH AND TEACHING ASSISTANT in Indian Institute of Science, Bangalore and now works as faculty in K.S. Institute of Technology, Bangalore. Her current research interest includes usage of genetic algorithms in physical design, biomedical VLSI.
Ms.Swetha is a life member in the Biomedical Society of India.

B. ShekarBabu, born in Bangalore, Karnataka on 28.05.1976 did his Bachelor's in Engineering from Banglaore University, India in the 1998 with major field of study in Electronics and Communication Engineering. He did his Master's in Engineering from Sathyabama University, Chennai, India in the year 2007 with major in Applied Electronics Engineering.
He worked as TECHNICAL SUPPORT ENGINEER at Hewlett Packard.
At present he is a faculty in Dayananda Sagar engineering college , Bangalore. Current research interests include placement techniques, physical design synthesis, analog mixed mode VLSI.

Dr.SumithraDeviK.A, born in Bangalore, Karnataka on 20.05.1961 has done her Bachelor's in engineering from Mysore university , Karnataka, in 1985 with the major field of study as Electronics and communication. She did her Masters in engineering at University Visweshwaraiya College of engineering ,Bangalore University in 1990.She did her Doctorate from Avinashilingam University for Women, Coimbatore, INDIA, in computer science and engineering with specialization in VLSI partitioning CAD tool
She is the Director, professer in master of computer applications department R.V.Collage of Engineering, Bangalore, Karnataka, INDIA.