A GENETIC ALGORITHM FOR VLSI FLOOR PLANNING

CD Rawat[1], Anmol Shahani[2], Nitish Natu[3], Abbas Badami[4], Ronak Hingorani[5]

[1]*Associate professor, Electronics & Telecommunication, V.E.S.I.T, Maharashtra, India,**chandansrawat@gmail.com***
[2]*Student, Electronics & Telecommunication, V.E.S.I.T, Maharashtra, India,**shahanianmol@gmail.com***
[3]*Student, Electronics & Telecommunication, V.E.S.I.T, Maharashtra, India,**nit90nat@gmail.com***
[4]*Student, Electronics & Telecommunication, V.E.S.I.T, Maharashtra, India,**badami.abbas@gmail.com***
[5]*Student, Electronics & Telecommunication, V.E.S.I.T, Maharashtra, India,**hingoranironak@gmail.com***

Abstract

The classical floor planning techniques use block packing to minimize chip area, by making use of algorithms like B-TREE representation, simulated annealing. To get an optimal solution it is imperative to choose an efficient, cost effective algorithm. This paper presents a genetic algorithm to provide a solution to the floor planning technique. It incorporates slicing tree construction process for the placement and area optimization of circuit modules. It uses a probabilistic selection approach within its evolutionary cycle. Genetic algorithm produces optimal or nearly optimal solutions to the floor planning process by using techniques inspired by natural evolution, such as inheritance, mutation, selection, and crossover.

Keywords: *Floor planning, genetic algorithm, slicing tree, binary tree.*

1. INTRODUCTION

One of the most important stages in the designing or fabrication of VLSI circuits is floor planning. It is computationally quite difficult. The process of determining the circuit modules and their position with the objective of area optimization is referred to as floor planning.

Genetic algorithm is a stochastic optimization technique inspired by the theory of evolution. According to the theory of evolution, living organisms change their characteristics in response to change in environmental conditions. The characteristics of individuals are based on the makeup of cell structure called chromosome. [4]

When individual reproduce in nature, the chromosome of the parents combine in a manner to achieve fitter individuals known as offspring. Fitter individuals have higher probability of produce fitter offspring thus climbing higher on the evolutionary ladder and maintaining the genetic makeup of that species.

Thus individuals evolve or mutate every generation to achieve the above stated goal. The continuous technology scaling over the years has led to the scale down of transistor length thereby reducing the effective size of any circuit module. As a result more number of modules can be packed in a fixed given area. This advancement adds on to the computational complexity of the floor planning process. To handle this, we use hierarchical design methods so that only portionsof the entire design have to be considered at a time. [7]

These advances of the nanometer era has made floor planning a crucial stage in VLSI design as it has a major influence on interconnect issues like wiring, congestion, crosstalk, and on performance. These issues are a bottleneck in achieving full potential of the technology.

This paper is organized as follows:
Section IIelucidates the Genetic algorithm as an optimization tool and compares it to the theory of evolution. Section III describes the advantages of Genetic Algorithm over other algorithms. Section IV states the proposed algorithm for providing an optimum solution to the VLSI floor planning. Section V illustrates the results and plots obtained on execution of the algorithm. Section VI summarizes the key points of floor planning using Genetic Algorithm.

2. GENETIC ALGORITHM

Genetic algorithm has the ability to simultaneously examine a set of possible solutions, manipulate them to achieve an optimized solution of the problem. The genetic algorithm starts with determining individuals in a given population. The individuals are encoded as binary strings called as chromosomal strings. The GA operates on these encodings during the optimization process.

The GA then selects individuals from the population (selection process can involve probabilistic select function or specific techniques like roulette wheelselection, tournament selection etc.) And evaluates them using a predefine fitness function. [2]. A fitness function is a complex mathematical function needed to evaluate each chromosome encountered by the GA. The fitness measure of the chromosome represents the quality of the solution being examined that ultimately decides the optimality of the solution. The fitness value of an individual determines whether or not it will survive through the generations. Hence it decides if an individual is fit to be chosen to participate in the operations.

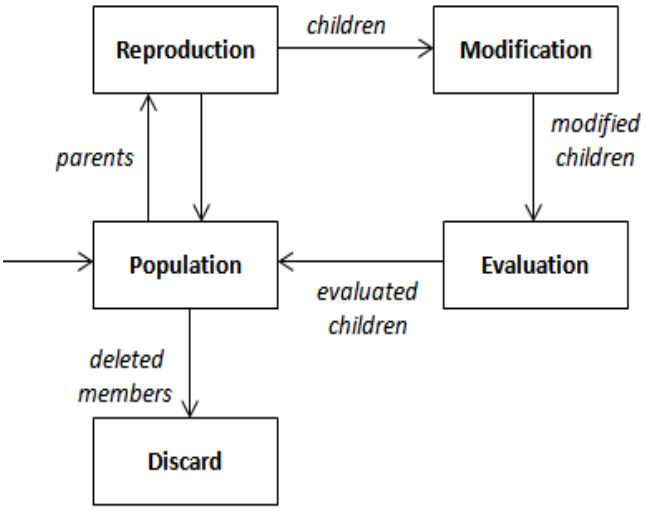

Fig1. The GA cycle of reproduction

The GA then uses these individuals to produce a new generation thereby moving upwards on the evolutionary chart. The algorithm then uses two operators namely crossover and mutation. The crossover performs an exchange of chromosomal information between two individuals to produce an offspring. It combines the good qualities from the parents to produce fitter offspring. Thus the offspring inherit the best qualities from both of its parents. But the crossover operator leads to sameness in the genetic characteristics of individual generation by generation. The offspring resemble the parents to a great effect. Then mutation operator plays an important role in restoring lost genetic information by providing diversity. [1]

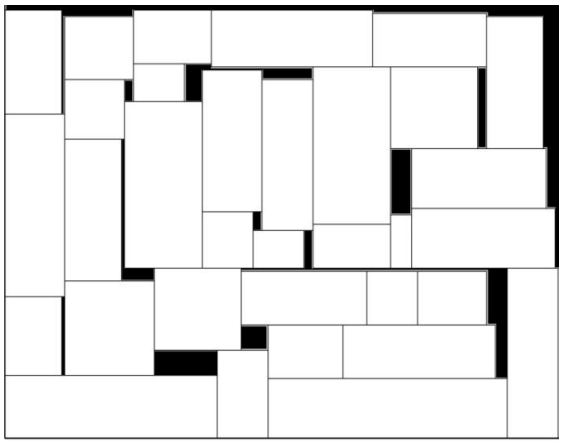

Fig2. Representational floor plan

Figure 2 shows a representational floorplan. The darker shaded regions in the figure illustrate the dead space. Our proposed genetic floorplanner minimizes the dead space to achieve optimum fitness of the circuit modules. The lesser the dead space obtained by the genetic algorithm more is the optimality of the solution.

3. OPTIMALITY OF GENETIC ALGORITHM

The advantages of using genetic algorithm as an optimization tool are elucidated in this section.
GA's are intrinsically parallel. Most other algorithms are serial and only extend their solution space in one direction. However since GA has many offspring in every generation they can explore the solution space in different directions. If one path in the gene hierarchy does not provide optimal or "acceptably good" solution, it can easily eliminate it and work on more promising offspring. [3]

It is because of their property of parallelism that they are better suited to solving problems where the space of all potential solutions is truly huge – too vast to search exhaustively in any reasonable amount of time. [8]

Genetic algorithm produces optimum solutions for noisy environments where other algorithms fail to give substantial results.GA do not require derivative information or other auxiliary knowledge; only the objective function and corresponding fitness levels influence the directions of search.

Since the genetic algorithm execution technique is not dependent on the error surface, we can solve multi-dimensional, non-differential, non-continuous, and even non-parametrical problems.

Another advantage of a hardware implementation of a GA is the elimination of the need for complex time and resource consuming communication protocols needed by an equivalent software implementation to interface with the main application.[5] This is particularly advantageous to real-time applications such as reconfiguration of evolvable hardware.

Another notable strength of genetic algorithms is that they perform well in problems for which the fitness landscape is complex - ones where the fitness function is discontinuous, noisy, changes over time, or has many local optima. GA has proven to be effective at escaping local optima and discovering the global optimum in even a very rugged and complex fitness landscape. However, even if a GA does not always deliver a provably perfect solution to a problem, it can almost always deliver at least a very good solution.

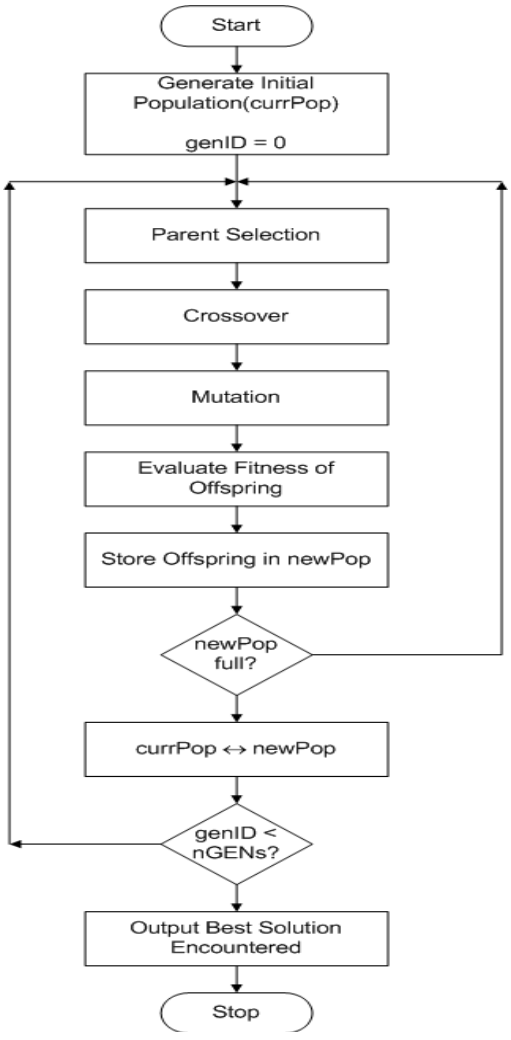

Fig3. Flowchart of GA. [3]

4. PROPOSED ALGORITHM

This paper proposes an algorithm to facilitate floorplanning using genetic algorithm. The method uses the efficiency of the genetic algorithm to optimize random inputs enabling us to work on wide patterns of floorplans that tend to appear in VLSI.

The population would be defined based on the input floorplan or rather the modules that are to be placed. The rectangle dimensions are to be provided by the user and also the number of modules to be placed and optimized.

The genetic algorithm, based on the principle of randomness, then would take customized number of iterations and continue to evaluate the input until the required optimization is achieved. The number of iterations can directly reflect on the accuracy and efficiency of the algorithm and thus should be chosen carefully. Another user input expected is the dimensions of the IC that are expected in the final placement.

The algorithm takes off by placing the blocks randomly along the defined rectangle of the IC. The rectangles are oriented either vertically or horizontally before the placement. Partitions are formed as one of the dimensions of the IC fills up. The whole arrangement forms the first population. User can decide the number of populations to be considered by the GA to compute the final floorplan. No. of partitions also play an important role in determining the end dead space.

The next step is the evaluation of cost for the given population. The cost function is a customized equation focusing on maximizing the unused area. The total area, as well as the cumulative area of the input modules, is fixed and thus, it won't matter if we maximize the unused area or minimize the dead space. Our cost function opts for the former and thus calculates the unused area which serves as the cost of the given population.

The genetic algorithm will assign indices to the population based on the cost and corresponding probability to pick up parents to be used for mating later. The assignment of probability is also random in nature. Two parents are picked up from different population and then crossover is performed using a pre-defined cross-over point. The cross-over point is a function of the number of matings to be performed and the bit-length of the population.

Mating involves crossover of the parents giving birth to a new offspring which will constitute the new population in the next iteration. The parents are chosen on basis of two variables, 'ma' and 'pa' which are a direct function of the probability of

the parent to produce a healthy offspring ultimately leading to optimization. Mutation is performed, again at a pre-defined rate, wherein a bit is chosen for a unit in the population using the mutation rate. The selected bit is then flipped giving rise to an entire new unit to be added to the new population.

The cost is evaluated for each and every population in every iteration and it minimizes as the number of iterations rise. There is provision to reset the whole process if the results are not headed towards the idea outcome and start over again. Also, if the minimum cost is achieved in less than the given number of iterations, the algorithm halts and presents the current population as the final optimized result.

5. RESULTS AND PLOTS

The following graphs were obtained with the layout of a 16-bit adder as the input. It comprises of 17 blocks, 16 of a 1-bit adder and a single block for computing the output carry bit. The dimensions of the 16 blocks are thus the same while the carry block is smaller. These plots were obtained using the software MATLAB R2007a.

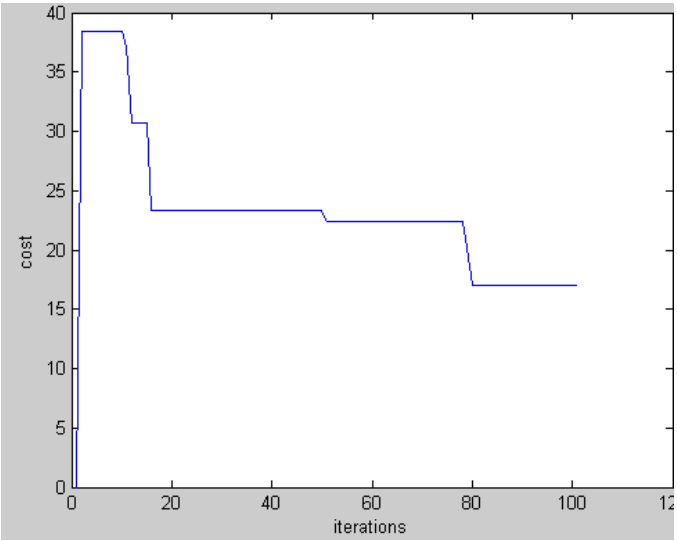

Fig4. Plot of cost v/s iteration

The layout, when designed using Cadence tool is 96.8uM2 placing the blocks in the most optimized way possible. The same input when fed to our proposed algorithm returns with an area of 80uM2. The height of the IC was fixed as 4uM while the length was varied according to the placements. The cost function calculates it as 20uM thus giving 80uM2 as the final area.

CONCLUSION

With recent advances in integration technology, in the nanometer era, VLSI floorplanning has transformed into multi objective optimization problem. Genetic algorithm facilitates in providing an optimum solutions to the problem as it is a more robust technique. It is applicable to both continuous and discrete optimization problems.Genetic algorithms are one of the best ways to solve a problem for which little is known. They are a very general algorithm and so work well in any search space.

REFERENCES

[1]. HameemShanavas .I and Gnanamurthy.R.K "Evolutionary Algorithmical Approach for VLSIFloorplanning Problem", International Journal of Computer Theory and Engineering, Vol. 1, No. 4, October2009 - 1793-8201.

[2]. D.JackulineMoni, S. Arumugam and D.GraciaNirmala Rani, "VLSI Floor Planning relying on Differential Evolution Algorithm", AIML Journal, Volume (7), Issue (1), June, 2007.

[3]. Pradeep Ruben Fernando , doctoral dissertation, University of South Florida "Genetic algorithm based design and optimization of VLSI ASICs and reconfigurable hardware".

[4]. PRATIBHA BAJPAI, "Genetic Algorithm – an Approach to Solve Global Optimization Problems", Indian Journal of Computer Science and Engineering.

[5]. D.F.Wong,andC.L.Liu,"A New algorithm forFloor plan Designs,"Proc.DAC.pp101-107,1986.

[6]. H. Murata, K. Fujiyoshi, S. Nakatake, and Y.Sajitani,"VLSI module placement based onrectangle-packing by the sequence-pair," IEEETrans. on Computer Aided Design, vol. 15, pp.1518-1524, Dec. 1996.

[7]. J. P. Cohoon, S. U. Hedge, W. N. Martin andD. S. Richards. Distributed Genetic Algorithms forthe Floorplan Design Problem. IEEE Transactionson Computer Aided Design, Vol. 10, No. 4, April1991.

[8]. C. L. Valenzuela and P. Y. Wang. VLSI Placement and Area Optimization Usinga Genetic Algorithm to Breed Normalized Post_x Strings. IEEE Transactions onEvolutionary Computations, Vol6.

[9]. Principles of Soft Computing – Shivanandam and Deepa, New Age Publications.

[10]. Neural Networks and Fuzzy Logics- SatishKumar, Prentice Hall of India.

[11]. Fuzzy Sets and Fuzzy Logics- Timothy, Prentice Hall of India.