# Genetic Algorithms based Partitioning of VLSI Circuit Systems

Nikhil Gupta
MTU, Noida, U.P.
Department of ECE
IIMT Engg. College, Meerut

Deepak Garg
MTU, Noida, U.P.
Department of ECE
IIMT Engg. College, Meerut

Soniya Gupta
MTU, Noida, U.P.
Department of AS
M.I.T., Meerut

## ABSTRACT
Circuit partitioning problem is a well known NP hard problem. The potential of Genetic Algorithm has been used to solve many computationally intensive problems (NP hard problems) because existing conventional methods are unable to perform the required breakthrough in terms of complexity, time and cost. The presented work deals with the problem of partitioning of a circuit using Genetic Algorithm. The program inputs the adjacency matrix, generates graph of the circuit and partitions the circuit based on crossover operator. The program produces a set of vertices that are highly connected to each other but highly disconnected from the other partitions.

## Keywords
*Genetic Algorithm, circuit partitioning, chromosomes, crossover, mutation*.

## 1. INTRODUCTION
With the advancement of VLSI technology the number of circuit components implemented on the VLSI chip is increasing day by day. Millions of transistors along with their numerous interconnections can be placed on a single chip today. With so many interconnections running over the chip area, it is important to find ways to reduce the overall length of the running wires across the chip area because this can affect various parameters of the interconnects like time delay in signal propagation, power consumption in the interconnections, area of chip acquired by the interconnections.

Circuit partitioning is an important step in VLSI physical design. This involves the breakup of a circuit into smaller parts for ease of design, layout and testability

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.

## 2. REVIEW OF PARTITIONING ALGORITHMS
In 1970, Kernighan and Lin [6] proposed a simple graph-based heuristic that iteratively improves an initial partitioning by using a *semi-greedy* graph search technique, since then numerous combinatorial optimization techniques have been applied to solve the graph and circuit partitioning problems like *constructive algorithms, iterative improvement algorithms.* Then *Fiduccia* and *Mattheyses*[7] proposed a more efficient method for implementing Kernighan and Lin's algorithm leading to a fast linear time algorithm for partitioning (the F-M algorithm). *Krishnamurthy* proposed Look-Ahead (LA) mechanism. *S. Dutt* and *W. Deng* proposed a new probability-based augmentation of the graph partitioning algorithm, called the probabilistic gain computation approach (*PROP*). L. A. Sanchis modified Krishnamurthy's algorithm to perform multiway partitioning. *Wei and Cheng*, introduced the *ratio-cut-algorithm*

Among the various well-know stochastic optimization methods, the *simulated annealing algorithm* has been widely used for solving numerous VLSI layout optimization problems. *Yih and Mazumder* [8] have presented a *neural network model* for circuit partitioning, using iterative improvement techniques. *Genetic Algorithm (GA)* was invented by Prof. John Holland [9] at the University of Michigan in 1975. Later, Prof. *David Goldberg* [5] at the University of Illinois made this topic popular.

## 3. CIRCUIT PARTITIONING
It is the task of dividing a circuit into smaller parts. The objective is to partition the circuit into parts such that the size of the components is within prescribed ranges and the number of connections between the components is minimized. Different partitioning results in different circuit implementations. Therefore a good partitioning can significantly improve the circuit performance and reduce layout cost.

## 4.CIRCIT PARTITIONING BY GENETIC ALGORITHM

### 4.1 Design Entry

Figure 1. shows the genetic algorithm for circuit partitioning. The algorithm starts with the design entry of the circuit to be partitioned, in to the program. An adjacency matrix is first prepared for the circuit to be partitioned. An adjacency matrix is a matrix which describes how the various components of the circuit are connected with each other. The components are treated as nodes and the interconnections are treated as edges of a graph. The program takes the Gate level implementation

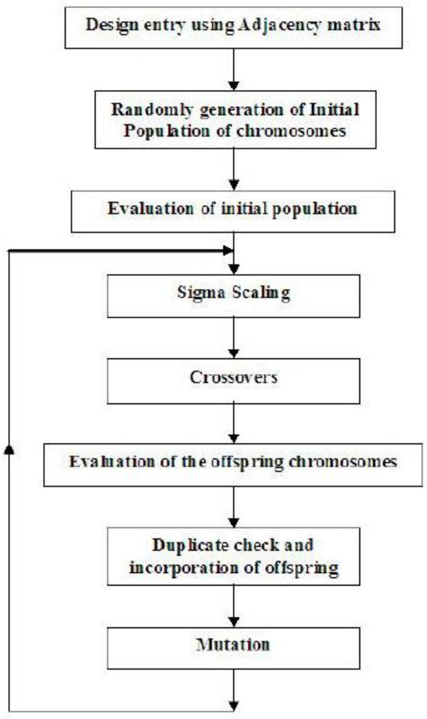
Design entry using Adjacency matrix
Randomly generation of Initial Population of chromosomes
Evaluation of initial population
Sigma Scaling
Crossovers
Evaluation of the offspring chromosomes
Duplicate check and incorporation of offspring
Mutation

**Figure 1. Flow chart of Genetic Algorithm for Circuit Partitioning**

of a 4-bit comparator circuit [10] as the object of analysis.

## 4.2 Initial Population Generation

A random initial population is generated. This population represents a number of possible solutions or ways to partition the circuit. Each of these ways is coded in to a binary string called chromosome. Thus a set of chromosomes are generated and termed as the initial population.

## 4.3 Population Evaluation

All the chromosomes are evaluated by a predefined fitness function. This function calculates the number of cuts imposed by the partition as suggested by the chromosome. The total cuts value is then converted into a fitness value using the sigma scaling function.

## 4.4 Sigma Scaling

In each generation, two parents are selected probabilistically, with the probability of selection proportional to their fitness. The evaluation function determines and stores the cost or badness of each chromosome. From this cost, the fitness of each individual is determined by sigma scaling as follows.

A reference **worst cost** is determined by

$C_w = C_{av} + S \times sd$. Where $C_{av}$ is the average cost,

**S** is the user defined sigma scaling factor, **Sd** is the standard deviation of the population cost

In case $C_w$ is less than the real worst cost in the population, then only the individuals with cost lower than $C_w$ are allowed to participate in the crossover.

The **fitness** of each individual is determined by

$F = C_w - C$ ; if $C_w < C$, where C is the cost of individual population. Two different individual are then randomly selected as parents, with selection probability directly proportional to the fitness.

If S is large, $C_w$ is large, and the fitness of the members of the population is relatively closer to each other. This causes the difference in selection probabilities to decrease, and the algorithm to be less selective in choosing parents. This may affect the algorithm in many complex ways, such as reduction of premature convergence of the population.

On the other hand, if S is small, the ratio of the least to the highest fitness in the population increases, and the algorithm becomes more selective in choosing parents. In fact, in this case, some high cost members of the population may not be able to participate in the algorithm at all. This may, in certain condition, speed up the algorithm, while compromising the final result quality.

## 4.5 Sorting

When all the chromosomes are evaluated and their fitness values are calculated, the chromosomes are sorted according to their decreasing value of their fitness value. Thus the chromosome with the highest fitness value is placed at the top and the lowest one at the bottom of the list.

## 4.6 Selection

In order to perform the crossover operation over the initial population of the chromosomes, it is required to select two chromosomes randomly; those are treated as parent chromosomes and are mated to obtain the offspring chromosomes. The chromosome with the highest fitness value is taken as the first parent and it is mated with other chromosomes one by one and a set of child chromosomes are obtained.

## 4.7 Crossover

Figure 2. shows the crossover operation. The crossover operator is the most important part of the genetic algorithm. In this the chromosome with highest fitness value is selected as parent chromosome and it is mated with other chromosomes to obtain two child chromosomes.

## 4.8 Evaluation of the offspring

The child chromosomes obtained after each crossover operation are evaluated by the fitness function and compared with the existing chromosomes of the population.

## 4.9 Duplicate check and incorporation

After the evaluation of each child chromosome obtained in the crossover operation, the fitness value of the child chromosome is compared with the chromosomes of the existing population. If the fitness value of the child chromosome is greater than the worst fitness value of the existing chromosome population, the child chromosome will replace the worst chromosome in the population, other wise the child chromosome is discarded.

Also if the child chromosome is same as the parent chromosome in the current population, the child chromosome is discarded.

## 4.10 Mutation

In this operation one bit of the chromosome is flipped. If it is ‘0’, it is changed to ‘1’ and vice-versa. In this way the chromosome characteristics is changed irrespective of their parent’s characteristics. Thus a new chromosome is generated. The mutated chromosome is evaluated and if its fitness value is greater than the others, it is included in the current population else it is discarded. If the decrement in the fitness value is within the predefined limits, then also the new chromosome may be included in the population. The probability of this has to be predefined, which is called the *probability of acceptance.*

## 5. RESULTS

The Genetic Algorithm for circuit partitioning is implemented in ‘C’ language. The program takes the *number of iterations* as input from the user and outputs the best chromosome and its fitness value after every iteration. The results are computed for three different parameters that can be altered in the program, namely *number of iterations, initial population* (*P)* and *sigma scaling factor (S).* Various graphs are drawn for the fitness value against different parameters like *number of iterations, initial population and scaling factor* for proper analysis of the results.

In figure 3, the fitness value is plotted for different number of iterations keeping scaling factor constant. As the scaling factor is increased the fitness value increases, the program convergence slows down. It shows that the number of iterations required is reduced if S is increased. In figure 4, the fitness value is plotted for different number of iterations taking different initial population. As the initial population of chromosomes is increased the program converges very rapidly.

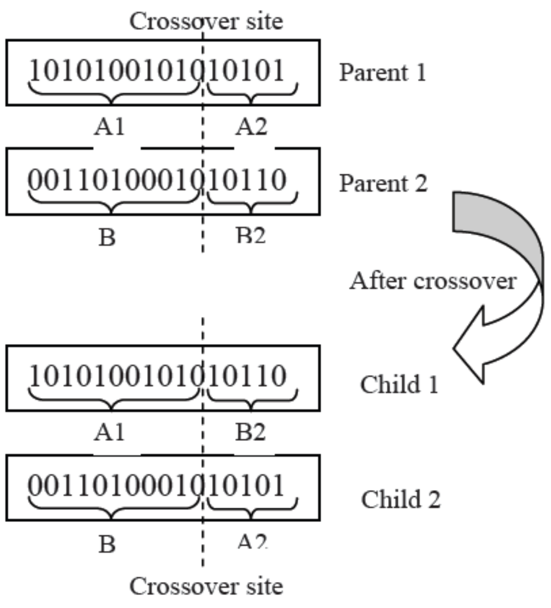

**Figure 2. Crossover operation**

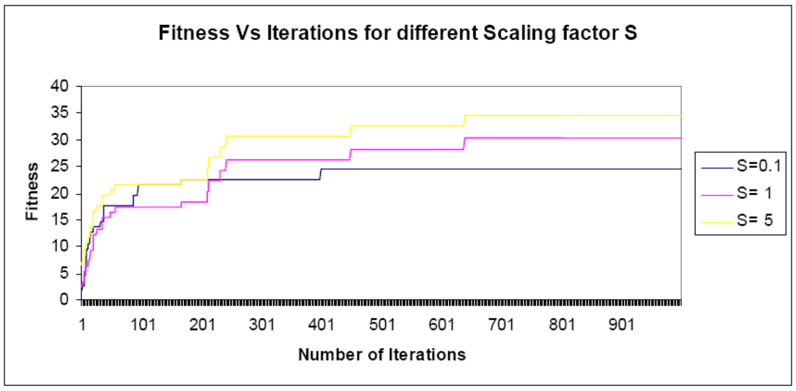

**Figure 3. Fitness Vs Iterations for different Scaling Factor (S)**

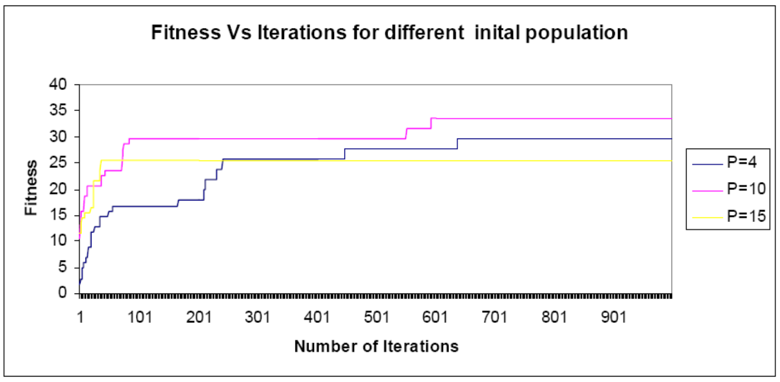

**Figure 4. Fitness Vs Iterations for Different Initial Population (P)**

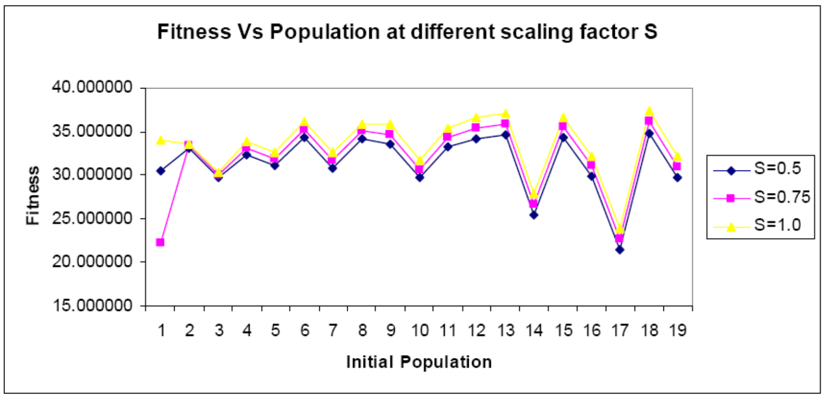

**Figure 5. Fitness Vs Iterations for different Initial Population (P)**

The quality of the results also decreases. Thus the initial population can not be increased abruptly. Typical value should be 10% of the number of node. In figure 5, the fitness value is plotted for different initial population keeping the Scaling factor constant. Scaling factor really does not matter so far as the dependence on initial population is concerned. However for higher values of scaling factor there can be a nominal advantage

## 6. CONCLUSIONS

The algorithm suggests a method for partitioning the circuit in to smaller sub-circuits such that the overall length of the various interconnects in the circuit can be minimized which in turn leads to reduction in delay, power dissipation and chip area. To achieve this, the partitioning should be such that the number of interconnects running between the partitioned groups should be minimized and thus reduce the overall length of the circuit interconnect.

## 7. ACKNOWLEDGMENTS

I would like to convey my heartfelt thanks to Prof. (Dr.) Shankar Sarkar (former professor ECE deptt. IIT Roorkee), Prof. (Dr.) Irene Sarkar (former professor Applied science dppt IIT Roorkee) and Dr. B. K. Kaushik (Asst. Professor department of Elecetronics & Computer Engineeinr. IIT Roorkee) for giving me continued encouragement throughout the research work.

## 8. REFERENCES

[1] Pinaki Mazumdar and Elizabeth M.Rudnick, "Genetic Algorithms for VLSI design, Layout & Test Automation," Pearson Education, Inc, 1999.

[2] Akash deep, Baljit Singh, Arjan Singh, and Jatinder Singh, "A Simple Efficient Circuit Partitioning by Genetic Algorithm," IJCSNS International Journal of Computer Science and Network Security, VOL.9 No.4, April 2009.

[3] Sarrafzadeh M. & C.K.Wong, "Introduction to VLSI Physical Design", McGraw Hill, 1996

[4] Sandeep Singh Gill, Dr. Rajeevan Chandel, Dr. Ashwani Chandel," Genetic Algorithm Based Approach To Circuit Partitioning," International Journal of Computer and Electrical Engineering, Vol. 2, No. 2, April, 2010 .

[5] Goldberg D.E., "Genetic Algorithms in Search, Optimization and Machine learning", Pearson Education, 2004.

[6] Kernighan B. W. and S. Lin, "An efficient heuristic procedure for partitioning graphs," Bell Systems Technical Journal, vol. 49, pp. 291—307, 1970.

[7] Fiduccia C. M. and R. M. Mattheyses, "A linear-time heuristic for improving network partitions," Proc. Design Automation Conf., pp. 175—181, 1982.

[8] Yih J S. and P. Mazumder, "A neural network design for circuit partitioning," IEEE Trans. Computer-Aided Design, vol. 9, no. 10, Oct. 1990.

[9] Holland J. H., Adaptation in Natural and Artificial Systems, Ann Arbor, M University of Michigan Press, 1975.

[10 M. Morris Mano, "Digital Design", 2nd edition, Prentice Hall of India, 2000.

[11] Sung-Mo kang and Yusuf Leblebici, " CMOS Digital Integrated Circuits", Analysis and design, 3rd edition, TMH, 2003.