International Journal of Computer and Electrical Engineering, Vol. 2, No. 2, April, 2010
1793-8163

Genetic Algorithm Based Approach To Circuit Partitioning

Sandeep Singh Gill, Dr. Rajeevan Chandel, Dr. Ashwani Chandel

Abstract— **In this paper multiway circuit partitioning of circuits using Genetic Algorithms has been attempted. Due to the random search, inherent parallelism, and robustness of genetic algorithms, the solution of a circuit partitioning problem is global optimum. Results obtained show the versatility of the proposed method in solving NP hard problems like circuit partitioning. Results obtained show an improvement over the results of UCLA Branch and Bound partitioner [27]. Information of the circuit has been given in accordance with circuit netlist files used in ISPD'98 circuit benchmark suite.**

Index Terms— **Partitioning, Genetic algorithm, NP Hard, Net list, Crossover, mutation.**

I. Introduction

With the advancements in VLSI technology the chip complexity is increasing, leading to more and more integration, increased design sizes, and huge chip estate being occupied by interconnects, which leads to increased delay. Improved physical design tools, are necessary to handle these issues. Circuit partitioning is an important step in VLSI physical design and involves the division of a circuit into smaller parts for ease of design and layout. The main objectives of circuit partitioning include minimization of number of interconnections between the partitions, minimization of delay due to interconnections between partitions, and ratio-cut minimization.

Efficient, easily applied algorithms for optimal clustering to minimize delay in digital networks were developed by Lawler et al. [1]. Kernighan and Lin [2] proposed a heuristic for two-way partitioning which was the first iterative algorithm based on swapping of vertices.

A more practical model based on hyper graphs was proposed, but was inefficient due to time complexity [3]. A new data structure bucket list for cell gains and proposed cell move with better time complexity was proposed [4]. Krishnamurthy [5] modified [4] to introduce the concept of look ahead to choose the cell move.

Various multiway partitioning algorithms were proposed by modifying [4] [5] and developing appropriate data structures [6], top down clustering and iterative primal-dual approach [7], dual intersection graph representation and ratio cut metric [8]. Areibi and Vannelli [9] described the application of Tabu search heuristic to circuit partitioning problem.

Mazumdar [10] proposed a GA based evolutionary approach for circuit partitioning giving a significant improvement in result quality. Comparative evaluation of genetic algorithm and simulated annealing was done with genetic algorithm giving better results [11]. A new hyper-graph partitioning algorithm hmetis was proposed, giving fast and better cutsize [12].

A genetic algorithm for partitioning of multi-FPGA system which uses problem specific encoding and fuzzy technique was developed [13]. Abeibi [14] discussed the implementation issues for applying memetic algorithm for VLSI physical design. A multi objective , hMetis partitioning for simultaneous cutsize and circuit delay minimization was proposed [15]. Various algorithms using different optimization techniques were developed for SoC and hardware software partitioning [16] [17]. Kolar et al. [18] developed a two way partitioning of a circuit, represented as a graph, using simulated annealing procedure, giving good results. Ghafari et al. [19] focused on minimizing the dynamic and sub threshold leakage power in CMOS circuits. Wang et al. [20] have given an algorithm for application partitioning on programmable platforms using Ant Colony optimization.

The different objectives [21] that may be satisfied by partitioning are:

1) The minimization of the number of cuts: The number of interconnections among partitions, have to be minimized. Reducing the interconnections not only reduces the delay but also reduces the interface between the partitions making it easier for independent design and fabrication. It is also called the mincut problem.
2) Minimization of delay due to partitioning: The partitioning of a circuit might cause a critical path to go in between partitions a number of times. As the delay between partitions is significantly larger than the delay within the partition, it is an important consideration in circuit partitioning. Important considerations for partitioning constraints [22] include.
3) The limit on number of terminals is decided by the maximum number of terminals available on PCB connector or the pin count or the number of terminals of a sub circuit.
4) Area of each partition is used as a constraint to reduce the fabrication cost with minimum area or as a balance constraint so that partitions are of almost equal size.
5) Number of partitions appears as a constraint as more number of partitions may ease the design but increase the cost of fabrication and number of interconnections

Manuscript received August 14, 2009.

Sandeep Singh Gill is working as Asstt. Prof., ECE Deptt., GNDEC, Ludhiana, Punjab, India.

Dr. Rajeevan Chandel is Asstt. Prof. & Head, E & CED, NIT, Hamirpur, H.P., India.

Dr. Ashwani Chandel is Asstt. Prof., EED, NIT, Hamirpur, H.P., India.

between partitions.

Various researchers have achieved varying levels of success using various optimization techniques. The current work attempts to use the simple genetic algorithm for multiway VLSI circuit partitioning.

The genetic algorithm has been used to minimize the interconnections, i.e. the mincut problem with a balance constraint. The coding has been done using MATLAB 7.0.

The proposed method gives excellent results for the partitioning problem, which brings out the versatility of genetic algorithms for solving such problems.

II. Mathematical Model/Problem Formulation

Problem of circuit partitioning is non polynomial hard and cannot be effectively solved by deterministic algorithms. Genetic algorithm being an evolutionary computational model, is stochastic in nature and can be effectively used for circuit partitioning. In this problem, the partitioning has been viewed as the task of clustering circuit elements in groups so that the objective function is optimized with respect to specified design constraints. Though a number of variants of original GA exist, simple GA with roulette wheel selection has been used. The objective function captures the interconnection information and partitioning solution is optimized with respect to interconnections between the partitions with the constraint of forming balanced partitions.

The mathematical representation of the objective function is given as

Minimize the cost function

$$C = \sum C_{ij}$$

Where i, j are vertices of an edge

$U^k_{n=1} \quad Vn = V$

C = cost of cut

Ci, = cost of an edge

k = number of partitions

Vi = Disjoint subsets of the net use

With the constraint

$V_1 = V_2 = V_n$

The applicability of the genetic algorithms has been illustrated by testing it on standard circuits based on ISPD'98 Benchmark suite. Keeping in mind its highly stochastic nature, performance of GA has been analyzed with respect to parameters:

1) Number of crossover points.
2) Probability of mutation.
3) Number of GA iterations.

III. Solution Methodology

Genetic Algorithms [23] are evolutionary computational models based on Charles Darwin's theory of natural evolution based on the concept of the survival of the fittest. Darwin observed that, as variations are introduced into a population with each new generation, the less-fit individuals tend to die off in the competition for food, and this survival of fittest principle leads to improvements in species. The concept of natural selection was used to explain how species have been able to adapt to changing environments and how, consequently, species that are very similar in 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 appearance and behavioral features of a living organism is contained in its chromosomes.

GAs are 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 out the bad features from individuals in the population.

The objective of the GA is then 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 optimum, but are able to find very good solutions for a wide range of problems [24].

The proposed algorithm follows the following steps:

Netlist: First step is to input the circuit to be partitioned. Circuit information is accepted in the form of circuit net list, in accordance with ISPD'98 benchmark suite. Netlist is commonly used in VLSI design to represent the circuit and can be considered as a hypergraph with vertices corresponding to cells (modules/ components/gates) and edges corresponding to signal nets [25]. Netlist processing is done so as to convert the circuit netlist in the form of chromosome. I/O pads are ignored in the netlist so that the partitioning process is reduced to partitioning of all components without any I/O constraints.

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

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 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.

In the figure 3.1, Value of j^{th} cell of chromosome is n1n2, where, n1 indicates the partition number assigned and n2 indicates the number of components clustered.

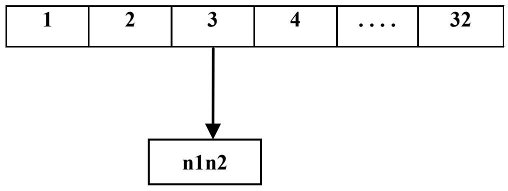

Figure 3.1. 32-bit Chromosome

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 [26]. 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.

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.

Crossover: Each individual is considered for selection as parent for crossover, with probability of selection proportional to its fitness value. 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. In this algorithm, new offsprings replace the equivalent number of worst solutions from previous population which helps in survival of the any better solutions over several generations.

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.

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.

The proposed algorithm is shown as flowchart in figure 3.2.

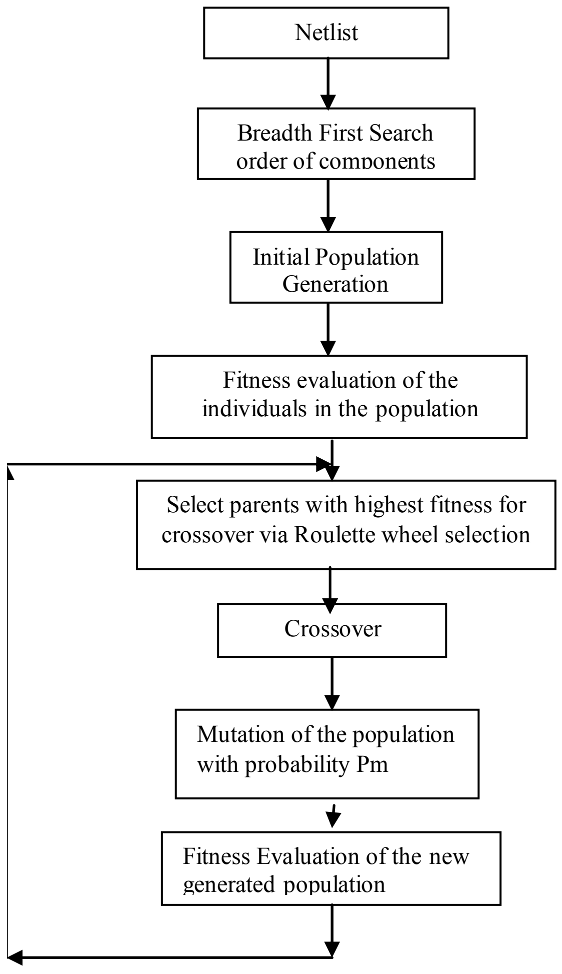
Netlist
Breadth First Search order of components
Initial Population Generation
Fitness evaluation of the individuals in the population
Select parents with highest fitness for crossover via Roulette wheel selection
Crossover
Mutation of the population with probability Pm
Fitness Evaluation of the new generated population

Figure 3.2 Flowchart of the Proposed Algorithm

Table 4.1 Results of partitioning with GA on circuit-1 (Np=10)

No. of iterations	Mincut	Average Cut	Max Fitness
50	25	32	0.0388
100	17	31	0.058
150	30	38	0.033
200	16	31	0.059
250	23	33	0.042
300	23	33	0.044

IV. Results And Discussion

The Proposed Algorithm is tested on two circuits to demonstrate the effect of variation various parameters in genetic algorithms on partitioning.

1) Circuit-1is a 4 bit full adder Circuit.
2) Circuit-2 is a Circuit for keyboard entry of 3-digit nos. in storage registers.

The Results of partitioning with GA on circuit 1 is shown in table 1. Here, No. of partitions, No. of Crossover points and No. of individuals are taken as 3, 1 and 10 respectively.

As seen from the results in table 4.1 and table 4.4 increasing the no. of iterations improves both mincut which is lowered as well as fitness which increases. But due to presence of local minimas and random nature of GA's it is not always possible to have lower values of mincut with increased no. of iterations e.g. mincut lowers from 25 to 17 and fitness of 0.058 is achieved if no of GA runs are increased from 50 to 100 runs. But if we further increase the runs to 150 mincut almost doubles to 30 and fitness drops to 0.033. this implies random nature of GA's an d also presence of local minimas of 17 and 25. At 200 runs global minima of mincut 16 is achieved . This shows that results given by GA in a specific no. of runs may not always be the global solution but it guarantees to gives the good solution at any stage.

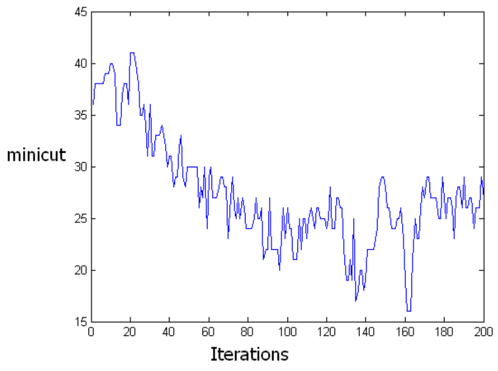

Fig 4.1 Mincut Plot for circuit 1 (Np=10)

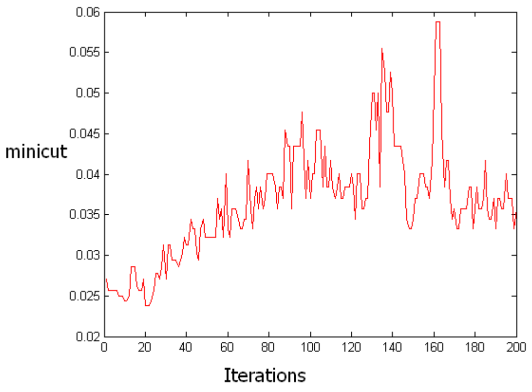

Fig4.2 Fitness plot circuit 1 (Np=10)

Table 4.2 Variation of mincut & fitness w.r.t. Pm for a 4-bit full adder Circuit Partitioning

Probability of Mutation (Pm)	Mincut	Average cut	Max fitness
0.0	32	38.22	0.0303
0.01	26	31.28	0.037
0.03	25	34.12	0.04
0.06	25	30.96	0.038
0.1	28	32.19	0.0344
0.2	24	35	0.04
0.3	22	32	0.044
0.4	21	33	0.046
0.5	16	23	0.059
0.7	23	33	0.042

No. of individuals in Population: 10
No of iterations: 150
No. of partitions: 3
Time of execution (min):28

Though mutation operator helps improve mincut and average cut but probability of mutation should be kept low (table 4.2). But mutation gives optimized or considerably lower value of mincut at cost of increased number of GA runs, which increase time of execution. For fixed number of runs, comparatively higher degree of mutation 30%-50% shows improvement in mincut. Very high values of probability of mutation around 1 mincut increases thereby giving lower fitness but the average cut increases with increase in probability of mutation which suggests mutation probability should be high around 0.5 and degree of crossover should be low for optimized results.

Table 4.3 Partitioning Results w.r.t. No. of individuals for circuit 1

No. of individuals	Min- Cut	Average mincut	Max Fitness
10	16	38	0.059
15	14	36	0.067
20	17	31	0.058
25	20	31	0.043

No. of Iterations: 200
No. of partitions: 3

As shown in table 4.3, with increase in no. of individuals in the population results are seen to show an overall improvement but improvement depends on whether the new individuals included in the population are closer to individuals giving good solutions or bad ones. Since this is also a random process, the dependency of GA results on no of individuals again is stochastic. But due to crossover and mutation operations involved even bad individuals undergo a change to yield better results and hence there is an improvement expected with increase in no of individuals in the population

Table 4.4 Partitioning Results w.r.t. No. of iterations on circuit 2

No. of iterations	Mincut	Maximum fitness	Avg Cut
100	32	0.0303	46.62
150	30	0.0322	44.23
200	30	0.0322	43.55
300	27	0.035	43.12

No. of Components in the circuit: 109
No. of nets: 156
No. of individuals in the Population: 10
No. of crossover points: 1 No of partitions: 3

As indicated by the results in table 4.5 the increase in number of crossover points does not necessarily increase the fitness, as fitness achieved besides crossover depends on points of crossover and mutation probability too. But in general with higher degree of crossover average mincut improves particularly for high mutation probability. Preferably mutation probability should be kept around 20-30% for good results.

Table 4.6 shows the results for partitioning w.r.t. the number of partitions. As the number of partitions is increased the mincut achieved is increased as expected because, if numbers of partitions required are more, optimization of interconnections between the partitions is required.

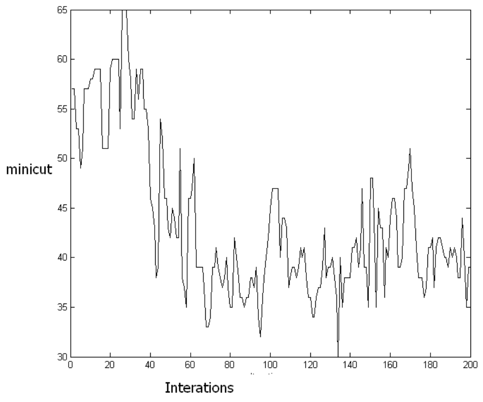

Fig 4.3 Mincut Plot for 100 iterations (circuit2)

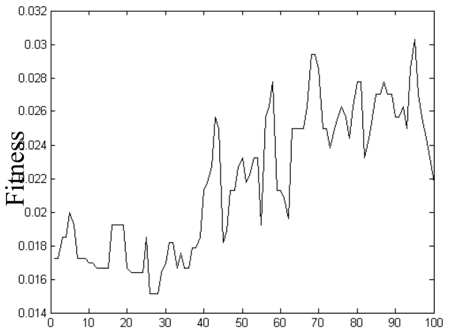

Fig 4.4 Fitness Plot for 100 iterations (Circuit 2)

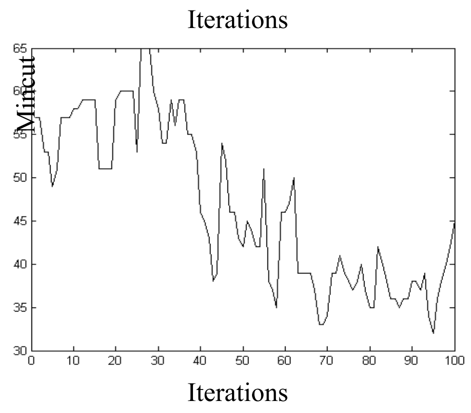

Fig 4.5 Mincut Plot for 200 iterations (circuit 2)

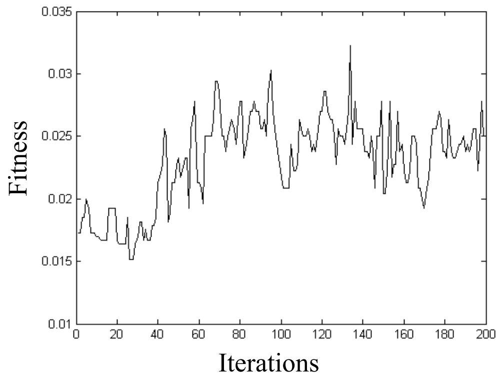

Fig 4.6 Fitness plot for 200 iterations (Circuit2)

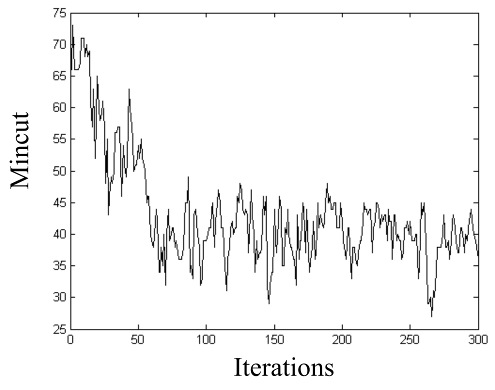

Fig 4.7 Mincut Plot for 300 iterations (Circuit2)

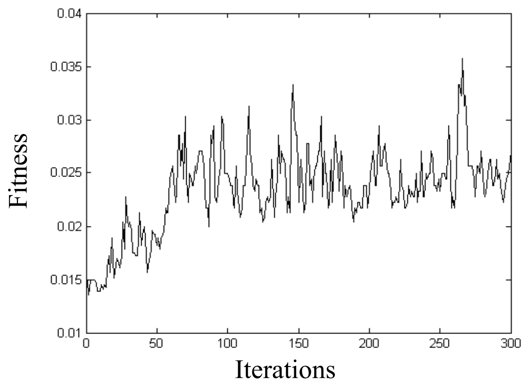

Fig 4.8 Fitness Plot for 300 iterations (Circuit2)

Table 4.5 Results of partitioning using GA on circuit 1 w.r.t. No. of crossover points

No. of crossover points	**Mincut**	**Avg. Cut**	**Time (min)**	**Max Fitness**
1	17	31	22	0.058
2	19	28	23	0.05
3	18	25	24	0.054
3	19	19	27	0.05

No. of Individuals: 10
No of partitions: 3
No. of Iterations: 100

Table 4.6 Results for partitioning w.r.t. No. of partitions

No. of partitions	**No of iterations**	**Min cut**	**Average Cut**	**Max. Fitness**
2	200	18	38.30	0.0526
3	200	30	43.55	0.032
4	200	31	45.2	0.031
5	200	27	46.69	0.0358

No. of Components: 109
No. of Individuals in population: 10
No. of crossover points: 1

Though exact number of partitions to be done depends on the area and I/O constraints of available chips into which partitions are to be mapped, the number of partitions to be done should not be very high as it will defeat the purpose of partitioning by consuming the resources inefficiently and resulting in higher circuit delays. It will give higher mincut as well as average mincut too will increase. But at the same time care should be taken that partition number should be a balance between area and I/O constraints as well mincut between the partitions.

The results obtained for circuit 1 and circuit 2 demonstrate the variation of various parameters and its effect. How ever, to compare the results with already available results, the method has been tried on circuit partitioning instances (net lists) available on MARCO GSRC VLSI CAD Bookshelf website. The circuit net lists are in the ISPD98 net list format (.Net D files). Results are compared with more obtained by UCLA Branch and Bound partitioner.

Table 4.7 Mincut results for GA partitioner and UCLA Branch and Bound Partitioner.

S. No.	CIRCUIT Series	umber f nodes	No. of net D files	inimum Cut using GA	inimum Cut using UCLA Branch & Bound Partitioner
1.	PP-N10.series	10	483	4.05	4.1
2.	PP-N15.series	15	184	5.29	5.4
3.	PP-N20.series	20	121	7.12	7.2
4.	PP-N25.series	25	107	8.0	7.6
5.	PP-N30.series	30	52	7.8	8
6.	PP-N35.series	35	31	10.32	10.4
7.	PP-N40.series	40	41	8.5	8.4
8.	PP-N45.series	45	28	10.8	11.2
9.	PP-N50.series	50	24	10.75	10.5
10.	PP-N55.series	55	20	11.5	11.7
11.	PP-N60.series	60	9	11.4	11.6
12.	PP-N65.series	65	6	10.8	11

Table 4.7 shows the comparison of average results obtained from the proposed GA partitioner and the UCLA Branch and Bound Partitioner [27]. The average results have been obtained on multiple number of partitioning instance groups in each size range. The partitioning instances have been generated by the top down partitioning based placement process employed by UCLA Capo placer [27]. The comparison of results shows that highly competitive results have been obtained through the GA process, which are either better or very near those obtained by the UCLA Branch and Bound Partitioner. Average time taken for simulation cannot be compared directly due to variation in hardware used.

V. Conclusion

The Genetic Algorithm applied to VLSI partitioning produces a significant improvement in result quality. In the GA, search is done from a population, not a single point. In conventional algorithms, a single point in the solution space is iteratively refined to obtain higher fitness values. By maintaining a population of well adapted points probability of reaching false peaks is reduced.

The results obtained show the random nature of GAs and the presence of local optima which need to be avoided. The global nature of GAs help in avoiding the above this problem. To obtain good solutions using GAs, the probability of mutation should be generally kept low. The increase in number of individuals may not always improve the results. The dependency of GA results on number of individuals is again stochastic. The increase in number of cross-over points does not necessarily increase the fitness, as the fitness achieved depends on cross-over as well as mutation probability.

The comparison of results obtained through the proposed GA based partitioner with UCLA Branch and Bound Partitioner [27] shows an improvement in cut size over most net list instances which proves the suitability of the proposed algorithm.

The main problem of a pure genetic-based partitioning algorithm is that its run time increases quickly as the problem size increases. In order to tackle the run-time complexity, a fast hybrid genetic algorithm that employs local optimization in every generation can be developed, and will help achieve a faster convergence without compromising the quality of the solutions.

References

[1] Eugene L. Lawler, Karl N. Levitt, and James Turner, " Module Clustering to minimize delay in Digital Networks", IEEE Transactions on Computers, Vol. C-18 , No.1,pp47-57,Jan, 1969.

[1] B.W. Kerhinghan , S. Lin, "An efficient heuristic procedure for partitioning graphs",Bell System Tech. Journal , Vol. 49, pp 291 – 307, Feb,1970..

[2] D.G. Schweikert and B.W. Kernighan, "A proper model for the partitioning of electrical circuits," Proc. ACM/IEEE Design Automation Workshop, pp. 57-62, 1972.

[3] C.M. Fiduccia and Mattheyses, "A Linear time heuristic for improving network partitions" , Proc. 19th IEEE Design and Automation Conference, IEEE Press, Piscataway, NJ, USA , pp 175-182, 1982..

[4] B. Krishnamurthy, "An improved min-cut algorithm for partitioning VLSI circuits",IEEE Trans. On Computers, Vol. C-33, May ,1989 .

[5] L.A. Sanchis, "Multiple way network partitioning", IEEE Trans. On Computers,,Vol 38, no. 1, pp 62-81 ,1989.

[6] Wei and Cheng, "Ratio-cut partitioning for hierarchical design", IEEE transc. on Computer Aided Design , , pp911-921 ,July 1991.

[7] Jason Cong,L.Hagen, Andrew Kahng , "Net partitions yield better module partitions" , 29th ACM/IEEE Design Automation Conference , Anaheim, CA, USA , pp47-52 ,1992.

[8] Shawki Areibi and Anthony Vannelli, "Circuit partitioning using a tabu search approach", IEEE International Symposium on Circuits and Systems , Chicago, Illinois,pp 1643-1646, March,1993.

[9] K.Shahookar and Mazumder "Genetic Multiway partitioning", IEEE 8th International Conference on VLSI Design, New Delhi, India , pp 365-369 ,4-7 Jan 1995.

[10] James Cane, Theodre Manikas, "Genetic Algorithms vs Simulated Annealing: A comparison of approaches for solving circuit partitioning problem", Technical report, University of Pittsburgh.

[11] George Karypis, Rajat Aggarwal, Vipin Kumar, and Shashi Shekhar, "Multilevel Hypergraph Partitioning: Applications in VLSI Domain",IEEE 34th Design Automation Conference., Anaheim,

California, United States, ,ACM Press New York, NY, USA., pp 526-529,1997.

[12] José Ignacio Hidalgo, Juan Lanchares, Roman Hermida, Graph partitioning methods for Multi-FPGA systems and Reconfigurable Hardware based on Genetic Algorithms, Computer Architecture Department, Spain.

[13] S. Areibi, "Memetic Algorithms for VLSI Physical Design: Implementation Issues", Genetic and Evolutionary computation Conference, San Fransisco, California, pp140-145, July,2001.

[14] C. Ababei, S.Navaratnasothie, K. Bazargan and G. Karypis, "Multi-objective Circuit partitioning for Cutsize and path-base delay minimization" , IEEE International Conference on Computer aided Design,2002.

[15] Maurizio Palesi,Tony Givargis, "Multi-Objective Design Space Exploration Using Genetic Algorithms", Proceedings of the 10th international symposium on Hardware/software codesign, ACM Press, Estes Park, Colorado , .pp 67-72 ,2002.

[16] Greg Stitt, Roman Lysecky, Frank Vahid, "Dynamic Hardwardsoftware Partitioning: A First Approach" ACM/IEEE Design Automation Conference 2003,, Anaheim, California, USA,pp 250-255, June 2-6, 2003.

[17] D. Kolar, J. Divokovic Puksec and Ivan Branica, " VLSI Circuit partitioning using Simulated annealing Algorithm", IEEE Melecon, Dubrovnik, Croatia,May 12-15,2004.

[18] P. Ghafari , E. Mirhard, M.Anis, S. Areibi and M. Elmary, " A low power partitioning methodology by maximizing sleep time and minimizing cut nets", IWSOC, Bauf, Alberta,Canada,pp368-371, July,2005.

[19] G. Wang, W. Gang and R.Kastner, "Application Partitioning on programmable platforms using Ant Colony Optimization", journal of Embedded Computing Vol.2 Issue 1, 2006.

[20] S.M.Sait, and H.Youseff, "VLSI Physical Design Automation", McGraw Hill Publishers, New Jersey, 1995.

[21] Naveed Sherwani, "Algorithms for VLSI Physical Design and Automation", third edition, Springer (India) Private Limited, New Delhi, 2005

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

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

[24] 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.

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

[26] http://www.gigascale.org/bookshelf