A Simple Efficient Circuit Partitioning by Genetic Algorithm

Akash deep[1], Baljit Singh[2], Arjan Singh[3], and Jatinder Singh[4]

BBSB Engineering College, Fatehgarh Sahib-140407, Punjab, India

Summary
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. This paper deals with the problem of partitioning of a circuit using Genetic Algorithm. The algorithm inputs the adjacency matrix generates graph of the circuit and partitions the circuit based on improved crossover operator. The algorithm produces a set of vertices that are highly connected to each other but highly disconnected from the other partitions and the results show that algorithm if far superior than the simple GA.

Keywords: *Genetic Algorithm, Circuit Partitioning, NP hard, Chromosome, Crossover*

1. INTRODUCTION

With the development in technology and growing demand for system-on-a-chip (SoC), integrated circuits had become more and more complicated. This creates a big challenge in IC design. Among steps in the design process, the circuit partitioning which is required as the first step in physical design has especially become very important[1][2]. A better circuit partition will reduce connection among sub-circuits and result in a better routing area of the layout. The challenge is that the circuit partitioning problem belongs to the class of well-known NP-hard optimization problem [3]. The problem can be viewed as a graph partitioning problem where each modules (gates etc.) are taken as vertices and the connection between them representing the edges between the nodes [4][5].

Since there may be many solutions possible for this problem therefore stochastic optimization techniques are utilized and until now many techniques have been known like Simulated Annealing Algorithm (SA) which combines the Local Search Algorithm with the Metropolis algorithm. SA is a simple and does not need much memory, but it takes a long time to reach the desired solution. Kemighan and Lin[12] proposed a two-way graph partitioning algorithm which has become the basis for most of the subsequent partitioning algorithms. Fiduccia and Manheyses [15] modified the K-L algorithm to a more efficient algorithm by suggesting moving one cell at a time and by designing a new data structure separately. As a kind of global optimization technique Genetic Algorithm (GA) which borrows the concept of generation from biological system had been used for circuit partitioning. Genetic Algorithms is an emerging technique. This technique has been applied to several problems, most of which are graph related because the genetic metaphor can be most easily applied to these types of problems. GA requires more memory but it takes less time than SA [6]. Lots of researchers have proposed their theories to partition circuit using GA. Work of [8] proposed hardware genetic algorithm by developing GA and local search processor that uses some external memory to overcome problem of local maxima/minima. Authors of [8][9] combined the advantages of both global search and local search algorithms. Authors of [8] expressed the probability of selection of chromosome as function of both the best and worst chromosome while [9] proposed different cost functions in order to achieve multiple objectives of minimizing delay cutsize area and power. The authors of [13] proposed two GA one based on 0-1 encoding and other based on integer encoding. Work done in [14] developed an adaptive strategy for partitioning and placement of circuits in which population size crossover rate and mutation rate is modified during the execution in order to enhance performance. Number of enhancements like crossover operator mutation or choosing different fitness functions can still be made to achieve optimal solutions. This means that theory of GA still provides opportunities for new inventions that can help in inventing new circuit partitioning problem solutions. This paper proposes a heuristic algorithm to solve the graph partitioning problem. The algorithm incorporates several genetic algorithm features namely selecting a population and crossover of the selected chromosomes to get better stable solutions. The algorithm starts by incorporating the circuit as unweighted connected graph in which each vertex represents gate and edge represents an interconnection between two gates and thereafter applying the GA metaphor to generate a partition that is highly interconnected within but disconnected from other sub partitions while trying to minimize the number of cuts and time consumed.

2. MATERIALS AND METHODS

Given un-weighted connected graph **G = (V, E)** on set of vertices V and edges E. Let k>=2 be a given integer, find a

Manuscript received April 5, 2009
Manuscript revised April 20, 2009

partition V_1, V_2, V_3……………..V_k of set of vertices V such that :

- The induced graph $G_i = (V_i, E_i)$, for all values i = 1,2,3………..k are connected.
- The following value is minimized $\min\{|V_1|,|V_2|,|V_3|\ldots\ldots\ldots\ldots |V_k|\}$

This k-Connected Problem is a particular instance of graph partitioning problem.

Because exact and approximation algorithms[3]that run in polynomial time do not exist for graph partitioning problems in general, it is necessary to attempt to solve the problem using heuristic algorithms. Genetic Algorithm is a heuristic technique that seeks to imitate the behavior of biological reproduction and their ability to collectively solve a problem. The GA starts with several alternative solutions to the optimization problem, which are considered as individuals in a population. These solutions are coded as binary strings, called chromosomes. The initial population is constructed randomly. These individuals are evaluated, using the partitioning-specific fitness function. The GA then uses these individuals to produce a new generation of hopefully better solutions. In each generation, two of the individuals are selected probabilistically as parents, with the selection probability proportional to their fitness. Crossover is performed on these individuals to generate two new individuals, called offspring, by exchanging parts of their structure. Thus each offspring inherits a combination of features from both parents. The next step is mutation. An incremental change is made to each member of the population, with a small probability. This ensures that the GA can explore new features that may not be in the population yet. It makes the entire search space reachable, despite the finite population size.

The basic foundation of the algorithm is to represent each vertex in the graph as a location that can represent a logic gate and a connection is represented by an edge [9][10][11]. To start the algorithm, *n* gates are placed on the graph as *n* vertex, and an initial population is chosen as the different permutations of the various vertices of the given graph. The problem reduces to associating to each chromosome each partition of the graph. An algorithm based on Genetic algorithm is proposed that can be used to partition an number of nodes.

Algorithm:-Proposed Algorithm for k–way partitioning using Genetic Algorithm

1. Input a connected graph G = (V,E) with |V| =n and an integer 1<k<n/4
2. Initialize randomly a population **P** of *2* X *n* elements.
3. **For** I = 0 to MAXGEN
4. **Do**
5. **For** each chromosome **p ∈ P**
6. call Create_Partition(p);
   fitness(p) = k.M(p) / n
   Comment: M(p) is number of nodes of partition with Maximum cardinality among *k* Partitions.

   End **For**.
7. Sort the elements according to fitness value.
8. Delete half of the population with lower fitness value.
9. **For** I=0 to n/2
10. Select two parents p_a , $p_b \in P$ randomly
11. Add the four individuals produced by crossover(p_a, p_b) to P.
12. End **For**
13. End **For**

Procedure Create_Partition (p)

1. **For** i=1 to k **Do**
2. Assign p_i to the partition i.
3. End **For**
4. **while** there are free vertices **Do**
5. **For** i= k +1 to n **Do**
6. **If** p_i is free
7. assign p_i to the smallest adjacent partition.
8. End **IF**
9. End **For**
10. End **while**

Procedure Crossover (p_a,p_b)

1. Compute intersection **I** of first k elements of p_a , p_b
2. **For** i=1 to k **Do**
3. **If** p_a [i] is not in I (intersection) then, choose randomly j >k
4. Swap (p_a[i], p_b[j])
5. End **If**
6. End **For**
7. **For** i=1 to k **Do**
8. **If** p_b[i] is not in I(intersection) then, choose randomly h>k such that p_b[h] is not in J
   Comment: J is set of first k elements of p_a.
9. Swap (p_b[i], p_b[h])
10. End **If**
11. End **For**
12. Copy the first k elements of p_a in q_1, q_3.
13. Copy the first k elements of p_b in q_2, q_4.
14. Create two vectors L, L' with 2(n-k) elements.
15. **For** j =1 to 2(n-k) **Do**
16. **If** (j mod 2 =1) then
17. L[j]=p_a[k + (j+1) / 2]
18. L**'** [j]=p_b[k + (j+1) / 2]
19. Else
20. L [i]=p_b [k + (j+1) / 2]
21. L**'** [j]=p_a[k + (j+1) / 2]
22. End **If**
23. End **For**.

24. **For** j =1 to 2(n-k) **Do**
25. **If** L [j] is not in q_1 then copy L [j] in q_1 else copy L [j] in q_2.
26. **If** L**'** [j] is not in q_4 then copy L**'** [j] in q_4 else copy L **'**[j] in q_4.
27. End **For**.

The algorithm starts by inputting a graph G(V,E) of the given VLSI circuit whose vertices |V| = n represent the gates of the circuit and whose edges represent the interconnection between different gates. The adjacency list of the graph representing the circuit is inputted. Also the number of partitions ie the leaders k are also inputted, after the execution of our algorithm all the different k leaders will be in different partitions. A population P of 2 X n elements is also initialized whose every element p known as chromosome is a permutation of the | V | integers representing the vertices of the given graph.

Two parents are selected randomly from the population and crossover of these chromosomes is performed to produce new off springs, the newly produced children are also added to the population and whole process repeated until desired results are reached ie until we find partitions whose total cardinality is close to average value.

2.1 Population

The first step in the algorithm consists of initiating a population P whose every element p such that $p \in P$ is known as chromosome is a permutation of the | V | integers representing the vertices of the given graph. Hence the population of our algorithm will be the different permutations of the vertices that represent the different gates of our circuit. Initial population is taken as twice the number of nodes. After each iteration population is updated as the chromosomes with low fitness value are deleted and new stable ones that are produced by cross over are added to the population.

2.2 Creating Partitions

After initial population of chromosomes is initiated we now proceed to produce partitions of our chromosomes. The first k elements of the chromosome represent the leaders of k partitions, ie two leaders cannot be in the same partition. The remaining |V| - k vertices are assigned to one of the k partitions by our procedure given above.

2.3 Fitness Function

The obtained partitions will be judged according to the following fitness function

Fitness(p)= k**.**M / |V| (I)

Where k is the number of partitions or the leaders and M is the number of vertices of the largest partition. That is (I) will give the fitness value of chromosome p. It is clear that k**.**M/|V| ≥ 1 and goal of our algorithm is to minimize such a value.

2.4 Crossover

After selection of the chromosomes our cross over operator is applied. This crossover operator is known as distance preserving operator, the distance between two chromosomes is defined as the number of leaders that are contained in one chromosome but not in other. Two chromosomes having no elements common among first k elements have a distance k. The aim of our crossover operator is to produce an offspring which has the same distance to each of its parents as one parent to the other. In the first chromosome we swap all the leaders that are not in common with the second chromosome with some of |V| - k elements chosen in a random way. Let

$C_{k+1}\ C_{k+2}\ \ldots\ldots\ldots\ldots\ldots\ldots C_{|V|}$

$D_{k+1} D_{k+1}\ \ldots\ldots\ldots\ldots\ldots D_{|V|}$

are the second parts of two chromosomes. Consider the list

$C_{k+1}\ D_{k+1} C_{k+2}\ D_{k+1}\ \ldots\ldots\ldots\ldots\ldots\ldots C_{|V|}\ D_{|V|}$

we use it to create two new chromosomes as follows:

Start from the first element and if the element does not belong to the new first chromosome, we put in there, otherwise we put it in the second chromosome. An example will demonstrate the crossover operator

3. RESULTS

The algorithm proposed is implemented to investigate the effectiveness of the algorithm using C programming language. A pseudo implementation of the algorithm is done. The results show that the algorithm produces results that are close to the optimal solution. Experiments have been performed by taking the graphs of variable number of nodes. The algorithm is made to run till the fitness function value falls within 10% of the ideal value and number of generations and CPU time required to reach this value is calculated for fixed number of partitions. Table 1 shows the results for the simple GA and our proposed algorithm. Two parameters measured are number of generations and CPU time. The results show that our method is far better than using simple GA method. Corresponding graphs are plotted that shows the effect on the values of CPU time and number of generations of Simple and Proposed method by increasing number of nodes.

Table 1: Number of generations and CPU time with fitness function value within 10% of ideal

Num of nodes	Avg Fitness Value	No of generations		CPU time	
		Simple	Proposed	Simple	Proposed
10	1.000	135	121	123	100
15	1.000	167	134	141	121
20	1.000	214	148	214	138
30	1.000	298	239	324	245
50	1.005	434	290	369	312
100	1.010	763	523	578	428
200	1.018	1976	1684	1273	973

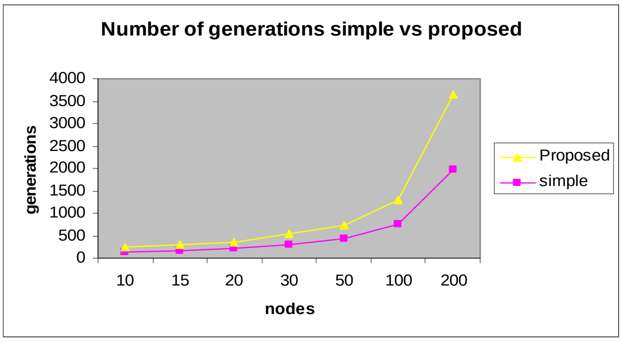

Fig.1. CPU Time Comparison between Simple and Proposed keeping Fitness Function values within ideal limits and partition accuracy within 95%

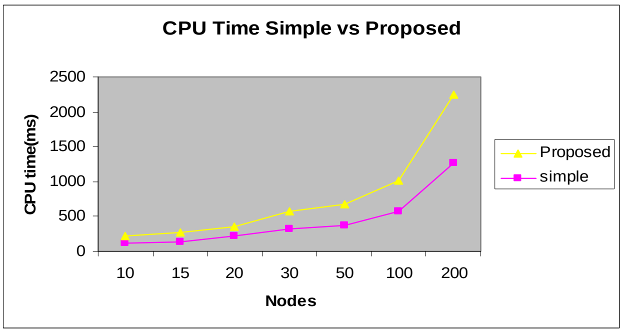

Fig.2. Number of Generations needed for Simple and Proposed keeping Fitness Function values within ideal limits and partition accuracy within 95%

When our genetic algorithm is made to run for a varying number of generations partitions are obtained with an accuracy of almost 95%.The results also show that the value of fitness function does not vary much even if the number of nodes are increased hence forth providing a better solution.

4. DISCUSSION

The heart of the algorithm is the crossover operator. We had used a simple but effective operator in this algorithm that produces new chromosomes that are more fit than their parents. Chromosomes that are less fit are discarded and only the fit chromosomes are used for further crossover.

5. CONCLUSION

Circuit partitioning is one of the key areas in chip designing. The algorithm proposed in this paper uses a new crossover operator that provides relatively good solutions for partitioning of circuit. The algorithm can partition circuit into a number of subcircuits. Our method calculates the fitness value and discards solutions with low fitness value. Space and Time complexity of our algorithm is much better than simple GA. The results demonstrate the effectiveness of the method. Our algorithm can be extended to partition standard circuits used in electrical and electronics chips and same algorithm can be extended to partition VLSI circuits.

REFERENCES

[1] Sadiq M sait and Habib youssef 1995. VLSI Physical Design Automation: theory and Practice Mcgraw Hill pp63-71. ISBN:0077077423

[2] Naveed A. Sherwani, 1995. Algorithm for VLSI Physical Design Automation, Kluwer Academic publishers, pp.141-171. ISBN:0792395921

[3] M. R. Garey and Johnson Computers and Intractability: A Guide to theory of NP-Completeness. Freeman Publishers, SanFrancisco. ISBN:0716710447

[4] A. E. Dunlop and B. W. Kernigham 1985 "A procedure for placement of standard-cell VLSI circuits" IEEE trans. CAD, Vol(4), pp. 92-98. http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=1270101 ISSN: 0278-0070

[5] Thang Nguyen Bui,and Byung Ro Moon, 1996. "Genetic Algorithm and Graph Partitioning, IEEE Transactions on Computers, vol. 45, no.7, pp.841-855. http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=508322 ISSN: 0018-9340

[6] Theodore W..Manikas,and James T.Cain, 1996 Genetic Algorithms vs Simulated Annealing:A Comparison of Approaches for Solving the Circuit Partitioning Problem Technical Report 96-101 ,The University of Pittsburgh. www.ee.utulsa.edu/~tmanikas/Pubs/gasa-TR-96-101.pdf

[7] Stephen Coe Shawaki Areibi and Medhat Moussa " A Genetic Local search Hybrid Architecture for VLSI Circuit Partitioning" Proceedings of 16th international conference on microelectronics, Dec 2004 pp 253-256. DOI:-10.1109/ICM.2004.1434259

[8] Apiradee Yodtean and Peerask Chantngarm " Hybrid Algorithm for circuit Partitioning" IEEE Region 10

conference Vol. D Nov 2004 pp 324-327. DOI 10.1109/TENCON.2004.1414935
[9] P. Subbaraj K. Sivasundari P. Siva Kumar 2007 An Effective Mementic Algoruthm for VLSI Partitioning Problem" Internatinal conference on ICTES Chennai India pp 667-670. Dec 2007
[10] F.M Johannes, "Partitioning of VLSI Circuits and Systems",Proceedings of 33rd design Automation Conference, USA June 1996, pp 83-87. ISBN: 0-7803-3294-6
[11] A. Cincotti, V. Cuttelo, M. Pavone, Graph Partitioning using Genetic Algorithms with OPDX".Proceedings of World Congresson Computational Intelligence IEEE 0-7803-7282.2002. pp 402 – 406
[12] Kernighan, B.W., Lin S 1970.An Efficient Heuristic Procedure for Partitioning Graphs, The Bell Sys. Tech. Journal, pp 291-307. http://www.cs.princeton.edu/~bwk/btl.mirror/new/partitioning.pdf.
[13] Guo-Fang Nan Min-Quang Li. Two Novel Encoding Strategies based Genetic Algorithms for circuit Partitioning. IEEE proceeding of 3rd International Conference on Machine Learning Shangai pp2182-2188. ISBN:0-7803-8403-2. http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=1382160
[14] George Ch. Sipakoulis and Adonios Thanailkis Genetic Partition and Placement for VLSI Circuits. Proceedings of 6th IEEE conference on Electronics Circuits and systems.1999 vol 3 pp 1647-1650.doi-10.1109/ICECS.1999.814490
[15] Fiduccia CM, Mattheyses RM 1982. A Linear-Time Heuristic for Improving Network Partitions. 19th ACM IEEE Design Automation Conference,1982 pp. 175-181.ISBN:0-89791-020-6. http://portal.acm.org/citation.cfm?id=800263.809204