COMPARATIVE STUDY OF DIGITAL CIRCUIT PARTITIONING ALGORITHMS BASED ON EVOLTUNARY COMPUTING

Maninder Kaur [#1], Kawaljeet Singh [*2]

[#1] *School of Mathematics and Computer Application, Thapar University, Patiala, INDIA*
[1] manindersohal@thapar.edu

[*2] *University Computer centre, Punjabi University Patiala, INDIA*
[2] singhkawaljeet@rediffmail.com

***Abstract*--- As the evolutionary computing based approaches are increasingly being used to solve different NP complete problems, the development of efficient parallel algorithms for digital circuit partitioning, circuit testing, logic minimization and simulation etc. is currently a field of increasing research activity. In some of these applications the circuit-partitioning problem occurs. That implies dividing a circuit into non-overlapping sub circuits while minimizing the number of cuts after the division and balancing the load associated to each one. This paper describes different evolutionary approach for solving circuit-partitioning problem and compares their effectiveness with existing classical approaches using benchmark circuit graphs/matrices. The extendibility of evolutionary approaches enables users to solve hardware/software aspects of partitioning instances and can be practiced for real purposes in VLSI/FPGA circuit's physical design.**

Keywords— **Evolutionary Computing, VLSI Circuits, Circuit Partitioning, Genetic Algorithm, Simulated Annealing.**

I. INTRODUCTION

A variety of devices is currently available for developing and implementing digital systems. Circuit partitioning is an important problem in many areas of VLSI and FPGA design, such as lay-out, placement, floor planning and routing etc. At the layout level, partitioning is used to find strongly connected components that can be placed together in order to minimize the layout area and propagation delay. In the design process with digital circuits, partitioning is used in the placement step, which assigns each node of the circuit network to a specific logic block in the FPGA device. Partitioning also plays an important role in rapid prototyping with multiple FPGA circuits. Bi partitioning of a circuit is done by dividing into two balanced components that minimizes the number of crossing connections [4]. This problem was shown to be NP-complete. Because of its importance, many heuristic algorithms have been proposed to solve the bi partitioning problem. Efficient designing of any complex system necessitates decomposition of the same into a set of smaller subsystem. Subsequently, each subsystem can be designed independently and simultaneously to speed up the design process [5]. The process of decomposition is called partitioning. The main aims at circuit partitioning, may be (a) the minimization of the number of sub-circuits, (b) the minimization of the number of interconnections between sub-circuits, (c) the minimization of the deviation in the number of elements (inputs, logical gates, outputs and fan out points) assigned to each partition.

II. DIGITAL CIRCUIT PARTITIONING PROBLEM

The partitioning problem is NP-complete problem this means it is unlikely that a polynomial-time algorithm exists for solving the problem. Therefore, one must use heuristic techniques generating approximate solutions. Partitioning is one of the first steps in VLSI circuit design. Partitioning has the important responsibility since it directly affects the rest of the steps in the process. A bad partitioning algorithm could leave us with a very well area-balanced chip, but with terrible wiring. We could also end up with a partition, which allows us to have the least complex wiring, but with the partitions being uneven in size. These results are undesirable, thus various partitioning algorithms have been built to create good partitions. Universally, all partitioning algorithms are expected to give good partitions, which we define as maintaining area constraints and minimizing wiring complexity. In fact, the perfect partition could always be found with any graph given an infinite amount of time. Time is limited, however; thus the algorithms must arrive at a reasonably good solution in polynomial time

III. CIRCUIT PARTITIONING ALGORITHMS BASED ON EVOLUTIONARY COMPUTING

A number of heuristic techniques are there to generate approximate solutions to the partitioning problem. This section discusses various evolutionary algorithms to solve circuit partitioning problem.

A. Divide and Conquer

The divide-and-conquer paradigm is widely used for solving large problems to reduce their complexity. The problem is recursively (top-down) partitioned into smaller sub problems. This process continues until sub problems are small enough to be solved directly. The solutions are combined hierarchically which yields, in general, suboptimal solutions

on the next higher level. A famous example for this successful solution strategy is the min-cut placement method in layout synthesis. Partitioning is applied recursively to the circuit's net list thereby generating a hierarchical neighbourhood (slicing) structure [6]. This structure is then interpreted as a placement for chip assembly. In addition to reducing problem size, solution quality is improved and heavy wiring congestion avoided by minimizing the number of wires cut by the partition.

B. The Kernighan-Lin Algorithm

The most basic approaches to the partitioning problem treat the circuit as a graph. This is true for the first, and most famous partitioning algorithm, called the Kernighan-Lin algorithm. This algorithm was originally designed for graph partitioning rather than circuit partitioning, so to apply the algorithm, one must first convert the circuit into a graph. The Kernighan-Lin algorithm works as follows. The initial partition is generated at random. Then the two sub circuits *S1* and *S2* are created. If the circuit has *n* gates, the first *n/2* are assigned to *S1*, and the rest are assigned to *S2*. Because the gates in a circuit description appear in what is essentially a random order, the initial partition appears to be random [8]. The technique for generating new solutions from old solutions is to select a subset of gates from *S1* and a subset of gates from *S2* and swap them. To maintain acceptability, we always select two subsets of the same size.

C. Genetic Algorithms

GA was developed by John Holland (Holland, 1975) and since then has been used in various fields of engineering. GA has been used quite successfully for combinatorial problems that are NP-complete. More recently GA has been used for solving some VLSI problems. A genetic algorithm is a randomized parallel search method modelled on natural selection and genetics. In contrast to more standards search algorithms, GA bases their progress on the performance of a population of candidate solutions, rather than on a single candidate solution. The motivation behind this is that by simultaneously searching many areas of the design space the risk of getting stuck at local optima is greatly reduced. GA are probabilistic in nature and start off with a population of randomly generated candidates and evolve toward better solutions by applying genetic operators, modelled on the natural genetic process. For solving any problem, a population of possible solutions is maintained by the GA and this population undergoes evolution. In each generation relatively good solutions survive and reproduce while bad solutions tend to die off, and are replaced by the offspring of the good, which are also likely to be good.

D. Simulated Annealing

Simulated Annealing (Kirkpatrick et al., 1983) belongs to the class of non-deterministic algorithms. Kirkpatrik, Gelatt and Vecchi first introduced this heuristic in 1983. Simulated Annealing (SA) is a general iterative improvement algorithm that can be used for many different purposes. In partitioning, SA starts with a random partition. A new state is computed by selecting a gate at random from each of the two subsets, and swapping them. As before, the swap remains tentative, until the quality of the new partitioning is computed. The number of nets cut is the measure of goodness. If the new state is better than the old state, it is accepted and the swap is made permanent. If the new state is worse than the old state, it might be accepted and it might not. The SA algorithm operates in a series of distinct phases called temperatures. An actual temperature value is assigned to each phase. The algorithm begins with temperature set to a high value, and proceeds to lower and lower temperatures. A predetermined number of moves are attempted at each temperature. When a bad move is attempted, the algorithm computes an acceptance value that is based on temperature and on the badness of the solution. This acceptance value is compared to a random number to determine whether the move will be accepted. The random number is used to guarantee that there is always a non-zero probability that any bad move will be accepted. The higher the temperature, the more likely it is that a particular bad move will be accepted, and at a given temperature, the worse the move, the less likely it is to be accepted.

E. The Tabu Search

The Tabu Search technique was originally proposed in 1990 by Glover as an optimization tool to solve nonlinear covering problems. Tabu Search has recently been applied to problems such as integer programming, scheduling, circuit partitioning and graph coloring. In general terms, Tabu Search is an iterative improvement procedure that starts from some initial feasible solution (i.e., assignment of cells to blocks for the partitioning problem) and attempts to determine a better solution in the manner of a steepest descent algorithm. However, Tabu Search is characterized by an ability to escape local optima which usually cause simple descent algorithms to terminate by using a short term memory of recent solutions [14]. Tabu Search permits back-tracking to previous solutions, which may ultimately lead, via a different direction, to partitions with fewer cut nets. A Tabu Search implementation for circuit partitioning requires an initial feasible solution (partition), an associated cost in this case, the net cut, a list of Tabu solutions and a maximum number of moves.

F. Ant Colony Method

ACO is a novel population-based metaheuristic framework for solving discrete optimization problems. It is based on indirect communication among individuals of a colony of agents, called ants, mediated by trails of a chemical substance pheromone used by real ants for communication. It is inspired by the behaviour of real ant colonies, in particular, by their foraging behaviour and their communication through pheromone trails. Pheromone trails are a kind of distributed numeric information modified by ants to reflect their experience accumulated while solving a particular problem. Typically, solution components which are part of better solutions or are used by many ants will receive a larger

amount of pheromone and, hence, will more likely be used by the ants in future iterations of the algorithm. The collective behaviour is a form of autocatalytic behaviour. The process is thus characterized by a positive feedback loop, where probability with which ant chooses a solution component increases with the number of ants that previously chose the same solution component.

G. Memetics Algorithm

MA Algorithm blends different search strategies in a combined algorithmic approach. Like Evaluation Algorithms. MAs are population based metaheuristics. This means that MAs maintain a *population* of solutions for the problem at hand. It is assumed that both repairing and extension processes can be performed faster, as to justify including them in the population. In the context of MAs, the denomination *agent* representing a processing unit that can hold multiple solutions, and has problem-domain methods that helps to improve them if required. Each individual/agent represents a tentative solution/method for the problem under consideration. When the agents adapt their methods we call the resulting strategy an *adaptive memetic algorithm [12]*. Adaptation may include a modification of the data as in due to the agents interactions, solutions are subject to processes of competition and mutual cooperation.

H. DNA Based Algorithm

To solve the instance of Partitioning problem with $G= (V, E)$ $(|v| = n)$ start with $2n$ identical single stranded DNA memory strands each with $2n$ bit regions. The first n bit regions will represent the presence/absence of vertex in the first partition and the rest n bit regions will represent the presence/absence of an edge crossing the partition. The method uses the Sticker model of DNA computation. The main idea of this method is grouping the strands according to the output value to be set for a particular bit, and then set the output value as 0 or 1 accordingly. This process is repeated until all the digits of output value are stored. The result tubes are the tubes, which contain the result strands after completion of the annealing process with stickers [15]. To perform a specific operation on given input and operand, first a particular tube is selected from the result tubes corresponding to the operation. Then a particular memory complex is selected from that tube corresponding to the input and operand value under consideration

IV. PROBLEM FORMULATION

Let us take the VLSI partitioning problem for the demonstration of the proposed approach. This problem can be expressed more naturally in graph theoretic terms. A graph $G=(V, E)$ representing a partitioning problem can be constructed as follows. Let $V=\{v_1, v_2 \ldots v_n\}$ be a set of vertices and $E=\{e_1, e_2 \ldots e_m\}$ be a set of edges. Each vertex represents a component. There is edge joining the vertices whenever the components corresponding to these vertices are to be connected. Thus, each edge is a subset of the vertex set i.e., $e_i \subseteq V$, $i=1,2 \ldots m$. Let ***edge*** represents a function which when called with first vertex of edge, returns the second vertex of edge. The modelling of partitioning problem into graphs allows us to represent the circuit-partitioning problem completely as a graph-partitioning problem. The partitioning problem is to partition V into $V_1, V_2 \ldots V_k$ where

$$V \cap V_j = \varnothing, \quad i \neq j$$
$$\cup V_i = V$$

Theses partitions can be obtained by first efficiently partitioning the graph into two parts and then recursively applying the same approach. Partition is also referred to as a cut. The cost of partition is called the cutsize, which is the number of edges crossing the cut [8]. The constraints and the objective functions for the partitioning algorithms vary for each level of partitioning and each of the different design styles used. However, at the chip level, the partitioning algorithms usually have interconnections between partitions as an objective function.

The number of interconnections at any level of partitioning has 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. A large number of interconnections increase the design area as well as complicates the task of placement and routing algorithms. Minimization of the number of interconnections between partitions is called the mincut problem. The minimization of the cut is a very important objective function for partitioning algorithms for any level or any style of design. This function can be stated as

$$\sum_{i=1}^{k} \sum_{j=1}^{k} c_{ij} \quad , \quad (i \neq j) \quad \text{is minimized } c_{ij}$$

The represent the crossing edge from node i to node j crossing a partition. The mincut problem is NP complete, it follows that general partitioning problem is also NP complete [4].

V. SIMULATION RESULTS

The different criteria used in this analysis are the following:

- Min cut;
- CPU Efficiency Time;
- Load Balancing.

The simulation has been performed by taking the different sets of graphs for different values of Benchmark circuits as shown in Table 1. The table shows the circuit names with having different Inputs and Outputs. The total gates in each Benchmark circuit are shown. The average simulation time to find the number of Input and Output is also shown in Table. The corresponding comparison has been made between Divide and Conquer, Kernighan-Lin, Simulated Annealing, Genetic Algorithm, Ant Colony Algorithm, Memetics Algorithm and DNA based.

TA BLE I
THE BENCHMARK CIRCUITS FOR COMBINATIONAL DIGITAL CIRCUITS

Circuit Name	Number of Inputs	Number of Outputs	Total Gates	Simulation Time
c432	36	7	160	34
c499	41	32	202	55
c880	60	26	383	76
c1355	41	32	546	105
c1908	33	25	880	115
c3540	50	22	1669	173
c6288	32	32	2406	488

Table II compares the results of evolutionary algorithms, for the ISCAS-85 circuits. In comparison with the computations are carried out with parameters like cut size, CPU Efficiency and Load Balancing. The new evolutionary algorithm is showing the good results. The effectiveness of the algorithm is demonstrated using the simulation setup with the help of Chaco 2.0 Algorithm.

To demonstrate the effectiveness of the approach another set of simulation results were taken for some standard benchmark circuit's c432 and c3540 in various domains shown in Figure 1 and Figure 2. DNA based approach produced a smaller edge-cut for each graph and was much faster than the other approaches on the average (in terms of number of iterations). The Figure 3 and Figure 4 shows the CPU Efficiency and Load balancing which in turn is used for Fan In and Fan Out of the other circuits for partitioning.

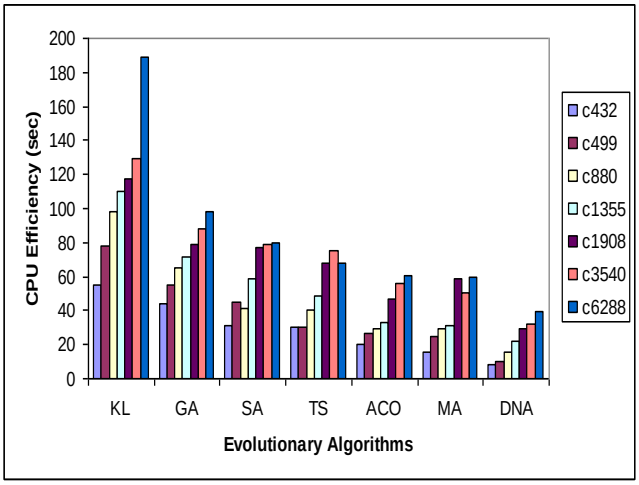


Fig. 3 CPU Efficiency using different Evolutionary Algorithms on Benchmark circuits.

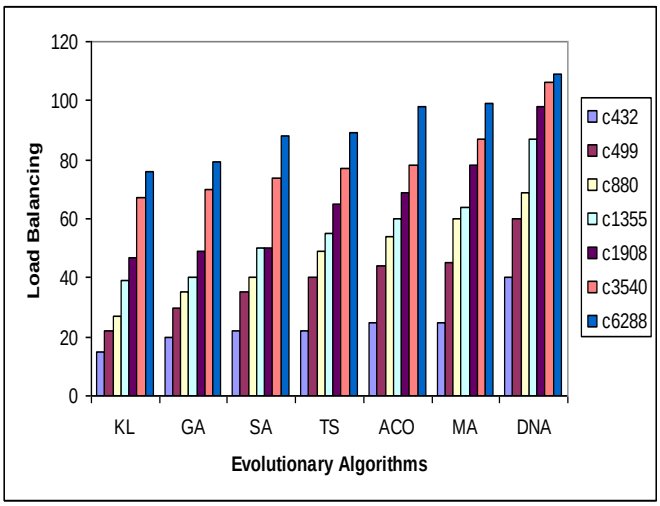


Fig. 4 CPU Efficiency using different Evolutionary Algorithms on Benchmark circuits

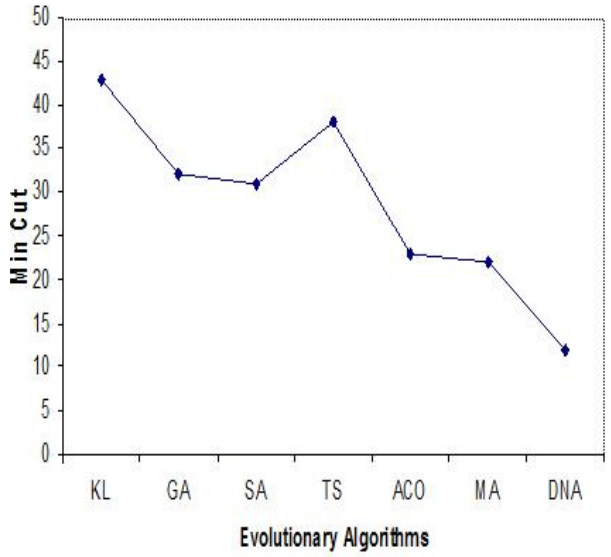
The outcomes clearly show that the new evolutionary approaches like DNA based approach is much faster and generates partitions with a smaller edge-cut for all graphs. Thus DNA based approach is much more effective, especially for large values of n as it always take less than n^2 iterations to find the required partition with a better edge cuts.

VI. CONCLUSIONS AND FUTURE SCOPE

The results show that the new evolutionary algorithms are able to partition the circuit graph taking less no. of iterations as compared to other available approaches. The new approaches might be found useful in VLSI circuit partitioning, circuit testing, logic minimization and simulation etc. In all of these applications the circuit-partitioning problem occurs. The graph-partitioning problem is an important component for mesh partitioning in the domain-decomposition method. The simulation results show that larger graphs, which are often encountered in mesh partitioning, we had to use a multilevel method to produce results that were competitive with the results given by other algorithms. There is a wide range of possibilities to be considered in the future. One of the most appealing is a merger of the non conventional computing methods like DNA approach with some other method through daemon actions and parallel implementation using parallelism. Furthermore, sequential circuits can be used to show the simulation results of these algorithms.

TABLE II
MIN CUT, CPU UTILIZATION AND LOAD BALANCING ON DIFFERENT BENCHMARK CIRCUITS

Circuit Name	Min Cut							CPU Efficiency Time (sec)							Load Balancing						
	KL	GA	SA	TS	ACO	MA	DNA	KL	GA	SA	TS	ACO	MA	DNA	KL	GA	SA	TS	ACO	MA	DNA
c432	43	32	31	38	23	22	12	55	44	31	30	20	16	08	15	20	22	22	25	25	40
c499	49	37	34	35	21	27	12	78	55	45	30	27	25	10	22	30	35	40	44	45	60
c880	79	55	45	41	37	32	19	98	65	41	40	29	29	16	27	35	40	49	54	60	69
c1355	89	78	66	44	39	32	19	110	72	59	49	33	31	22	39	40	50	55	60	64	87
c1908	97	78	55	66	35	40	21	117	79	77	68	47	59	29	47	49	50	65	69	78	98
c3540	124	100	92	78	55	54	25	129	88	79	75	56	50	32	67	70	74	77	78	87	106
c6288	178	116	98	89	78	77	30	189	98	80	68	61	60	39	76	79	88	89	98	99	109



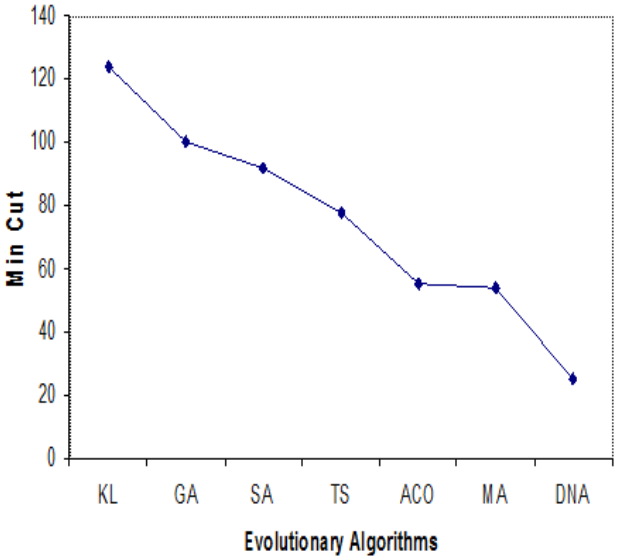


FIG. 1 AND FIG 2 SHOWS MINCUTS ON DIFFERENT ALGORITHMS WITH BENCHMARK CIRCUITS C432 AND C3540

REFERENCES

Trifunovi´c and W.J. Knottenbelt. (2004) "Towards a parallel disk-based algorithm for multilevel *k*-way hypergraph partitioning." In *Proc. 5th Workshop on Parallel and Distributed Scientific and Engineering Computing*, Santa Fe, NM, USA

Trifunovic and W. J. Knottenbelt. (2004) ".Parkway 2.0: A parallel multilevel hypergraph partitioning tool" In *Proc. 19th International Symposium on Computer and Information Sciences (ISCIS 2004)*, volume 3280 of *LNCS*, pages 789–800. Springer

C.A.C. Coello, G.T.Pulido and M.S. Lechuga (2004)., "Handling multiple objectives with particle swarm optimization", *IEEE Trans. on Evolutionary Computation*, Vol. 8, Issue 3, pp. 256-279.

J. Alpert and A. B. Kahng (1995). "Recent directions in netlist partitioning: A survey". *Integration: The VLSI Journal*, (19):1-81

J. Alpert, A. E. Caldwell, A. B. Kahng, and I. L. Markov, (2000), "Hypergraph Partitioning with Fixed Vertices*"*, *IEEE Transactions on Computer–Aided Design of Circuits and Systems*, 19 pp. 267–272.

Rayside, S. Reuss, E. Hedges and K. Kontogiannis (2000). "The effect of call graph construction algorithms for object-oriented programs on automatic clustering." In *Proc. of 8th International Workshop on Program Comprehension*, pages 191-200, IEEE.

Garey, M.R.,Johnson, D.S.,(1979) "Computers and Interactability" *A Guide to the Theory of NP-Completeness*, W.H. Freeman & Company. San Francisco.

Holland, J.,(1975)."Adaptation in Natural and Artificial Systems". Ann Arbor: University of Michigan Press.

Kernighan, B.W., Lin S.,(1970)." An Efficient Heuristic Procedure for Partitioning Graphs", *The Bell Sys. Tech. Journal*, pp 291-307.

Kirkpatrick, S., Gelatt, C.D., Jr, Vecchi, M.P.,(1983). "Optimization by Simulated Annealing. Science", Vol. 220, pp. 671-680.

Kumar, V., Grama, A., Gupta, A., Karypis, G.,(1994). "Introduction to Parallel Computing. Design and analysis of algorithms", *The Benjamin/Cummings*

Publishing company.

Lipton, R., (1996). "Speeding Up Computations via Molecular Biology". *1st DIMACS workshop on DNA based computers*. Princeton. In DIMACS series. vol.27, pp 67-74.

N. Krasnogor and J.E. Smith (2005). "A tutorial for competent memetic algorithms: Model, taxonomy and design issues". *IEEE Transactions on Evolutionary Algorithms*, 9(5):474–488,

R. Koschke and T. Eisenbarth. (2000) ".A framework for experimental evaluation of clustering techniques". *In Proc. of 8th International Workshop on Program Comprehension*, pages 201-210, IEEE.

Roweis, S., Winfree, E., Burgoyne, R., Chelyapov, N., Goodman, M., Rothemund, P., Adleman, L., (1998). "A Sticker-Based Model for DNA Computation". *Journal of Computational Biology*, 5(4) pp 615-629.