Digital Circuit Layout based on Graph Partitioning Technique using DNA Computing

Maninder Kaur
Assistant Professor, SMCA
Thapar University, Patiala

Kawaljeet Singh
Director, University Computer Center
Punjabi University, Patiala

ABSTRACT
After Adleman and Lipton have described the potential power of DNA computing, researchers have developed an interest in DNA computing for solving difficult computational problems like NP-complete problems. Partitioning of digital circuits also come under this category. Partitioning plays a key role in the design of a computer system. Existing conventional methods are unable to achieve the required breakthrough in terms of complexity, time and cost. This paper discusses how the potential of DNA can be used to solve the instances of circuit partitioning problem in an efficient way. A prototypical algorithm named DBACP is developed for solving the partitioning problem which is compared with SCAP approach on the small scale instances of circuit Benchmarks.

General Terms
DNA Computing, Tube, VLSI Circuits, Circuit Partitioning, Balance constraints

Keywords
Memory Strands, Extraction, Crossover, Mutation,

1. INTRODUCTION
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. The procedureat of decomposition is called partitioning. It plays a key role in the design of a computer system in general, and VLSI chips in particular. A computer system is comprised of tens of millions of transistors. It is partitioned into several smaller modules/blocks for facilitation of the design process. Each block has terminals located at the periphery that are used to connect the blocks. A VLSI system is partitioned at several levels due to its complexity. At the top level, it is partitioned into a set of sub systems whereby each subsystem can be designed and fabricated independently on a single PCB. The circuit partitioning problem arises in many VLSI applications [1, 2]. At any level of partitioning, the input to the partitioning algorithm is a set of components and a netlist. The result is a set of subcircuits which when connected, function as the original circuit and terminals required for each subcircuit to connect it to the other subcircuits. Other than maintaining the original functionality, partitioning technique optimizes certain parameters subject to certain constraints. The constraints for the partitioning problem include area constraints and terminal constraints [9, 11]. The objective function for a partitioning problem includes the minimization of the number of nets that cross the partition boundaries. Partitioning efficiency can be improved within three broad parameters.

• The system must be decomposed carefully so that the original functionality of the system remains intact.
• An interface specification is generated during the decomposition, which is used to connect all the subsystems. The system decomposition should ensure minimization of the interface interconnection between any two subsystems.
• Finally, the decomposition technique should be simple and efficient so that the time required for the decomposition is a small fraction of the total design time.

The present work concentrates on the DNA computing approach to solve the circuit partitioning problem.DNA computing like much of today's cutting-edge research is multidisciplinary. It involves mathematics, biology and computer science. In many papers, [6,7,8,13,14,15] it is stressed that high levels of collaboration between academic disciplines will be essential to gear up the progress in DNA computing. Many of the search problems in computer science are unsolvable not theoretically but because of astronomical resources required for their solution. DNA Algorithms can be applied to these problems. They can be used to extract statistics such as mean, median, minimum element from an unsorted data that can be used to speed up algorithms for NP problems. It can also be used to speed up the search for keys to crypt systems such as DES, PGP etc. The main disadvantage of these algorithms is the lack of hardware support and measurement difficulties. Though the current difficulties found in translating theoretical DNA computing approaches into real life solutions are never sufficiently overcome, there is still a great potential for other areas of development. Future applications might utilize the error rates and instability of DNA based computation approaches as a means of predicting and simulating the emergent behavior of complex systems. This could relate to weather forecasting, economics, and lead to more a scientific analysis of similar problems. Such a system might rely on inducing mutation and increased error rates through exposure to radiation and deliberately inefficient encoding schemes. The methods of DNA computing can serve as the most obvious medium for the use of evolutionary programming for applications in design of expert systems.

2. PROBLEM FORMULATION
The partitioning 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…m.

The modeling of partitioning problem into graphs allows us to represent the circuit-partitioning problem completely as a graph-partitioning problem. The Bipartitioning problem is to partition V into V_1, V_2 where

$$V_1 \cap V_2 = \Phi \; for \; i \neq j$$

$$r - \delta \leq \frac{|V_1|}{|V_1| + |V_2|} \leq r + \delta$$

In the proposed work the values r, the balance factor, =0.5 and ϭ=10%, the tolerance limit respectively.

|V1| and |V2| denotes the size of partitions such that

$$|V_1| = \sum_{v_i \in V_i} a(v_i) \; for \; j = 1,2$$

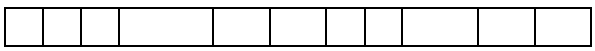
Where a(vi) with denotes the area of cell .The cost of partition is called the cutsize, which is the number of edges crossing the cut. The constraints and the objective functions for the partitioning algorithms vary for each level of partitioning and each of the different design styles used. However, 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. Minimizing 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 boost the design area as well as complicates the task of placement and routing algorithms. Reduction of the number of interconnections between partitions is called the mincut problem. The cut minimization is a very important objective function for partitioning algorithms for any level or any style of design. The mincut problem is NP complete [3].

3. THE PROPOSED METHOD DBACP

The basic idea behind the proposed algorithm DBACP (A DNA Based Approach for Circuit Partitioning) for the minimum balanced bisection problem is as follows:
Step 1: Construct a set T of all possible 2^n bipartition solutions of given circuit with n gates.
Step 2: Discard the solutions from the set T that don't satisfy balance constraint (reading weighted file), thereby generating a tube T of all possible feasible solutions.
Step 3: Based on to T and incidence matrix M of graph G, determine the cut edge set C of the partitions for all solutions in T.
Step 4: Retain the strands in tube T that have minimum edge cut.

To solve the instance of Partitioning problem with G=(V, E) (|v| = n,nodes, |E|=e ,interconnections) start with 2^n identical single stranded DNA memory strands each with (n+m) bit regions. The first n bit regions will represent the presence/absence of vertex in the first partition and the rest m bit regions will represent the presence/absence of an edge crossing the partition.



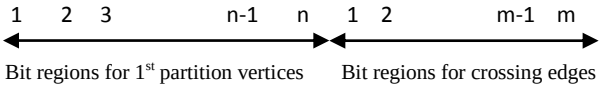


Fig 1: Bit regions of a DNA Strand

As shown in Fig 1, the n+m bit regions of DNA strands are represented by $v_1, v_2 \ldots v_n, e_1, e_2 \ldots e_{m-1}, e_m$. v_i is 1 if the vertex i is present in the first partition and otherwise it is zero. Similarly e_i is 1 if the edge is crossing the partition otherwise it is zero. The summation of $e_1, e_2 \ldots e_{m-1}, e_m$ will represent the total number of edges crossing the partition.

3.1 Algorithm

Step I Prepare a library .Design 2^n DNA strands, each with n+m bit regions (Initialize (T0,n)).

Step II Read the weight file and separate the strands that don't satisfy the balance constraints such that the tube T contains only strands of k feasible solutions i.e. a nonempty tube of k strands.

Step III Find the edge cut costs of every strand by using incidence matrix M of graph G, for all solutions in T by calling edge_cut() function
Step IV
Extracts the strands from the tube using dna_opt(T) function which has minimum edge cut count.
bool :=check(T) : Given a tube T , the function gives yes or no according as there is at least one DNA strand in T or not.
b := get(T): Given a tube T containing at least one strand, get randomly one strand s in T .

***Procedure**: edge_cut(T)*
begin
for every strand in a tube of feasible solution
for k:= 1 to n
if(get(k)=0)
for e:=1 to m
sum(e)=m[k][e]+sum(e);
for e:=1 to m
if sum(e)==1
set(edgebit(e))=1
else
set(edgebit(e))=0
end of for statement
end of for statement
end

***Procedure** s := dna_opt.(T)*
Input: T : a nonempty tube of (k) strands.
Output: s: a strand in T such that s contains a minimum edge cut cost.
begin
1. for i = 1 to k
2. (T1, T0) := extract(T , n+ i);
3. if check(T0)= yes, then
4. T:=rename(T0);
5. else T :=rename(T1);
6. s :=get(T);
end.

This improved efficiency is achieved due to the ability of DNA computing to evaluate all possible combinations simultaneously.

Table 1. shows the comparison of average cut(average runtime of SCAP and the proposed DBACP algorithm for the set of spp benchmark circuits series

No. of Processors	1		2		3		4	
Circuit Series ↓	SCAP	DBACP	SCAP	DBACP	SCAP	DBACP	SCAP	DBACP
spp-N10	4.293(0.00031)	3.06(0.000112)	3.889(0.000121)	3.06(0.000102)	3.40056(0.000109)	3.06(0.000092)	3.2034(0.000103)	3.06(0.000088)
spp-N15	5.23(0.000462)	4.36(0.0001192)	5.128(0.000317)	4.36(0.000118)	4.9818(0.000278)	4.36(0.000101)	4.567(0.00019)	4.36(0.000091)
spp-N20	7.39(0.001638)	5.204(0.002328)	6.16(0.001232)	5.204(0.001623)	6.009(0.001219)	5.204(0.001287)	5.89(0.001205)	5.204(0.000505)
spp-N25	8.07(0.00441)	6.18(0.013371)	7.94(0.00334)	6.18(0.010234)	7.88(0.003112)	6.18(0.009232)	7.62(0.002112)	6.18(0.002306
spp-N30	9.2890(0.009085)	7.75(0.0239083)	8.806(0.007902)	7.75(0.010915)	8.256(0.006805)	7.75(0.009649)	7.942(0.004964)	7.75(0.004818)

The figure 2 shows a random strand formed of circuit with 4 nodes and 3 edges after the initialization process having stickers annealed to v_1, v_3, v_4 and with last three bits for edges, with sticker annealed to e_2 representing the edge cut.

Fig 2: A random strand generated during the initialization process with annealed stickers

4. SIMULATION RESULTS

The DBACP algorithm is implemented in Linux environment running on parallel computing environment[12]. The performance of the proposed algorithm is tested on UCLA small circuit partitioning instances [4] generated by the top-down partitioning-based placement process employed by the UCLA Capo placer. The circuit net lists are in the net list format. (.net,.nodes,.wts,.blk files). The DBACP was tested against the circuits with the problem sizes range from 10 cells to 30 cells. DNA notations are developed utilizing the properties of massive parallelism and replications. A DNA Simulator software is developed which is used to obtain the results for the proposed DNA computing method.

The results of DBACP are compared with that of SCAP algorithm [5]. The SCAP approach was run using the parameters: population size as 10, crossover probability 0.6, mutation probability 0.02, with 50 as number of generations. Table 1 shows the comparison of average results of SCAP and DBACP algorithm. The algorithms were simulated using 1,2,3,4 processors and results present the average cut and average runtime. The average results have been obtained on multiple number of partitioning instance groups in each size range.

As seen from Table I, average results obtained by DBACP based partitioner are consistently better than these obtained by SCAP for small problem instances upto 20 nodes .With the increase in the size of problem instance the DBACP execution time increases in comparison to SCAP algorithm. The DBACP algorithm gives excellent quality of solution at the cost of running time of algorithm.

5. CONCLUSIONS AND FUTURE SCOPE

The results show that the proposed DNA algorithm is able to partition the circuit graph taking less no. of iterations as compared to SCAP approaches for small instances. SCAP Algorithm takes more amount of CPU time .In these situations DNA algorithms provide the better solution by taking the help of massive parallelism and recombination properties of DNA .at the cost of little increase in running time.

6. REFERENCES

[1] Alpert, C.J., Kahng, A., (1995). "Recent Developments in Netlist Partitioning: A survey", in Integration: the VLSI Journal, vol. 19, 1-18.

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

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

[4] http://vlsicad.ucsd.edu/GSRC/bookshelf/Slots/Partitioning/

[5] K.,Maninder, S., Kawaljeet (2011). "Soft Computing Approach for Digital Circuit Layout based on Graph Partitioning". International Journal of Computer Applications, Volume 15– No.1.

[6] J. Watada and R. B. A. Barkar, "DNA computing and its applications," in Proc. 8th Int. Conf. Intell. Syst. Des. Appl., Nov. 2008, pp. 288–294.

[7] J. Y. Lee, S. Y. Shin, T. H. Park, and B.-T. Zhang, "Solving traveling salesman problems with DNA molecules encoding numerical value," Biosystems, vol. 78, no. 1–3, pp. 39–47, Dec. 2004.

[8] Kahan, M.; Gil, B.; Adar, R.; Shapiro, E. (2008). "Towards molecular computers that operate in a biological environment". Physica D: Nonlinear Phenomena 237 (9): 1165–1172.

[9] Fiducia, C.M., and Mattheyses, R.M., (1982),"A linear time heuristic for improving network partitioning", in Proc ACM/IEEE Design Automation, 175-181.

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

[11] Kernighan, B.W., Lin S., (1970), "An Efficient Heuristic Procedure for Partitioning Graphs", the Bell Sys. Tech. Journal, 291-307.

[12] David Padua "Encyclopedia of Parallel Computing," Volume 4, 2011.

[13] S. Shin, I. Lee, D. Kim, and B. Zhang, "Multiobjective evolutionary optimization of DNA sequences for reliable DNA computing," IEEE Trans.Evol. Comput., vol. 9, no. 2, pp. 143–158, Apr. 2005.

[14] M. Darehmiraki and H. M. Nehi, "Molecular solution to the 0–1 knapsack problem based on DNA computing," Appl. Math. Comput., vol. 187, no. 2,pp. 1033–1037, 2007.

[15] 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, 4, 615-629.