National Conference on Intelligent Systems
Hyderabad, 24-25 August 2007

A GENETIC ALGORITHM FOR NON-SLICING FLOORPLAN REPRESENTATION

Debarshi Chatterjee and Theodore W. Manikas
The University of Tulsa
600 South College Avenue, Tulsa, OK-74104, U.S.A.
{debarshi-chatterjee, theodore-manikas}@utulsa.edu

Abstract. Floorplanning is one of the most significant stages of the Very Large Scale Integrated (VLSI) circuit design cycle. Genetic algorithms (GA) provide immense opportunity to efficiently solve the floorplanning problem. Previous applications of GA to the floorplanning involve slicing floorplan representation and are therefore limited in application. We develop a GA for floorplan area optimization by applying permutation type crossover and mutation operators to the integer string encoding generated by Sequence Pair (SP), a non-slicing floorplan representation. In addition, we carry out a comprehensive comparative study of the performance of GA operators and selection schemes. In order to select the best crossover operator, we compare the efficacy of Partially Mapped, Order1 and Cycle crossovers. We also study the performance of Swap Mutation, Insert Mutation and Invert Mutation. Finally, we compare the convergence of Roulette Wheel, Rank and Binary Tournament Selection and determine the optimal values for population size and other GA parameters. It is observed that our floorplanner generates better results for 3 out of 5 MCNC benchmark circuits as compared to Parquet.

Key Words: Genetic Algorithm, Optimization, Computer Aided Design.

1. INTRODUCTION

Floorplanning determines the relative location of circuit components (commonly known as blocks or modules) on a chip. It plays a vital role in determining the performance and cost of a chip. Let $p = \{B_1, B_2, ...B_N\}$ denote a set of N rectangular blocks with constant areas denoted by $A(B_i)$. Then, a *packing of* p is defined as a non-overlapping placement of the blocks, and the minimum bounding rectangle of the is called the *chip*. The difference between the chip area and the sum of all block areas $\sum_{i=1}^{N} A(B_i)$ is known as the *dead-space* and is usually represented as a percentage of the total chip area. The floorplanning problem aims at finding a packing of p that minimizes the chip area. Reducing the chip-area reduces the silicon cost required to manufacture the chip.

There are two major types of floorplans: slicing floorplans and non-slicing floorplans. Slicing floorplans are those that can be partitioned by recursively bisecting using horizontal and vertical cuts. A slicing floorplan can be represented using a slicing tree, as shown in Fig. 1. The slicing tree can be encoded into a normalized Polish expression that hastens the search procedure [1]. Other floorplan representation such as SP [2], Bounded Slicing Grid [3], O-tree [4] and B* tree [5] are more general in the sense that they are capable of handling non-slicing floorplans.

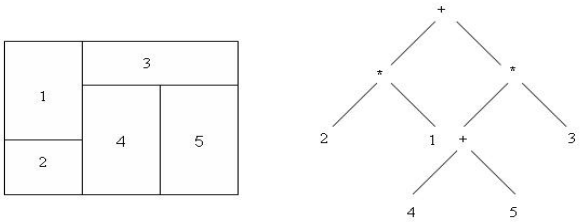

Fig. 1. A Slicing floorplan and the corresponding slicing tree.

Simulated Annealing (SA) [6] [7], Force directed approach [8] and numerical optimization techniques [9] are popular in determining the optimal floorplan. The application of GA to floorplanning was first done by Cohoon et al. [10]. In [10] the authors studied four different crossover operators when applied to post fix expressions. The GA developed by Schnecke et al. [11] carried out a direct manipulation of the slicing tree to maintain legality. Valenzuela et al. [12] bred normalized Polish expressions using GA and carried out a comparative study of the performance of four crossover operators. Till date, all the studies relating to the application of GA to floorplanning uses slicing tree representation. This renders the solution space P-inadmissible because the minimum area floorplan may not be slicing and hence may not be present in the solution space. Our research applies GA to SP, a non-slicing floorplan representation. The advantage of using SP for our purpose is two fold. First, it makes the solution space P-admissible. Second, it makes the comparison of our results with those obtained from Parquet [6] more meaningful, since Parquet also uses SP representation.

2. SEQUENCE PAIR REPRESENTATION

It is possible to represent a floorplan with N blocks by a pair of sequences (S_1, S_2) each having N elements. This pair imposes certain constraints on the relative positioning of the blocks on chip. For every element $x \in S_1, S_2$ we can find a set $M^{bb}(x)$ where:

$$M^{bb}(x) = \{x' \mid x' \text{ is before } x \text{ in } S_1 \text{ and } S_2\} \quad (1)$$

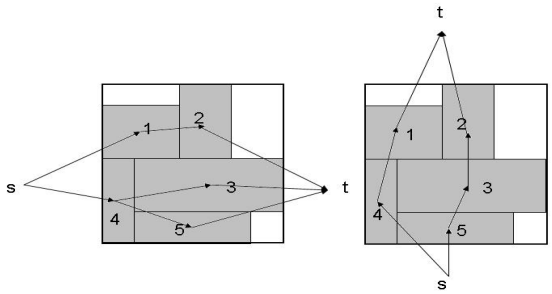

Fig. 2. Horizontal and Vertical Constraint Graphs corresponding to a floorplan

Any element $x' \in M^{bb}(x)$ is located left of x in the floorplan. Based on the "left of" constraint imposed by the set $M^{bb}(x)$, a horizontal constraint graph $G_H(V,E)$ can be constructed as follows: The vertex set consist of a source ***s***, sink ***t*** and N vertices labeled with module names. Directed edges are drawn from source to each of the vertices, which in turn are connected to the sink ***t***. A directed edge exists between two vertices x_i and x_j in $G_H(V,E)$ iff $x_i \in M^{bb}(x_j)$. Vertex weights are zero for source and sink and equal to the module width for the rest of the vertices. Fig. 2 shows a floorplan along with its corresponding horizontal and vertical constraint graphs from which the chip width and height can be computed.

3. GENETIC ALGORITHM

GA was first proposed by Holland in 1975 [14]. In nature only fittest individuals survive and reproduce, a natural phenomenon known as "the survival of the fittest". GAs mimic the natural evolution process by suppressing inferior genotypes and breeding offspring from superior population members. The cyclic process is continued for several generations and the best member is selected. The performance of a GA can be drastically improved by using elitism, which ensures that best-known solution is preserved and passed on to future generations. A pseudo code for our elitist GA is given below (Algorithm 1).

3.1 Fitness Assignment

In order to create the mating set, each member in the population is assigned a fitness value given by:

$$F = \frac{\sum_{i=1}^{N} A(B_i)}{A(p)} \quad (2)$$

Where $A(B_i)$ is the area of the ith block and $A(p)$ is the area of the minimum packing of p imposed by the sequence pair (S_1, S_2). $A(p)$ can be computed from (S_1, S_2) using the longest path algorithm [15] that has a time complexity of $O(n^2)$.

3.2 Selection Schemes

In this paper, we study the performance of the following selection schemes:

(i) *Roulette Wheel Selection (RWS):* In this scheme, the probability of selection of a member in a population is proportional to its fitness value.
(ii) *Rank Selection (RS):* In this scheme, the members are ranked first, with the highest numerical rank being assigned to the fittest member. The members are then selected for mating on the basis of their ranks.
(iii) *Binary Tournament Selection (BTS):* In this scheme, two population members are chosen at random and their fitness values are compared. The member with higher fitness value is included in the mating set. For details on operators and selection schemes refer to [13].

Algorithm 1. Floorplan Area Optimization

Begin
input: Block Dimensions, $w_i \times h_i \quad \forall i \in \Pi$ /*MCNC Benchmark files*/
output: Block Coordinates, $x_i \times y_i \quad \forall i \in \Pi$ /*Layout Files*/
1: Initialize population by assigning random pair of permutations.
2: Create empty external population P_{ext} with max. size N_{ext} to store the best members.
3: **if** number of members in $P_{ext} > N_{ext}$
4: Delete the worst member in P_{ext}
5: Select members from the current population
6: Apply crossover
7: Apply mutation
8: Replace a small proportion of the new population by random members of P_{ext}.
9: **if** (no of generations < constant1 or fitness of best member in P_{ext} < constant2)
10: Go to step 3.
end

3.3 Crossover Operators

Using SP representation, a floorplan can be encoded into a pair of strings. However, instead of binary bit strings, the problem gives rise to an integer string (permutation of N integer without repetition). In case of SP representations, we can use any one of the following permutation crossover mechanisms for generating a valid offspring [13]:
(i) *Order1 Crossover (Order1X):* In order1X an arbitrary portion of the first parent (P1) is copied to the first child (C1). Next, starting from the end point of the copied part, elements are copied from the second parent (P2) to C1 in the order in which they appear in P2, as shown in Fig. 3(a). An element is dropped in case it matches any of the elements copied from P1. The same procedure is repeated for the second offspring (C2) with the role of the parents reversed.
(ii) *Partially Mapped Crossover (PMX):* The entire procedure is explained in Fig. 3(b). Please refer to [13] for details.
(iii) *Cycle Crossover (CycleX)*: is explained in Fig. 3(c). Please refer to [13] for details.

3.4 Mutation Operators

The three mutation operators suitable for SP representation are as follows:
(i) *Swap Mutation:* Swap Mutation exchanges two arbitrary elements in a chromosome.

(ii) *Insert Mutation:* Insert Mutation picks two allele values at random and shifts the second allele until it abuts the first one while maintaining the relative order of all other alleles.
(iii) *Inversion Mutation:* Inversion mutation reverses the order of elements lying between two randomly selected alleles.

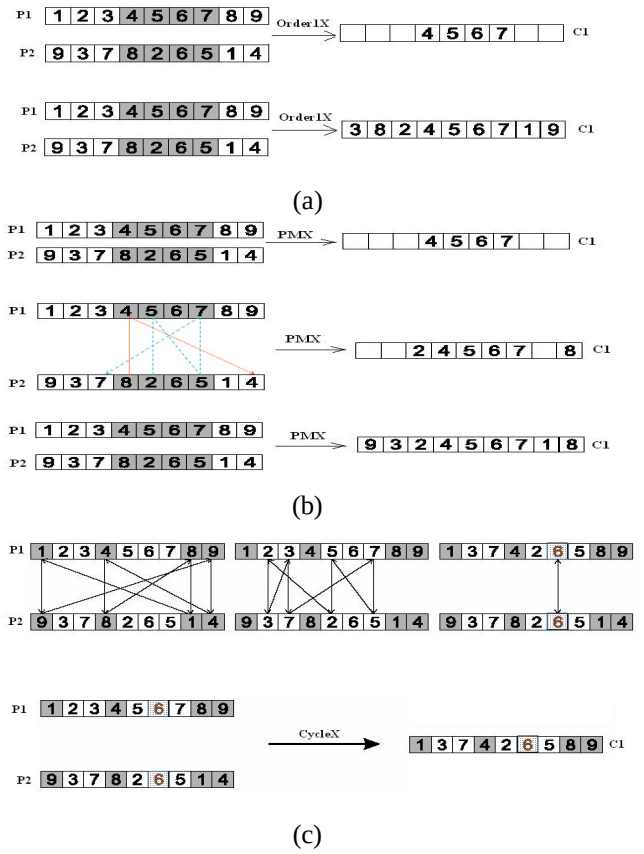

Fig 3. Crossover operators (a) Order1X (b) PMX and (c) CycleX

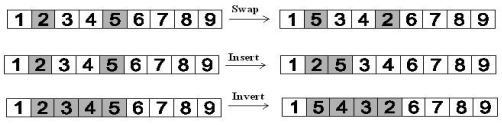

Fig 4. Various Mutation operators

4. SIMULATION RESULTS

We test the performance of Algorithm 1 for floorplan area optimization. The GA was applied on MCNC benchmark circuits. Details of each circuit including the number of modules and the total block area are given in Table 1. Simulations were carried out on a Pentium IV computer running at 1.8GHz with a 1GB RAM. Our GA generated compact layouts for each benchmark circuit. One such layout for ami33 is shown in Fig. 5.

In order to compare crossover operators, mutation operators and selection schemes, the GA was run several times, each time using a different operator or selection scheme within the GA and the minimum percentage deadspace (PDS) out of 50 independent runs of the GA was recorded. For the crossover operators, we compared the performance of Order1X, PMX and CycleX [11]. The minimum PDS produced by the various crossover operators are compared in Fig. 6. PMX produces the smallest value for minimum PDS when the results are averaged over the MCNC benchmarks. We also studied the performance of Swap Mutation, Insert Mutation and Inversion Mutation. Fig. 7 demonstrates that for the majority of the benchmarks, swap mutation produces the smallest value of minimum PDS. Hence, it is most effective in guiding the GA towards a global minimum. Finally, we study the effects of RWS, RS and BTS in generating the optimal floorplan. All three selection schemes work equally well for circuits having smaller number of modules. A major drawback of RWS is that it causes the GA to converge prematurely. This effect becomes prominent for larger circuits like ami33 and ami49. Further, it was found that the sorting overhead in RS slightly increases the runtime of the GA without guaranteeing an equally promising reward in terms of decreasing the area. BTS produced the minimum area in 4 out of 5 benchmarks as compared to the other selection schemes (Fig. 8).

Table 1. Details of the benchmark circuits.

MCNC Benchmarks	# Modules	Area(μm^2)
apte	9	46.5616
xerox	10	19.3503
hp	11	8.83058
ami33	33	1.15645
ami49	49	35.4454

Table 2. Comparison of the area and percentage deadspace obtained using Parquet and our floorplanner

MCNC Bench-marks	Parquet Results		Our Floorplanner	
	Area (mm^2)	Percentage deadspace	Area (mm^2)	Percentage deadspace
apte	51.81	10.13	47.52	2.03
xerox	22.09	12.4	20.55	5.85
hp	9.59	7.92	9.23	4.32
ami33	1.25	7.48	1.32	12.68
ami49	38.89	8.86	39.33	9.87

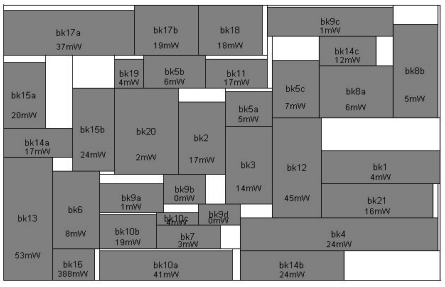

Fig. 5. Floorplan of Ami33 benchmark generated by our GA

In order to study the impact of population size on the performance of GA, we varied the population size from 4-50

in equal steps. A crossover probability of 0.7 and mutation probability of 0.01 were used in all the experiments. For each population size, we plotted the runtime of the GA per iteration (in ms) and the corresponding value of mean PDS. It is evident from Fig. 9 that increasing the population size caused a linear increase in the runtime of the GA. On the other hand, it improved the final result due to an increased diversity of the initial population. From Fig. 9, it can be observed that if the population size is increased beyond 20, then the improvement in PDS < 0.1%, but the runtime increases drastically. The optimal population size is thus identified as 20.

Based on the results obtained so far, we refined our Elitist GA by selecting the best operators and selection scheme. The final version of our floorplanner uses PMX, Swap Mutation, BTS and a population size of 20 members. We now compare results obtained by our floorplanner with those obtained from Parquet [6]. Parquet is a fast floorplanner that uses SP representation and SA. The chip area and PDS produced bu our floorplanner and Parquet are compared in Table 2. Simulation results show that our GA based floorplanner generates significantly better floorplans for apte, xerox and hp benchmark circuits and comparable results for ami33 and ami49.

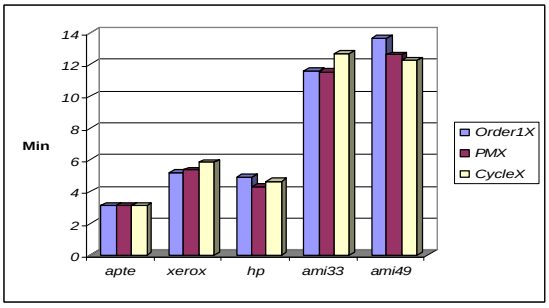

Fig. 6. Comparison of the performance of various crossover operators

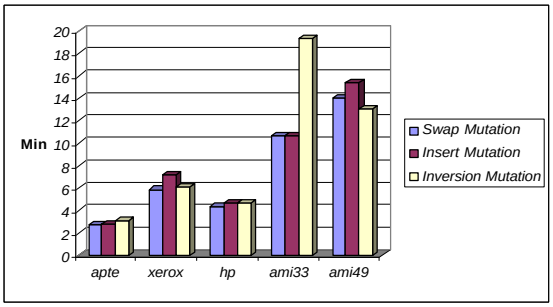

Fig. 7. Comparison of the performance of various mutation operators

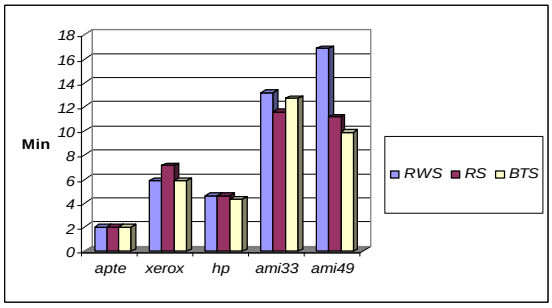

Fig. 8. Comparison of the performance of various mutation operators

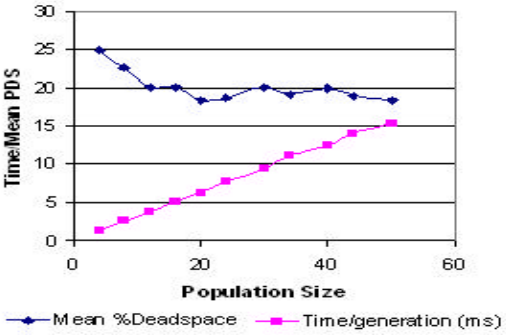

Fig.9. Run time and minimum percentage deadspace versus population size.

5. REFERENCES

[1] D.F. Wong & C.L Liu, A new Algorithm for Floorplan Design, *Design Automation Conference*, 1986, 101-107.

[2] H. Murata, K. Fujiyoshi, S. Nakatake and Y. Kajitani, VLSI Module Placement Based on Rectangle-Packing by Sequence Pair, *IEEE TCAD, Vol. 15*(12), 1996, 1518-1524.

[3] S. Nakatake, K. Fujiyoshi, H. Murata and Y. Kajitani, Module Placement on BSG Structure and IC Layout Applications, *ICCAD*, 1996, 484-491.

[4] P.-N Guo, C. -K. Cheng and T. Yoshimura, An O-tree Representation of Non-slicing Floorplan and its Applications, *DAC*, 1999, 268-273.

[5] Y.-C. Chang, Y.-W. Chang, G.-M. Wu and S.-W. Wu, B*-Trees: A new Representation for Non-Slicing Floorplans, *DAC*, 2000, 458-463.

[6] S. N. Adya and I.L Markov, Fixed Outline Floorplanning: Enabling Hierarchical Design, *IEEE Transactions on VLSI*, *11*(6), 2003, 1120-1135. http:/vlsicad.eecs.umich.edu/BK/parquet

[7] C. Sechen and A. Sangiovanni-Vincentelli, TimberWolf Placement and Routing Package, *IEEE Journal of Solid State Circuits*, *20*(2), 1985, 510-522.

[8] F. M. Johannes, K. M. Just and K. J Antreich, On the force Placement of Logic Arrays, 6th *European Conference on Circuit Theory and Design*, 1983, 203-206.

[9] F.Y. Young, Chris C.N. Chu, W.S Luk and Y.C. Wong, Floorplan Area Minimization using Lagrangian Relaxation, *International Symposium on Physical Design*, 2000.

[10] J. P. Cohoon, Distributed Genetic Algorithms for the Floorplan Design Problem, *IEEE Transactions on Computer Aided Design*, *10*(4), 1991, 483-492.

[11] V. Schnecke and O. Vornberger, Genetic Design of VLSI Layouts, *International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications*, 1995, 430-435.

[12] C.L. Valenzuela & P.Y. Wang, VLSI Placement and Area Optimization using a Genetic Algorithm to Breed Normalized Postfix Expressions, *IEEE Transactions on Evolutionary Computation*, *6*(4), 2002, 390-401.

[13] A.E Eiben and J.E Smith, *An introduction to evolutionary computing* (New York: Springer, 2003).

[14] J. H Holland, *Adaptation in natural and artificial systems* (Ann Arbor: University of Michigan Press, 1975).

[15] Nicos Christofides, *Graph theory- an algorithmic approach* (London: Academic Press, 1975).