Genetic Algorithm for Circuit Partitioning

ZOLTAN BARUCH, OCTAVIAN CREȚ, KALMAN PUSZTAI

Computer Science Department, Technical University of Cluj-Napoca,
26, Barițiu St., 3400 Cluj-Napoca, Romania
{Zoltan.Baruch, Octavian.Cret, Kalman.Pusztai}@cs.utcluj.ro

Abstract. In computer-aided logic design, partitioning is the task of clustering circuit elements into groups so that a given objective function is optimized with respect to a set of design constraints. In this paper we present a genetic algorithm for the circuit partitioning problem. The objective of the proposed algorithm is not only to balance the size of the two portions, but also to evenly distribute the connections among them. The implemented algorithm has been compared to a simulated annealing partitioning algorithm. Experiments show that the execution time of the genetic algorithm is lower, the results being comparable to that obtained by simulated annealing.

1. Introduction

In computer-aided design, partitioning is the task of clustering objects into groups so that a given objective function is optimized with respect to a set of design constraints. Partitioning is used frequently in physical design; for example, at the layout level it is used to find strongly connected components that can be placed together in order to minimize the layout area and propagation delay. In the layout synthesis process of FPGA circuits, the partitioning is used in the placement step, which assigns each node of the Boolean network to a specific logic block in the FPGA device. Partitioning can also be used to divide a large design into several chips to satisfy packaging constraints.

The partitioning techniques are based on a graph model of the design. Application of these basic partitioning methods requires the mapping of design structures into graph models. Each node in the graph represents a physical component, and each edge represents a physical connection between two components. The main objective of partitioning is to decompose a graph into a set of subgraphs to satisfy the given constraints, such as the size of the subgraph, while minimizing an objective function, such as the number of edges connecting two subgraphs.

The partitioning problem is shown to be NP-complete [1]. Exact algorithms for solving this problem, using techniques like dynamic programming, are found to be computationally expensive. Hence, approximate solutions are generally obtained in polynomial time using various heuristics. Simulated annealing (SA) has been employed for such hard optimization problems. A logarithmic temperature scheduling for SA is proved to ensure a global optimum solution. This requires a very slow cooling schedule, and often this makes SA prohibitively slow. Therefore, an efficient algorithm for the partitioning problem is desirable.

An efficient paradigm for solving hard optimization problems is the genetic algorithm (GA). J. H. Holland proposed that the simulation of natural evolution in which population undergoes adaptation, under controlled evolution strategy, can be an efficient alternative approach for solving complex optimization problems. Genetic algorithms work with a population of solutions. By the simulated evolution process of GA, over a number of generations, the candidate solutions retain the better characteristics of multiple solutions of earlier generations. This intrinsic parallelism provides the GA with an advantage over simulated annealing [2] which uses one single solution.

Because of such advantages, we are motivated to attempt a solution to the circuit partitioning problem by GA. The experimental results show the efficacy of GA for the partitioning problem.

The rest of this paper is organized as follows. In the next section, we formulate the circuit partitioning problem. In Section 3, we describe some related work. In Section 4, we describe the genetic algorithm for circuit partitioning. Conclusions are presented in Section 5. References are listed at the end.

2. The Circuit Partitioning Problem

The circuit partitioning problem can be formulated as a graph partitioning problem. A standard mathematical model of the circuits associates a graph $G = (V, E)$ with the circuit netlist, where vertices in V represent modules, and edges in E represent signal nets. The vertices and edges of G may be weighted to reflect module area or importance of a wiring connection.

Two basic formulations for circuit partitioning are the following:

- *Minimum Cut:* Given $G = (V, E)$, partition V into disjoint subsets U and W such that $e(U, W)$, i.e., the number of edges in $\{(u, w) \in E \mid u \in U, w \in W\}$, is minimized.
- *Minimum-Width Bisection:* Given $G = (V, E)$, partition V into disjoint subsets U and W, with $|U| = |W|$, such that $e(U, W)$ is minimized.

Because minimum-width bisection divides modules so that their number is equal in each partition, it is a more desirable objective for real circuits.

The more general partitioning problem is when k disjoint subsets are formed. This is called k-way partitioning and can be defined as follows:

- Given $G = (V, E)$, where each vertex $v \in V$ has a size $s(v)$, and each edge $e \in E$ has a weight $w(e)$, partition V into k subsets $V_1, V_2, \ldots, V_k$, such that a given objective function is optimized, taking into account a number of constraints.

Because circuit nets often have more than two pins, the netlist is more generally represented by a *hypergraph* $H = (V, E')$, where hyperedges in E' are the subsets of V contained by each net. A large segment of the literature treats graph partitioning instead of hypergraph partitioning since the formulation is simpler, and many algorithms are applicable only to graph inputs. In this paper, we will discuss graph partitioning.

3. Related Work

Many approaches have been proposed for the circuit partitioning problem. They include group swapping [3], [4], simulated annealing [2], network flow [5], eigenvector decomposition [6], etc.

Kernighan and Lin proposed a two-way partitioning algorithm with constraints on the subset size. This algorithm randomly starts with two subsets, and pairwise swapping is iteratively applied on all pairs of nodes. Subsequently, many improvements have been made to this method. Schweikert and Kernighan proposed the use of a net cut model so that the algorithm can handle multipin net cases [3]. Fiduccia and Mattheyses improved this algorithm by reducing time complexity to $O(P)$ with respect to the number of pins P, and Krishnamurthy [4] further added in lookahead ability. The Kernighan-Lin based algorithm is quite efficient but it needs a predefined subset size to start with.

Simulated annealing [2] is another method based on iterative improvement. The objective function in simulated annealing is analogous to energy in a physical system, and each move is analogous to changes in the energy of the system. The Metropolis Monte Carlo method is used to decide whether a move is accepted. Simulated annealing usually produces good results at the expense of very long running time.

The maximum-flow-minimum-cut algorithm was presented by Ford and Fulkerson. They transformed the minimum cut problem into the maximum flow problem [5]. In order to separate a pair of nodes into two subsets, the minimum number of crossing edges is equal to the maximum amount of flow from one node to the other. Although this algorithm can find the optimum solution between any pair of nodes in a network, there is no constraint on the sizes of resultant subsets. In practice, the result is not useful if two very unevenly sized subsets are generated.

In eigenvector decomposition [6], connections are represented in a matrix. The eigenvectors of the matrix define the locations of all components and thus derive partitioning results. This method requires the transformation of every multipin net into several two-pin nets in real circuits before establishing the matrix.

Several studies using genetic algorithms (GAs) for the partitioning problem have been done [7], [8]. These algorithms are motivated by the principles of natural selection. GAs generate potential solutions derived from solutions which were previously well ranked and mutated by genetic operators such as crossover, mutation, and inversion. GAs are effective for solving NP-complete problems, but their effectiveness is greatly dependent on the representation of the solution space. They are inherently parallelizable since numerous solutions can be generated and tested simultaneously.

Chandrasekharam *et al.* [7] described a GA for the node partitioning problem (NPP), with applications in VLSI design. This algorithm was used to solve several graph-theoretic problems, including graph colouring, clique cover, and maximal clique, which may be viewed as instances of a general NPP. Three applied problems in VLSI design, which are instances of NPP, were solved using this algorithm. The problems include test scheduling for “built-in self test” (BIST) schemes of VLSI circuits, input encoding of finite state machines or digital logic in control synthesis, and the determination of row and column folding to obtain an optimal area PLA.

Bui and Moon [8] described a hybrid GA for the graph partitioning problem. The algorithm include a fast local improvement heuristic. One of the features of the algorithm is the schema preprocessing phase that improves the solution space searching capability of the GA.

4. Genetic Algorithm for Circuit Partitioning

4.1. Overview of the Algorithm

The GA for the circuit partitioning problem (CPP) may be briefly described as follows. The algorithm starts with an initial population of partitions. In each generation, offspring partitions are obtained by a process of crossover between two parent partitions, at a specific crossover rate. A process of mutation is also used for changing the structure of few selected partitions in the population in each generation. From the initial partitions and the offspring partitions a set of partitions are selected as the initial population of the next generation. This process is continued for a number of generations. At the end the best partition available in the population is chosen as the solution for the CPP.

The only metric in the cost function in traditional partitioning algorithms, applied for the placement of FPGA circuits, is the cut size. However, the cut size alone is not a good metric for architectures with limited routing resources, such as FPGA circuits. Since the algorithm tries to place connected blocks close together, it is possible to obtain a partition with a small cut size, with one portion being heavily connected and the other being very sparse [9]. The proposed GA not only balances the size of the two portions, but also evenly distributes the connections among them. We define the unbalancing number of a net to be the number of connecting paths needed to connect all the terminals in the left portion minus the number of connecting paths needed in the right portion. The unbalancing number of a bi-partition is defined to be the sum of the unbalancing numbers of all nets.

The following fitness function is used to incorporate the effect of congestion distribution in a given partition:

$$F_i = 1 / (Cut\_size + WEIGHT \times Unbalancing\ number)$$

where *WEIGHT* is a constant. If *WEIGHT* is set to zero, then the algorithm is similar with a conventional partitioning algorithm. By setting the value of *WEIGHT* appropriately, we can control the importance of balancing the congestion.

The process of genetic evolution essentially modifies the structure of elements in the population. Therefore the structural representation of the solution is important. The structure contains smaller building blocks, referred to as 'genes'. For an efficient GA, in addition to a proper solution structure, it is necessary that the initial population of solutions be a diverse representative of the search space. The genetic operators crossover and mutation should be efficient. Also the strategy of next generation population selection should ensure the survival of better solutions in the long term and, at the same time, allow every new solution to compete for survival in the short term. These aspects of the GA are detailed as follows.

4.2. Solution Representation

Solution representation has a major role in designing an efficient GA. This representation should permit a large diversity in a small population. In the same time, the solution representation should allow easy crossover and mutation operations, and easy computation of the objective function.

Based on these criteria, we have chosen a string of subsets of compatible nodes as the solution structure. The subsets in a solution are disjoint. Since the objective function is primarily determined by a cover of valid subsets, the number of subsets in the partition are chosen as one field in the solution structure. This is shown in Fig. 1.

P_1	P_2	.	.	P_l	.	.	P_k	No. of subsets

Fig. 1. Solution representation of CPP.

4.3. Initial Population Generation

The GA starts with a random initial population of solutions. This population is obtained by generating random covers of compatible subsets of nodes. A random choice of size of each subset in the cover contributes to some diversity in the population. Experimentally the population of solutions are verified for diversity, and the method of initial population generation is observed to give adequate diversity.

4.4. Genetic Operators

Crossover is defined as an operator by which an offspring solution is obtained from two candidate solutions of a population (parent solutions). If Π_a and Π_b are two candidate solutions, representing valid partitions, the crossover operation is defined as $\Pi_a \times \Pi_b \rightarrow \Pi_o$, where Π_o, the offspring, is another valid partition. The crossover combines schemata from both parents, and therefore the offspring inherits some of the characteristics of the parents.

The crossover operator with solution structures as strings of subsets is given in Fig. 2. The offspring partition is constructed in three parts. The first part includes all subsets from Π_a up to the random cut l. The second part is from Π_b from the random cut m forward. The third part is from the subsets of the remaining nodes to be added for making the partition a valid cover. No preliminary knowledge is necessary regarding the number of disjoint subsets of nodes. This makes the choice of solution representation and corresponding crossover operator suitable for this GA.

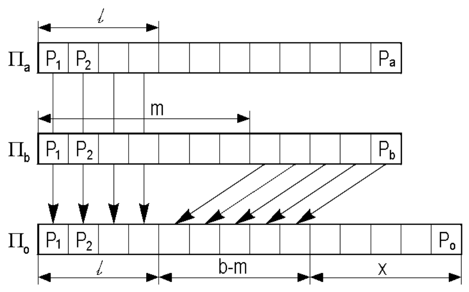

Fig. 2. Crossover operator.

If a certain solution reproduces for a large number of times in the early stages of the GA, it is likely that all solutions in the population would inherit some features of this solution and so tend to be the same. Hence, the parent selection strategy employed should avoid this premature convergence of the GA to a local optimum. We achieved this by not permitting any better fit parent to reproduce more than once in a single generation.

Mutation is implemented as follows. Depending on the mutation rate, a few nodes are selected randomly from subsets, and placed in the other possible compatible subsets. If a compatible subset is not found, a new subset is added with such a node. If this results in emptying any subset, a compaction in the subset list is performed. Such low level mutation would disturb the solution structure by only a small amount. This mutation process would permit population diversity to be maintained in later stages of the GA. Mutation also helps the GA to surmount any local optimum.

4.5. Next Generation Population Selection

The conventional GA employs the approach of generating two offspring from two parents and replacing the parents by the two offspring for the next generation process. However, we employed a different evolution strategy. In our approach, all offspring's replace an equivalent number of worst solutions in the current population. This strategy helps the survival of any better solution over many generations. The solutions generated by the genetic operators compete for at least one generation. This approach to next population selection provides an efficient performance for the GA.

5. Conclusions

In this paper we presented a GA for the circuit partitioning problem. The algorithm not only balances the size of the two portions, but also evenly distributes the connections among them. As the solution structure, a string of subsets of compatible nodes has been chosen. The parent selection strategy employed avoids the premature convergence of the GA to a local optimum. The mutation process permits population diversity to be maintained. The algorithm has been implemented and compared to a simulated annealing partitioning algorithm. Experiments show that the execution time of the genetic algorithm is lower, the results being comparable to that obtained by simulated annealing.

6. References

[1] S.M. Sait and H. Youssef, "VLSI Physical Design Automation", McGraw-Hill Book Company, 1995

[2] S. Kirkpatrick, C.D. Gelatt and M.P. Vecchi, "Optimization by Simulated Annealing", Science, No. 220, pp. 671-680, May 1983

[3] Y.C. Wei and C.K. Cheng, "Ratio Cut Partitioning for Hierarchical Designs", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 10, No. 7, pp. 911-921, July 1991

[4] B. Krishnamurthy, "An Improved Min-Cut Algorithm for Partitioning VLSI Networks", IEEE Transactions on Computers, Vol. 33, No. 5, pp. 438-446, May 1984

[5] H.H. Yang and D.F. Wong, "Efficient Network Flow Based Min-Cut Balanced Partitioning", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 15, No. 12, pp. 1533-1540, December 1996

[6] S.W. Hadley and B.L. Mark, "An Efficient Eigenvector Approach for Finding Netlist Partitions", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 11, No. 7, pp. 885-892, July 1992

[7] R. Chandrasekharam, S. Subhramanian and S. Chaudhury, "Genetic Algorithm for Node Partitioning Problem and Applications in VLSI Design", IEE Proceedings-E, Vol. 140, No. 5, pp. 255-260, September 1993

[8] T.N. Bui and B.R. Moon, "Genetic Algorithm and Graph Partitioning", IEEE Transactions on Computers, Vol. 45, No. 7, pp. 841-855, July 1996

[9] Z. Baruch, O. Creț and K. Pusztai, "Partitioning for FPGA Circuits", MicroCAD '97 International Computer Science Meeting, Miskolc, Hungary, Section D, pp. 113-116, 1997