PCB Layout Design Using a Genetic Algorithm

This paper presents an approach to find the optimal design layout of chips on a circuit board in a manner that minimizes the area covered on the board and the connections between the various chips. In addition, there are no major heat sources next to each other and certain physical constraints are satisfied while finding a layout design. In this approach, the whole circuit board area is divided into a finite number of cells for mapping it into a Genetic Algorithm (GA) chromosome. The mutation and crossover operators have been modified and are applied in conjunction with connectivity analysis for the chips to reduce the creation of a lot of faulty connections. Examples of GA-based chip layout are presented to show how each of the objectives are obtained separately followed by example to arrive at layouts using multiple objectives.

1 Introduction

The advent of electronics marked a new phase in the development of the methods of communication and scientific research. The latest trend is to make all the devices as compact as possible without affecting the performance of the system. All this work got a major thrust by the introduction of integrated circuits. By connecting these ICs on a circuit board very compact electronic equipment have been developed. Size reduction is still a major research area for all the current researchers. One means of achieving this is to come up with an optimal design layout for all the chips on a circuit board once all the connections have been finalized. This would require minimizing the total area of the circuit board and the total wiring required while ensuring that all major heat sources are separated by a minimum distance and that certain physical constraints are satisfied at the same time.

Layout problems are found in various kinds of design problems such as packing and component placement problems in plant and VLSI design. These layout problems include many combinatorial conditions and hence, a lot of heuristic search procedures and probabilistic approaches have been applied for many cases. Techniques like simulated annealing (Seymour and Cagan, 1994) and heuristic methods (Dai and Cha, 1994; Dai et al., 1994) have been extensively used in solving layout problems because of this combinatorial nature of these problems as no gradients are available. Shahrkavar and Mazumdar (1990) used Genetic Algorithms to solve VLSI cell placement problems by reducing the bounded rectangle wire length. Lee and his co-workers (1993) used Genetic Algorithms to solve PCB assembly planning problems. Hybrid approaches (Fujita et al., 1993; Fujita et al., 1994) have also been used to solve layout problems for general shaped objects. Certain methods (Goto, 1981; Wong et al., 1988) have been used to solve the chip nryout problem.

This paper presents a new approach to solve the layout problem of chips on a PCB using genetic algorithms. In this approach, the mutation and crossover operators are modified and a new operation competition is also introduced, not only the minimum board area but various constraints such as connections, heat sources, and certain physical constraints are considered. The paper begins with a brief introduction to the problem in Section 2. In Section 3, the method of discretizing the board area to map it into a parent chromosome is presented. The importance of this method in realizing a physical picture of the layout while using the mutation and crossover operators is also discussed. In Section 4, a couple of examples demonstrating the GA at work are shown and it is discussed how the algorithm can be used to solve single and multiple objective problems. Finally, the last section gives the conclusion.

2 Chip Layout Design

In the process of chip layout design, the positions of the chips are determined in a layout space under various layout conditions. There might be some conditions which have to be necessarily satisfied. These include physical constraints which might restrict all chips from occupying a certain area of the board or some kind of heat constraints which require the certain high heat-generating chips must be separated by some minimum spacing. There might be some conditions which give an indication as how to a better layout might be achieved e.g., conditions which specify the amount of wiring connections between various chips.

2.1 Design Conditions on Chip Layout

The layout problem can be formulated as a problem with simple or multiple objectives and constraints. The objectives for the layout problem may be categorized as:

- Minimizing the total board area required. This is one of the major concerns as size reduction directly implies using the smallest possible board area for any given layout configuration.
- Minimize the length of the connections between various chips on the board. This is necessary to reduce the complications in making a PCB by reducing the number of connections to be etched onto the board.
- The design considerations for the layout problem can be tabulated as:
  - Size of the chip: It constrains the minimum cell size of the circuit board such that all the chips can be represented by a fixed number of cells greater than or equal to one. All the chips are assumed to be rectangular.
  - Physical Constraints: Constraints which restrict the chips from occupying a certain area of the design space. This is needed as in many cases certain areas of the board are reserved for things like power sources etc. and cannot be occupied by any of the chips.
  - Separation of heat chips: It is required that all the high heat generating chips are kept apart from each other. This

Journal of Electronic Packaging

Contributed by the Electrical and Electronic Packaging Division for publication in the Journal of Electronic Packaging. Manuscript received by the IEEE March 12, 1995; revised manuscript received October 26, 1995. Associate Technical Editor: P. Gogate.

MARCH 1996, Vol. 118 / 11
The total design space is selected to be much larger than the expected final layout area. This gives enough freedom to the chips to try all different kinds of layout possibilities. Figure 1 shows how a two-dimensional design space is discretized along with its chromosome representation in the given configuration.

On the board, chips are represented as a group of cells lying adjacent to each other. It is convenient to represent the cells as a board of cells as it gives them the freedom to be rotated while trying to find a layout. The total board area required in a particular layout configuration can be evaluated by noting the maximum and minimum co-ordinates attained by any chip. The distances between various chips can also be easily evaluated. For convenience, the context of the chips have been used to calculate all the distances and wiring connections.

3.2 Genetic Operators

The genetic algorithm used here was a self-developed code using the C programming language. The genetic operators of mutation and crossover used in this algorithm have been modified to suit this problem. Applying these operators in the traditional ways (Goldberg, 1989) would create a lot of faulty generations as the integrity of some chips may be violated in some cases. For example, using the mutation of a single bit instead of all the bits comprising a chip might destroy the integrity of a chip. Similarly, a single point crossover may also affect the integrity of the chips. These operators were hence modified so that they could be applied in conjunction with the assurance of integrity of the chips.

A new operator called comparison has been added to the genetic algorithm. This operator makes this approach similar to the hybrid approach used by (Fujita et al., 1990) where they combine genetic algorithms with a traditional gradient optimization method. In the present case, the traditional gradient method has been replaced by a new genetic operator called comparison.

3.3 Mutation

In the present implementation of GA, mutation is done in three different ways. Firstly, any chip is selected arbitrarily from a randomly selected parent. This chip is removed from its current position and is replaced at a randomly selected location on the chipboard. Secondly, the mutation of a cell-chromosome having improved fitness attributes to a new cell-chromosome. We begin this section with a description of the coding used to represent a layout in the design space.

3.1 Design Space Representation

The entire design space is discretized into a finite number of equal-sized cells. The size of a cell is determined by the least common multiple of the lengths of the sides of the chips. This enables the representation of each chip precisely by grouping a couple of cells lying adjacent to each other. Each of the cells can take a value of 0 up to the number of chips. A value of zero indicates that the spot on the board is empty while any other number indicates that a particular chip number is occupying that space on the board. This two-dimensional design space is mapped into a parent chromosome which is a long one-dimensional string. This is formed by concatenating the values of all the cells in a step order as shown in Fig. 1.

Transactions of the ASME
parent and its orientation is changed (e.g., chip 2 in Fig. 4). The orientation can be changed by rotating the chip through 90 degrees about any one of its corners. The corner is selected randomly. The purpose of introducing this kind of an operator is to avoid rapidly loosing the variety of layouts from the parent chromosomes and to approach the global optimal solution.

In all the three types of mutation, if the chip in its current orientation does not fit in the position selected for it, a different orientation is tried for the chip as the same solution. If the chip does not fit in that position any orientation, then the positions around that position are tried. If still the chip is unable to fit because of the surrounding chips, then a random position is selected for it.

5.4 Crossover. Two children are generated by mating a randomly selected pair of parents with the crossover operator. This operator creates offsprings from the parents by trying to combine some traits of parent 1 with certain other traits of parent 2. The crossover operator used here is similar to the cycle crossover operator (Goldberg, 1989) in the sense that it tries to copy the maximum from the two parents, but because of chip sizes of multiple cells it is not always possible to do so for all the chips.

Two parents are selected at random. These are named as parent 1 and parent 2. The four points are selected randomly to mark off a certain rectangular area on the design domain as the region for crossover (area marked by the thick dark rectangle in Fig. 5). A check is applied to see if any of the chips cuts the boundary of this region (chip 1 in the example shown for parent 1). If it so happens then the area occupied by this chip outside the region is also included in the crossover region (as indicated by the small dark rectangle in the figure). After this has been successively done in both the parents an area is marked off as the region for crossover. For the first child, all the chips lying within the crossover region are simply inherited from parent 1. For all the remaining chips, they are tried to be inherited from parent 2. In case any of these lie in an area already occupied by some chip in the child that is being formed, we go back to parent 1 to check for the position of that chip. In case this also lies on an occupied area in child 1, a random position is selected for this chip. A similar procedure is followed for the second child by exchanging the positions of parent 1 and parent 2. The mechanism of this crossover operator has been shown in Fig. 5.

The advantage of using this kind of an operator for crossover is that firstly it results in no faulty off springs and increases the efficiency of the algorithm and secondly it tries to inherit the maximum from either of the parents.

5.5 Compaction. This is a new operator that has been used to increase the efficiency of the genetic algorithm. It is an operator that takes a given configuration and compacts all the chips in the configuration to give a very compact layout. The purpose of introducing this operator is to get a measure of the goodness of a partial layout. After the crossover and mutation operators, we obtain layouts with new relative positioning of the chips. However these configurations are not very closely placed. This operator brings all the chips close to each other in order to get the most compact layout while maintaining all the positional features of the layout.

Journal of Electronic Packaging
MARCH 1996, VOL. 118 / 13

3.6 Evaluation. The fitness value of an individual is found by calculating the minimum board area (F1) required to hold all the chips in the given configuration and by noting the total length of the wiring connections between the various chips (F2). These two objective functions are generally quite different as regards to the numerical value, so we normalize them to f1 and F2. In addition to these, the constraints of maximum generating chips and fixed chip positions are introduced as penalty functions. If f1 is the number of penalty constraint violations (high heat generating chips lying closer than the minimum separation requirement, fixed chips not in the desired locations or physical constraints being violated) then, we can combine these objectives into a single objective function as J = f1 + f2 + F1 + jf2, where P is sufficiently large number.

3.7 Selection. The next generation of parents is formed based on the objective function values. Two strategies have been used for this selection procedure: "expected value plan" and " elitist plan" (Goldberg, 1989). The expected value plan is basically a survival of the fittest plan. The number of copies of a particular parent is determined according to the expected number calculated from the fitness value with the low function value individuals having a high expected number. The elitist plan maintains the individual with the highest fitness value, i.e., the best layout in the current generation is directly copied into the next generation without applying crossover and mutation operators.

4 Examples to Show GA at Work

In this section, a couple of examples are presented to show how this algorithm can be applied to different problems. In the beginning, there are three examples where different objective functions have been applied to the same layout problem. This is to indicate the effectiveness of the algorithm to achieve different objectives separately. Later, two more examples are presented where the algorithm has been applied to much bigger problems with physical constraints. First, there is an example where all the objectives and constraints have been introduced at a multi objective layout. Then, there is another example to show how all of these objectives can still be attained for a non-rectangular shaped board.

4.1 Examples to Test the Algorithm. A couple of examples have been presented to demonstrate how GAs can be used to achieve one or more different objectives for the same layout problem. In each of these examples, heat constraints were applied in addition to the objective function which varied in each case. It is very convenient to represent all the wiring connections and the heat constraints in a matrix form. The heat generating chips for each of the examples were chips 2, 4, and 7. The separation requirements for these chips are shown in Fig. 6 e.g., the entry in the second row and fourth column is four and it indicates that the centers of chips 4 and 7 (the chips corresponding to its row and column) have to be separated by a minimum
distance of 4 units. The heat constraints were always satisfied as they were introduced as penalty functions in all the examples. The wiring connections requirements are also shown in Fig. 6. They can be interpreted as follows. The entry corresponding to the ith row and the jth column indicates how many wiring connections run between the ith and jth chips, e.g., the entry in the 4th row and 5th column is 6. It indicates that chip 4 has 6 wiring connections with chip 5.

Example 1 (Fig. 7) indicates the results when the objective was to minimize the board area only. As seen, the final layout is very compact and is still satisfying all the heat constraints. Figure 8 indicates the results when the same problem was solved with the objective function of minimizing the wiring connections with no consideration for minimum board area (no conduction operator was used in this case as area reduction was not required). In this case, the wiring connections input clearly indicates that chip 4 has wiring connections with all the other chips. It can be seen from the final layout that chip 6 is almost in the center with all the other chips surrounding it. Also, chips 7, 8, and 9 form a tric with some connections amongst them. As seen from the final layout all three of them lie very close to each other thus ensuring low wiring length. Finally, the chips 1, 3, 4, and 3 have a lot of interconnections and are again close to each other in the final layout.

Figure 9 depicts the results of multiple objectives at work where both the board area and the connections between the various chips were to be minimized. In this case, the board area required for the final layout was found to be the same as in the case for example 1 (15 square units in both the cases) but the chips have been rearranged in a manner to reduce the wiring connections to a great extent while still satisfying the heat constraints.
could result in a very long wait to test and therefore call for very expensive completion. Hence, some approximation may be required for implementing this algorithm for a very large scale problem. Further work can also be done on this if care is taken of the fact as to which side of a chip carries connections to another chip rather than calculating the chip distances from the center.

Since GAs are a randomized search algorithm, the final solution attained depends on the starting random seed. Starting from a new random seed results in a different solution. In this work, different random seeds were tried on the same problem. The final layouts attained differed in the configuration but the final objective function values in all of these cases were very close to each other. A better use of GAs may be made by getting results from several runs parallelly starting from different seeds and getting several good layouts. Then the good features of these results can be combined by applying GA on these group of results to arrive at an even better result.

References