ISSN: 2277-9655
Impact Factor: 1.852

IJESRT

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

A Genetic Approach for Area Reduction in VLSI Layout

J.Allwyn Vinoth[*1], K.Batri[2]
Department of Electronics and Communication, PSNA College of Engineering and Technology, Dindigul, Tamil Nadu, India
allwyn.vinoth@gmail.com

Abstract

Very-large-scale-integration (VLSI) is defined as a technology that allows the construction and interconnection of large numbers (millions) of transistors on a single integrated circuit. Integrated circuit is a collection of one or more gates fabricated on a single silicon chip. The major objective in designing of VLSI integrated circuits is overall chip area reduction. Genetic Algorithm is an iterative and evolutional approach that could be applied to VLSI module placement problem. In this paper a Genetic Algorithm based approach is proposed to reduce the chip area by means of effective placement of the modules. Major placement constraints are considered such that the modules are placed based on best fit position values. As an idea to improve the result of final floor plan, a condition is given such that the modules whose heights are greater than the width in their dimensions are rotated 90 degrees (i.e.) the height is converted into width and the width into height. This yield an area optimized floor plan.

Keywords: Floorplanning, Genetic Algorithm (GA), Integrated Circuit (IC) design, layout, macrocell, placement, VLSI.

Introduction

Nearly all the advances in the modern day electronic systems and devices are a direct outcome of VLSI technology. VLSI is defined as a technology that allows the construction and interconnection of large numbers (millions) of transistors on a single integrated circuit. Integrated circuit is a collection of one or more gates fabricated on a single silicon chip.

The designing of VLSI microchips is a process of many successive steps that includes specification, functional design, circuit design, physical design, and fabrication. Macro-cell layout generation is step in the physical design cycle.

Due to complexity the circuit is partitioned into sub circuits and the components are grouped in as functional units, called the macro-cells which have to be placed on the chip. These cells are described as rectangular blocks with terminals along their borders.

The positions of the cells and the information for the routes of the interconnections between them are defined by the layout. During placement it has to be ensured that enough space is reserved for the completion of all interconnections. In the routing phase, pins on the border of the modules are to be connected. The final step in the physical design is the compaction of the layout where it is compressed in all dimensions such that the total area is being reduced.

Related Works

Floorplanning is an important step in the physical design of VLSI circuits to plan the relative positions of a set of circuit modules on a chip so as to optimize the circuit performance. In this step, it is common that a designer may want to control the positions of some the modules in the final packing due various reasons. The designer may want to restrict the separation between two modules if they have many interconnections between them. This will also happen in design re-use in which the designer will keep the positions of some modules unchanged in the new floorplan. The designers may also be interested in a particular kind of placement constraint known to be symmetry, and some recent literature on this problem can be found from [1], [2]. However, an effective method to control the absolute or relative positions of the modules in floorplanning is non-trivial and this inadequacy has also limited the application and usefulness of many floorplanning algorithms in practice.

There are no universally accepted criteria for measuring the quality of floor plans, possible criteria are: [5]

1. Minimize area 2. Minimize wirelength 3. Maximum routability 4. Minimize delays or 5. A combination of two or more of such criteria.

Several previous works have been carried out to handle some particular kind of placement constraints. The floor planners in [3], [5] and [4] can handle preplaced constraint in which some modules are fixed in position. They work on boundary constraint, in which some of the modules are constrained to be placed along one of the four sides of the chip for I/O connection purpose. To demonstrate the effects of an input set of infeasible constraints, an experiment is performed in [1], where the required sets of constraints were contradictory to each other. These contradictory requirements will mostly lead to positive cycles in the constraint graphs.

Recently, there are some researches activities are carried out in the direction of non-slicing floorplan. Two methods, bound-slice line-grid (BSG) [5] and sequence-pair (SP) [3], are being proposed. Those methods are originally designed for the placement of modules, which have no flexibility in the shape (hard modules). The sequence-pair (SP) method is recently extended to handle the soft modules [4]. In order to handle soft modules, we need to solve an expensive convex programming problem to determine the exact shape of each soft module for numerous times, and this may result in long runtime. For the same set of benchmark data in [4], the slicing floorplan algorithm in [2] and may obtain comparable results by using only a fraction of the runtime. In fact, it may have less than 1% dead space, using no more than 7 s for all the test problems. In floor plan, it is useful if the users are allowed to specify some of their desired placement constraints in the final packing. There are some previous works on floorplanning with preplaced modules [6]. A preplaced module is fixed in position, height and width.

The placement constraint consider in [2] is called boundary constraint: some modules are constrained to be packed on one of the four sides: on the right, on the left, at the top, or at the bottom of the final floor plan. This may be due to the reason that the designers may want to place some modules along the boundary for input-output connections. Other than these, floorplanning is usually done in hierarchical manner in which the modules are grouped into different units and the floorplanning is done purely independently for each unit of the chip. It will help, if some of the modules are constrained to be packed along the boundary of the unit such that they can abut with some other modules in the neighboring units. In [2] using the simulated annealing (SA) process, the normalized Polish expression in each of the iteration is checked to see either the boundary constraints are satisfied. This can also be done efficiently in linear time by just scanning the expression once. Then fix the violated constraints as much as possible, and then include it in the cost a boundary constraint term to penalize the remaining violations.

Slicing representations have some advantages like smaller encoding cost and solution space brings faster runtime for packing. Further it is more flexible to deal with hard, pre-placed, soft and rectilinear blocks. However in real designs, optimal solution may not be in the solution space of the slicing structure. While with non-slicing representation, optimal solution might be achieved but it will need more evaluating runtime for packing than slicing approach. In [5] the number of feasible solutions for a given stage of a floorplanning problem is very large. Besides the great reduction in the search of a floorplanning solution, the introduction of an objective function will allow to select superior floor plans. More over this will change the problem to an optimization problem.

In cluster growth-based method [6], the floorplanning is carried out by iteratively adding the blocks until all blocks have been assigned. An initial block is chosen and is placed in the lower-left (else any other) corner. Successive blocks are then added, one at a time, and they are merged either vertically, horizontally, or diagonally with the cluster. The orientation and location of the next block, depends on the current shape of the cluster and it will be placed to the best position of the objective function of the floorplan. In controversy to the floor plan-sizing algorithm, only the different orientations of all the individual blocks are taken into consideration. Methods that may be used directly and simultaneously to optimize both the shapes of each block and the floor plan are not suitably known.

The floor plan problem is said to be NP-complete. Different heuristic approaches are taken to solve this problem. Those approaches can be categorized as Genetic Algorithm (GA), Simulated Annealing (SA) and Hybrid approach (SAGA: Simulated Annealing and Genetic Algorithm).Other than these various other methods are also available. These types of algorithm, searches through the feasible solution space for better floor plan. Both simulated annealing and genetic algorithm are computation intensive. The difference is one that the simulated annealing operates on only a single solution at a time while genetic algorithm deals a large population of solutions which are optimized simultaneously. Thus the genetic algorithm takes the advantage of the experience gained in the past exploration of the solution space. Both the genetic algorithm and simulated annealing have mechanisms to avoid entrapment at local optima. In SA, this is

ISSN: 2277-9655
Impact Factor: 1.852

accompanied by discarding a superior solution occasionally and accepting the inferior one. The genetic algorithm also relies on the inferior individuals so as to avoid false optima. Since it deals with the whole population of individuals, the genetic algorithm can hold and process inferior individuals without losing the best one. Simulated annealing is an inherently serial algorithm while the GA can be parallelized even on a loosely coupled distributed computer network with 100% processor utilization.

A solution is being described in [7], such that incorporates a novel encoding system with a simple GA. It utilizes an order-based representation that encodes the rectangles and the binary operations into a simple permutation of structures, and a decoder that converts the permutation of structures into a normalized postfix expression. The normalized postfix expression representation is a non-redundant because it provides a unique postfix representation for every different slicing floor plan. If the postfix expressions are not constrained by normalization, a single layout can be expressed by very many equivalent postfix expressions.

Thus the genetic algorithm explores the space of encodings rather than the solution space itself [8]. For continuous parameter optimization problems both spaces are identically. A straightforward genotype encoding in this case is a string of genes which are simple floats. Each gene represents an element of the vector decoding a point in the solution space. The standard mutation operator randomly modifies single genes and crossover is done by direct merging of two genes strings which results in two offspring. All offspring represent correct encodings and these encodings define admissible solutions to the given optimization problem because of the one to one genotype to phenotype mapping between both spaces.

Problem Description

A circuit is laid out according to a set of layout rules (or geometric design rules). The layout rules may be in the form of minimum allowable values for certain widths, distance of separations, and overlaps. A circuit layout problem involves a collection of cells (or modules). These modules may be very simple elements (e.g. ., a transistor or a gate) or may contain more complicated structures (e .g. a multiplier).

Layout architecture refers to the way devices are organized in the chip area. Different layout architectures achieve different trade-offs among speed, packaging density, fabrication time, cost, and degree of automation. The fabrication technology for these layout architectures is generally identical.

The placement constraints may be relative or absolute. The relative placement constraint defines the relationship between two modules and an absolute placement constraint describes the relationship between a module and the chip. Three common types of placement constraints are pre-placed constraint, boundary constraint and range constraint.

In pre-placed constraint, a specific block is being placed exactly at a certain position in the final packing. While on considering the boundary constraint, a block is required to be placed along one particular side of the final floorplan: on the left, on the right, at the bottom, or at the top. For range constraint, a module is needed to be placed within a given rectangular region in the final packing.

Here the inputs of placement problems being specified, the main requirement is to find an optimal layout for the given set of modules such that i) the modules do not overlap and ii) the length of interconnection used is minimal. The module has defined length and width .For the purpose of simple representation and manipulation, the modules considered here are only rectangular shapes which use the same grid structure.

Implementation Issues

During floorplanning the macro cells are described by certain information. That information includes their width, length and cell number. The main objective of the proposed method is to plan all the cell positions over a chip area.

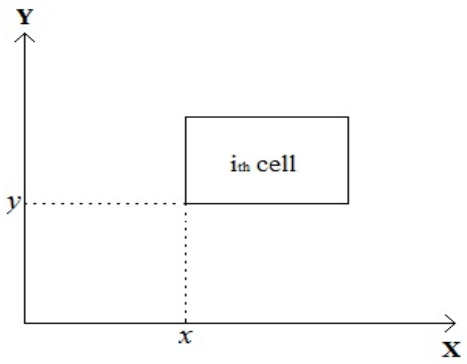

Fig.1. Cell Definition

The shape of a cell is defined as the lower area bound of all possible rectangles of the cell. It is expressed as the x and y dimensions of the rectangles. If we assume that each rectangle can be extended in either the x or y dimension by empty space, we get a cell with one fixed shape. The cells are to be placed such that non overlap constraints are satisfied and the

ISSN: 2277-9655
Impact Factor: 1.852

area cost is minimized. Here the cell positions are defined as the lower left corner of the cell.

A. Non-Overlapping Constraint

To place the modules over a minimum chip area, they have to be placed adjacent to each other. On placing so, the main issue to be considered is that the modules do not get overlapped at any instance. So they have to be separated by minimum distance to avoid overlapping.

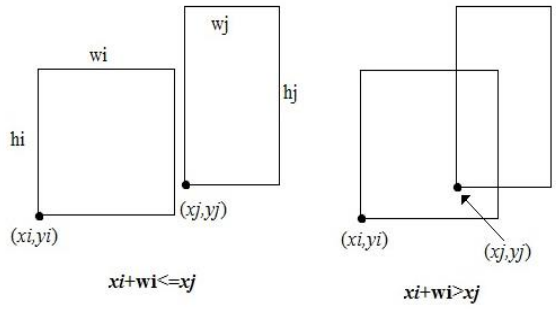

Fig.2.Seperate and Overlapped Modules

The spacing could be utilized for routing in the final layout.

B. Wire Length and Area Estimation

Wire length between two modules is calculated by measuring the distances between centers of two modules, which are connected by a forward path. The overall wire length between the connected modules is added to give the total wire length.

Wire Length (F) =∑i,j (cij * dij) ..(1)

cij : connectivity between blocks i and j.
dij : distances between the centers of rectangles of blocks i and j.

Total chip area is the area of minimum rectangle that encloses all the modules within itself. Thus the area is expected to be minimum, satisfying the main objective of reduced chip area. The total chip area is calculated by the product of difference between the maximum and minimum values of the x and y-axis.

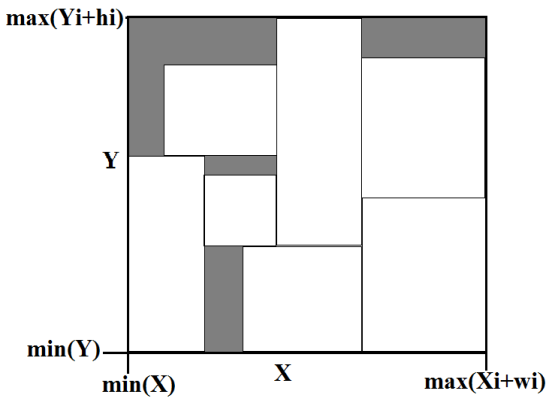

Fig.3.Area Estimation

The total area is
Area (F) = {Max (xi+wi)-Min (xi)}*
{Max (yi+hi)-Min (yi)} ... (2)

Genetic Layout Optimization

The performance of each Genetic Algorithm depends on a set of control parameters like population size, crossover and mutation rates.

A. Localization Algorithm

The initial localization algorithm used here searches the partially covered grid for a free space, where the next module could be placed. It starts at the bottom left corner of the grid and moves the modules top, till it either finds the available space or it reaches the top of the grid.

If an empty space is found, then the module will be placed there. Otherwise the algorithm will return the module to the bottom edge of the grid, shifts to the adjacent position and tries to find the free space for the placement. The search is carried out until any legal position for the modules is found.

B. Cost Function and Fitness Function

VLSI floorplan is a minimization problem whose objective is to minimize the cost of floorplan F, i.e., Cost (F). Generally a floorplan has an area cost and an interconnection cost. Here the area cost is measured by the area of the smallest rectangle that encloses all the modules and the interconnect cost is the total length of the wires fulfilling the interconnections between the modules. Thus the cost of a floorplan F could be given by,

Cost (F) =Area (F) +Wire length (F) ... (3)

The fitness measures of each module relays on maximal compactness of layouts and non-overlapping of objects. The objective of reduced area

ISSN: 2277-9655
Impact Factor: 1.852

is taken in the fitness value such that it is essential to have bigger fitness for shorter layouts.
Thus the fitness of the individuals in the whole population is taken by

$$\text{Fitness (F)} = 1/\text{Cost (F)} \quad \ldots (4)$$

C. Genetic Operators

After the construction of the initial floor plan, the genetic algorithm will start the optimization by modifying the individuals (i.e. mutation) and by combining the building blocks (i.e. crossover). The main mutation operator modifies the location of the modules over the floor plan area. The operator exchanges the blocks which corresponds to exchanging cells or partial layouts on the layout surface. Here storing all important implementations for the blocks will enhances the performance of the genetic algorithm because for a moved partial layout a different implementation may be better in its new environment. Here the number of iterations is taken to be 1000, which could be changed. The crossover rate and the mutation rate may taken to be the number equal to the number of modules to be place.

The implementation of the crossover operator is carried out by choosing two individuals randomly to produce an offspring. Due to this, the number of transmitted genes from the parents is smaller than for problems where cross over directly leads to a correct individual. The recombination operator, which is the only means to generate new solutions, plays important role in genetic algorithm. The basic idea behind such operator it should be designed in order to allow any useful inheritance.

When designing a genetic algorithm for a specific problem, it is very important that a global optimum can be reached starting from any set of individuals by the application of the genetic operators.

The main goal is to order the modules in such a way that the compact parts of the layout will refer to the compact group of objects within the available solution. This means that once some good-looking solution sequence is found out from the population, then it is hard to replace it with any other sequence which seems to be less fit than the former one. The process of evolution of the best solution can be found as that it starts with competition among various solutions to find the best beginning of the layout, and then search gradually towards the end of the layout.

D. Algorithm Description

The steps of the detailed working of Genetic Algorithm can be described as follows:

Step 1: Load modules data and the initial parameters of the GA (such as generations, cross over & mutation probabilities, etc.).
Step 2: Generate the initial population, initialize the position of each module by initial placement and calculate the floor plan area.
Step 3: Using the genetic operators and their genetic probabilities, generate the next module floor plan layout.
Step 4: Check each module for its best fitness position and if its fitness is better than the initial fitness then update it.
Step 5: Check the overall floor plan layout and calculated the fitness using the equation (4), if its fitness value is better than the population‘s value, update it as optimal value.
Step 6: If termination condition is satisfied, the algorithm stops and the inputs which gave optimal fitness is given as output; otherwise, go to Step 3.

To improve the final module floor plan, for the condition

```
{
    if (hi>wi), then
    ti=hi;
    hi=wi;
    wi=ti;
}
```

Where hi & wi are height and width of the corresponding module. Then carry out the algorithm from step 3.

Generations	1000
Crossover Probability	0.5
Mutation Probability	0.1

Table1. Initial Parameters

Simulation Results

The floor plan result has been observed and it is shown in Fig.4 which is enclosed by a rectangle with minimum area that contains all the modules.
In Fig.4.a, the initial placement of the modules is done which is not an optimal one. Though the modules do not overlap, it is not more area efficient.

ISSN: 2277-9655
Impact Factor: 1.852

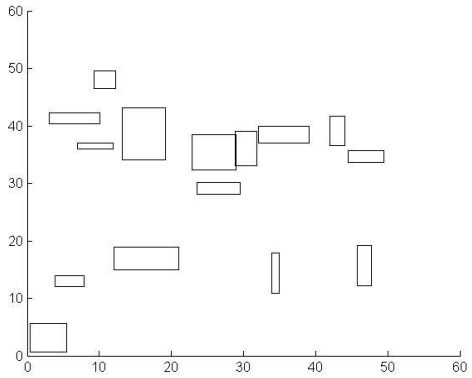

Fig.4.a. Initial Module Floorplan Layout

In Fig.4.b the final module floor plan layout is achieved which is more area efficient than the initial floor plan.

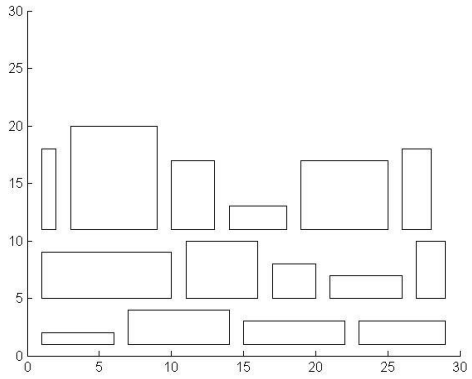

Fig.4.b.Final Module Floorplan Layout

As an idea to improve the result of final floor plan, certain conditions were given such that the area is further reduced which is considered to be near optimal floor plan. According to it, the modules whose heights are greater than the width in their dimensions are rotated 90 degrees (i.e.) the height is converted into width and the width into height. This yield an area optimized floor plan and is shown in Fig.4.c.

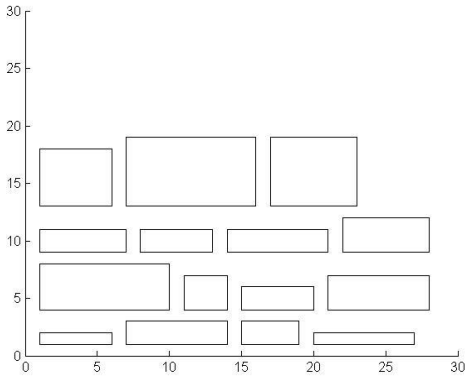

Fig.4.c.Improved Final Module Floorplan Layout

Stage	Estimated Area	Approximate Wirelength
Initial Floorplan	2400 sq. u l	98 u l
Final Floorplan	621 sq .u l	50 u l
Improved Final Floorplan	532 sq .u l	39 u l

Table 2.Floorplan Output

Conclusion

Floorplanning is one of the process manually crafted which results in time consumption and less efficient. The problem of the floorplanning has been considered as a problem of constraint optimization to take care of non overlapping requirement. The concept of GA has been used because it holds the advantages like, simplicity, not much problem dependency and efficient solution.

The solution has been proposed to define the floorplanning by means of genetic algorithm, always there is a scope of having some improvement in the solution. In this regard evolutionary programming can be considered as one of the future possibility. Evolutionary programming (EP) is most widely used approach for optimization problems, which gives the desirable results by using the given constraints to fetch the optimal solution in a reasonable time even when problem size increases.

References

[1] Mazumder P., Rudnick E., "Genetic Algorithm for VLSI Design, Layout and

Automation" Addison-Wesley Longman Singapore Pvt.Ltd., Singapore

[2] Sathiamoorthy S. and Andaljayalakshmi.G "Hybrid Genetic Algorithm for VLSI Macro Cell Layout" in proceedings of International Workshop on Information Retrieval, June 2004, Madurai.

[3] Evangeline, F.Y. Young, Chris C.N.Chu, and M.L. Ho "Placement Constraints in Floor plan Design" IEEE transactions on VLSI Systems, 2004.

[4] D. F. Wong, and Hannah H. Yang "Slicing Floorplans with Boundary Constraints" IEEE transactions on Computer- Aided Design of Integrated Circuits and Systems, Vol. 18, No. 9, September 1999.

[5] B Sowmya, Sunil MP "Minimization of Floorplanning Area and Wire Length Interconnection Using Particle Swarm Optimization" International Journal of Emerging Technology and Advanced Engineering Volume 3, Issue 8, August 2013.

[6] Christine L. Valenzuela and Pearl Y. Wangy "Layout Problem Optimization Using Genetic Algorithms" in proceedings of the third ICGA, Kaufmann publishers, pp. 133-140, 1989

[7] Schnecke V., Vornberger O (1996) "An Adaptive Parallel Genetic Algorithm for VLSI Layout Optimization" 4th Int. Conf. on Parallel Problem Solving from Nature (PPSN IV), Berlin, Germany, September 22-27, 1996.

[8] Heming Chan, P.Mazumeder and K.Shahkoor "Macro cell and Module Placement by Genetic Adaptive Search with Bit Map-Represented Chromosome" Elsevier Science Publishers Integration, the VLSI journal 12 , p.p 49-77, 1991.

[9] Christine L. Valenzuela and Pearl Y. Wang "A Genetic Algorithm for VLSI Floorplanning" Department of Computer Science , George Mason University, Fairfax, USA. International Symposium on Physical Design, 1998.

[10] Volker Schnecke Oliver Vornberger "A Genetic Algorithm For VLSI Physical Design Automation" Dept of Math Computer Science, University of Osnabruck, Dosnabruck Germany 3rd Conf. on Parallel Problem Solving from Nature Springer Lecture Notes in Computer Science 866,78-87,1994.

[11] Xiao-Dong Wang, Tora Chen " On Performance and Area Optimization of VLSI Systems Using Genetic Algorithm"Dept of Electrical Engineering, Colorado State University Ft Collins 1993.

[12] M.Sarrafzadeh and C.K.Wong "An introduction to VLSI physical design" The McGraw-Hill Companies, Inc . United States of America.

[13] Sabhi H.Gerez"Algorithms for VLSI design automation "JOHN WILEY & SONS (ASIA)