ISSN (Print) : 2320 – 3765
ISSN (Online) : 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering
Vol. 2, Issue 4, April 2013

AN O-TREE FOR FLOORPLAN

Anil Krishna Manne[1], B. Srinath[2]

M.Tech Scholar [VLSI Design], Department of ECE, SRM University, Tamilnadu, India[1]

Assistant professor, Department of ECE, SRM University, Tamilnadu, India[2]

ABSTRACT*:* Floorplanning is the process to arranging the number of blocks in to boundary. Conventionally different trees are used for Floorplanning. Here a new representation called O-Tree is used. This O-tree is the representation of admissible placement and used for Floorplanning. The admissible placement is to move blocks either left or bottom of the boundary and it reduces the white spaces in the Floorplanning. A deterministic algorithm will propose for the Floorplanning. This algorithm is generating from the O-Tree representation. The effectiveness of the proposed algorithm will be tested in MCNC benchmark suites.

Keywords: Floorplan, MCNC benchmarks, O-Tree Admissible Placement

I. INTRODUCTION

The integration technology has been (and continues to be) made possible by the automation of various steps involved in the design and fabrication of VLSI chips. The integrated circuits consist of a number of electronic components and layering several different materials in a well-defined fashion on a silicon base is called wafer. The VLSI Physical Design Automation is the research, development and productization of algorithms and data structures related to the physical design process. The steps for VLSI design automation are partition, floorplan, placement, power planning, routing and etc.., in the VLSI design important part is the floorplan. Floorplan is the arrangement of logical blocks with an efficient area, speed, time. The arrangement of blocks is done in two phases; Floorplanning phase, which consists of planning and sizing of blocks and interconnect and the Placement phase, which assign a specific location to blocks. The blocks are positioned so as to minimize the total area of the layout. In addition, the locations of pins on each block are also determined.

The input to the Floorplanning phase is a set of blocks, the area of each block, possible shapes of each block and the number of terminals for each block and the net list. If the layout of the circuit within a block has been completed then the dimensions (shape) of the block are also known. Thus we need to determine an appropriate shape for each block (if shape is not known), location of each block on the layout surface, and determine the locations of pins on the boundary of the blocks.

The floorplanning problem is an NP-hard problem. This is an obvious statement since floorplanning is a generalization of the placement problem, which is NP hard and there are no universally accepted criteria for measuring the quality of floorplan. The works and algorithms that have already been already introduced determine the optimal orientation of the blocks in a given slicing floorplan or non-slicing floorplan.

The classical floorplanning problem seeks to shape and pack all blocks, such that no blocks overlap and the enclosing layout region has minimum area while satisfying aspect ratio constraints. This corresponds to a minimum whitespace objective. But they mainly retarget the top-down placement. To overcome this, the proposed algorithm uses B* tree for modeling slicing and non-slicing floorplan [4]. Given the length and the width of each block we can construct an O-tree in linear time to model floorplanning efficiently. We represent each block as a node. We can move these blocks, that is, the nodes easily to find out the most effective way to place it.

Floor planning in VLSI design is to arrange the modules on a chip under the constraint that no two modules are overlap while controlling the area, wire length, and other performance indices to be optimal. Wong and Liu proposed an algorithm for slicing floor plan designs [5]. They presented a normalized polish expression to represent a slicing structure. For non-slicing floor plans, representations such as sequence pair bounded slicing grid and o-tree, B*tree are found in the literature. Sequence pairs can be used to floor plan hard rectangular blocks by simulated annealing [9]. This approach has very high time complexity and is thus feasible for only small problem sizes. Murata et al. in proposed the sequence-pair representation for rectangular module placement [8].

ISSN (Print) : 2320 – 3765
ISSN (Online) : 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering
Vol. 2, Issue 4, April 2013

Here a rough floorplan is considered to be already done and the length and the width of the modules are obtained from MCNC benchmark. The floorplan can be done by representing the O-Tree from the representation of the admissible placement. One of the key benefits in this is that criteria such as wire length and congestion have already been initially optimized. However significant improvements have to be to achieve zero dead space. Hence a new algorithm is proposed to make a floorplan which is very feasible and it will demonstrate that it is fast and effective in practice.

II. FLOORPLAN

As stated earlier, Floorplanning is the placement of flexible blocks, that is, blocks with fixed area but unknown dimensions. It is a much more difficult problem as compared to the placement problem. In Floorplanning, several layout alternatives for each block are considered. Usually, the blocks are assumed to be rectangular and the lengths and widths of these blocks are determined in addition to their locations. The blocks are assigned dimensions by making use of the aspect ratios. The aspect ratio of a block is the ratio of the width of the block to its height. Usually, there is an upper and a lower bound on the aspect ratio a block can have as the blocks cannot take shapes which are too long and very thin. Initial estimate on the set of feasible alternatives for a block can be made by statistical means, i.e., by estimating the expected area requirement of the block. Many techniques of general block placement have been adapted to Floorplanning. The only difference between Floorplanning and general block placement is the freedom of cells' interface characteristic. Like placement, inaccurate data partly affects Floorplanning. In addition to the inaccuracy of the cost function that we optimize, the area requirements for the blocks may be inaccurate. Floorplanning algorithms are typically used in hierarchical design. This is due to the fact that, although the dimensions of each leaf of the hierarchical tree may be known, the blocks at the node level in the tree are *flexible,* i.e., they can take any dimension. Hence, the Floorplanning algorithms are used at each of the nodes in the tree so that the area of the layout is very less and the position of all the blocks are identified.

The floorplan is having two types fixed outline and free outline and it can be done by two methodologies slicing and non-slicing floorplan. The arrangement of logical blocks is shown in figure 1.

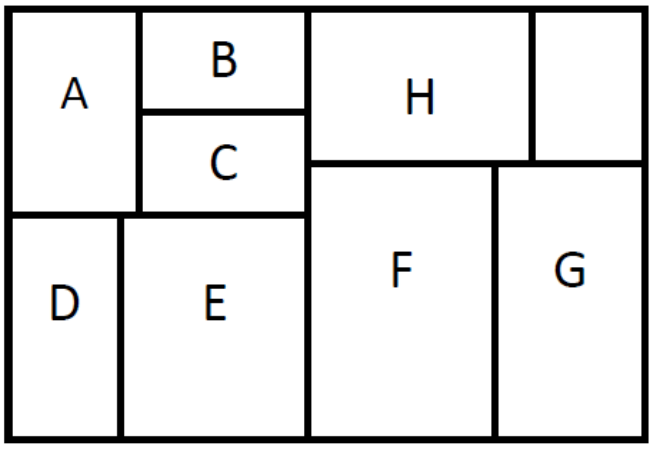

Fig.1 Floorplan

In the figure, Partitions are labeled with letters and cutline's are labeled with numbers. Several new trends are emerging in floorplanning. Interactive floorplanning can improve productivity, improve performance and reduce die size. In [EK96a] an interactive floor planner based on the genetic algorithm is presented. Layout area, aspect ratio, routing congestion and maximum path delay are optimized simultaneously. The design requirements are refined interactively as knowledge of the obtainable cost tradeoffs is gained and a set of feasible solutions representing alternative and good tradeoffs is generated. Experimental results illustrate the special features of the approach.

In [WC95], a new approach to solve a general floorplan area optimization problem is proposed. By using the analogy between a floorplan and a resistive network, it has been shown that a class of zero wasted area floorplan can be achieved under the shape constraint of continuous aspect ratio. However, in many practical designs, each module may have constraints on its dimensions such as minimum length or width. Here, the authors have defined the floorplan area minimization problem under the constrained aspect ratio and give necessary conditions for the realization of zero wasted area floorplan under the shape constraints. A set of optimization methods is developed to minimize the wasted area if no

ISSN (Print) : 2320 – 3765
ISSN (Online) : 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering
Vol. 2, Issue 4, April 2013

zero wasted area floorplan is achievable. Here representing a new representation an O-Tree, this is represented by the admissible placement for the non-slicing floorplan.

III. ADMISSIBLE PLACEMENT

An admissible placement is a compacted placement by compacting the blocks to the left and to the bottom edges [6]. The results of the previous research show that the complexity of problem increases a lot from slicing floorplan to non-slicing floorplan. It challenges to find a comparable or even better representation for non-slicing floorplan. Our thought is encouraged by the observation that any floorplan is bounded on a 2-D plane and could be represented by a planar graph [1]. That might be some means to reduce the redundancy of the floorplanning representation. We first focus on a class of placement defined as admissible. Given a placement, can derive an admissible placement by compacting the blocks to the left and to the bottom edges. An admissible placement is a compacted placement where all the blocks can either move down or move left. Given a placement P, can find a corresponding LB-compact placement Q by a sequence of x-direction and y-direction the overall area of P is less then area of Q. It reduces the some white spaces in the floorplan [5].

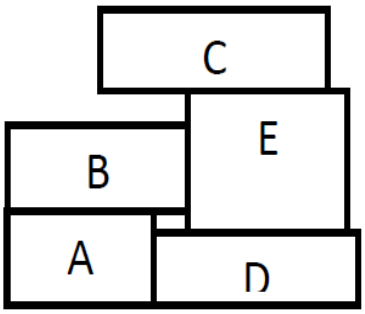

Fig.2 Before admissible placement

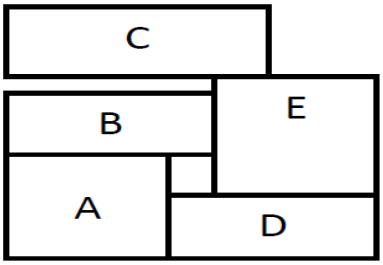

Fig.3 After admissible placement

A constraint graph for placement are a graph G = (V, E), where the nodes in V are placement blocks with additional four nodes used for the boundary of the placement, and the edges in E are the geometric constraints between two blocks. A geometric constraint exists when can draw a horizontal or vertical line between two blocks without passing through the blocks. The edges in E are directed. There are two kind edges one with the direction from a left node to right node and another with the direction from bottom node to the top node. Since, consider geometric constraints in two dimensions, the edges in the constraint graph can be divided in to two sets; Eh for horizontal constraint and Ev for vertical constraint. Then the horizontal constraint graph G= (V, Eh) and the vertical constraint graph G= (V, Ev). Both the graphs are set to planar directed acyclic graphs(s-t PDAG). Both constraint graphs are planar because the placement is planar and there is no edge crossing other edge. According to graph theory, the number of edges in planar graph is less than or equal to three times of nodes minus six.

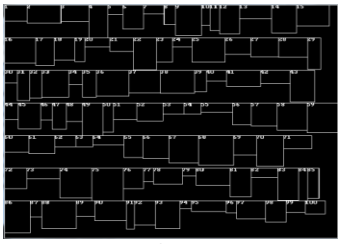

Fig.4 Arrangement of logic blocks compiling

ISSN (Print) : 2320 – 3765
ISSN (Online) : 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering
Vol. 2, Issue 4, April 2013

The fig.4 shows the arrangement of logic blocks in free outline, this is the initial floorplan. The width and length of each logic block is taken from the MCNC benchmark circuits. All the blocks are placed in the free outline randomly, there is no rule to place the blocks.

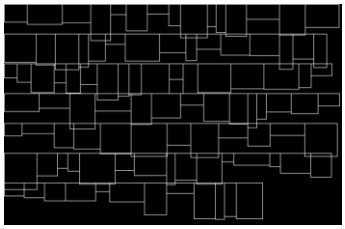

Fig.5 admissible placement compiling

The fig.5 shows the admissible placement of logical blocks. Here the admissible has done by the top compaction of the logical blocks. Due to this admissible placement some amount of dead space is reduced as observed in the compiling results.

IV. O-TREE

An ordered tree (a "tree" of the first definition) has the sub trees in a particular order, and the sub trees sequence cannot contain the same tree twice, because the sub trees must be disjoint. A node connected to an ordered sequence of (sub) trees. It's a recursive definition: if the sequence is empty, the node is called *leaf*, and if not, each of the trees in the sequence is also "a node connected to an ordered sequence of (sub) trees." By disjoint the author means that the sub trees don't have nodes in common. The definition means that the sub trees of a rooted tree are not in a particular order and that they may be repeated. A multi set is kind of like a set that allows multiples.

An ordered tree is a rooted tree in which the children of each vertex are assigned ordering. In a standard plane drawing of an ordered tree, the root is at the top, the vertices at each level are horizontally aligned, and the left-to-right order of the vertices agrees with their prescribed order. In an ordered tree, the prescribed local ordering of the children of each vertex extends to several possible global orderings of the vertices of the tree. One of them, the level order, is equivalent to reading the vertex names top-to-bottom, left-to-right in a standard plane drawing. Level of order in tree is global orderings, pre-order, post-order and in-order. The ordered tree on the left stores the expression a - b *c, where as the one on the right stores c-a* b.

Fig.6 example of o-tree

The ordered rooted tree can be represents for the floorplan from the admissible placement. The generalization of ordered rooted tree in three steps, those are

1. Orthogonal constraint graph.
2. Constraint graph.
3. Admissible O-tree transformation.

i. ORTHOGONAL CONSTRAINT GRAPH

The result of the planarization phase is a planar representation P of the planarized input graph. Now we have to perform the orthogonalization phase which extends P to a quasi-orthogonal representation H. Given an O-tree, to build up its orthogonal constraint graph by using DFS and maintain a contour structure. Based on the definition of O-tree, to develop an algorithm which first find the correspond placement of the O-tree and then builds up its orthogonal constraint graph. By use this orthogonal constraint graph to find the coding of the structure graph. By this code, the area should be minimized.

ISSN (Print) : 2320 – 3765
ISSN (Online) : 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering
Vol. 2, Issue 4, April 2013

To encode O-tree with n nodes, need of 2(n-1) bit string T to identify the branching structure of tree and a permutation P as labels of n nodes. The bit string T is a realization of tree structure. To write '0' for traversal which descends an edge and to write '1' when it subsequently ascends that edge in tree in DFS order. The element in permutation is the root of tree.

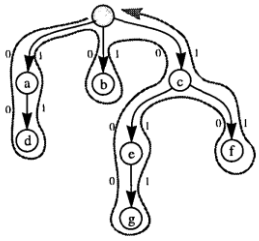

Fig.7 Orthogonal constraint tree example

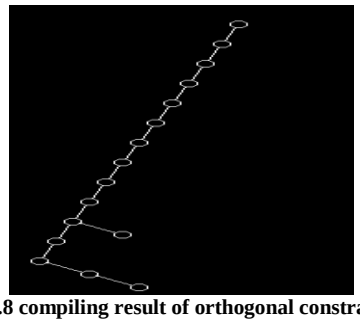

Fig.8 compiling result of orthogonal constraint tree

The fig.8 shows the result of orthogonal constraint tree which is represented by the admissible placement. The figure shows the tree structure which is having the nodes these nodes are represented as logic blocks, these nodes are used to developing a sequence which is as shown in figure 7.

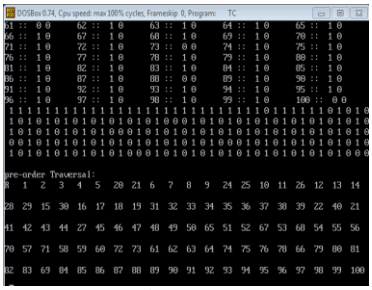

Fig.9 The sequence of the orthogonal constraint tree

The fig.9 shows the sequence of the orthogonal constraint tree for 100 logic blocks the sequence is having the binary code and the pre-order traversal code. The binary code is developing from the tree by visiting the nodes. The pre-order traversal is the order of the each node in the tree.

ii. CONSTRAINT GRAPH

In some tasks of integrated circuit layout design a necessity arises to optimize placement of non-overlapping objects in the plane. In general this problem is extremely hard, and to tackle it with computer algorithms, certain assumptions are made about admissible placements and about operations allowed in placement modifications. Constraint graphs capture the restrictions of relative movements of the objects placed in the plane. These graphs, while sharing common idea, have different definition, depending on a particular design task or its model. The constraint graph for a given floorplan is a directed graph with vertex set being the set of floorplan blocks and there is an edge from block b1 to b2 (called horizontal constraint), if b1 is completely to the left of b2 and there is an edge from block b1 to b2 (called vertical constraint), if b1 is completely below b2.

ISSN (Print) : 2320 – 3765
ISSN (Online) : 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering
Vol. 2, Issue 4, April 2013

If only horizontal constraints are considered, one obtains the horizontal constraint graph. If only vertical constraints are considered, one obtains the vertical constraint graph. Under this definition, the constraint graph can have as many as $O(n^2)$ edges, where *n* is the number of blocks. Therefore other, less dense constraint graphs are considered. The horizontal visibility graph is a horizontal constraint graph in which the horizontal constraint between two blocks exists only if there is a horizontal line segment which connects the two blocks and does not intersect any other blocks. In other words, one block is a potential "immediate obstacle" for moving another one horizontally. The vertical visibility graph is defined in a similar way.

The constraint graph can build its O-tree by SPST algorithm. Because the constraint graph created by algorithm orthogonal constraint graph is either L-compact or B-compact, then construct an O-tree whose edges all have weights equal to zero. Instead of using a breadth first search algorithm as a traditional approach of SPST and use DFS algorithm which has the same performance and needs less explicitly memory space. The run time of this algorithm is linear to the number of blocks.

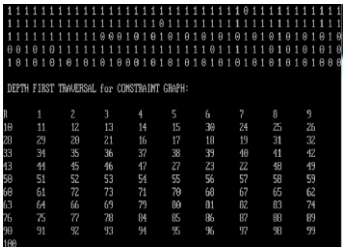

Fig.10 DFS Sequence result after compiling

The fig.10 shows the DFS sequence which is generated from the orthogonal constraint graph. By this constraint graph to find the total width and the length of the overall floorplan by using the shortest path spanning tree algorithm.

iii. ADMISSIBLE O-TREE TRANSFORMATION

AOT is the very important step to generate O-tree. After the constraint graph, an admissible O-tree by invoking orthogonal and constraint iteratively in sequence until convergence is achieved. Given a horizontal O-tree T, because vertical constraint graph is B-compact, we get a vertical constraint graph by orthogonal graph. can get vertical O-tree by constraint graph, similarly we can get a horizontal O-tree which represents an L-compact placement. All moves in the compaction are monotone because all blocks are either moving down or moving left. Therefore, the convergence of above iteration is assured and we get admissible O-Tree. The floorplan can be represented by this O-Tree.

V. PERTURBING ALGORITHM

A new deterministic floorplan algorithm using the O-Tree structure described in above section. Systematically perturbing a given O-Tree, we can find an optimized solution according to preset cost function. To perturb the O-Tree by following steps (a) select a block in O-tree. (b) Delete block from O-Tree. (c) Insert block in the position with the best value of cost function among all possible inserting positions in O-Tree as an external node.(d) perform (a)-(c) on its orthogonal O-Tree.

The selection of inserting position from all above will create many useful configurations for the perturb operation. We may also select other method to insert a node to the tree, but it needs more additional operation to split and merge its descending sub trees. It is one of our choices to not include them in our approach. Given any O-Tree with n nodes, the number of possible inserting positions as external nodes is 2n-1. A deterministic algorithm is derived by perturbing O-Tree in sequence. We select nodes in sequence and find the best perturb position for each of them. Given a fixed sequence of node, we can always find a best O-Tree and its corresponding placement. The advantage of this deterministic algorithm is that its implementation is straightforward and easy to comprehend as shown in fig.11.

ISSN (Print) : 2320 – 3765
ISSN (Online) : 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering
Vol. 2, Issue 4, April 2013

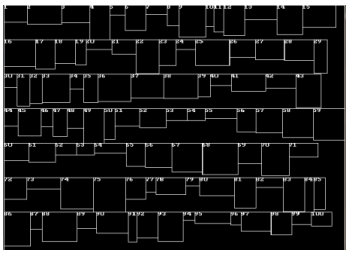

Fig.11 Floorplan

VI. EXPERIMENTAL RESULTS

The circuit characteristics in MCNC benchmarks can be found in [12]. Our program is written in c++ graphics language. The core part of O-Tree operation is around 1000 lines of code and the overall package including an X windows interface is a little more than 6000 lines of source codes. The program is running on the platform of 200MHz Ultra-1 Sparc station with 512MB memory.

We compare the wire length and chip area with the result of cluster refinement [12]. Table 1 show the results of initial placement using the given sequence order and the results after one run of deterministic algorithm. The cost function here is solely by the wire length, which is the sum of half bounding box of all nets in the circuit. In each table there are three numbers; the first number is chip area, second number is wire length, and the third number is CPU time in seconds.

circuit	Cluster refinement	Initial placement	Deterministic algorithm
ami33	1.21/64/603	1.69/61.9/2.83	1.34/50.9/24.3
n100	9.58/185/18	14.3/178/0.26	9.91/167/6.32

Table.1 area/wire length/CPU comparisons

VII. CONCLUSION

We have successfully found a simpler layout representation and incorporated the capability for geometric compaction into the structure itself. O-Tree, a simplified structure to present the geometric relation, is developed and proposed to replace the commonly used constraint graph which we find is complicated and has expensive operations. The tree structure is well known in applied mathematics and computer science and the properties of tree are very straightforward and simple. Our algorithm shows improvement in both chip area and wire length.

REFERENCES

[1] K. Keeler and J. Westbrook, Short Encoding of Planar Graphs and Maps, Discrete Applied Mathematics, vol. 58, pp. 239-252, 1995
[2] D.E. Knuth, The Art of Computer Programming, 2nd Ed, vol. 1, Addison -Wesley Publishing Co, pp. 385-395, 1973
[3] B.S.Baker, E.G.Coffman, and R.L.Rivest, "Orthogonal Packings in Two Dimensions," *SIAM J. Comput.*, vol. 9, no. 4, pp. 846-855, 1980.
[4] L. Sha and R. W. Dutton, "An Analytical Algorithm for Placement of Arbitrarily Sized Rectangular Blocks," in *PTOC2. 2th ACM/IEEE Deaign Automation Conf.*, pp. 602-608, 1985.
[5] A.Alon and U.Ascher, "Model and Solution Strategy for Placement of Rectangular Blocks in the Euclidean Plane," *IEEE Trans. on CAD,* vol. 7, no. 3, pp. 378-386, 1988.
[6] Y.G.Saab and V.B.Rao, "Combinatorial Optimization by Stochastic Evolution," *IEEE Trans. on CAD,* vol. CAD-10, no. 4, pp. 525-535, 1991.
[7] D.F,Wong and C.L.Liu, "A New Algorithm for Floorplan Designs," in *Proc. 23rd ACM/IEEE Design Automation Conf.*, pp. 101-107, 1986.
[8] W.M.Dai and E.Kuh, "Simaltaneous Floorplanning and Global Routing for Hierarchical Building Block Layout," *IEEE Trans. on CAD,* vol. CAD-6, no. 5, vol. 59, pp. 91-101, 1983.
[9] T.C.Wang and D.F.Wong, "An Optimal Algorithm for Floorplan Area Optimization,'' in *PTOC. 27th ACM/IEEE Design Automation Conf..*, pp. 180-186, 1990.
[10] H.Onodera, Y.Taniguchi, and K.Tammu, "Branchand- Bound Placement for Building Block Layout ," in *PTOC. 28th ACM/IEEE Design Automation Conf.*, pp. 433439,1991.
[11] L. Stockmeyer, "Optimal Orientations cif Cells in Silicing Floorplan Designs," Information and Control, P.Pan, W. Shi, and C. L. Liu, "Area Minimization for Hierarchical Floorplans," in IEEE International Con$ on Computer Aided Design, pp.435-440, 1990