A Packing Algorithm for Non-Manhattan Hexagon/Triangle Placement Design by Using an Adaptive O-tree Representation

Jing Li[1] lijing0530@sina.com
Tan Yan[1] tyan@uestc.edu.cn
Bo Yang[1] boyang@263.net
Juebang Yu[1] jbyu@uestc.edu.cn
Chunhui Li[2] chunhui@cadence.com

School of Electronic Engineering, University of Electronic
Science and Technology of China, Chengdu 610054 China[1]
555 River Oaks Parkway, Building 4 MS 4.2.314
San Jose, CA 95134 USA[2]

ABSTRACT

A non-Manhattan Hexagon/Triangle Placement (HTP for short) paradigm is proposed in the present paper. Main feature of this paradigm lies in adapting to the Y- architecture which is one of the promising non-Manhattan VLSI circuit layout architectures. Aim of the HTP is to place a set of equilateral triangles with given size onto a hexagonal chip with maximal chip area usage. Based on the O-tree representation, some adaptive packing rules are adopted to develop an effective placement algorithm for solving the HTP problem in BBL mode. Two examples with benchmark data transformed from the Manhattan BBL mode placement (ami33/49) are presented to justify the feasibility and effectiveness of our algorithms. Experiment results demonstrate that the chip area usage of 94% is achieved through simulated annealing optimization.

Categories and Subject Descriptors

J.6 [**Computer-Aided Engineering**]: *Computer-aided design (CAD), Auto CAD.*

General Terms

Algorithms

Keywords

VLSI circuit physical design, non-Manhattan layout, Y-architecture, O-tree representation, placement, diagonal wiring

1. INTRODUCTION

With the virtue of shorten the wire length of interconnect substantially, Non-Manhattan layout, which means to route diagonally instead of orthogonally, has recently attracted the interest of industry. For example, the X architecture, which is a non-Manhattan integrated-circuit (IC) wiring architecture based on the so-called 4-geometry, demonstrates a wire length reduction of more than 20% and a via reduction of more than 30%[8].

On the other hand, a new performance competitive non-Manhattan routing paradigm called Y architecture is proposed recently [2]. Y-architecture can achieve almost the same throughput and wire length as X-architecture with one routing direction less [2]. The Y-architecture has been proved to be of interest in the on chip routing of processor arrays [3,6].

Though the two kinds of non-Manhattan architecture (X and Y) are both promising in the next generation of nanometer realm ICs, so far the major concern of the researchers and developers in this regard is focused on the routing aspect of the X/Y architecture[1,4,5]. To take full advantage of the non-Manhattan architecture, it is necessary to develop X/Y-aware floorplan and placement tools. In [8], Teig proposed optimizing octilinear wire length instead of Manhattan wire length when designing the placer. But that is not enough. Cheng *et al* demonstrate in [2] that a hexagonal chip could produce 7.6% more throughput than a square one for the Y architecture, and an octagonal chip could achieve an improvement of more than 5.2% in throughput over a square one for the X architecture. Moreover, Zhou[3] placed their innerconnect architecture on hexagonal shaped processor array. To fully utilize the virtues of Y architecture, it is necessary to develop a Y-aware placer that could arrange the placement of modules on a hexagonal chip.

In a preliminary work, we attempt to solve the HTP problem by a specifically developed non-Manhattan placement representation called *Helical Sequence* (HS for short). An HS based packing algorithm called as *Corner Coincidence Compact Placement* (CCCP for short) is also designed to perform the HTP packing. To test the effectiveness of the HS/CCCP algorithm, elaborated are two benchmarks htp33and htp49, which are transplanted from the renowned Manhattan placement benchmarks MCNC ami33 and ami49 respectively, then a GA (Genetic Algorithm) optimization iterations are conducted on these two non-Manhattan benchmarks htp33 and htp49. The experimental simulation results show that the chip area usage is about 87%, far behind the 95 – 96% for the Manhattan placement results easily obtained for ami33 and ami49. (This preliminary results obtained by the HS/CCCP approach has been summarized in a Tech Memo of EDA Group/UESTC, and is available under request by contacting jbyu@uestc.edu.cn) Motivated by getting better result for the non-Manhattan HTP problem, in this paper we will use the O-Tree representation that is pervasively used in Manhattan floorplan. Based on the O-tree representation and some packing rules, an effective HTP algorithm is developed and tested by htp33 and htp49, The experimental results show that the rate of area usage could achieve the level of about 94%, very closed to the 95-96% usage for the Manhattan placement results on ami33 and ami49.

2. PRELIMINARIES

2.1 HTP Problem Formulation

To formulate the non-Manhattan HTP problem, we firstly build a triangle grid (refer to Figure1) instead of a Manhattan one. Every edge in the grid must be at either 0, 60 or 120° angle, that means we use the so-called 3-Geometry as mentioned in [2]**.** Triangle blocks must be placed on the grid. Then the HTP problem we are formulating could be defined as below:

Let $\boldsymbol{B} = \{b_1, b_2, \ldots, b_n\}$ be a set of n equilateral triangle blocks with each block b_i defined by the length of its edge e_i. The task of hexagonal/triangle placement is to place the blocks on the grid without any overlap and find out a smallest equilateral hexagon that encloses all the blocks.

Here we do not consider the interconnection among blocks, but it's easy to integrate it into our system. An example of triangle grid and hexagonal placement is shown in Figure1.

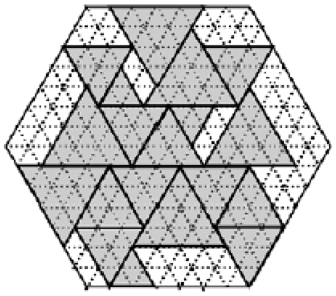

Figure 1. Hexagonal Placement

2.2 The O-tree Representation

An n-node O-tree is a tree with $n+1$ nodes encoded by (T, π), where T is a $2n$-bit string that identifies the branching structure of the tree, and π is a permutation of the n node labels (excluding the root). When traversing the tree, we write a '*0*' for a descending edge and a '*1*' for subsequently ascending that edge. Figure2 displays a 7-node O-tree *(00110100011011,adbcegf)* and corresponding Manhattan placement. Details can be found in [7].

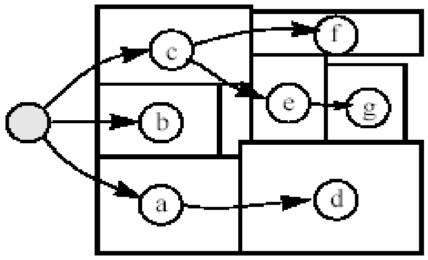

Figure 2. A 7-node O-tree and its placement

2.3 The O-tree's Application in HTP Problem

In the HTP design, finding an efficient method to handle triangle block's packing is extremely important. In this paper, we adopt the O-tree, (which has been proven to be a powerful non-slicing representation for Manhattan BBL placement design) as the placement representation in our packing algorithm by taking account of its following advantages [7]:

(1) Less space to store a placement's O-tree code $(n(2+\lceil \lg n \rceil))$ bits, where n is the number of blocks)

(2) Linear time for transforming an O-tree to its representing placement ($O(n)$)

(3) Smaller upper-bound of possible configurations at $O(n!2^{2n-2}/n^{1.5})$

3. AN O-TREE REPRESENTATION ADAPTED TO HTP AND TWO RELEVANT ALGORITHMS

In this section we will present an O-tree representation adapted to the HTP problem. We adopt the original coding mode of O-Tree, *i.e.*, using a $2n$-bit *0-1* string and a block label's permutation π to represent a placement. Since packing of equilateral triangle blocks is much more complicated than that of rectangle blocks, some different but necessary packing rules, which are tailored for equilateral triangle block's placement, are introduced in our algorithm.

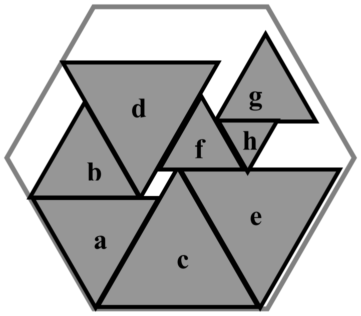

Figure 3(a). Compact Hexagonal Placement

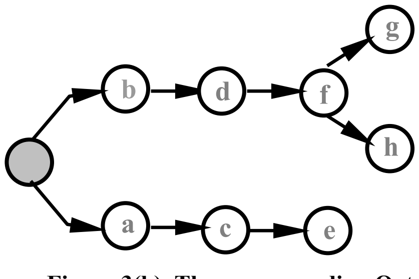

Figure 3(b). The corresponding O_tree (T=*0001110000101111*, π =*acebdfhg*)

Definition 1 [Orientation of Blocks]: There are two kinds of equilateral triangle blocks in a placement. We call those triangles with one corner pointing directly upward as Corner UP (UP for short) while those with one corner pointing directly downward as Corner DOWN (DOWN for short). For example, in Figure 3(a), d is a DOWN module, while g is UP.

Definition 2 [Compact Hexagonal/Triangle Placement]: A HTP is compact if and only if no block can be shift leftward and downward along the 0° or 60° axis with other components fixed. Figure 3(a) gives an example of a compact HTP.

3.1 Compact HTP to Its O-tree Representation

Given any compact HTP, we can build its O-tree by the SPST (Shortest Path Spanning Tree) algorithm. Starting from the module at the left-bottom corner of the hexagonal chip, we visit every block in the placement and encode it into the O-Tree by a depth first search (DFS) strategy. For each block *i*, its right abutting blocks whose left edges abut *i*'s right edge are encoded as its children in the O-Tree. The run time of this procedure is linear to the number of blocks. We are able to address the CH2O (Compact HTP to O-tree) algorithm as below:

```
Input: a compact hexagonal placement including n blocks
Output: O-tree(T[0:2n], Pi[0:n])
set all mark[n]=0
set code=0;
set perm=0;
DFS traverse on the compact hexagonal placement
    set m=current block;
    set p=parent[m] (can be left border of chip);
    if mark[m]==0 and weight[edge(p, m)]==0
        set mark[m]=1;
        set T[code++]=0;
        set Pi[perm++]= m;
        for block c whose left edge abuts the right edge of block m
                traverse (c);
         end for
        set T[code++]=1;
  end if
```

Where the array variable "weight" is to indicate the relationship between the corresponding two blocks. A weight of "0" means these two blocks abut against each other, and any other value means they are isolated. Another array variable "mark" is to avoid processing the handled blocks repetitively.

According to this algorithm, Figure3(b) shows the corresponding O-tree of the compact hexagonal placement in Figure3 (a).

3.2 O-tree Representation to Corresponding Compact HTP

Similar to the original O-tree for Manhattan placement problem, the edge in (T, π) determines the horizontal relations of blocks. However, in HTP problem the block is triangle rather than rectangle and the chip is hexagonal instead of rectangle. Therefore, the packing of triangle blocks should comply with some specifically formulated rules to be described below.

3.2.1 Packing for Roots of Directed Subtrees

Unlike the root node which represents one left vertical border of a rectangle chip in the original O-Tree, the root node of the O-tree for an HTP placement represents two left diagonal borders (60° and 120°). Moreover, because the location of the bend is likely to affect the final area optimizing result, we can not deliberately locate the bend of these two diagonal (60° or 120°) borders. Hence, in the packing procedure the location of the bend is decided by the structure of O-Tree.

Assume set $\boldsymbol{O}=\{o_1,o_2,\cdots,o_n\}$ stands for the roots of the directed subtrees[7] of the O-Tree, namely the blocks originating from the root node in the O_tree. The elements $o_i \in \boldsymbol{O}$'s order are determined by the DFS searching procedure, in which they should be placed along the left border in order.

For the first block $o_1 \in \boldsymbol{O}$ which is also the first block in the O-tree, we place it at the left-bottom corner of the HTP with this corner's coordinate recorded as (0, 0).

For other blocks in $\boldsymbol{O}$, assuming block o_j is the current block to be placed, we pack it according to the orientations of o_j and its previous block o_{j-1}. We can determine the location of o_j according to the following three rules:

a. If o_j is UP (shown as Figure4(a) and (c)), we can draw a 60° line *l* through the leftmost corner of o_{j-1}, and then let o_j slide downward along *l*'s right hand side until it is cumbered by a block which has already been packed.

b. When o_j is DOWN but o_{j-1} is UP (shown as Figure4(b)), we can also draw a 60° line *l* through the leftmost corner of o_{j-1}, then let o_j slide downward along *l*'s left hand side until it is cumbered by an already placed block.

c. When both o_{j-1} and o_j are DOWN (shown as Figure4(d)), we will draw a 120° line *l* through the leftmost corner of o_{j-1}, then make o_j slide downward along *l*'s right hand side until it is cumbered by a packed block.

As can be seen in Figure4, when we pack a block at the root of a directed subtree in the O-tree, the bend of the left border is determined adaptively to the O-tree.

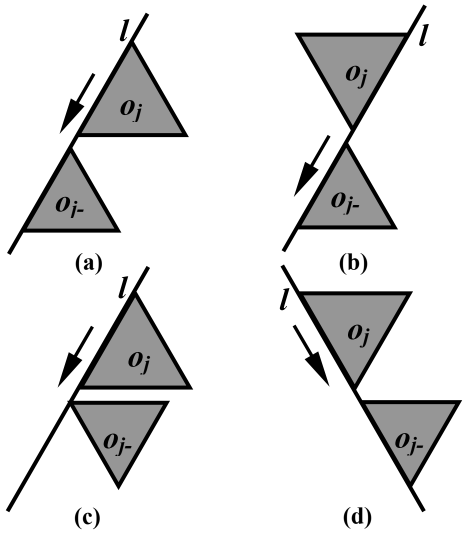

Figure 4. Rules for packing of blocks originating from the root node in O_tree

3.2.2 Packing for General Blocks

In this section, we pack other general blocks that are not the roots of directed subtrees. Assume block *s* is the current block that we want to pack and *p* is its parent whose location has already been fixed. According to the orientations of *p* and *s* we can determine the location for *s* by the following two rules:

a. When *p* is UP (shown as Figure 5(a) and (b)). Firstly we draw a

60° line l_1 through the rightmost corner of *p*, then let block *s* slide downward along l_1's right hand side until it is cumbered by a block which has already been packed. *s* is then located if the Y coordinate of its horizontal edge is smaller than that of *p's*. If *s'* horizontal edge fails to reach a position lower than *p's*, we place it according to another rule: we draw a 120° line l_2 through the rightmost corner of *p*, and then let *s* slide downward along l_2's right hand side until it is cumbered by a placed block.

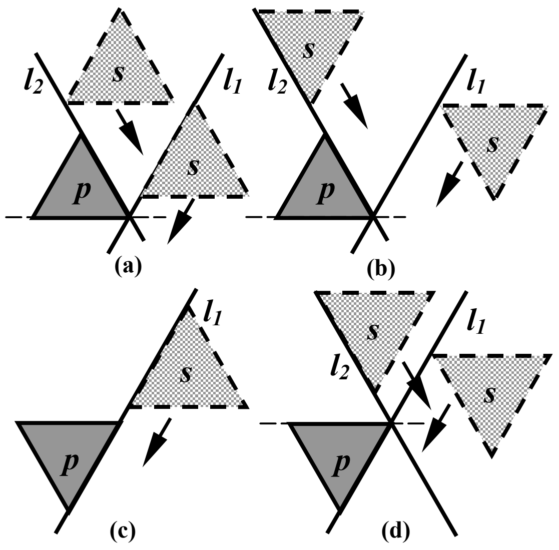

Figure 5. Rules for packing of general blocks in O_tree

When both *p* and *s* are DOWN (shown as Figure 5(d)), the placing process is similar.

b. When *p* is DOWN while *s* is UP (shown as Figure 5(c)). We draw a 60° line *l* through the rightmost corner of block *p*, and then let block *s* slide downward along *l*'s right hand side until it reaches any packed block.

According to the above two rules, every node in the O-Tree could be placed if its parent is located. Thus we could locate every node in the O-Tree by a DFS.

3.2.3 O2CH Algorithm Description

Given any O-tree, we can build the corresponding compact HTP by a DFS and the rules mentioned in section 3.2.1 and 3.2.2.

As is done for the original O-tree, in order to reduce the run time for locating a block, a contour structure[7] is used to record the block where we want to insert next block within the contour. In fact, this contour structure is a variable and double linked list of blocks, which describes the contour line in current compact direction. As can be seen from section 3.2.1 and 3.2.2 of this paper, the compact direction is 60° or 120°.

The run time is linear to the number of blocks without the contour structure. By maintaining a contour structure, the amortized time of finding any Y coordinate becomes constant. Thus, the time complexity of this algorithm is O(*n*), where *n* is the number of blocks.

Now we are able to address the O2CH (O-tree to compact HTP) algorithm by the following pseudo-code:

```
Input: O_tree (T[0:2n], Pi[0:n])
Output:a hexagonal/triangle placement including n blocks
set perm = 0;
set contour = NULL;
set parent = 0;
for code=0 to 2n-1
   if T[code] = 0 then
      set current-block = Pi [perm];
      if parent != 0 (0 represents left border)
          pack current-block by the rules mentioned in 3.2.2
      else
          pack current-block by the rules mentioned in 3.2.1
      end if
      update-contour(contour, current-block)
      set parent = current-block ;
      set perm = perm + 1;
   else
      set parent= prev(parent);
   end if
end for
```

4. EXPERIMENTAL RESULTS

To search the optimal solution, we develop an optimizing algorithm based on the SA (Simulated Annealing) algorithm. Unlike the original deterministic algorithm using O-tree, in which only the external nodes inserting operation is included when perturbing the O-Tree, we adopt four kinds of operation to search an optimal solution: *swap*, *rotating*, *external node's inserting* and *internal nodes inserting*, which can enhance the chance to get a good packing result within a comparatively shorter time.

We implemented our packing algorithm in C language. The simulation experiments are conducted on a platform consisting of a PC machine with 256M memory. The OS is Windows 2000.

Since no benchmark for non-Manhattan HTP is available for the time being, for testing our algorithm we elaborate two benchmarks referred to as htp33 and htp49, which are transplanted from their Manhattan BBL mode placement counterparts MCNC ami33 and ami49 respectively [9]. Every block in htp33 and htp49 is an equilateral triangle and is exactly of the same area as the corresponding block in ami33 and ami49. Figure 6(a) and (b) demonstrate the best packing results, the data of the area usage and the runtime for the corresponding benchmark are given in Table 1.

Note that the orientation of routing channel lies at 0°, 60° or 120° angle which is more suitable for the Y architecture. Hence the correctness of our HTP approach has been verified.

Table 1 Area usage

Benchmark	Area Usage	Runtime
htp33	93.9%	166minutes
htp49	94.1%	310minutes

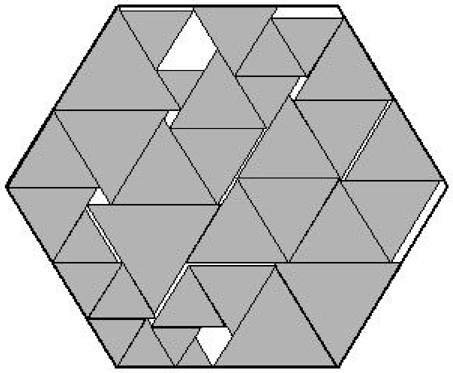

Figure 6(a). Packing result of htp33

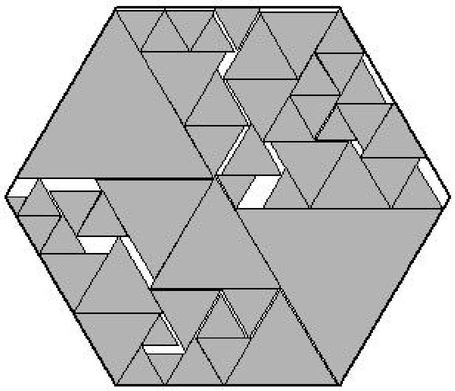

Figure 6(b). Packing result of htp49

5. CONCLUDING REMARKS

In this paper, based on the O-Tree representation together with some adaptive packing rules, we develop an optimization packing algorithm to successfully solve the Hexagonal /Triangle Placement problem.

The algorithm is tested by benchmarks htp33 and htp49. As compared to the HS/CCCP approach, the chip usage rate increases from 87% to 94% by using our new HTP algorithm. This improvement might be mainly attributed to : i) The O-tree representation seems be more powerful than the HS representation; ii) The adaptive packing strategy used in our new algorithm is more effective than that used in the CCCP approach.

The problems remained in the present work can be summarized as the following:

1) Theoretically prove the optimum solution is within the solution space created by the O-tree representation when used in the non-Manhattan placement problem, in spite of the fact that the O-tree representation is able to include the optimal solution in its solution space for the Manhattan floorplan case;
2) Develop efficient algorithm to evaluate the wire length when using the O-tree representation to solve the HTP problem since the wire length estimation is another important measure of the layout result assessment;
3) Decrease the run time of the iterative packing algorithm by seeking more powerful representations as well as more efficient packing rules for the HTP problem;
4) Create more suitable non-Manhattan placement benchmarks besides the htp33 and htp49 such that people could test a newly developed placement algorithms more reliably.

We are confident that any progress on the mentioned above problems will be helpful in promoting the non-Manhattan floorplan research efforts. For example, one of our recent projects in this research direction is try to solve another non-Manhattan placement problem called as OTP (Octagonal/Triangle Placement), a placement paradigm suitable for the X-architecture. Since the X-Architecture chip design is so far limited in the routing layers, while our efforts in this regard are concentrated in the bottom circuit layer, by integrating the diagonal routing into this layer would be of interest in alleviating the routing pressure of the upper routing layers, and further enhancing the performance of the X/Y Architecture based IC chips.

The non-Manhattan layout research area is full of academic and technological challenges such as non-Manhattan floorplan, global routing, gridless, non-preferred direction wiring (both for the X-architecture and Y-architecture), X/Y-aware geometric data structures for graphic visualization and layout databases, etc. Further efforts in these research areas are worthy being encouraged.

6. ACKNOWLEDGMENTS

The authors would like to thank Prof. Ernest S. Kuh of UC Berkeley, Prof. Y. Kajitani of Kitakyushu University and Prof. C. K. Cheng of UCSD for their encouraging and helpful comments on this paper's earlier version.

Many thanks to the paper referees who offered constructive suggestions to help us improving in many aspects the final version of our manuscript.

We also would like to thank Dr. Jin Huang, CEO of Ambow Corp, USA and Fang Shen, CEO of Fangtek, USA, their comments on our work are quite helpful for us to prepare the present paper.

Finally, constructive and inspiring discussions and comments come from global members of the UESTC EDA Web Seminar during the past year are gratefully acknowledged.

7. REFERENCES

[1] C. Chiang, M Sarrafzadeh, Wirability of Knock-Knee layouts with 45° wires, *IEEE Trans on CAS*, Vol.38, No. 6, pp. 613-624, 1991

[2] Cheng-Kok Koh, Patrick H. Madden, Manhattan or non-Manhattan?: a study of alternative VLSI routing architectures, *Proceedings of the tenth Great Lakes Symposium on VLSI*, pp. 47-52, March 2000

[3] Feng Zhou, Esther Y. Cheng, Bo Yao, Chung-Kuan Cheng, Ronald Graham, A Hierarchical Three-way Interconnect Architecture for Hexagonal Processors, *Proceedings of the*

2003 International Workshop on System-level Interconnect Prediction, pp.133-139, April 2003

[4] G. H. Lin G.L. Xue D.F. Zhou, Approximating Hexagonal Steiner Minimal Trees by Fast Optimal Layout of Minimum Spanning Trees, *International Conference on Computer Design (ICCD '99)*, pp. 392 –398,10-13 October 1999

[5] G. H. Lin G.L. Xue, Optimal Layout of Hexagonal Minimum Spanning Trees in Linear Time, *ISCAS 2000-IEEE International Symposium on Circuits and Systems*, pp. IV633-IV636, May 2000

[6] Hongyu Chen, Bo Yao, Feng Zhou, Chung-Kuan Cheng, The Y-architecture: Yet Another On-chip Interconnect Solution, IEEE Design Automation Conference, 2003. *Proceedings of the ASP-DAC 2003. Asia and South Pacific*, pp. 840-846, 21-24 January 2003

[7] P.N. Guo, C.K. Cheng, T.Yoshimura, An O- tree Representation of Nonslicing Floorplan and Its Application, *Proc. 36th ACM/IEEE design Automat- ion Conf.,* pp.268-273, June 1999.

[8] Steven L. Teig, The X Architecture: Not Your Father's Diagonal Wiring , *Proceedings of the 2002 International Workshop on System-level Interconnect Prediction*, pp. 33 –37, April 2002

[9] http://www.cbl.ncsu.edu/