A Non-Slicing Floorplanning and Placement Algorithm Using a New Constraint Graph Based Model*

DONG She-qin[1], HONG Xian-long[1], HUANG Gang[1], GU Jun[2]

[1](*Department of Computer Science and Technology, Tsinghua University, Beijing* 100084, *China*);

[2](*Department of Computer Science, Hong Kong University of Science and Technology, Hong Kong, China*)

E-mail: dongsq@mail.tsinghua.edu.cn

http://www.tsinghua.edu.cn

Received September 27, 2000; accepted May 9, 2001

Abstract: To use a stochastic optimization algorithm to search an optimum placement, the representation of the configuration of a placement is the most important and fundamental issue. A new Constraint Graph based representation *SL* was devised in the paper to represent the non-slicing structure of placement. A nearly $O(n)$ placement algorithm can be designed over the *SL* representation. With the assumption of the meta-grid, we can derive a new concise representation for non-slicing structures from *SL*. We name the new representation as Stairway Grid (*SG*) model. It needs $n(3+\lceil \lg n \rceil)$ bits for a placement of n rectangular blocks. The solution space of *SG* is $n!\,2^{3n-3}$. For a *SG* representation, it takes only $O(n)$ time to transform it to its corresponding placement. It had been proved that all slicing structures could be represented by *SG*. And *SG* model also can represent non-slicing structure. Experimental results on *SG* model demonstrated that it is a concise and effective representation of non-slicing structure.

Key words: building block layout; placement; foorplanning; non-slicing structure

Placement and Floorplaning of blocks on a 2D surface is the first and the most critical phase in layout design of VLSI circuits. It has received much more attention recently due to the increased importance of hierarchical design and IP (Intellectual Property module) blocks. The major objectives of VLSI placement are chip area minimization and interconnection wire length minimization. Since the number of possible placements increases explosively with the number of blocks, even subsets of the problem have been shown to be NP-complete or NP-hard. To optimize the placement or floorplan, many heuristic algorithms were proposed to perturb the floorplan and search for better

* Supported by the National Natural Science Foundation of China under Grant No. 60076016 (国家自然科学基金); the National Grant Fundamental Research 973 Program of China under Grant No. G1998030411 (国家重点基础研究 973 发展规划)

DONG She-qin was born in 1964. He received his Ph. D. degree in mechantronic control and automation in 1996 from Harbin Institute of Technology. He is currently an associate professor at the Department of Computer Science and Technology of Tsinghua University. His current research interests are CAD for VLSI, parallel algorithms, multi-media ASIC and hardware design. **HONG Xian-long** was born in 1940. He is a professor in the Department of Computer Science & Technology, Tsinghua University. His research interests are VLSI layout algorithms and DA systems. **HUANG Gang** was born in 1963, received Ph. D. degree from Tsinghua University in 1999. His current research interests are VLSI layout design algorithms. **GU Jun** received his Ph. D. degree from Utah University in 1989. He was a professor of Calgary University in Canada and currently he is a professor of the Department of Computer Science in Hong Kong University of Science & Technology. His research interests are the optimization algorithms, local search and global optimization, and their application in VLSI CAD, system engineering, communication and multi-media fields.

solutions, e. g. simulated annealing and genetic algorithms. To use such a stochastic optimization algorithm to search an optimum placement, the representation of the configuration of a placement is one of the most important and fundamental issues. The representation structure affects the effectiveness and the efficiency of the optimization algorithm directly.

To devise a representation, the key properties are

1. Completeness of representation: A representation exists for each placement. Hence, the search based on the representation does not miss the optimal solutions subject to various cost functions.
2. Representation of topology: The description of the topology is independent of the block widths and heights. Thus, we can use the topology as a template to optimize the aspect ratios of the blocks.
3. Compactness of representation: The ratio of the number of possible floorplans and the number of configurations of the representation needs to be small. Otherwise, it wastes time to try many different representations that correspond to an identical floorplan.
4. Efficiency of transformation: The transformation between the representation and the floorplan is efficient. It takes linear time, $O(n)$, to read a floorplan of n blocks. Thus, a linear time computational complexity is optimal.
5. Conciseness of representation: The representation requires small memory storage. Thus, we are able to specify multiple floorplans with limited memory space. This is useful for certain algorithms such as genetic approaches.

Two classes of floorplan, slicing and non-slicing, have been identified. The slicing structure obtained by recursive applications of "slicing a rectangle". Choices of horizontal or vertical slicing line as well as its position produce a variety of structures[1]. However, the slicing structure is very limited since trivially most of the packings are non-slicing, but it has been very popular for the sake of its simpleness. A slicing floorplan can be represented by an oriented rooted binary tree. The number of possible configurations for the tree is $O(n!\,2^{3n-3}/n^{1.5})$. There are efforts that have been published to extend the binary tree to the representation of non-slicing structure.

Onodera *et al.* presents a non-slicing placement approach which employs a branch-and-bound strategy to search for one optimal solution within the whole solution space[2]. The maximum number of blocks which can be placed in a reasonable amount of CPU time is about six.

H. Murata et al propose a sequence pair representation to handle non-slicing floorplan[3]. They use two sets of permutations to represent the geometric relation of blocks. The p-admissible solution space is $(n!)^2 8^n$, where n is the total number of blocks. Clearly, the space is so large that there is no guarantee of finding an optimal solution in a reasonable amount of computation time. Its time complexity is $O(n^2)$.

Nakatake *et al.* devised a bounded-sliceline grid structure (BSG) to cope with non-slicing floorplan[4]. An n by n grid plane is used for the placement of n blocks. The representation itself has much redundancy. Its time complexity also is $O(n^2)$.

J. Xu *et al.* develop a 2D branch-and-bound cluster refinement algorithm for block placement to optimize area and interconnection at the same time[5]. For a small k as the cluster size, the run time complexity for each iteration is $O(n^{2+k/2})$. It is CPU intensive and difficult to handle if we choose a larger cluster size.

It is obvious that the complexity of problem increases a lot from slicing floorplan to non-slicing floorplan. More recently, Pei-Ning Guo et al devised a rooted ordered tree, O-tree, to represent the admissible placement which is non-slicing, in which the blocks were compacted to the left and to the bottom boundaries[6]. The run time for transforming an O-tree to its representing placement is linear to the number of blocks. A deterministic algorithm is derived by perturbing O-tree in sequence, but only external nodes can be the possible inserting positions. Unfortunately, O-tree structure is not a topological representation. Given an O-tree, the exact

floorplan topology depends on the widths and heights of the blocks.

Any placement or floorplan is bounded on a 2D plane and can be represented by a planar graph. In such a planar graph, there are only three kinds of relationship, such as "direct horizontal constraint", "direct vertical constraint" and "having no direct relation", between two nodes (blocks). Based on this observation, we devised a new equivalent representation of a planar graph. The new representation is a triple (S,L_1,L_2). We call it a constraint graph (CG) model SL. In the triple, S is a permutation of the nodes, L_1 denotes the relationship between two adjacent nodes in S, L_2 denotes the relationship between a node and its other connected nodes in a planar graph. From a triple (S,L_1,L_2), we can get its corresponding planar graph, and therefore its corresponding placement. The solution space of the triple (S,L_1,L_2) is $n!\,3^{(n(n-1)/2)}$, where n is the number of the blocks(nodes) in S. The transformation operation takes $O(n(n-1)/2)$ time. Through a re-encode operation on L_2, the transformation time could be reduced to nearly $O(n)$.

Our aim is to find an effective non-slicing representation which has reasonable solution space and the runtime for transforming a representation to its representing placement is linear to the number of blocks. If we add meta-grid assumptions to the CG model, we can derive a new Stairway Grid(SG) model. The solution space of the SG model is $n!\,2^{3n-7}$, the transformation operation takes exactly $O(n)$ time. The placement algorithm based on SG can be embedded in a simulated annealing process.

The rest of the paper is composed as follows. Section 1 defines the CG model, SL. Section 2 derived the SG model from SL representation. Section 3 presents the algorithm based on SG and shows our experimental results for MCNC benchmarks. Section 4 is the conclusions.

1 Constraint Graph Based Model

Problem Description. The set $B=\{B_1,B_2,\ldots,B_n\}$ of rectangular blocks lie parallel to the coordinate axes. A placement $P=\{(x_i,y_i)/1\leqslant i\leqslant n\}$ is an assignment of coordinates to the lower left corners of n rectangular blocks such that there are no two rectangular blocks overlapping. Each B_i is defined by a tuple (h_i,w_i), where h_i and w_i are the height and the width of blocks B_i, respectively. The objective of the placement is to find an assignment so that the chip area and interconnection wire-length between blocks are minimized while satisfying the given constraints.

1.1 Constraint graph

A constraint graph for a placement is a graph $G=(V,E)$, in which the nodes in V are placement blocks with additional four nodes used for the boundaries of the placement, and the edges in E are the geometric constraints between two blocks. A geometric constraint exists when we can draw a horizontal or vertical line between two blocks without passing through other blocks.

The edges in E are directed. There are two kinds of edges: one with the direction from a left node to a right node, another with the direction from a bottom node to a top node. The weight $d(e)$ for an edge $e=(B_i,B_j)$ is the separation distance between two nodes, $d(e)$ is equal to $x_j-x_i-w_i$ for horizontal edge and $y_j-y_i-h_i$ for vertical edge. $d(e)$ is equal to zero when two nodes B_i and B_j are adjacent to each other, otherwise it is positive. The edges in constraint graph can be divided into two sets: E_h for horizontal constraints and E_v for vertical constraints. Then, we have the horizontal constraint graph $G_h=(V,E_h)$ and the vertical constraint graph $G_v=(V,E_v)$. Both G_h and G_v are $s-t$ planar directed acyclic graphs. Because the placement is planar and there is no edge crossing other edges, both constraint graph G_h and G_v are planar. According to Graph theory, we have

Lemma 1. G_h and G_v are planar graph, both sizes E_h and E_v are less than $3|V|-3$ for $|V|>3$.

Definition (Admissible Placement). A placement is admissible if any block in a placement has one nether

adjacent block and one left adjacent block except boundary blocks. Two blocks are adjacent to each other if there is a constraint edge e between them and the weight $d(e)=0$.

Lemma 2. From an admissible placement we can get a corresponding constraint graph; from a constraint graph we also can get an admissible placement.

For an admissible placement in Fig. 1, its corresponding constraint graph will be the one in Fig. 2.

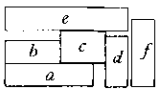
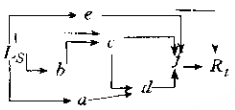


Fig. 1 Admissible placement



(a) G_h (b) G_v

Fig. 2 Constraint graph of Fig. 1 admissible placement

1.2 Constraint graph based representation-SL

In a constraint graph, if there is an edge (B_i,B_j) in G_h, B_j is the right "direct horizontal constraint" block of B_i; if there is an edge (B_i,B_j) in G_v, B_j is the above "direct vertical constraint" block of B_i; if there is no edge between two blocks in G, we call that the two blocks "have no direct relationship". There is no other kind of relationship between two blocks except above three. We first delete the left and bottom boundary node and its output edges from constraint graph. After this step, each time we choose one of the nodes which have no input edges in G and delete it and its output edges in G, until there is no node left in G. During this node deleting process, at each step, we record the deleted node name in order in S, if the current deleted node has an input edge in G_h from the previous deleted node, then record a "1" in L_1, if the edge is in G_v, then record a "0" in L_1, if there is no edge between them in G, then record a "*" in L_1. Similarly, we record the relationship between the current deleted node and each of all the nodes deleted before the previous deleted node from the back in S in L_2. After the process stated above, we get the new representation SL of a constraint graph, this process can be described as the algorithm CG-TO-SL.

```
/* Algorithm CG-TO-SL */
Input:Constraint Graph G=(V,E)
Output:S[0:n-1] /* permutation of n blocks */
L1[0:n-1] /* relationship between a block and its adja-
             cent front block in S */
L2[0:(n-1)(n-2)/2] /* relationship between a block
    and all Its front block from the back in S except its
    adjacent front block */
  mark the root nodes and all its output edges;
  pointer=0
  for i=0 to n-1
    S[i]=find_node(G,node); /* find node in G with
         all its input edges marked and mark all its out-
         put edges */
    if (i!=0) then
      if (true=find_edge (S[i-1],S[i],Gh)) then
        L1[i]="1";
      else if (true=find_edge (S[i],S[i-1],Gv)) then
        L1[i]="0";
      Else
        L1[i]="*";
      end if
    if (i>1) then
      for j=i-2 to 0
        if (true=find_edge (S[i],S[j],Gh)) then
          L2[pointer]="1"
        if (true=find_edge (S[i],S[j],Gv)) then
          L2[pointer]="0";
        else
          L2[pointer]="*";
        end if
        pointer=pointer+1
      end for
    end if
    else
      L1[i]="*";
    end for
  /* in the algorithm, we use mark operation to replace the
     delete operation */
```

For an admissible placement in Fig. 1, we get its corresponding constraint graph in Fig. 2. By applying the

algorithm CG TO-SL, the produced SL representation is (S, L_1, L_2)=(*abcdef*,01101,0 * 100 * 11 * *).

In algorithm CG-TO-SL, every time when we select a node in G which is not recorded in S and all its input edges had been marked (deleted), we may search all the nodes that are still not recorded in S. Therefore, the total search time is $O(n(n+1)/2)$.

From a representation (S, L_1, L_2), we also can derive an algorithm SL-TO-CG to get its corresponding constraint graph.

```
/ * Algorithm SL-TO-CG * /
  Input: S[0:n], L1[1:n-1], L2[1:(n-1)(n-2)/2]
  Output: Constraint Graph G=(V,E)
Pointer=1;
Add S[1] to V;
for i=2 to n;
  add S[i] to V;
  if (L1[i]= '1') then
    add an edge (S[i-1],S[i]) in Gh
  else if (L1[i]= '0') then
    add an edge (S[i-1],S[i]) in Gv
  end if
  if (i≥3) then
    for j=i  2 to 1
    if (L2[pointer]= '1') then
      add an edge (S[j],S[i]) to Gh
    else if (L2[pointer]= '0') then
      add an edge (S[j],S[i]) to Gv
    end if
    pointer=pointer+1
    end for
    end if
  end for
```

Obviously, the runtime of algorithm SL-TO-CG is $O((n-1)(n-2)/2)$. The runtime of the two algorithms CG-TO-SL and SL-TO-CG show that the (S, L_1, L_2) representation is equivalent to constraint graph for admissible placement. Given an (S, L_1, L_2) representation, we can find its corresponding admissible placement by applying the algorithm SL-TO-P.

Lemma 3. The run time of the algorithm SL-TO-P is $O((n-1)(n-2)/2)$.

```
/ * Algorithm SL-TO-P * /
Input: S[1:n], L1[1:n-1], L2[1:(n-1)(n-2)/2]
Output: A placement P={(xi,yi)|1≤i≤n}
  Pointer=1
  S[1].x=0
  S[1].y=0
  for i=2 to n
    S[i].x=0
    S[i].y=0
    if (L1[i-1]= '1') then
      S[i].x=S[i-1].x+S[i-1].width
    else if (L[i-1]= '0') then
      S[i].y=S[i-1].y+S[i-1].height
    end if
if (i≥3) then
  for j=i-2 to 1
    if (L2[pointer]= '1') then
      if (S[i].x<S[j].x+S[j].width) then
        S[i].x=S[j].x+S[j].width
      end if
    else if (L2[pointer]= '0') then
      if (S[i].y<S[j].y+S[j].height) then
        S[i].y=S[j].y+S[j].height
      end if
    end if
  pointer=pointer+1
  end for
 end if
end for
```

1.3 Block placement based on SL representation

For a SL representation we can find its corresponding admissible placement. If we exchange any two blocks in S, another new placement can be produced. If we randomly choose one element in L_1 or in L_2, and change its present "state" to another one of the three, this operation also produces a new solution. Based on this property of SL representation, we can design a reasonable stochastic search algorithm over the SL representation.

For the SL representation, its solution space is $n!\ 3^{(n(n-1)/2)}$, it needs $n((n-1)+\lceil \lg n \rceil)$ bits to describe, and takes $O((n-1)(n-2)/2)$ searches to construct the placement. To reduce the run time complexity of SL

representation, we must reduce the number of searches in the L_2.

On the observation that any block in an admissible placement has not many blocks which have a directed constraint edge connected to it, therefore there are many strings only included the "*" relationship in L_2. For example, from a floorplan result of the MCNC benchmark *hp*, we can derive its SL representation from its constraint graph. The derived SL representation is

$$S=(ACDBEFGHIJK),\ L=(011001*111),$$

$$L_2=(0*11*****01*0*****0*****0*****0*****00*******010*****)$$

Obviously, if we can skip the "*" string in L_2 effectively, the total number of searches will be reduced to nearly $O(n)$, it is true because the constraint graph is planar and its placement is also planar. We rewrite the derived L_2 as follows:

$$L_2=(0\text{-}111\text{-}3**01\text{-}10\text{-}4****0\text{-}3**02*0\text{-}6*****00\text{-}7******010\text{-}4***)$$

the negative number is the length of the "*" string. Although the length of L_2 is not reduced, based on the new encode, an effective search can be design on it. Therefore the run time complexity is reduced. Random change of one element "state" in L_2 only needs to adjust at most 2 related "*" strings.

To further reduce the solution space and the run time complexity of the SL representation, we add more restrict assumptions on the admissible placement.

2 Stairway Grid Model

In SL representation, if two blocks have not any constraint edges between them, we record their relationship as "*". In an admissible placement, blocks transfer their vertical relationship through edges in G_v among them. The horizontal relationship has the similar property. It is the "*" relationship that breaks off the transfer and makes the SL representation solution space become larger. Because the relationship cannot convey in SL, we always need to record all the relationship between a block and each of blocks which are in front of it in S. This makes the run time of transforming a SL representation to its corresponding placement become $O(n^2)$. According to the definition of admissible placement, a block in the placement has an up adjacent block and a right adjacent block except boundary blocks. We may explicitly designate the relationship between two adjacent blocks in S in L_1. Because the vertical or horizontal relationship can be transferred among blocks in a placement, relationships recorded in L_2 also may be designated explicitly.

2.1 Meta-grid assumption

To repair the "broken" transfer relationship in SL, we assume that two adjacent blocks in S only have one of the two relationships "direct vertical constraint" and "direct horizontal constraint". Recall the definition of the admissible placement, it means that for a block in S, its above block will take all the "room" above it and its right block will take all the "room" right to it. If the block itself takes "full" of its "room", there will be no gap between any two "rooms" of the three. Note that the permutation order in S just describes the blocks order in a placement from its left bottom to its top-right. For an admissible placement in which any block will take a full "room" of the same size, given a permutation S, we place the blocks in the order recorded in S from left-bottom to top-right in a stairway grid. This situation is depicted in Fig. 3.

With the above assumption, any two adjacent blocks in S have one of the two relationships such that the L_1 in SL representation will not include the "*" element, so the relationship in L_1 becomes conveyable. In Fig. 3, anyblock has a room "above" it and a room "right to" it. L_1 just depicts the next block in which one of the two rooms, Which block will take the other one room must describe in L_2.

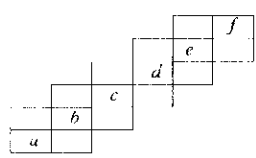


Fig. 3 Stairway grid

Given a permutation S and L_1, with the "room" assumption, we can get a "room packing". The minimal rectangle surrounding the "packing" comprise $(p+1)*(q+1)$ rooms, where p is the number of "0" in L_1 and q is the number of "1" in L_1. For instance, if $S=(abcdef)$, $L_1=(01101)$, the packing will be the one in Fig. 4.

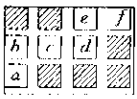


Fig. 4 Relationship determined by $L_1=(01101)$

In such a packing, it is reasonable that if a block "a" in S is above its adjacent front block "b", it needs only to determine the relationship between block "a" and blocks which are located on the left of block "b". If the left block of block "a" is determined, the decision process will stop. For the "right" relationship in S, we have the similar decision process. If we just record the decision process in L_2 which describes how to determine one of the two relationships having not been recorded in L_1, we will eliminate the "*" element from L_2.

For example, if $(S,L_1,L_2)=(abcdef,01101,01001)$, the packing determined by S and L_1 is depicted in Fig. 4, an overall decision process described in L_2 can be pictured in Fig. 5. The bold lines in Fig. 5 clearly describe the decision process. In L_2, we need to record the relationship between blocks such as (ac,ad,ce,be,df). If f is above d, the decision process for f may extend to include cf, or/and bf. For the decision process, we have:

Lemma 4. Given a n-block permutation S and its L_1, with the "room" assumption described above, to determine the other relationship "above (direct vertical constraint)" or "right to (direct horizontal constraint)" between a block and any other block in S which has not been described in L_1, the maximal length of L_2 is $2n-6$ and L_2 only includes "0" and "1" elements.

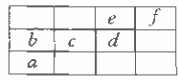


Fig. 5 Decision process determined by $L_2=(01001)$

Lemma 4 has been proved in Ref. [7]. According to Lemma 4, we obtain the Stairway Grid model. It has the similar form to the SL representation. The difference between these two models is L_2. In SG model, L_2 just describes a decision process. The possible configuration of SG model is $n!\,2^{3n-7}$.

Lemma 5. In SG model, given a n-block permutation S and its L_1, and that any block will take the same size of rooms, we can get only one corresponding constraint graph of these rooms. (S,L_1) representation is equivalent to its constraint graph.

Lemma 6. The conversion time between (S,L_1) and its corresponding constraint graph is $O(n)$, where n is the number of blocks in S.

2.2 Block placement based on SG model

From a SG representation, we can determine the above block room and the right block room for any block room. If we have the width and height of all the blocks, through a placement algorithm, we can construct a corresponding admissible placement for a SG representation. This algorithm can be easily derived from the algorithm SL-TO-P. Because the maximal comparison operations in the decision process is $|L_1|+|L_2|=3n-7$, we have:

Lemma 7. The placement algorithm based on SG model is $O(n)$, where n is the number of blocks.

Lemma 8. The solution space of SG model includes all the slicing structure placement.

Lemma 8 has been proved in other place[7]. Obviously, the SG model can represent non-slicing placement, but it has the same time complexity as slicing representation.

3 Experimental Results

We have implemented a placement algorithm based on SG model in C programming language. Our algorithm is embedded in a simulated annealing process. A new admissible placement can be obtained through the operations on SG model listed below: 1) exchange randomly two blocks in S; 2) randomly change the state of one element in L_1 or L_2; 3) randomly rotate a module 90°, 180°, 270°; 4) reflect a module when the wire length is optimized; 5) for

floorplan including soft-blocks, randomly select one candidate in candidate set (in the set, all blocks have the same area, but their aspect ratio is different) to substitute the corresponding block.

We use a cost function like (C_1 * area+C_2 * wire-length), where C_1 and C_2 are the weights for area and wire-length respectively, to optimize a placement or floorplan. The wire-length model used here is the half-perimeter model. On a SUN SPARK20 workstation, we obtained our experimental results on MCNC benchmarks (a set of standard test circuits for floorplanning and placement devised by Microelectronics Center of North Carolina). Table 1 is the results of placement. Table 2 is the results of floorplan including soft-blocks. Table 3 is Minimum/average experiment results with different weights. The last column of Table 3 is the comparison results with O-tree. Our algorithm is better in most cases and it is effective for dealing with soft blocks in floorplanning, while O-tree is not. For ami33, when $C_1=1$, and $C_2=5$, the placement result is area = 1.2583 $(mm)^2$, wire-length =48.23mm, CPU time increased reasonable compared with the situation when $C_1=1$, $C_2=0$. Figure 6 is the floorplan of ami49. Figure 7 is the placement of ami33. We use ami33 to create an example including 198 modules, and the algorithm reaches a placement of this example in 165 seconds with area usage 0.926.

Table 1 BBL placemenmt results based on SG

Example	Usage	Time (sec.)
Xerox	0.922	30
Hp	0.932	32
Ami33	0.963	36
Ami49	0.918	65

Table 2 Floorplan results based on SG

Example	Usage	Time (sec.)
Ami33	0.983	87
Xerox	0.979	76
Hp	0.980	75
Apte	0.966	78
Ami49	0.982	179

Table 3

Examples	Area+0.1 * wirelength (minimum/average)		Area+wirelength (minimum/average)		0.1 * area+wirelength (minimum/average)		Improve over O-tree area/wire(%) (average) ($C_1=C_2=1$)
	area	wire	area	wire	area	wire	
apte	46.07/46.73	405.84/429.24	47.4/47.9	194.95/202.83	47.23/48.20	103.8/177.6	10/45
xerox	20.12/20.34	604.7/667.7	20.2/20.3	403.46/504.10	20.68/20.95	403.4/533.9	9/−12
ami33	1.214/1.218	54.07/61.64	1.22/1.23	51.67/56.72	1.24/1.25	39.43/48.94	9.2/5.2
ami49	38.377/38.477	973.8/1164.6	38.37/39.4	732.84/870.43	41.74/45.72	710.22/775.5	6/−12

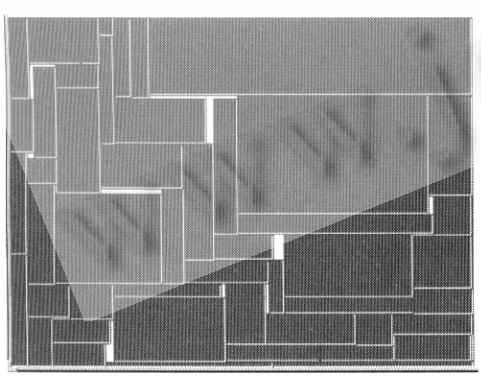


Fig. 6 Floorplan of ami49

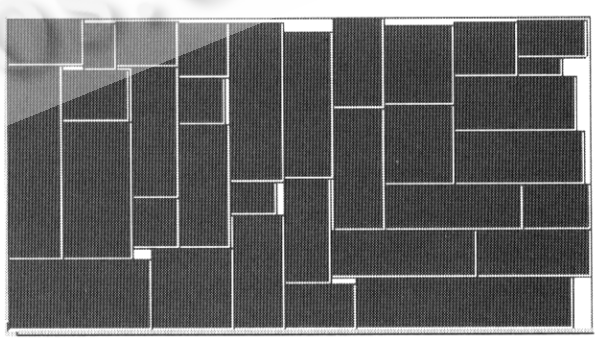


Fig. 7 Placement of ami33

4 Conclusion

We have devised a new non-slicing representation from the commonly used constraint graph. Operations on

the new SL representation are simple and effective. On the observation that any node in a constraint graph corresponding to an admissible placement has reasonable limited number of input edges, through a new encode of L_2 in the SL representation, we can get a placement algorithm and its time complexity is nearly $O(n)$. In order to reduce the solution space of SL representation and get a concise $O(n)$ placement algorithm, we add the meta-grid assumption to the SL representation of CG model such that we derive out the so-called SG model. Experimental results have proved the SG model is a new effective representation for non-slicing structure. Although the generality of SG representation is affected by the mata-grid assumption, it is worth to introduce the assumption to get a $O(n)$ placement algorithm. Research on CG and SG models and their applications, such as routability of the placement and timing driven placement based on SG model, is ongoing. It is promising to obtain new research results on these two models.

References:

[1] Wong, D.F., Liu, C.L. A new algorithm for floorplan design. In: Proceedings of the 23rd ACM/IEEE Design Automation Conference. IEEE Press, USA, 1986. 101~107.

[2] Onodera, H., Taniguchi, Y., Tamaru, K. Branch-and-bound placement for building block layout. In: Proceedings of the 28th ACM/IEEE Design Automation Conference. IEEE Press, USA, 1991. 433~439.

[3] Murata, H., Fujiyoshi, K., Nakatake, S., *et al.* VLSI module placement based on rectangular-packing by the sequence pair. IEEE Transactions on Computer Aided Design, 1996,15(12):472~479.

[4] Nakatake, S., Fujiyoshi, K., Murata, H., *et al.* Module placement on BSG-structure and IC layout applications. In: Proceedings IEEE International Conference on Computer Aided Design. IEEE Press, USA, 1996. 484~491.

[5] Xu, J., Guo, P.-N., Cheng, C.-K. Cluster refinement for block placement. In: Proceedings of the 34th ACM/IEEE Design Automation Conference. IEEE Press, USA, 1997. 762~765.

[6] Guo, P.-N., Cheng, C.-K., Yoshimura, T. An O-tree representation of non-slicing floorplan and its applications. In: Proceedings of the 36th ACM/IEEE Design Automation Conference. IEEE Press, USA, 1999. 258~273.

[7] Huang, G. VLSI floorplanning and block placement algorithms [Ph.D. Thesis]. Beijing: Tsinghua University, 1999.

[8] Huang, Gang, Hong, Xian-long, *et al.* Buiding block placement optimization based on sequence pair model considering area, aspect ratio and wire length. Chinese Journal of Advanced Software Research, 1999,6(4):311~318.

基于新约束图模型的布图规划和布局算法

董社勤[1], 洪先龙[1], 黄 钺[1], 顾 均[2]

[1](清华大学 计算机科学与技术系,北京 100084);

[2](香港科技大学 计算机系,香港)

摘要: 布图规划和布局构形的表示是基于随机优化方法的布图规划和布局算法的核心问题.针对Non-slicing结构的布图规划和布局,提出了一种新的基于约束图表示的模型.基于该模型及其性质,可以得到近似$O(n)$时间复杂度的有效的布局算法.通过引入变形网格的假设,得到了一种新的更加精确的Non-Slicing结构的表示模型:梯形网格模型.其空间复杂度为$n(3+\lg[n])$,时间复杂度为$O(n)$,解空间规模为$n!2^{3n-7}$.已经证明,梯形网格模型可以表示所有的Slicing结构的布局,同时又可以有效地表示Non-Slicing结构的布局,而时间复杂度与Slicing表示相同.实验结果表明,该表示优于刚刚发表的O-tree模型.梯形网格模型是一种拓扑模型,而O-tree的表示依赖于模块的尺寸,因而梯形网格能更有效地处理含有软模块的的布图规划问题.

关键词: 积木块布图;布局;布图规划;Non-Slicing结构

中图法分类号: TP301 **文献标识码:** A