On Handling Fixed Blocks in Incremental Fixed-outline Floorplanning*

Zhigang He[1,3], Yuchun Ma[2,3], Ning Xu[1], Yu Wang[2,4], Xianlong Hong[2]
[1] School of Computer Science and Technology, WuHan University of Technology, WuHan, China
[2] Tsinghua National Laboratory for Information Science and Technology
[3] Department of Computer Science and Technology, Tsinghua University, Beijing, China
[4] Department of Electronic Engineering, Tsinghua university,Beijing, China, 100084

Abstract —**Unlike classical floorplanning that usually only handles block packing to minimize silicon area, modern VLSI floorplanning typically needs to pack blocks within a fixed die(outline) with various user defined constraints. Many algorithms of floorplanning now are with fixed-outline constraint, and these algorithms handle this constraint in the process of stochastic iterative optimization. But the process not only costs too much run time but also is really hard to converge when additional constrains are included. To meet the incremental design requirements, an algorithm to handle fixed-blocks constraints based on given fixed-outline packing with post process is proposed in this paper. By adjusting critical path iteratively in graph of TCG, the violated fixed-outline constraints can be fixed. At the same time, virtual nodes are added into TCG for the constrained blocks with fixed positions so that both fixed-outline constraints and fixed-blocks constraints can be satisfied during the incremental process. Experimental results on MCNC benchmarks show that our algorithm can fix all the violations effectively that for ami49 with 49 blocks, it only takes less than 0.3s to handle the fixed-outline constraints. The degradation on area and wirelength are controlled to about 1.3% and 19.5% respectively.**

I. INTRODUCTION

As the design complexity increases dramatically, modern VLSI floorplanning incurs more sophisticated constraints with the die outline, interconnect plan and block positions[3]. Kahng[4] pointed out that modern VLSI design is based on a fixed-die (fixed-outline) floorplan, rather than a variable-die one. Classical floorplanning that usually handles only block packing to minimize silicon area and wire length. Modern floorplanning should be formulated as a fixed-outline floorplanning, but also include various design constraints such as fixed-blocks[3].

Capturing the designer's intent during floorplanning plays a critical role to improve design productivity of systems-on-chip (SoC). Most of traditional floorplanning approaches use simulated annealing based stochastic optimization techniques, which suffer from long run times with poor performance scalability. In current design, in order to avoid the iterative process between high-level and physical-level design and favor the converge of the design flow, efficient incremental algorithms and methodologies are urgently required. Though stochastic optimization, such as Simulated-nnealing(SA) based approaches, can explore the initial topology, but such approaches usually have limited usage in incremental iterations since they rely on regenerating the solution each time, discarding the manual refinement that always takes place between the initial topology and the subsequent iterations. Especially, during the iteration, the designer may change the design locally such as manually move some blocks. Hence, the incremental algorithms which can handle constraints in modern floorplanning are needed, such as incremental floorplanning methods to handle fixed-blocks and fixed-outline constraints.

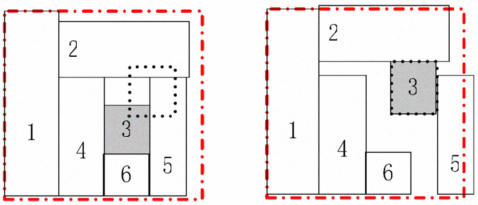
A. Previous Work

To handle fixed-outline constraint within floorplanning, Reference [10] suggests an objective functions considering the violation of fixed outline constraints to drive simulated annealing and moves that better guide local search in the new context. Reference [3] presents an adaptive Fast-SA that can dynamically change the weights in the cost function to optimize wire length under the outline constraint for fixed-outline floorplanning. Most of the previous works treat the fixed-outline as a constraint or suggest new objective functions to drive simulated annealing.

Sometime during the incremental floorplanning process, some modules are moved to certain positions due to some specific constraints. As shown in Fig.1, block 3 should be moved to the dashed box by some manually modification. Reference [9] proposed a method to handle the blocks with fixed positions by exchanging the pre-placed block with another so that the constrained block can be located at its fixed position. But the exchanging may not be effective since the blocks have various shapes and sizes. Especially with multiple constraints, it is very difficult for floorplanning to balance various objectives and constraints at the same time. As shown in Fig.1(b), if we move the constrained block 3 to the required position, then some other blocks afterward will be moved at the same time. Hence, the packing of the whole chip will exceed the given fixed outline. But most of the previous works did not consider these two constraints simultaneously.

In this paper, we propose an effective method to consider the fixed blocks constraints with the given fixed die. During the iterative design process, the local design changes may invoke the physical layout movement.

*This work is supported by NSFC 60606007, 60870001 and 60720106003, 863 project (No. 2009AA01Z130) and TNList Cross-discipline Foundation



(a) fixed-blocks constraint is violated (b) fixed-outline constraint is violated

Fig 1. A floorplan with fixed-blocks. Block 3 is to be fixedthe dashed box. If block 3 is moved to meet the fixed-block constraint directly, the fixed-outline constraint will be violated.

B. Our Contributions

In this paper we study an algorithm to satisfy both fixed-outline constraint and fixed-blocks constraints in a post process based on TCG[2] representation for packings, the iterative modifications on graphs are proposed to fix the violations. The main contributions can be concluded as:

We work out the critical paths of the TCG, and reduce the length of the critical path by repeatedly moving one edge/block away from the critical path. With the length of the critical path being reduced, the width/height of the floorplan is being decreased, eventually it will satisfy fixed-outline constraint,

To handle the position constraint based on given fixed-outline packing the concept of the virtual node is proposed on the TCG graphes. With virtual nodes, there are some paths from source to sink in the TCG. So if the fixed-outline constraint is satisfied, the fixed-blocks constraints are also met.

Experimental results on MCNC benchmarks show that our algorithm can fix all the violations effectively. For ami49 with 49 blocks, it only takes less than 0.3s to handle the fixed-outline constraints, and 1.7s to handle the fixed-blocks constraints. And for ami33 with 33 blocks, it only takes less than 0.09s to handle the fixed-outline constraints, and 1.35s to handle the fixed-blocks constraints. To handle the fixed-outline constraints, the degradation on area and wirelength are controlled to about 1.3% and 19.5% respectively. To handle the fixed-blocks constraints, the degradation on area and wirelength are controlled to about 6.6% and 48.4% respectively.

II. Problem Definition

A. Formulation

Given a packing with the fixed-outline constraints, the additional constraints may be added to some blocks that they are moved to some specific positions. Therefore, we need to find a feasible non-overlapped packing which not only satisfy the fixed-outline constraints, but also the fixed-blocks constraints. In this problem, we are given the following:

1.a set of n rectangular blocks B={b_1, vb_2 ,……, b_n } and each block *bi* is associated with an size (w_i, h_i). The set of $N=\{ n_1, n_2, \ldots\ldots, n_n \}$ corresponded to the B represent the nodes in the qraphs of TCG .

2. fixed-outline of the packing by defining the chip size as *(W, H).*

3. The blocks to be fixed with the fixed positions, such as b_i to be located at (x_i, *yi*).

The aim is to decide the position of each block with no overlapping occurs and no slopping over even with the additional fixed-blocks constraints. To maintain the advantage of the given packing, we want the changes to be locally and the results such as area and wire length can be maintained.

B. TCG representation

We use TCG[2] as the representation.

TCG describes the geometric relations between blocks based on two graphs—namely, a horizontal transitive closure graph *chg* and a vertical transitive closure graph *cvg*, in which a node n_i represents a block b_i and an edge (n_i, n_j) in *chg* (*cvg*) denotes that block b_i is left of (below) block b_j.

Fig.2 shows a placement with five blocks, *1,2,3,4,* and *5*, and the corresponding TCG graphs. The value associated with a node in *chg* (*cvg*) is the width (height) of the corresponding block, and the edge (n_i,n_j) in *chg* (*cvg*) denotes the horizontal (vertical) relation of b_i and b_j. Here, *S* and *T* are the dummy nodes representing the source node and target node. For clarity, we omit the transitive edges connecting the dummy nodes in Fig.2. Since there is an edge (n_1, n_3) in *chg*, block b_1 is left of b_3. Similarly, b_4 is below b_3 since there exists an edge (n_4,n_3) in *cvg*. Therefore, by traversing the constraint graphs finding the longest path, the positions for each block are determined. Given a packing, we will generate the corresponding TCG representation.

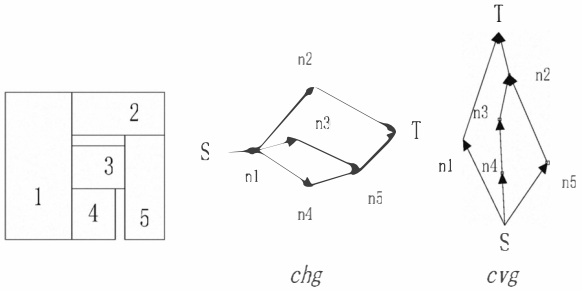


Fig. 2. A packing and the corresponding TCG

III. Virtual Nodes for Fixed-Blocks

Sometimes we need to fix the positions of some blocks in a floorplanning .Through the setting of virtual nodes, we fix the position of fixed-blocks. So for fixed-block *bi, the* virtual nodes in TCG are defined as following:

- Here are four virtual nodes, each of them is *vnl* (left), *vnr*(right), *vna*(above), *vnb*(bellow). The *vnl* and *vnr* are set in the horizontal graph. In TCG, node *vnl* is only adjoining to the source node and node n_i. , node *vnr* is only adjoin to the sink node and node n_i. The node *vna* and *vnb* are set into vertical graph .Node *vna* is only adjoin to the source node and node n_i, node *vnb* is only adjoin to the sink node and node n_i. There is no need to insert the transitive edge from virtual blocks to all blocks. And virtual nodes and the edges adjoin to it should not be involved in perturbing of TCG.

- Virtual nodes are with weight. The weight are defined as the following equations:

$$w_{vnl} = x_i, w_{vnr} = W - w_i - x_i;$$

$$w_{vnb} = y_i, w_{vna} = H - h_i - y_i;$$

Shown as Fig.3(a), here is a floorplan, and the block 3 is to be fixed at (x_3, y_3). Fig.3(c) shows the TCG with the virtual nodes. Here are path 1 *{source, vnl , n3 ,vnr , sink}* and path 2 *{source , vnb, n3 ,vna , sink}* to fix *n3*. *vnl* and *vnr* are only in *chg*, and *vnb* and *vna* are only in *cvg*.

In horizontal direction, if the fixed-outline constraint is satisfied, then the width of the floorplan is *W*, and the path 1 is a critical path. So the length of the path 1 must be W. As there is a virtual node *vnr* at the right of node *n3,* the horizontal coordinate of node *n3* is *(W-* w_{vnr} *-* w_i*)* at most. As there is a virtual node *vnl* at the left of node *n3,* the horizontal coordinate of node *n3* is w_{vnl} at least. In fact, here are the following equations:

$$w_{vnl} = x_3;$$

$$(W - w_{nvl} - w_i) = W - (W - w_{nvl} - x_3) - w_i = w_{nvl} = x_3;$$

So the horizontal coordinate of *n3* must be x_3. The vertical coordinate can be obtained similarly.

If the fixed-block is not located at the required position, then the fixed-outline constraint will be violated. Shown as Fig.3(b). In this graph, *vnr* and *vna* slop over. Therefore the fixed-block can be translated to fixed-outline with virtual nodes.

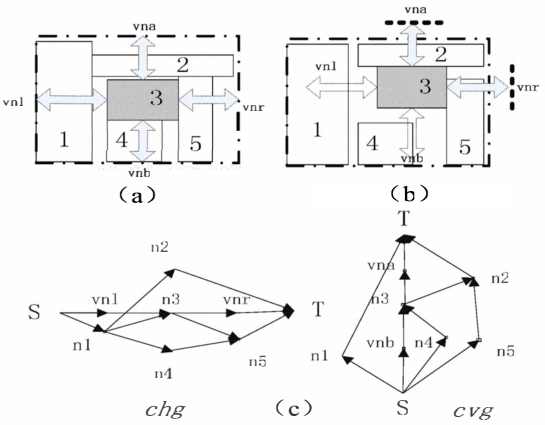
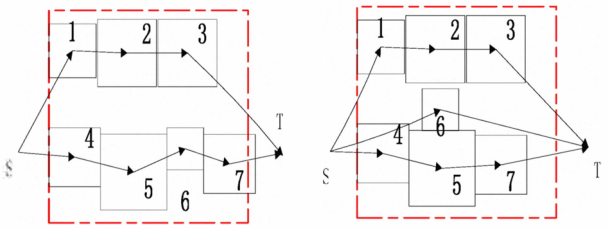


Fig.3. TCG with b3 to be fixed and its virtual block

IV. ADJUSTMENT ON CRITICAL PATH

For a floorplan in TCG, the width/height of a flooplan equals to the length of a critical path of *chg/cvg*. Hence, the most effective way to fix the slopping is to reduce the length of critical path. When some nodes are moved to some other positions, the length of the critical path will be reduced. And the width/height of floorplan will be reduced.

As shown in Fig.4, there is a path *{4,5,6,7},* and the length of it is longer than the width of thechip region. If block 6 is moved away from the path, the length of the cirtical path can be reduced. So the modified packing will satisfy the fixed-outline constraints. By iteratively adjusting the critical path, the violations of constraints can be fixed. Fig.5 shows the flow how we adjust the critical path.



(a) floorplan over flow (b) floorplan reprocessed Incrementally

Fig.4. Graph for adjustment of critical path

Here are the specific steps.

1. Setup the target critical path

For this algorithm we should work out the critical path first.

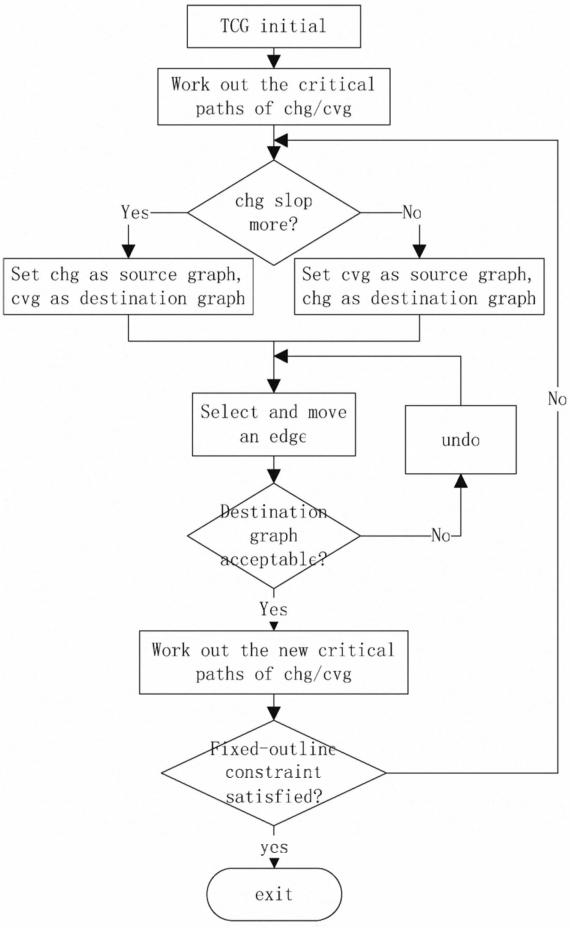


Fig.5. Flow of adjusting critical path

One graph maybe has many critical paths owning the same longest length. We should get the critical path variability.

When the target critical paths of both vertical graph and horizontal graph are worked out, we compare the part of the two critical paths over the outline. The graph which over the outline more should be selected as the source graph, and the other graph as destination graph.

2. Choose an edge to move

With the source graph and the destination graph being ready, an edge in critical path of source graph should be

selected and moved from source graph to destination graph. This means moving one block from above/below of the other to the left/right and vice versa.

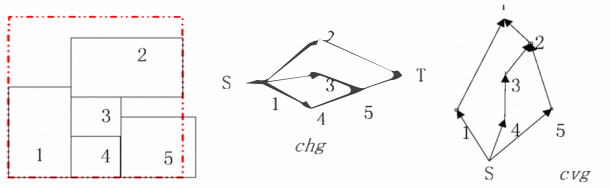
Here we just choose an edge at random. As the critical path is a variable, we can get a large search space with random choosing edge.

3. Probability acceptation

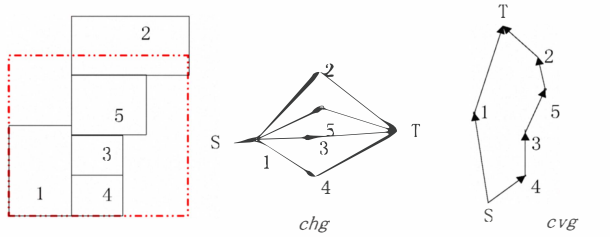
Then here is a new TCG. A new destination graph is got and a new critical path of it should be worked out. We will accept the new destination graph either the length of the new critical path is shorter than the older one, or the chip size is smaller. Otherwise, on the condition that the length of the new critical path is longer than the older one and the difference between them is acceptable, we all accept it by a probability. If the new graph is not accepted, we should undo the movement, and try to choose another edge and move again or work out a critical path again. If the new destination graph is accepted, we should set the source graph and destination graph again and redo the movement.

As shown in Fig.6, it is a floorplan violating the fixed-outline constraints at horizontal direction. In the form of Fig.6 (a), *chg* is the source graph and there are two critical paths. The first is *{1,3,5}* which we set as path 1, and the other is *{1,4,5}* which we set as path 2. The width of the floorplan is the length of those critical paths. Path 1 is chosen as the target random. Edge *(3,5)* is selected for the first time. After the movement, the floorplan violating the fixed-outline constraints at vertical direction is shown as Fig.6 (b), which is not acceptable by random, so we undo this movement and chose an edge again.

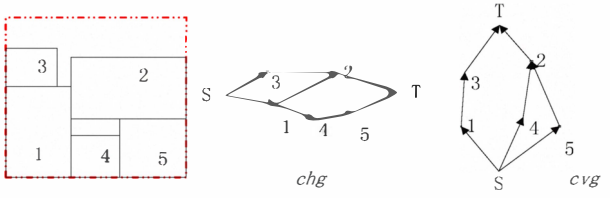
The edge *(1,3)* is selected. After this movement, the floorplan satisfies the fixed-outline constraints on both vertical and horizontal direction. The destination graph is accepted. Hence, the floorplan is legal, as shown as Fig.6 (c).



(a) Initial floorplan



(b) Floorplan with edge(3, 5) moved, illegal



(c) Floorplan with edge(1, 3) moved, legal

Fig 6. Adjustment of critical path

V. EXPERIMENTAL

Here we use btree_v2b_20050611_src[9] as the base program, ami33 and ami49 as the base testcase. All result is given by comparing the program with ours. We give two groups of result. The first involves fixed-outline constraint only. We calculate the time cost and the cost on total area, wire length for handling fixed-outline respectively. The second involves both the fixed-block and the fixed-outline constraints, we compare the results between the initial flooplan and the floorplan after being handled with fixed-outline and fixed-block constraints.

We modify the base program by adding a handling fixed-outline function into it and execute the function after SA floorplaning. All data sets are run on the same machine, an AMD ALTHON 3800+ 2.2G HZ CPU with 2GB memory. Each testcase is run one hundred times and then we calculate the average of one hundred results.

A. fixed-outline constraints

Table I gives the result for the first group. As there are not many blocks and at the same time, then we set a dead space ratio as 0.25. Floorplanning is a multi-objective optimization problem. Those objectives include area and wire length. The base program takes linear weight sum method to get the main objective and sets a parameter *alpha*. The weight to area is *alpha* and the wire length is *(1- alpha)*. The weight of the area influences much on whether satisfying the fixed-outline constraints or not. We give three values like 0.1,0.5,0.9 for alpha. For the probability of missing satisfying the fixed-outline constraints is too low, we don't set the iteration time for every temperature in SA as 400 but 10.

We can see from the table I that we can handle the fixed-outline constraint within the limited time at little cost of wire length and area.

When the wire length gets more heave weight in the floorplan, our algorithm can even reduce the size of the area. In fact, the runtime of handling fixe-outline constraint is irrelevant with the total time of the SA. The average total time is 39.24 for ami49 and 7.83 for ami33 with the iteration time 400[9]. There is an item in the table I whose value is 0.000 which means it costs time too little that the computer to signify the data in a decimal with 6 digits.

B. fixed-blocks constraints

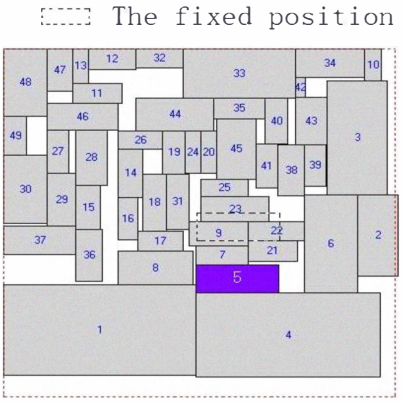
Table II gives the result for the second groups. We also use ami33 and ami49 as the testcases and set the parameters the same as the first group.

Block 5 is limited to be fixed at the center of the floorplan.

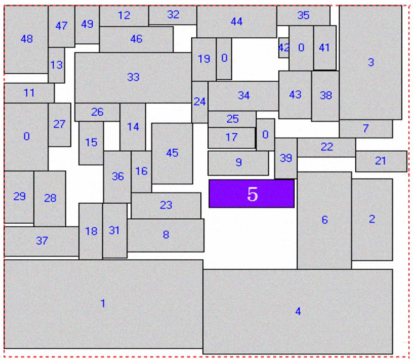
The violation row is the distance of block 5 from target position to the fixed position. We get the initial floorplan by the SA process without the fixed-block constraint.

As we fix block by adding virtual blocks to floorplan to handle the fixed block based on the fixed outline constraints, there are much more blocks in floorplan. And as the fixed position of the block to be fixed is too far from target position, the floorplan slop much and should do incremental changes dramatically. Both of those cost too much time.

Fig 7 gives an example of floorplan. It satisfies fixed-outline constraint and fixed-block constraint by our algorithm.



(a) Initial floorplan, block 2 is out and block 5 is not at the fixed position, illegal



(b)Fixed-blocks and fixed-outline floorplan, legal

Fig 7. A solution of ami49, block 5 is fixed at center of the floorplan, max dead space is 0.25, aspect ratio is 1.

REFERENCES

[1] MICHAELD.MOFFITT, JARRODA.ROY,IGORL.MARKOV,And MARTHAE.POLLACK, " Constraint-driven floorplan repair", Design Automation Conf. (DAC), 1103-1108, July, 2006.

[2] J.-M. Lin and Y.-W. Chang, "TCG: A Transitive Closure Graph-Based. Representation for Non-Slicing Floorplans," in Proc. DAC, 2001,

[3] Tung-Chieh Chen , Yao-Wen Chang,"Modern Floorplanning Based on Fast Simulated Annealing", IEEE Transactions on computer-aided design of integrated circuits and systems, 2006 - Citeseer

[4] A.B.Kahng, "Classical floorplanning harmful?" Proceedings of ACM International Symposium on Physical Design,pp.

[5] D.F.Wong,H.W.Leong,andC.L.Liu, "SimulatedAnnealing for VLSI Design", Kluwer academic pulishers,Boston,1988.

[6] S.N.AdyaandI.L.Markov,"Fixed-outline floorplannin through better local search", Proceedings of IEEE International Conference on Computer Design,pp.328-334, 2001.

[7] H.Murata,K.Fujiyoshi,S.Nakatake,and Y.Kajitani,"Rectangle-packing based module placement",Proceedings of IEEE/ACM International Conference on Computer-Aided Design,pp.472-479,1995.

[8] C.-T.Lin,D.-S.Chen,andY.-W.Wang,"Robust fixed-outline floorplanning through evolutionary search", Proceedings of IEEE/ACM Asia and South Pacific Design Automation Conference,pp.42-44,2004

[9] YC Chang, YW Chang, GM Wu,SW Wu,"B*-Trees: a new representation for non-slicing floorplans",Proceedings of the 37th Annual Design Automation Conference ,Pages: 458 - 463,2000

[10] S. N. Adya and I. L. Markov, "Fixed-outline floorplanning through better local search," Proceedings of IEEE International Conference on Computer Design, pp. 328-334, 2001

TABLE I. RESULT OF EXAMPLES WITH FIXED-OUTLINE

testcase	alpha	Initial		Handle fixed-outline		
		area	wirelength	area	wirelength	time
ami49	0.1	48.152	807.466	43.689	1094.932	0.254
	0.5	42.118	923.026	44.129	1029.630	0.074
	0.9	40.617	1092.513	43.434	1131.845	0.000
ami33	0.1	1.481	56.715	1.411	77.918	0.004
	0.5	1.364	64.803	1.417	72.981	0.055
	0.9	1.330	79.014	1.416	91.643	0.089
ratio	-	1.000	1.000	1.013	1.195	-

TABLE II. RESULT OF EXAMPLES WITH FIXED-OUTLINE AND FIXED-BLOCKS

testcase	alpha	Initial			Handle fixed-outline			
		area	wirelength	violation	area	wirelength	time	violation
ami49	0.1	47.624	814.797	2589.102	43.801	1432.155	1.675	0.000
	0.5	40.997	908.094	2290.071	43.651	1397.444	0.988	0.000
	0.9	38.652	1159.439	2545.630	43.475	1488.054	1.074	0.000
ami33	0.1	1.423	56.485	555.464	1.415	89.137	1.032	0.000
	0.5	1.302	62.305	583.277	1.418	91.323	1.352	0.000
	0.9	1.248	76.191	523.488	1.414	97.566	0.905	0.000
ratio	-	1.000	1.000	1.000	1.066	1.484	-	-