Twin Binary Sequences: A Non-Redundant Representation for General Non-slicing Floorplan

Evangeline F.Y. Young
Dept. of CSE
The Chinese Univ. of HK
Shatin, N.T., Hong Kong
fyyoung@cse.cuhk.edu.hk

Chris C.N. Chu
Dept. of ECPE
Iowa State University
Ames, IA 50011-3060
cnchu@iastate.edu

Zion Cien Shen
Dept. of ECPE
Iowa State University
Ames, IA 50011-3060
zionshen@iastate.edu

ABSTRACT

The efficiency and effectiveness of many floorplanning methods depend very much on the representation of the geometrical relationship between the modules. A good representation can shorten the searching process so that more accurate estimations on area and interconnect costs can be performed. Non-slicing floorplan is the most general kind of floorplan that is commonly used. Unfortunately, there is not yet any complete and non-redundant topological representation for non-slicing structure. In this paper, we will propose the first representation of this kind. Like some previous work [9], we have also made used of mosaic floorplan as an intermediate step. However, instead of including a more than sufficient number of extra dummy blocks in the set of modules, our representation allows us to insert an *exact* number of *irreducible empty rooms* to a mosaic floorplan in such a way that *every* non-slicing floorplan can be obtained by this method uniquely from one and only one mosaic floorplan. The size of the solution space is only $O(n!2^{3n}/n^{1.5})$ but every non-slicing floorplan can be generated uniquely and efficiently in linear time without any redundant representation.

Categories and Subject Descriptors

B.7.2 [**Integrated Circuits**]: Design Aids—*Layout*; J.6 [**Computer Applications**]: Computer-Aided Engineering—*Computer-aided design*

General Terms

Algorithms, Design, Performance

1. INTRODUCTION

As technology moves into the deep-submicron era, circuit sizes and complexities grow rapidly. Floorplanning has become ever more important than before. Unfortunately, floorplanning problems are NP-complete. Many floorplanners employ methods of perturbations with random searches and heuristics. The efficiency and effectiveness of these methods depend very much on the representation of the geometrical relationship between the modules. A good representation can shorten the searching process and allows fast realization of a floorplan so that more accurate estimations on area and interconnect costs can be performed.

The problem of floorplan representation has been studied extensively. There are three kinds of floorplan: slicing, mosaic and non-slicing. A slicing floorplan is one that can be obtained by recursively cutting a rectangle into two by using a vertical line or a horizontal line. Normalized Polish expression [7] is the most popular method to represent slicing floorplan. This representation can describe any slicing structure with no redundancy. An upper bound on its solution space is $O(n!2^{3n-3}/n^{1.5})$. For general non-slicing floorplan that is not necessarily slicing, there was no efficient representation other than the constraint graphs until the sequence pair (SP) [5] and the bounded-sliceline grid (BSG) [6] appeared in the mid 90's. The SP representation has been widely used because of its simplicity. Unfortunately, there are a lot of redundancies in these representations. The size of the solution space of SP is $(n!)^2$ and that of BSG is $n!C(n^2, n)$. This drawback has restricted the applicability of these methods in large scale problems. O-tree [3] and B*-tree [1] are later proposed to represent general non-slicing floorplan. They have very small solution space of $O(n!2^{2n-2}/n^{1.5})$ and can give a floorplan in linear time. However, they can only describe partial topological information and module dimensions are needed to give a floorplan exactly.

The paper [4] proposes a new kind of floorplan called mosaic floorplan. A mosaic floorplan is similar to a general non-slicing floorplan except that it does not have any unoccupied room (Figure 1(a)) and there is no *crossing cut* in the floorplan (Figure 1(b)). A representation called Corner Block List (CBL) is proposed to represent mosaic floorplan. This representation has a relatively small solution space of $O(n!2^{3n})$[1] and the time complexity to realize a floorplan

ISPD'02, April 7-10, 2002, San Diego, California, USA.

[1] In [4], the paper claims without proof that the size of solution space for Corner Block List is $O(n!2^{3n}/n^{1.5})$. However, we believe that the correct size of CBL solution space should be $\Theta(n!2^{3n})$. In the CBL algorithm, the corner block list (S, L, T) are perturbed randomly and independently in the simulated annealing process. There are $n!$ combinations for

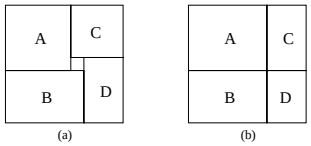

Figure 1: Structures that cannot be represented in a mosaic floorplan

from its representation is linear. However, some corner block list does not correspond to any floorplan. As a remedy to the weakness that some non-slicing structures cannot be represented (e.g., Figure 1(a)), CBL is extended by including dummy blocks of zero area in the set of modules. In order to represent all non-slicing structure, $O(n^2)$ of such dummy blocks are used, which have increased the size of the solution space significantly [9].

1.1 Our Contributions

Although the problem of floorplan representation has been studied extensively, it is still practically useful and theoretically interesting to find a complete (i.e., every non-slicing floorplan can be represented) and non-redundant topological representation for general non-slicing structure. In this paper, we will present such a representation, the twin binary sequences (TBS) representation. This will mark the first of this kind. Like some previous work [9], we have made use of mosaic floorplan as an intermediate step to represent a non-slicing structure. However, instead of including extra dummy blocks in the set of modules, TBS allows us to insert an exact number of *irreducible empty rooms* to a mosaic floorplan such that every non-slicing structure can be generated uniquely and non-redundantly. Besides, the representation can give a floorplan efficiently in linear time. We have also studied the relationship between mosaic and non-slicing floorplan and have proved that the number of empty rooms needed to be inserted into a mosaic floorplan to obtain a non-slicing structure is tightly bounded by $\Theta(n)$ where n is the number of modules.

In the next section, we will define and study this new representation. In section 3, we will show some interesting relationships between mosaic and general non-slicing floorplan. In section 4, experimental results will be shown and discussed.

2. TWIN BINARY SEQUENCES (TBS) REPRESENTATION

In the paper [8], Yao, et al. first suggest that twin binary trees (TBT) can be used to represent mosaic floorplan. They have shown a one-to-one mapping between mosaic floorplan structure and TBT. We have made use of TBT in our representation. Recall that the definition of twin binary trees comes originally from the paper [2] as follows:

DEFINITION 1. *The set of twin binary trees with n nodes* $\mathrm{TBT}_n \subset \mathrm{Tree}_n \times \mathrm{Tree}_n$ *is the set:*

$$\mathrm{TBT}_n = \{(b_1, b_2) | b_1, b_2 \in \mathrm{Tree}_n \text{ and } \Theta(b_1) = \Theta^c(b_2)\}$$

where Tree_n is the set of binary trees with n nodes, and $\Theta(b)$ is the *labeling* of a binary tree b as follows. We begin with an empty sequence, and perform an in-order traversal on the tree. When a node with no left child is reached, we will add a bit 0 to the sequence, and when a node with no right child is reached, we will add a bit 1 to the sequence. The first 0 and the last 1 in the sequence will be omitted. Θ^c is the complement of Θ obtained by interchanging all the 0's and 1's in Θ. An example of a twin binary trees is shown in Figure 2.

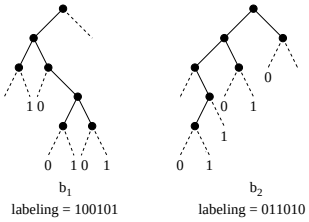

Figure 2: An example of a twin binary trees

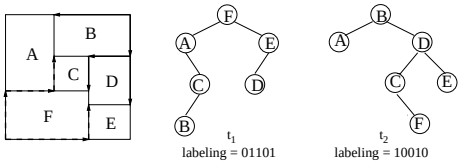

Figure 3: Building a twin binary trees from a mosaic packing

Instead of using an arbitrary pair of trees (which may not be twin binary to each other) directly, we used a 4-tuple $s = (\pi, \alpha, \beta, \beta')$ called a twin binary sequences (TBS) to represent a mosaic floorplan. This 4-tuple can be one-to-one mapped to a pair of binary trees t_1 and t_2 such that t_1 and t_2 must be twin binary to each other and they together represent a mosaic floorplan uniquely. In order to motivate the idea of our new representation, we will first show how a twin binary trees can be obtained from a mosaic floorplan in the following sub-section.

2.1 From Floorplan to Twin Binary Trees

Given a mosaic floorplan F, we can obtain a pair of twin binary trees t_1 and t_2 by traveling along the slicelines of F. An example is shown in Figure 3. To construct t_1, we start from the module at the lower left corner and travel upward (left subtree) and to the right (right subtree). Whenever the lower left corner of another module x is reached, a node labeled x is inserted into the tree and the process will be repeated starting from module x until all the modules in the floorplan are visited. The tree t_2 can be built similarly by starting from the module at the upper right corner and travel downward (right subtree) and to the left (left subtree). The paper [8] has shown that the pair of trees built in this way must be twin binary to each other, and there is a one-to-one mapping between mosaic floorplan structure and twin binary trees. We observed that the inorder traversal of the two binary trees constructed from a mosaic floorplan must be the same. Let us look at the example in Figure 3. We can see that the inorder traversal of both t_1 and t_2 are $ABCFDE$. We have proved the following observation that has helped in defining the Twin Binary Sequences (TBS) representation:

S, 2^{n-1} combinations for L, and 2^{2n-3} combinations for T. So the total number of combinations is $\Theta(n!2^{3n})$.

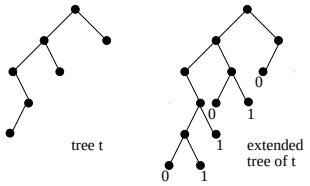

Figure 4: An example of an extended tree

Observation 1 A pair of binary trees t_1 and t_2 can be constructed from a mosaic floorplan if and only if (1) they are twin binary to each other, i.e., $\Theta(t_1) = \Theta^c(t_2)$, and (2) their inorder traversals are the same.

If we extend a tree t by adding a left child of bit 0 to every node (except the leftmost node) that has no left child and by adding a right child of bit 1 to every node (except the rightmost node) that has no right child, the tree obtained is called an *extended* tree of t. An example of an extended tree is shown in Figure 4. Notice that the inorder traversal of the extended tree of t will be $m_1\alpha_1 m_2\alpha_2 \ldots \alpha_{n-1}m_n$ where $m_1 m_2 \ldots m_n$ are the inorder traversal of t and $\alpha_1\alpha_2 \ldots \alpha_{n-1}$ are the labeling of t. Observation 1 can be restated as follows:

Observation 2 A pair of binary trees t_1 and t_2 can be constructed from a mosaic floorplan if and only if the inorder traversal of their extended trees are the same except that all the bits are complemented.

2.2 Definition of Twin Binary Sequences

From observation 1, we know that a pair of binary trees t_1 and t_2 are *valid* (i.e., corresponding to a packing) if and only if their labeling are complement of each other and their inorder traversals are the same. However, the labeling and the inorder traversal are not sufficient to identify a unique pair of t_1 and t_2. Given a permutation of module names π and a labeling α, there can be more than one valid pairs of t_1 and t_2 such that their inorder traversals are π and $\Theta(t_1) = \Theta^c(t_2) = \alpha$. In order to identify a pair of trees uniquely, we need two additional bit sequences β and β' for t_1 and t_2 respectively such that the i^{th} bit in β and β' tells whether the i^{th} module in α is the left child (when the bit is 0) or the right child (when the bit is 1) of its parent in t_1 and t_2 respectively. These bits are called the *directional bits*. If module k is the root of a tree, its directional bit will be assigned to zero.

For a binary tree t, its labeling sequence $\alpha = \alpha_1\alpha_2 \ldots \alpha_{n-1}$ and its directional bit sequence $\beta = \beta_1\beta_2 \ldots \beta_n$ must satisfy the following conditions:

(1) In the bit sequence $\beta_1\alpha_1\beta_2 \ldots \alpha_{n-1}\beta_n$, the number of 0's is one more than the number of 1's.

(2) For any prefix of the bit sequence $\beta_1\alpha_1\beta_2 \ldots \alpha_{n-1}\beta_n$, the number of 0's is more than or equal to the number of 1's.

We proved the following lemmas which shows that conditions (1) and (2) are necessary and sufficient for a pair of sequences α and β to correspond to a binary tree. The proof is not shown here because of the limitation in space.

LEMMA 1. *For any binary tree, its labeling sequence α and directional bit sequence β must satisfy conditions (1) and (2).*

LEMMA 2. *For any binary sequences α of $n-1$ bits and β of n bits satisfying conditions (1) and (2), there exists a unique binary tree t such that the labeling sequence of t is α and the directional bit sequence of t is β.*

Now, we can define a twin binary sequences representation. A twin binary sequence s for n modules is a 4-tuple:

$$s = (\pi, \alpha, \beta, \beta')$$

where π is a permutation of the n modules, both α and β, and α^c (the complement of α) and β' satisfy conditions (1) and (2). From the above observations and lemmas, we can prove the following two theorems:

THEOREM 1. *There is a one-to-one mapping between twin binary sequences and twin binary trees.*

THEOREM 2. *There is a one-to-one mapping between twin binary sequences and mosaic floorplan.*

2.3 Insertion of Empty Rooms

A twin binary sequences s represents a mosaic floorplan F. Now we want to insert an exact number of empty rooms at the right places in F to obtain a corresponding non-slicing floorplan F' such that every non-slicing floorplan can be generated by this method from one mosaic floorplan non-redundantly. There are two kinds of empty rooms. One is resulted because a big room is assigned to a small module. This kind of empty room is called *reducible empty room*. An example is shown in Figure 5(a). Another kind of empty room is called *irreducible empty room* and is defined as follows:

DEFINITION 2. *An irreducible empty room is an empty room that cannot be removed by merging with another room in the packing.*

An example of an irreducible empty room is shown in Figure 5(b). We observed that an irreducible empty room must be of *wheel* shape and its four *adjacent rooms* (the rooms that shares a T-junction at one of its corners) must not be irreducible empty rooms themselves:

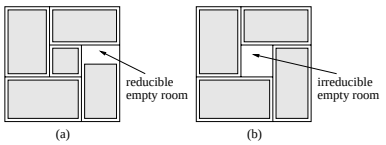

Figure 5: An example of reducible and irreducible empty rooms

LEMMA 3. *The T-junctions at the four corners of an irreducible empty room must form a wheel structure (Figure 6).*

PROOF. If an empty room X does not form a wheel structure, there is at least one slicing cut on one of its four sides. By removing this slicing cut, we can merge X with the room on the other side of the slicing cut and X can be removed. □

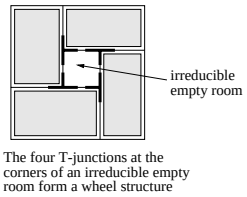

Figure 6: A wheel structure

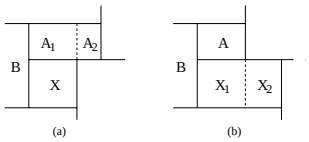

Figure 7: Two cases in Lemma 4

LEMMA 4. *The adjacent rooms at the four T-junctions of an irreducible empty room must not be irreducible empty rooms themselves.*

PROOF. Without lost of generality, we consider an irreducible empty room X of a clockwise wheel shape, and assume that its adjacent room A sharing with X the T-junction at its upper left corner is also an irreducible empty room (Figure 7). Then A must be an anti-clockwise wheel. There are two cases: (1) If $width(A) \geq width(S)$, X can be merged with A_1 (Figure 7(a)) to form a new empty room. This empty room $X + A_1$ is reducible, and can be removed by combining with the modules on the right hand side (labeled B). (2) If $width(A) \leq width(S)$, A can be merged with X_1 (Figure 7(b)) to form a new empty room and similar argument follows. In both cases, we are able to reduce the number of irreducible empty rooms by one. By repeating the above process, we will either end up with only one irreducible empty room that must satisfy the condition, or the situation that every remaining irreducible empty room does not share a T-junction with each other. □

2.3.1 Mapping Between Mosaic Floorplan and General Non-slicing Floorplan

In this section, we will show how a non-slicing floorplan F' can be constructed from a mosaic floorplan F by inserting some irreducible empty rooms at the right places in F. For simplicity, we will make use of twin binary trees for explanation. That means, given a mosaic floorplan F represented by a twin binary trees t_1 and t_2, we want to insert the minimal number of empty rooms (represented by X) to t_1 and t_2 appropriately so that they will correspond to a valid non-slicing floorplan F', and the method should be such that every non-slicing floorplan can be constructed by this method uniquely from one and only one mosaic floorplan. To construct a non-slicing floorplan from a mosaic floorplan, we only need to consider those irreducible empty rooms, because all reducible empty rooms can be removed by merging with some neighboring rooms. From Lemma 3, we know that an irreducible empty room must be of the shape of a wheel, so its structure in the twin binary trees must be of the form as shown in Figure 8. In our approach, we will use the following mapping M_x to create irreducible empty rooms from a sliceline structure.

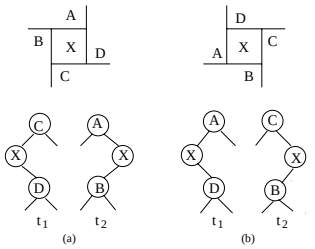

Figure 8: Tree structure of an irreducible empty room

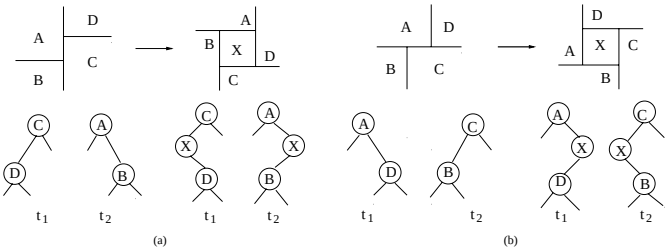

Figure 9: Mapping between mosaic floorplan and non-slicing floorplan

DEFINITION 3. *The mapping M_x will map a vertical (horizontal) sliceline with one T-junction on each side to an irreducible empty room of anti-clockwise (clockwise) wheel shape (Figure 9).*

It is not difficult to prove the uniqueness of this mapping as stated in the next Lemma:

LEMMA 5. *Every non-slicing floorplan can be mapped by M_x from one and only one mosaic floorplan.*

From Lemma 4, we know that the adjacent rooms of an irreducible empty room must be occupied. Therefore if we want to insert X's into the twin binary trees t_1 and t_2 of a mosaic floorplan, the X's must be inserted between some module nodes as shown in Figure 10. Given this observation, we will first insert as many X's as possible (i.e., $n-1$) into t_1 and t_2 to obtain another pair of trees t_1' and t_2'. An example is shown in Figure 11(b). Now, the most difficult task is to select those X's that are inserted correctly. According to Observation 2, a pair of twin binary trees are valid (correspond to a packing) if and only if the inorder traversal of their extended trees are equivalent except that all the bits are reversed. Therefore, in order to find out those valid X's, we will write down the inorder traversal of the extended trees of t_1' and t_2' and try to match the X's. The matching is not difficult since there must be an equal number of X's between any two neighboring module names (Figure 11(c)). We may need to make a choice when there are more than one X's between two modules. For example, in Figure 11(c), there is one X between C and D in the first sequence and there are two X's in the second sequence. In this case, we can match one pair of X's. There are two choices from the second sequence, and they will correspond to different non-slicing structures as shown in Figure 11(c). Every matching will correspond uniquely to a valid floorplan, and each non-slicing floorplan can be constructed uniquely by this method from one and only one mosaic floorplan.

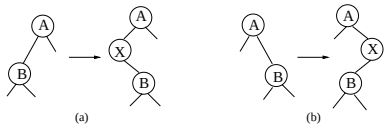

Figure 10: The only two ways to insert X into a tree

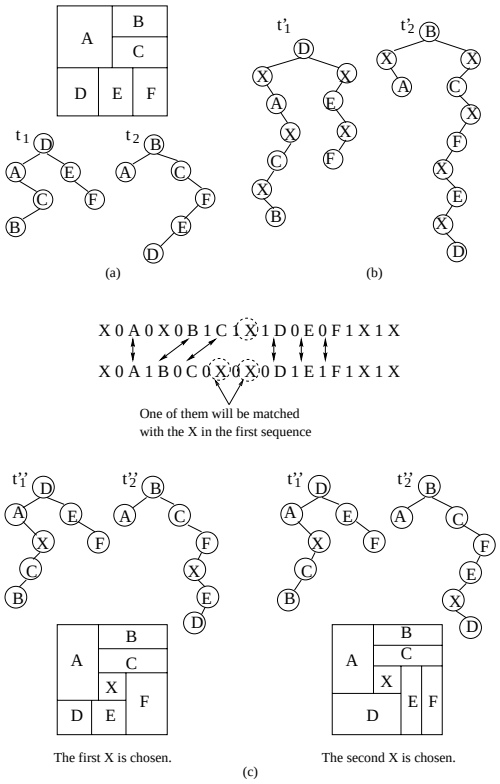

Figure 11: A simple example of constructing a non-slicing floorplan from a mosaic floorplan

In our implementation, we do not need to build the trees explicitly to insert the empty rooms. We can scan the twin binary sequences $s = (\pi, \alpha, \beta, \beta')$ once to find out the positions of the X's in the inorder sequences of t_1 and t_2 after insertion. This is possible because of the following observation. Consider an X inserted at a node position B in a tree. If B is the left child of its parent A (Figure 10 (a)), this inserted X will appear just before the left subtree of B in the inorder sequence of the extended tree. Similarly, if B is the right child of its parent A (Figure 10 (b)), this inserted X will appear just after the right subtree of B in the inorder sequence of the extended tree. A simple algorithm can be used to break down the subtree structure of a tree and find out the positions of all the X's in the inorder sequence in linear time. The details are omitted here because of the space limitation. After that, matching can be done as described above.

2.4 Floorplan Realization

In order to realize a floorplan from its TBS representation efficiently, we devised an algorithm that only needs to scan the sequences once from right to left to construct the packing:

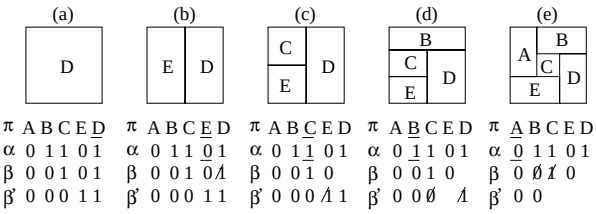

Figure 12: A simple example of constructing a floorplan from its TBS

Algorithm TBStoFloorplan
Input: A TBS $s = (\pi, \alpha, \beta, \beta')$
Output: Packing P *corresponding to* s

Begin
1. *Append* α *with bit '1', i.e.,* $\alpha_n = 1$.
2. *Initially, we have only module* π_n *in* P.
3. *For* $i = n-1$ *downto* $i = 1$:
4. *If* $(\alpha_i = 0)$:
5. *Find the largest* k *s.t.* $i < k \leq n$ *and* $\beta_k = 1$.
6. *Add module* π_i *to* P *from the left, pushing modules* $\pi_{i+1}, \pi_{i+2}, \ldots, \pi_k$ *to the right. Note that modules* $\pi_{i+1}, \pi_{i+2}, \ldots, \pi_k$ *will be lying on the left boundary of* P *at this moment.*
7. *Delete* $\beta_{i+1}, \beta_{i+2}, \ldots, \beta_k$ *from* β.
8. *If* $(\alpha_i = 1)$:
9. *Find the largest* k *s.t.* $i < k \leq n$ *and* $\beta'_k = 1$
10. *Add module* π_i *to* P *from above, pushing modules* $\pi_{i+1}, \pi_{i+2}, \ldots, \pi_k$ *down. Note that modules* $\pi_{i+1}, \pi_{i+2}, \ldots, \pi_k$ *will be lying on the top boundary of* P *at this moment.*
11. *Delete* $\beta'_{i+1}, \beta'_{i+2}, \ldots, \beta'_k$ *from* β'.
End

We have proved the correctness of the above algorithm, but the proof is is not shown here because of the space limitation. A simple example illustrating the steps of the algorithm is given in Figure 12.

2.5 Size of Solution Space

Consider a TBS $(\pi, \alpha, \beta, \beta')$ for n modules. α and β uniquely specify a rooted ordered binary tree. So the number of combinations of α and β is given by the Catalan number. Since the number of combinations for π is $n!$, the number of combinations for β' is upper-bounded by $O(2^n)$, and the Catalan number is upper-bounded by $O(2^{2n}/n^{1.5})$, the number of different TBS configurations (i.e., on the Baxter number) is bounded by $O(n!2^{3n}/n^{1.5})$.

3. TIGHT BOUND ON NUMBER OF IRREDUCIBLE EMPTY ROOMS

In order to describe non-slicing structure by a mosaic floorplan representation, some previous work [9] include dummy blocks of zero area in the set of modules. The method described in section 2.3 is very efficient but it is applicable to the TBS representation only. In general, we only need to have $n-1$ extra dummy blocks in order to represent all non-slicing structures by a mosaic floorplan representation. This is an improvement to the previous result [9] in which $O(n^2)$ dummy blocks are needed. We have proved an upper bound of $n-1$ and a lower bound of $n - 2\sqrt{n} + 1$ on the number of irreducible empty rooms in a general non-slicing floorplan. (An example with 49 modules and 43 irreducible

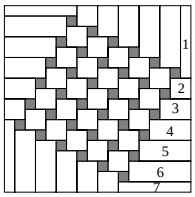

Figure 13: An example with many irreducible empty rooms.

empty rooms is shown in Figure 13.) It means that $n-1$ dummy blocks are needed and we cannot use much less.

THEOREM 3. *In a non-slicing floorplan P, there can be at most $n-1$ irreducible empty rooms.*

THEOREM 4. *There exists a non-slicing floorplan P of n modules and $n-2\sqrt{n}+1$ irreducible empty rooms.*

4. EXPERIMENTAL RESULTS

All experiments are performed on a PC with 1400MHz Intel Xeon Processor and 256Mb Memory. Simulated annealing is used to search for a good TBS. The initial temperature is set to 1.5×10^6 initially and is lowered at a constant rate of 0.95 to 0.97 until it is below 1×10^{-10}. The number of iterations at one temperature step is 30.

In every iteration of the annealing process, we will modify the TBS by one of the following five kinds of moves:

- **M1:** Exchange two modules in π.
- **M2:** Flip one bit in α.
- **M3:** Exchange two bits of different values in β.
- **M4:** Exchange two bits of different values in β'.
- **M5:** Change the width and height of a module.

We design the moves such that all TBS's are reachable. In addition, we will make sure that the sequences obtained after each move is a TBS (i.e., satisfying conditions (1)-(2)). Moves M1 and M5 takes $O(1)$ time and the moves M2, M3 and M4 takes $O(n)$ time.

We test our algorithm with empty room insertion on six MCNC benchmarks. We also run the algorithm with empty room insertion disabled. In other words, only mosaic floorplan can be generated. For each case, two objective functions are considered. The first one aims at minimizing area only, while the second one will minimize a weighted sum of area and wirelength. The weights are set such that the costs of area and wirelength are approximately equal. For each experiment, ten runs are performed and the result of the best run is reported. The results for area minimization is shown in Table 1. The results for area and wire length minimization is shown in Table 2.

As the results show, our floorplanner can produce high-quality floorplans in a very short runtime. We also notice that empty room insertion is very effective in reducing the floorplan area. If empty room insertion is disabled, the deadspace is worse for all but two cases. The deadspace is 32.84% more on average. However, with empty room insertion, the floorplanner is about 40.8% slower.

MCNC benchmark	TBS (with X)		TBS (no X)	
	% Dead-space	Run-time (s)	% Dead-space	Run- (s) time (s)
apte	1.89	0.86	1.30	0.73
xerox	2.17	1.30	2.46	1.20
hp	2.10	0.76	2.22	0.63
ami33a	3.05	1.26	4.05	0.98
ami49a	4.05	2.55	4.38	2.08
playout	6.20	2.58	7.60	1.09

Table 1: Area minimization.

MCNC benchmark	TBS (with X)			
	% Dead-space	Wire-length	Cost	Run-time (s)
apte	1.79	12652	95492	0.89
xerox	2.64	14937	39295	1.36
hp	1.32	4246	18291	0.73
ami33a	8.41	6078	26000	1.30
ami49a	9.40	29668	80660	2.60
playout	5.19	2.373	9341	2.50
MCNC benchmark	TBS (no X)			
	% Dead-space	Wire-length	Cost	Run-time (s)
apte	3.45	13267	98642	0.62
xerox	4.41	14738	39405	1.22
hp	3.43	4292	18587	0.61
ami33a	7.25	6488	26742	1.02
ami49a	10.82	30256	82107	2.14
playout	6.32	2.265	9454	1.08

Table 2: Area and wirelength minimization.

5. REFERENCES

[1] Y.C. Chang, Y.W. Chang, G.M. Wu, and S.W. Wu. B*-Trees: A New Representation for Non-Slicing Floorplans. *Proceedings of the 37th ACM/IEEE Design Automation Conference*, 2000.

[2] S. Dulucq and O. Guibert. Baxter Permutations. *Discrete Mathematics*, 180:143–156, 1998.

[3] Pei-Ning Guo, Chung-Kuan Cheng, and Takeshi Yoshimura. An O-Tree Representation of Non-Slicing Floorplan and Its Applications. *Proceedings of the 36th ACM/IEEE Design Automation Conference*, pages 268–273, 1999.

[4] Xianlong Hong, Gang Huang, Yici Cai, Jiangchun Gu, Sheqin Dong, Chung-Kuan Cheng, and Jun Gu. Corner Block List: An Effective and Efficient Topological Representation of Non-Slicing Floorplan. *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pages 8–12, 2000.

[5] H. Murata, K. Fujiyoushi, S. Nakatake, and Y. Kajitani. Rectangle-Packing-Based Module Placement. *Proceedings IEEE International Conference on Computer-Aided Design*, pages 472–479, 1995.

[6] S. Nakatake, K. Fujiyoushi, H. Murata, and Y. Kajitani. Module Placement on BSG-Structure and IC Layout Applications. *Proceedings IEEE International Conference on Computer-Aided Design*, pages 484–491, 1996.

[7] D.F. Wong and C.L. Liu. A New Algorithm for Floorplan Design. *Proceedings of the 23rd ACM/IEEE Design Automation Conference*, pages 101–107, 1986.

[8] B. Yao, H. Chen, C.K. Cheng, and R. Graham. Revisiting Floorplan Representations. *Proceedings of International Symposium on Physical Design*, pages 138–143, 2001.

[9] S. Zhou, S. Dong, X. Hong, Y. Cai, and C.-K. Cheng. ECBL: An Extended Corner Block List with Solution Space Including Optimum Placement. *Proceedings of International Symposium on Physical Design*, pages 156–161, 2001.