Vol.2014-AL-148 No.6
2014/6/13

Compact Codes of Slicing Floorplans

Takafumi Ohmori[1,a)] Katsuhisa Yamanaka[1,b)] Takashi Hirayama[1] Yasuaki Nishitani[1]

Abstract: A floorplan is a partition (dissection) of a rectangle into small rectangles, called *blocks*, by horizontal and vertical line segments such that no four rectangles meet at the same point. Floorplans are used to design the layout of very-large-scale integration (VLSI) circuits. Since modern VLSI circuits are extremely large, it is necessary to design a compact representation of floorplans. Slicing floorplans are one of important classes of floorplans. It is also desirable to design a compact representation of the slicing floorplans. We therefore address a problem of designing compact code for the slicing floorplans. We first propose a code for a slicing floorplan of $3n-3$ bits. We then propose a more compact code for a slicing floorplan. Finally, we experimentally compare the two codes.

Keywords:
floorplan, slicing floorplan, coding, decoding, slicing tree, skewed slicing tree

1. Introduction

A floorplan is a partition (dissection) of a rectangle into small rectangles, called blocks, by horizontal and vertical line segments such that no four rectangles meet at the same point. Floorplans are used to design the layout of very-large-scale integration (VLSI) circuits. Since modern VLSI circuits are extremely large, it is necessary to design a compact representation of floorplans. Slicing floorplans are one of important classes of floorplans. It is also desirable to design a compact representation of the slicing floorplans. We therefore address a problem of designing compact code for the slicing floorplans.

Recently, compact codes for floorplans have been proposed. Table 1 shows some results on compact codes for floorplans and their information-theoretic lower bounds. A (normal) floorplan represents adjacency relations among blocks. On the other hand, a mosaic floorplan represents adjacency relations among maximal line segments. For example, the two floorplans in Fig.1 (a) and (b) are non-isomorphic as floorplans, but are isomorphic as mosaic floorplans. See [1] for formal definitions of floorplans and mosaic floorplans. To the best of our knowledge, the most compact code for a (normal) floorplan was proposed by Takahashi *et al.* [6]. The length of their code is $4n-4$ bits, where n is the number of blocks of a floorplan. For mosaic floorplans, He [1] and Talahashi [7] proposed codes of $3n - O(1)$ bits, independently. Since the information-theoretic lower bounds of mosaic floorplan is $3n - o(n)$ bits, both codes are asymptotically optimal.

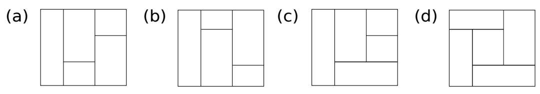

Fig. 1 Examples of floorplans.

[1] Iwate University, Ueda 4-3-5, Morioka, Iwate, 020-8551, Japan
a) t-ohmori@kono.cis.iwate-u.ac.jp
b) yamanaka@cis.iwate-u.ac.jp

Table 1 Existing results on coding floorplans.

	The lengths of the existing codes (bits)	Information-theoretic lower bounds (bits)
Floorplan [6]	$4n-4$	$3.53n$
Mosaic floorplan [1],[7]	$3n-3$	$3n-o(n)$
Slicing floorplan [This paper]	$5n/2+m_1-p_{10}-4$	$2.543n-o(n)$

A floorplan is a slicing floorplan if it is obtained by recursively cutting a rectangle into two blocks by a vertical line or horizontal line. We will give a definition of the slicing floorplan in Section 2. The class of slicing floorplans is a proper subset of the class of mosaic floorplans. Since a slicing floorplan has the simple structure, it is important from mathematical and application points of views. For example, a bijection between separable permutations and slicing floorplans are known [4], and slicing floorplans are used in VLSI layout design [3].

In this paper, we address a problem of designing a compact code for slicing floorplans. It is known that the information-theoretic lower bound of slicing floorplans with n blocks is $2.543n - o(n)$ bits [8]. Thus, any code of slicing floorplan with n blocks needs at least $2.543n$ bits on average.

The paper is organized as follows. Section 2 gives definitions. In Sections 3, we propose a code of $3n$ bits for a slicing floorplan with n blocks. In Section 4, we improve the code. In Section 5, we compare two codes by experiments. Finally, Section 6 is the conclusion and future work.

2. Definition

In this section, we give some definitions.

2.1 Slicing floorplan

We recursively define a *slicing floorplan* as follows.

(1) The floorplan with exactly one block is a slicing floorplan.
(2) Let F_1, F_2 be any two slicing floorplans. The floorplan obtained by merging the rightmost line segment of F_1, and the leftmost vertical line segment of F_2 is a slicing floorplan.

Similarly, the floorplan obtained by merging the uppermost line segment of F_1, and the lowermost horizontal line segment of F_2 is a slicing floorplan.

Fig.2 (a), illustrates a recursive structure of the floorplan in Fig.1 (a). Note that the floorplan in Fig.1 (d) has no recursive structure, and hence it is a non-slicing floorplan.

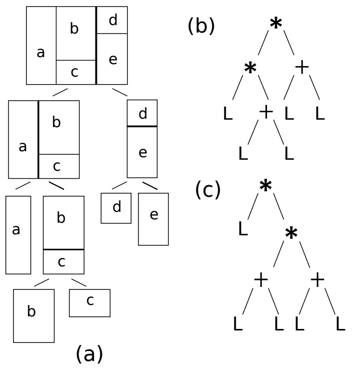

Fig. 2 Illustration for recursive structure of a slicing floorplan.

The *base* of a slicing floorplan is the lowermost horizontal line segment. In this paper, we draw the base of a slicing floorplan as the lowermost horizontal line segment.

Now we explain isomorphism of slicing floorplans. Two slicing floorplan F_1 and F_2 are *isomorphic* if there exist a one-to-one correspondence between maximal vertical line segments and a one-to-one correspondence between maximal horizontal line segments such that the set of blocks located to the top and the bottom of each maximal horizontal line segment, and the set of blocks to the left and the right of each maximal vertical line segment are preserved, respectively. Intuitively, slicing floorplan are isomorphic if and only if they can be converted to each other by sliding some maximal horizontal and vertical line segments, preserving the sets of blocks located to the top, bottom, left and right of each maximal line segment. For example, the slicing floorplan in Fig.1 (a) is obtained from the slicing floorplan in Fig.1 (b) by sliding one horizontal line segment and hence the two slicing floorplans are isomorphic.

2.2 Slicing tree

A *slicing tree* is a binary tree and represents the recursive structure of slicing floorplan. Each vertex of a slicing tree has a label '+', '*', or 'L'. '+' represents a horizontal merge and '*' represents vertical merge in a recursive structure. 'L' represents a single block. Note that the label of a vertex is 'L' if and only if the vertex is a leaf of a slicing tree. For example, the recursive structure of the slicing floorplan in Fig.1 (a) is shown in Fig.2 (a), and its slicing tree is shown in Fig.2 (b).

A slicing floorplan has one or more corresponding slicing tree. For example, the slicing trees in Figs.2 (b) and (c) correspond to the same slicing floorplan. Thus, a relation between slicing floorplans and slicing trees is not bijective. On the other hand, Wang and Lui [1] defined skewed slicing trees such that they have a bijection to slicing floorplans. A slicing tree is a *skewed slicing tree* if the label of each inner vertex of the slicing floorplan is different from the label of its right child. For example,the slicing tree in Fig.2 (b) is the skewed slicing tree of the slicing floorplan in Fig.1 (a).

2.3 Slicing string

Let F be a slicing floorplan with n blocks, and let $T(F)$ be a skewed slicing tree of F. We traverse $T(F)$ with breadth-first manner. In the traverse, we output the label of each vertex when the vertex is visited. We call the obtained string the *slicing string* of F. For example, the slicing string of the slicing flooorplan in Fig.1 (a) is "**+L+LLLL".

It is obvious that we can construct the slicing string from a skewed slicing tree, and reconstruct the skewed slicing tree from a slicing string. We therefore from now on address to code the slicing string of a skewed slicing tree.

3. Breadth-first code

In this section, we propose the code of $3n - O(1)$ bits for a slicing floorplan with n blocks.

3.1 Coding

A label appeared in a slicing string is '+', '*', or 'L'. We code the three labels with `00`, `01`, and `1`, respectively. Table 2 shows the correspondence between the labels and their codes. We code each label in a slicing string according to Table 2. We call the obtained code a *breadth-first code* of a slicing floorplan F, denoted by $B(F)$.

Table 2 Correspondence between the labels and their codes.

label	code
+	00
*	01
L	1

3.2 Decoding

We can reconstruct a slicing string from the breadth-first code straightforwardly. If we first read `0`, then we read the next bit. If the next bit is `0`, then the current two bits represent the label '+.' Otherwise, if the next bit is `1`, then the current two bits represent the label '*.' If we first read `1`, then this bit represents the label 'L.' By repeating this process, we can reconstruct the original slicing string.

3.3 Estimation of the length of breadth-first code

Let F be a slicing floorplan with n blocks, and let $B(F)$ be the breadth-first code of F. We denote by $|B(F)|$ the length of $B(F)$. Since the skewed slicing tree of F has $n-1$ inner vertices and n leaves, we have

$$|B(F)| = n + 2(n-1) = 3n - 2. \qquad (1)$$

Therefore, we have the following theorem.

Theorem 1 Given a slicing floorplan with n blocks and its skewed slicing tree, we can code the slicing floorplan by $3n-2$ bits. Each of coding and decoding takes $O(n)$ time.

4. Slicing-pair code

In this section, we improve the breadth-first code. The idea of our improvement is to code a pair of vertices in a skewed slicing tree. The same approach is appeared in [7] to code an unordered binary tree.

4.1 Coding

Let F be a slicing floorplan with n blocks, and let $S(F) = (s_1, s_2, \ldots, s_{2n-1})$ be the slicing string of F. $S'(F) = (s'_1, s'_2, \ldots, s'_{2n-2})$ is the string obtained by removing s_1 in $S(F)$. A *slicing-pair* is a pair (s'_i, s'_{i+1}) for each $i = 1, 3, \ldots, 2n-3$. Note that two vertices corresponding s'_i and s'_{i+1} in a slicing-pair (s'_i, s'_{i+1}) have the same parent. See Fig.3 for an example. Let P_i be the parent of s'_i, and s'_{i+1} in a slicing-pair (s'_i, s'_{i+1}). If the label of P_i is '+', then $s_{i+1} \neq$'+' holds. In this case, the possible patterns of labels of (s'_i, s'_{i+1}) are six patterns. Table 3 shows the possible patterns. Otherwise, the label of P_i is '*', then the possible patterns of (s'_i, s'_{i+1}) is the six patterns as shown in Table 3. We assign two or three bits to each pattern of a slicing-pair, as shown in Table 3. The label of root is either '+' or '*', and hence we can encode it by one bit. The last slicing-pair is always (L,L), and hence no bits is assigned to it. We call the *slicing-pair code* the code obtained by the above assignment of bits.

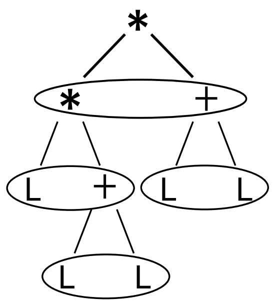

Fig. 3 Slicing-pair in a skewed slicing tree.

Table 3 Patterns of slicing-pair and their codes.

Label of parent	Label of slicing-pair pair	Assigned bits
+	(+,*)	000
	(+,L)	001
	(*,*)	010
	(*,L)	011
	(L,*)	10
	(L,L)	11
*	(+,+)	000
	(+,L)	001
	(*,+)	010
	(*,L)	011
	(L,+)	10
	(L,L)	11

4.2 Decoding

Now we show how to reconstruct the original slicing string from the slicing-pair code. Basic idea is to reconstruct each slicing-pair using Table 3. The detail is as follows. Let $SP(F)$ be a slicing-pair code for a slicing floorplan F. The first bit of $SP(F)$ represents whether the label of the root is '+' or '*'. Suppose that a slicing-pair (s'_i, s'_{i+1}) for each $i = 1, 3, \ldots, j-2$ is reconstructed then the slicing-pair (s'_j, s'_{j+1}) is reconstructed using Table 3, because we know the label of the parent of (s'_j, s'_{j+1}). We repeat this process, and we finally insert the omitted (L,L). By the above process, we obtain a sequence of slicing-pairs in the order of a slicing string and hence we can obtain the original slicing string.

4.3 Estimation of the length of slicing-pair code

In this section, we estimate the length of the slicing pair code. We first give some definitions. Let F be a slicing floorplan with n blocks, and let m be the number of inner vertices of a slicing tree of F. We denote by $m_i (i = 0, 1, 2)$ the number of vertices each of which has i leaves as its children. Then we have the following two equations:

$$m = m_0 + m_1 + m_2, \tag{2}$$

$$n = m + 1 = 2m_2 + m_1. \tag{3}$$

By Eqs. (2) and (3), we have

$$m_0 = m_2 - 1. \tag{4}$$

Let p_{00} be the number of slicing-pairs in which both labels correspond to inner vertices, let p_{01} be the number of slicing-pairs in which the left label corresponds to an inner vertex and the right label corresponds to a leaf, let p_{10} be the number of slicing-pairs in which the left label corresponds to a leaf and the right label corresponds to an inner vertex, and let p_{11} be the number of slicing-pairs in which both labels correspond to leaves. We immediately obtain the following three equations:

$$p_{00} = m_0 = m_2 - 1, \tag{5}$$

$$p_{01} + p_{10} = m_1, \tag{6}$$

$$p_{11} = m_2. \tag{7}$$

Now we estimate the length of the slicing-pair code $SP(F)$ of F, denoted by $|SP(F)|$. From Table 3, we can observe that each label for p_{00} uses three bits; each label for p_{01} uses three bits; each label for p_{10} uses two bits; each label for p_{11} uses two bits. We therefore obtain the following equation:

$$|SP(F)| = 3p_{00} + 3p_{01} + 2p_{10} + 2(p_{11} - 1) + 1$$

Recall that the label of the root uses one bit and the code for the last slicing-pair is saved. By Eqs.(3), (5), (6), and (7), we obtain the following estimation:

$$\begin{aligned} |SP(F)| &= 3p_{00} + 3p_{01} + 2p_{10} + 2(p_{11} - 1) + 1 \\ &= 3(m_2 - 1) + 3(m_1 - p_{10}) + 2(m_1 - p_{01}) + 2(m_2 - 1) + 1 \\ &= 5m_2 + 5m_1 - 3p_{10} - 2p_{01} - 4 \\ &= 5/2(2m_2 + m_1) + 5m_1/2 - 2(p_{01} + p_{10}) - p_{10} - 4 \\ &= 5n/2 + 5m_1/2 - 2m_1 - p_{10} - 4 \\ &= 5n/2 + m_1/2 - p_{10} - 4. \end{aligned}$$

This gives our main result as in the following theorem.

Theorem 2 Given a slicing floorplan with n blocks and its skewed slicing tree, we can code the slicing floorplan by $5n/2 + m_1/2 - p_{10} - 4$ bits. Each of coding and decoding takes $O(n)$ time.

4.4 Range of length

We proposed two codes for slicing floorplans: breadth-first code and slicing-pair code. Which code is more compact? We now estimate the range of the length of $SP(F)$ of a slicing floorplan F. We first consider the best case of $|SP(F)|$. If the skewed slicing tree has $n - 2$ inner vertices each of which has a leaf as its left child and has an inner vertex as its right child. Fig.4 (a) shows an example of such a skewed slicing tree. In this case, we have

$$\begin{aligned}|SP(F)| &= 5n/2 + (n-2)/2 - (n-2) - 4\\ &= 2n - 3.\end{aligned}$$

On the other hand, if a skewed slicing tree has $n - 2$ vertices each of which has an inner vertex as its left child and has a leaf as its right child (See Fig.4 (b) for an example), then $|SP(F)|$ takes the maximum number of bits:

$$\begin{aligned}|SP(F)| &= 5n/2 + (n-2)/2 - 0 - 4.\\ &= 3n - 5\end{aligned}$$

We therefore obtain the following inequality:

$$2n - 3 \leq |SP(F)| \leq 3n - 5. \qquad (8)$$

Eq.(8) implies that in worst case the lengths of the breadth-first code and the slicing-pair code are almost equal if we ignore constant terms. In the next section, we experimentally compare the average length of the two codes.

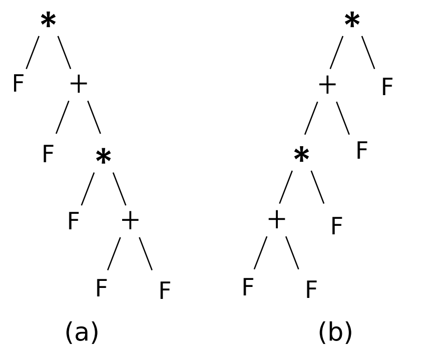

Fig. 4 Minimum and maximum saved slicing tree.

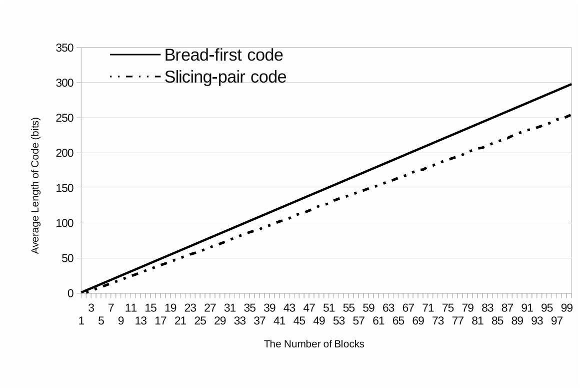

Fig. 5 Comparison of the two codes.

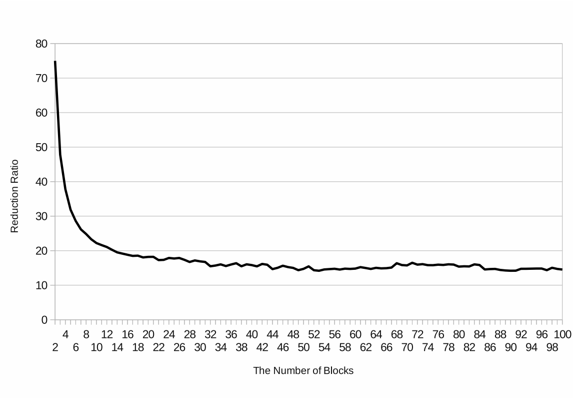

Fig. 6 Reduction rate of slicing-pair code.

Table 4 Experiment environment.

CPU	Intel Core i7-4770S(3.1GHz/4C8T)
Memory	DDR3-1333 8GBx2
OS	FreeBSD 9.1-RELEASE amd64
Language	Haskell
Compiler	GHCi version 7.6.3

5. Computational experiment

In this section, we compare the average length of the breadth-first code and the slicing-pair code. We first generate 10,000 slicing floorplans with n blocks for each $n = 1, 2, \ldots, 100$ uniformly at random. We code each slicing floorplan; then calculate the average length for each $n = 1, 2, \ldots, 100$. Table 4 shows the environment of our experiment.

For each $n = 1, 2, \ldots, 100$, the average lengths of the two codes are shown in Fig.5 which shows a graph with the average lengths on the y-axis and the number of blocks on the x-axis. For every n, the slicing-pair code is more compact than the breadth-first code. Fig.6 shows a graph with reduction ratio on the y-axis and the number of blocks on the x-axis. From the graph, it can be conjectured that the reduction ratio couerges to roughly 14 %.

6. Conclusion

In this paper, we first proposed a breadth-first code. The breadth-first code represents a slicing floorplan with n blocks by $3n - 2$ bits. We then proposed a slicing-pair code by improving the breadth-first code. The length of the slicing-pair code satisfies the inequality $2n - 3 \leq |SP(F)| \leq 3n - 5$, where $|SP(F)|$ is the length of the slicing-pair code of a slicing floorplan F. In our computational experiments, we investigated the average lengths of the two codes. The experiment showed that the slicing-pair code is more compact than the breadth-first code. Reduction ratio of the two codes was roughly 14% for $n = 100$, where n is the number of blocks.

In this paper, we suppose a slicing floorplan and its skewed slicing tree are given. Thus, our analysis of coding time and decoding time contain no running time for constructing the skewed slicing tree from a given slicing floorplan. It can be observed that a naive method can construct the skewed slicing tree of a slicing floorplan with n blocks in $O(n^2)$ time. Our future works includes to improve this running time to be linear time. One of other future works is to design an asymptotically optimal code.

Acknowledgments This work is partially supported by JSPS KAKENHI Grant Numbers 25330001.

References

[1] B.D. He, Optimal binary representation of mosaic floorplans and Baxter permutations, In *Proc. The 6th International Frontiers in Algorithmics ,and The 8th International Conference on Algorithmic Aspects in Information and Management* (*FAW − AAIM*2012), Vol.7285 of Lecture Notes in Computer Science, pp.1-12, 2012.

[2] B. Yao, H. Chen, C. K. Cheng, and R.L. Graham, Floorplan representations: complexity and connections, *ACM Transactions on Design Automation of Electronic Systems*, Vol.8 (1) pp.55-80, 2003.

[3] D.F. Wong and C.L. Lui, A new algorithm for floorplan design, *Proc. ACM/IEEE Design Automation Conf.*, pp.101-107, 1986.

[4] E. Ackerman, G. Bargnet, and R.Y. Pinter, A bijection between permutations and floorplans, and its applications, In *Discrete Applied Mathematics*, Vol.154, pp.1674-1684, 2006.

[5] K. Iwata, S. Ishiwata, and S. Nakano, A compact encoding of unordered binary trees, In *Theory and Applications of Models of Computation* Vol.6648 of Lecture Notes in Computer Science, pp.106-113, 2011.

[6] T. Takahashi, R. Fujimaki, and Y. Inoue, A (4n-4)-bit representation of a rectangular drawing or floorplan, In *Proc. The 15th International Computing and Combinatorics Conference* (*COCOON*2009), Vol.5609 of Lecture Notes in Computer Science, pp.47-55, 2009.

[7] T. Takahashi, (3n-4)-bit representation of rectangular partitions, In *Proc. The 27th International Technical Conference on Circuits/Systems, Computers and Communications*, 2012.

[8] Z.C. Shen, Chris C.N. Chu, Bounds on the number of slicing, mosaic, and general floorplans, *IEEE Council on Electronic Design Automation*, pp.1354-1361, 2003.