# Optimal tile partition for space region of integrated circuits geometry

P.-Y. Hsiao
C.-Y. Lin
P.-W. Shew

*Indexing terms: VLSI layout, Channel partition, Corner stitching, Tile generation*

**Abstract:** An optimal tile partition (OTP) is presented for partitioning the space region of a VLSI layout plane into rectangular space tiles. It modifies the corner stitching data structure to optimise the space tile partition. There is a serious restriction in the original corner stitching data structure, i.e. the solid rectangles cannot overlap each other, whereas our OTP allows overlapping. This paper also shows three theorems with rigorous proofs and experimental results to obtain the minimal number of the space tiles through the OTP. Moreover, a dynamic plane-sweep algorithm based on region query for the OTP has been developed. Using the OTP, the memory efficiency and the local query operations of the original corner stitching data structure have been enhanced.

## 1 Introduction

To represent a number of 'solid rectangles' in a two-dimensional VLSI layout plane, many efficient spatial data structures [2–4] have been presented for the design of various layout tools. 'Corner stitching' [1] is one of the well known data structures for designing layout editor [5], router [6], compactor and plower [7]. It provides a variety of powerful local region query operations, such as neighbour finding, point finding, area searches, directed area enumeration and channel finding. Corner stitching was originally incorporated into the Magic VLSI system [5] and has contributed to the success of the system.

The primary idea of corner stitching is to divide the space region of a layout plane, which consists of various nonoverlapped solid rectangles, into 'space tiles' by using 'horizontal partition edges' [1] or by using 'vertical partition edges'. For example, a layout with five non-overlapped solid rectangles can be represented by horizontal partition (HCSP), or vertical partition (VCSP) as illustrated in Fig. 1 and Fig. 2, respectively. Since the number of space tiles from HCSP or VCSP ($|HCSP| = 13$, $|VCSP| = 14$) is not minimal, neither HCSP nor VCSP is an optimal tile partition (OTP). An OTP should partition the space region into a minimal number of space tiles [21]. Fig. 3 and Fig. 4 show the horizontal oriented OTP (HOTP) and the vertical oriented OTP (VOTP) for the same layout, where the number of space tiles, $|HOTP| = |VOTP| = 11$.

In corner stitching representation, as described in Reference 1, the space region is partitioned into space tiles

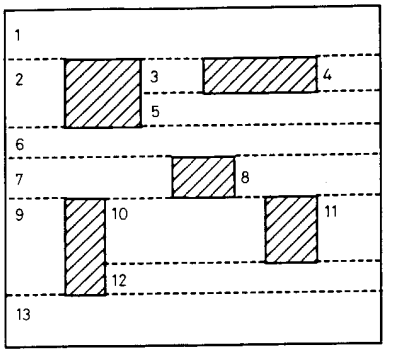



**Fig. 1** *Horizontal partition by corner stitching, HCSP*

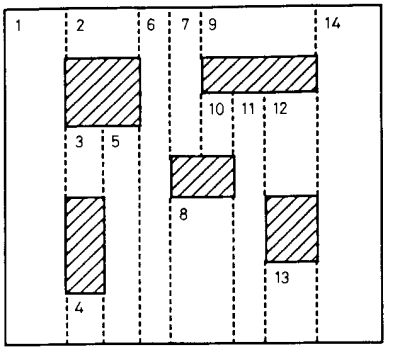



**Fig. 2** *Vertical partition by corner stitching, VCSP*

© IEE, 1993

Paper 9361E (E10), first received 23rd December 1991 and in revised form 21st September 1992

P.-Y. Hsiao and C.-Y. Lin are with the Department of Computer and Information Science, National Chaio Tung University, Hsinchu, Taiwan, Republic of China

P.-W. Shew is with the Department of Electrical Engineering, National University of Singapore, 10 Kent Ridge Crescent, Singapore 0511

This work was supported in part by the National Science Council, Republic of China, under contract number NSC 80-0404-E-009-73 and NSC 81-0404-E-009-525. Thanks are also due to Ms. Fang Fang, Mr. Fang-Chi Li and Mr. Jun-Ren Tzong for their contributions in the preliminary stages of this work.

such that no space tile has other space tiles immediately to its left or right. In a layout with $N$ nonoverlapped solid rectangles, there will be no more than $3N + 1$ space tiles. Hence, in the worst case, the memory requirement for corner stitching is three times more than that for linked lists. To obtain a minimal number of space tiles, as presented in the following sections, will promote the memory efficiency and the local search speed.

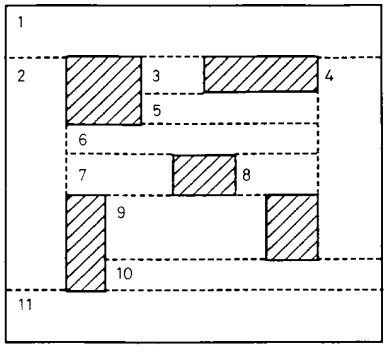



**Fig. 3** *The horizontal oriented optimal tile partition, HOTP*

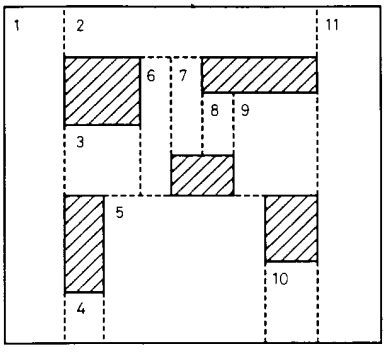



**Fig. 4** *The vertical oriented optimal tile partition, VOTP*

The neighbouring solid rectangles and space tiles are linked together at their corners (hence the name 'corner stitching'). Referring to Fig. 5, the four corner stitches (pointers) are the top-right stitch (tr), the right-top stitch (rt), the left-bottom stitch (lb), and the bottom-left stitch (bl), respectively. When a solid rectangle is deleted from or inserted into the layout, these pointers are easily updated.

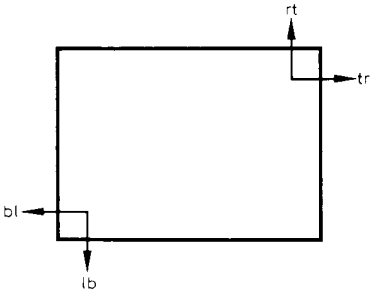



**Fig. 5** *Using four corner pointers to connect the neighbouring rectangles*

In this paper we present the general rules for creating the OTP (HOTP and VOTP) along with theoretical discussion and rigorous proofs. Also, we shall present our algorithm, the plane-sweep OTP. In our approach, we use the quad list quad tree (QLQT) data structure [9] to store the layout information of the solid rectangles. This data structure provides a fast region query speed. The plane-sweep OTP algorithm first uses the plane-sweeping technique and region queries on the QLQT, to determine some parameters (such as those defined in Section 5) and enumerate a set of candidates for critical partition edges (CPE). A CPE is basically a space tile edge extending between corners of two solid rectangles and will certainly appear in both the HOTP and VOTP (refer to Fig. 6 and Section 2 for its formal definition). Second, an elimination algorithm is applied on the enumerated set to obtain a subset of non-intersected CPEs. Next, the outer edges of all clusters of intersected rectangles are traced. Then with the nonintersected CPEs and outer edges, the HOTP and VOTP are obtained by applying the plane-sweep algorithm again.

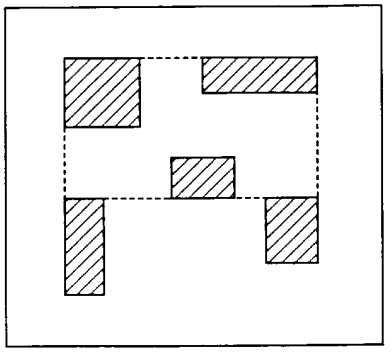


**Fig. 6** *The critical partition edges by the HOTP and the VOTP*

## 2 Preliminary definitions and modelling

Before presenting the OTP, we assume the layout plane is large enough so that the solid rectangles are included entirely inside the layout plane. In other words, the edges of the solid rectangles are never located outside or exactly at the boundary of the layout plane. Some formal definitions used in later parts of this paper now follow.

*Definition 1: partition edge (PE).* The 'partition edge', as shown in Fig. 7, is a vertical or horizontal line segment,

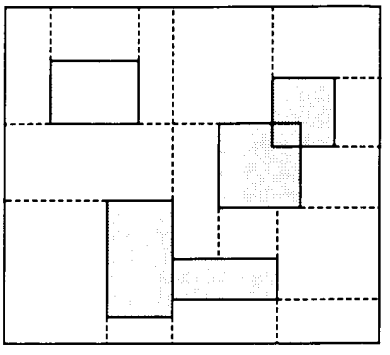


**Fig. 7** *Each dashed line segment representing a reliable partition edge. The whole partition in the figure shows an example of reliable partition edge*

which satisfies the following two conditions, used to partition the spacing region of a given layout:

(i) All the interior points of the line segment must be included inside the space region.

(ii) The end points of the line segment must locate exactly at the boundary rectangle of the layout plane, the edge of a solid rectangle, the corner of a solid rectangle, or the interior point of another partition edge. ■

*Definition 2: reliable partition edge (RPE).* A partition edge satisfying the condition that at least one of its end points is locating at the corner of a solid rectangle is defined as a 'reliable partition edge'. ■

*Definition 3: reliable tile partition (RTP).* For a given layout, the 'reliable tile partition' (see Fig. 7) is a partition of space region by reliable partition edges only and each corner of the solid rectangles should intersect with at least one end point of the reliable partition edge except those corners overlapped with other solid rectangles. ■

Note that for a given layout, there may be a number of different reliable tile partitions. From definition 3, it is obvious that the partition of original corner stitching partition is a kind of reliable tile partitions. As such, Fig. 1 to Fig. 4 are reliable tile partitions of the same layout. Among the possible reliable tile partitions there must be some which possess the minimal number of partitioned space tiles. Those reliable tile partitions with minimal number of space tiles are called 'optimal tile partitions' (OTPs). In this paper, we will focus our argument on two of the most important OTPs: the horizontal oriented OTP (HOTP) and the vertical oriented OTP (VOTP).

*Definition 4: horizontal oriented optimal tile partition (HOTP).* A horizontal oriented optimal tile partition is defined as a reliable tile partition which has a minimal number of partitioned space tiles and satisfies the condition that changing any one of its vertical reliable partition edges into a horizontal reliable partition edge (leaving the end point intersected with the corner of the solid rectangle unchanged) will increase the number of space tiles. ■

Note that most of the reliable partition edges of the HOTP are horizontal. If any of the horizontal reliable partition edges is changed into a vertical one, the number of space tiles ($|SPT|$) must be the same. Moreover, both of the original horizontal reliable partition edge and the updated new vertical reliable partition edge should have a same end point intersected with the corner of the solid rectangle.

*Definition 5: vertical oriented optimal tile partition (VOTP).* VOTP is defined as a reliable tile partition which has a minimal number of space tiles and satisfies the condition that changing any of its horizontal reliable partition edges into vertical reliable partition edge (keeping the end point intersected with the corner of a solid rectangle unchanged), will increase the number of space tiles (see Fig. 4). ■

*Definition 6: critical partition edge (CPE).* For a given layout, the vertical reliable partition edges appeared in the HOTP and the horizontal reliable partition edges appeared in the VOTP are defined as 'critical partition edges' (refer to Figs. 3, 4 and 6). ■

It is obvious that the CPEs play a key role in finding the general partition rules for HOTP and VOTP. Notice that an OTP is obtained from a set of nonintersecting CPEs and other reliable partition edges. We know that the minimal $|SPT|$ can be calculated from the minimal total number of edges of space tiles ($|Espt|$). That means the minimal $|SPT|$ equals the minimal $|Espt|$ divided by four. As a result, to solve the problem of the HOTP and the VOTP, we should focus our argument on $|Espt|$ instead of directly counting $|SPT|$. And the following theorem gives a classification of the Espt. It would be helpful for the discussion in the following Section.

*Theorem 1:* For a given layout with Nsor overlapped solid rectangles, $\boldsymbol{L} = \{R_i \mid i = 1, 2, \ldots, N_{sor}\}$, the minimal number of the edges of the space tiles ($|Espt|_{min}$) can be calculated from the number of the boundary edges of the whole layout ($|Ebr|$), the number of the edges of the solid rectangles ($|Esor|$) and the number of reliable partition edges for the HOTP or the VOTP ($|Erpe|_{otp}$). In other words,

$$|\mathrm{OTP}| = |Espt|_{min}/4, \tag{1}$$

$$= f(|Ebr|, |Esor|, |Erpe|_{otp}) \tag{2}$$

Where $|\mathrm{OTP}|$ denotes the minimal $|SPT|$ partitioned by the HOTP or the VOTP and $f(\cdot)$ means an algebraic function with three variables. ■

*Proof:* As each space tile has four orthogonal boundary edges, eqn. 1 is obvious.

Each edge of space tile may come from six different sources, namely *Ebr*, *Esor*, *Erpe*, segment of *Ebr*, segment of *Esor*, and segment of *Erpe*. However, the segment of *Ebr* is produced by the intersection between *Ebr* and *Erpe*, hence the number of partitioned segments of *Ebr* can be determined by $|Ebr|$ and $|Erpe|_{otp}$. For the same reason, the number of partitioned segments of *Esor* can be determined by $|Esor|$ and $|Erpe|_{otp}$. Similarly, since the segment of *Erpe* is produced by the crossover occurred between a couple of *Erpe*, the number of partitioned segment of *Erpe* can be determined by $|Erpe|_{otp}$ itself. Therefore, we conclude that

$$|Espt|_{min} = \text{function of } |Ebr|, |Esor|, \text{ and } |Erpe|_{otp}. \tag{3}$$

From eqns. 1 and 3, eqn. 2 is proven. QED

## 3 Horizontal oriented OTP and vertical oriented OTP

Our discussion is now limited to finding out the general rules for the HOTP and the VOTP but not for the other kinds of OTP. To obtain the final equation of the general rules shown in theorem 3 in Section 5, first, the given layout should be restricted by some assumptions, and secondly, those assumptions are removed step by step so that the final partition rules presented in theorem 3 can solve any kind of the given layout with a large number of overlapped rectangles.

*Lemma 1:* For a given layout with *Nsor* solid rectangles satisfying the following three assumptions:

*Assumption 1:* The solid rectangles are disjointed (nonoverlapped).

*Assumption 2:* The reliable partition edges are disjointed (nonintersected).

*Assumption 3:* Each reliable partition edge, starting from the corner of solid rectangle must end at the edge of solid rectangle or at the boundary edge of the layout plane but not end at the corner of the other solid rectangles or at the other reliable partition edge.

Eqns. 4–7 must hold:

$$|OTP| = |HCSP| \tag{4}$$

$$= |VCSP| \tag{5}$$

$$= [4 + (4 * Nsor) + (2 * (4 * Nsor))]/4 \tag{6}$$

$$= [|Ebr| + |Esor| + |Erpe^*|]/4 \tag{7}$$

where $|Erpe^*| = 2 * |Erpe|_{otp}$. ■

*Proof:* According to theorem 1, the $|Espt|_{min}$ comes from $|Ebr|$, $|Esor|$, $|Erpe|$, |segment of $Ebr$|, |segment of $Esor$| and |segment of $Erpe$|. Here, it is obvious that $|Ebr| = 4$ and $|Esor| = 4 * Nsor$. From definition 3 and assumption 1 and 3, we have $|Erpe| = 4 * Nsor$. Furthermore, assumption 2 implies that |segment of $Erpe$| = 0.

According to assumption 3, consider line segments shown in Fig. 8, $p$–$p1$, $r$–$r1$ and $i$–$i1$ which are reliable partition edges starting from the corner of solid rectangle

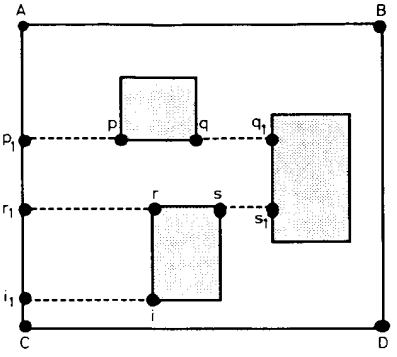



**Fig. 8** *Example for proving that the number of new generated Espt is equal to |segment of Ebr| + |segment of Esor| = |Erpe|*

and ending at the *Ebr*. Each of them partitions the *Ebr* or segment of *Ebr* into two edges of space tiles. For the same reason, each reliable partition edge of $q$–$q1$ and $s$–$s1$ will partition the *Esor* or segment of *Esor* into two edges of space tiles. Consequently, the number of new space tiles generated from the intersections between reliable partition edge and *Ebr* (or *Esor*) is equal to $|Erpe|$. In other words, the sum of the |segment of *Ebr*| and the |segment of *Esor*| equals the $|Erpe|$. Hence, we have

$$|OTP| = \{|Ebr| + |Esor| + |Erpe| + [|\text{segment of } Ebr| + |\text{segment of } Esor|] + |\text{segment of } Erpe|\}/4 \tag{8}$$

$$= [4 + 4 * Nsor + |Erpe| + |Erpe| + 0]/4$$

$$= [4 + 4 * Nsor + 2 * (4 * Nsor)]/4 \tag{6}$$

$$= [|Ebr| + |Esor| + |Erpe^*|]/4 \tag{7}$$

Furthermore, from definition 6, every reliable partition edge under the above three assumptions is not a CPE. Thus we conclude that the HOTP and the HCSP are equal and, likewise, the VOTP and VCSP are equal. Hence eqns. 4 and 5 hold. QED

If there is a reliable partition edge violating assumption 3 and whose end point is exactly intersected with the corner of the other solid rectangle, then the reliable partition edge is called the candidate critical partition edge (CCPE). Most of the CCPEs will become CPEs as shown in Fig. 6. However, some of them will disappear in the HOTP or the VOTP.

*Lemma 2:* For a given layout with *Nsor* solid rectangles satisfying assumptions 1, 2′, 4, and 5, eqns. 8 and 9 must hold.

*Assumption 2′:* The reliable partition edges do not cross over with each other. But the end point of reliable partition edge may touch the other reliable partition edge.

*Assumption 4:* All the CCPEs existing in the given layout must be built up.

*Assumption 5:* There is no double corner for any pair of CCPEs. A double corner as shown in Fig. 9 is a corner of

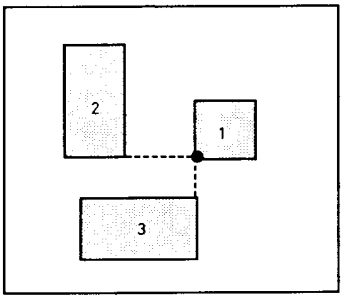



**Fig. 9** *The bottom-left corner of solid rectangle 1 representing a double corner*

solid rectangle such that both of its orthogonal reliable partition edges are CCPEs.

$$|OTP| = \{4 + (4 * Nsor) + [2 * (4 * Nsor) - (4 * Nccpe)]\}/4 \tag{8}$$

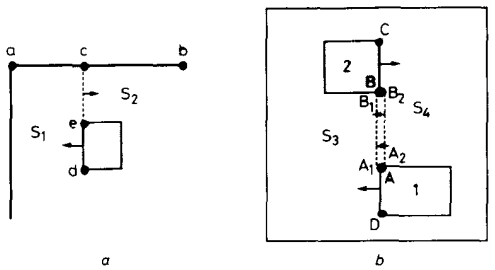
$$= (|Ebr| + |Esor| + |Erpe^{**}|)/4 \tag{9}$$

where $|Erpe^{**}| = 2 * [(4 * Nsor) - (2 * Nccpe)]$ and *Nccpe* is the total number of CCPEs in the given layout. ■

*Proof:* The discard of assumption 3 and the modification from assumption 2 to 2′ imply that some of the reliable partition edges become the CCPEs. The difference between lemma 1 and lemma 2 comes from the influence of the CCPEs. Assumption 4 says that every CCPE must be built up. However, assumption 5 gives a restriction so that the complicated situation (the existence of the double corner) is excluded from this lemma. As a result, the proof should focus on the influence of each simplified and separated CCPE.

From lemma 1, each corner of solid rectangle will project exactly one reliable partition edge for the OTP. Consider Fig. 10*a*, each projected reliable partition edge, *ec*, will increase two edges of space tiles: splitting *ab* to *ac* for *S*1 and *bc* for *S*2, and extending *de* to *cd* for *S*1 and *ce* for *S*2, where *S*1 and *S*2 stand for two new generated space tiles. However, if the projected reliable partition edge is a CCPE, then the CCPE will nullify two corners

of different solid rectangle. According to Fig. 10*b*, each CCPE generates zero edge of space tiles: both *BC* and *AC* for *S*4, and both *AD* and *BD* for *S*2. Hence, consequently, we obtain that the $|Espt|$ generated from the




a b

**Fig. 10** *Example for proving lemma 2*

*a* A normal reliable partition edge increases two edges of space tiles
*b* A CCPE will not generate any new edge of space tiles

reliable partition edges (some of them are CCPEs), which defined as $|Erpe^{**}|$, is equal to $2 * (|Erpe| - 2 * Nccpe)$. Then, from theorem 1 and eqns. 6 and 7, we have

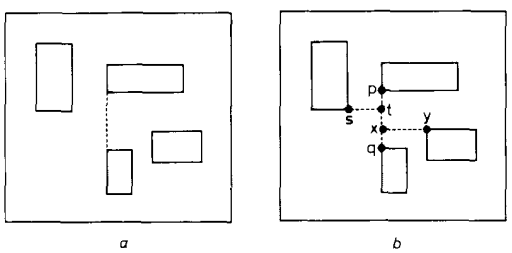
$$|OTP| = (|Ebr| + |Esor| + |Erpe^{**}|)/4 \quad (9)$$

$$= \{4 + (4 * Nsor) + [2 * |Erpe| - 4 * Nccpe]\}/4$$

$$= \{4 + (4 * Nsor) + [2 * (4 * Nsor) - (4 * Nccpe)]\}/4 \quad (8)$$

QED

In lemma 2, firstly, all the CCPEs are built up, and then the normal reliable partition edges are linked up. All the reliable partition edges are allowed to touch the other reliable partition edge by their end points, but are forbidden to cross over each other. Fig. 11 shows the partition process of the HOTP, where the source layout has one CCPE.




a b

**Fig. 11** *An HOTP example to illustrate the partition process of lemma 2*

*a* The source layout and the CCPE
*b* After the HOTP, the reliable partition edge $\overline{xy}$ and $\overline{st}$ should not cross over the other reliable partition edge $\overline{pq}$

To completely solve the problem of OTP, the above assumptions (1, 2', 4, 5) must be removed one by one. In other words, we have to solve the following three subproblems

(i) how to eliminate the crossovers among the CCPEs

(ii) how to correctly maintain the OTP while the double corners exist

(iii) how to partition a group of overlapped solid rectangles.

## 4 Eliminating crossovers and redundant double corners

### 4.1 Algorithm for optimising crossovers and double corners

The crossover of any pair of reliable partition edges (including CCPEs) does inherently increase the number of space tiles except one horizontal CCPE cross over one vertical CCPE only (including 'single double corner' which means that a double corner does not connect with the other double corners through the CCPEs). For an OTP, to remove assumption 2' will not affect the number of space tiles. However, if we try to remove assumption 2' and 4 at the same time, the conflict that for a random layout there may be some CCPEs which cross over each other does exist.

To remove assumption 2', 4 and 5 at the same time, this Section presents an elimination algorithm to discard some of the CCPEs to make all of the remaining CCPEs not crossing and all of the double corners disappear. This algorithm is designed to reserve a maximal number of disjointed CCPEs, because the more the amount of disjointed CCPEs is, the less the amount of space tiles will be.

*Elimination algorithm:* This algorithm will eliminate some of the redundant CCPEs to make all of the remaining CCPEs disjointed. That means after elimination there will be no more crossover and no more intersection at the same double corner for each pair of CCPEs.

*Step 1*

Build up all the CCPEs for the given layout.

*Step 2*

Put the intersected CCPEs into a group ***Gi***.

Put the non-intersected CCPEs into a group ***Gn***.

(All the CCPEs intersected at their end points or interior points are put together into ***Gi***. For example, the set of the dashed line segments shown in Fig. 12 is partial of ***Gi***.)

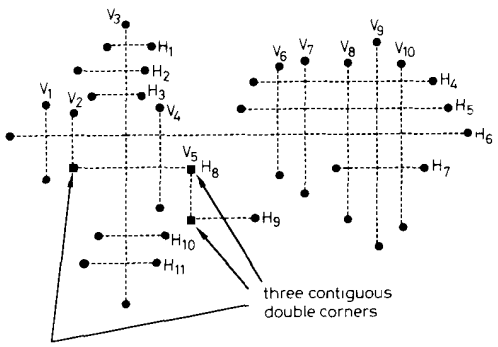



**Fig. 12** *Example for illustrating a group of connected CCPEs*

Each dashed line segment represents one CCPE
The intersection points occurred between ($V_2$, $H_8$), ($H_8$, $V_5$), and ($V_5$, $H_9$) are double corners

*Step 3*

Let all the vertical CCPEs in ***Gi*** labelled $V_1$, $V_2$, ..., $V_n$, be a set ***Sv***.

Let all the horizontal CCPEs in ***Gi*** labelled $H_1$, $H_2$, ..., $H_m$, be a set ***Sh***.

*Step 4*

Mark ***Sv***, do **reduction function**. We obtain a new group ***Gv*** of disjointed CCPEs.

Mark ***Sh***, do **reduction function**. We obtain a new group ***Gh*** of disjointed CCPEs.

*Step 5*

If $|Gh| > |Gv|$, then the set of maximal disjointed CCPEs, ***Sd***, is the union of ***Gh*** and ***Gn***; else, ***Sd*** is the union of ***Gv*** and ***Gn***.

*Step 6*

Stop.

*Reduction function:* This function will be used in the **elimination algorithm**.

*Step 1*

Let the marked set be ***Sm***, the unmarked set be ***Su*** and the temporary group ***Gt*** be the group ***Gi***.

*Step 2*

Classify the group ***Gt*** into several subgroups such that

(i) Each CCPE in ***Su*** and all of its intersected CCPEs, ***Gs***, in ***Sm*** are put in one subgroup.

(ii) Any pair of CCPEs in ***Su*** which their ***Gs*** are the same must be put into same subgroup. So that each CCPE in ***Su*** should belong to one and only one subgroup.

(From Fig. 12, there are six subgroups, such as $\{V_1, H_6\}$, $\{V_2, V_4, H_6, H_8\}$, $\{V_3, H_1, H_2, H_3, H_6, H_{10}, H_{11}\}$, $\{V_5, H_8, H_9\}$, $\{V_6, V_7, H_4, H_5, H_6\}$, and $\{V_8, V_9, V_{10}, H_4, H_5, H_6, H_7\}$.)

*Step 3*

For each subgroup, do the following statement:

If (| unmarked CCPEs | ≥ | marked CCPEs |) then:

(i) Delete the marked CCPEs from ***Gt*** and all of the subgroups.

(ii) Mark the homologous unmarked CCPEs in ***Gt***.

(iii) Remove this subgroup.

*Step 4*

Go to step 3 until there is no subgroup satisfying the condition described in step 3.

(Fig. 13 shows how the process is undergoing in steps 3 and 4 for the example illustrated in Fig. 12).

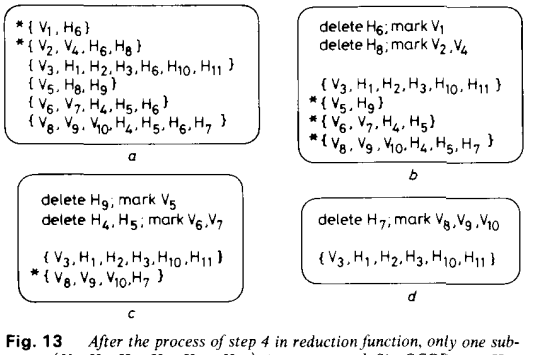



**Fig. 13** *After the process of step 4 in reduction function, only one subgroup, $\{V_3, H_1, H_2, H_3, H_{10}, H_{11}\}$, is not removed. Six CCPEs, viz. $H_6$, $H_8$, $H_9$, $H_4$, $H_5$, and $H_7$, are deleted in this step*

*Step 5*

Delete all the unmarked CCPEs in ***Gt***, the remaining CCPEs in ***Gt*** are disjointed now.

(For the group shown in Fig. 12, $V_3$ will be deleted. Fig. 14 shows the final result of Fig. 12 after the elimination of this algorithm).

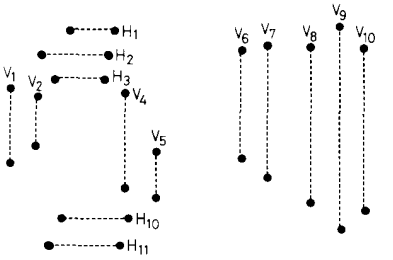



**Fig. 14** *After the process of the elimination algorithm, 14 disjointed CCPEs are reserved from the 21 connected CCPEs shown in Fig. 12. Note that the double corners have disappeared*

*Step 6*

Stop.

### *4.2 The OTP is superior to the corner stitching partition*

According to lemma 1, 2, and the elimination algorithm shown above, lemma 3 (below) is proposed to remove assumptions 2', 4, and 5 at the same time. Subsequently, theorem 2 will prove that for a nonoverlapped layout (assumption 1), our OTP (HOTP or VOTP) is indeed superior to the corner stitching partition (HCSP or VCSP).

*Lemma 3:* For a given layout with *Nsor* solid rectangles satisfying assumption 1, eqns. 10 and 11 must hold.

$$|OTP| = \{4 + (4 * Nsor) + [2 * (4 * Nsor) - (4 * Nccpe^*)]\}/4 \quad (10)$$

$$= (|Ebr| + |Esor| + |Erpe^{***}|)/4 \quad (11)$$

where $|Erpe^{***}| = 2 * [(4 * Nsor) - (2 * Nccpe^*)]$, and $Nccpe^*$ is the maximal number of disjointed CCPEs which are obtained by applying the elimination algorithm to all of the CCPEs in the given layout. ■

*Proof:* For an OTP, because each corner of solid rectangle needs one and only one reliable partition edge, hence one of the CCPEs of the double corner is redundant. It has to be deleted from the OTP. On the other hand, the crossover of any pair of CCPEs will increase or not change the number of space tiles. Each crossover also have to be avoided in the OTP. As a result, by applying the elimination algorithm to obtain the maximal value of $Nccpe^*$ from the original *Nccpe*, eqn. 8 has to be updated to eqn. 10. Thus the lemma QED

*Theorem 2:* For a given layout with a number of nonoverlapped solid rectangles, the number of the space tiles partitioned by the OTP is not larger than that partitioned by the corner stitching partition. Hence, we have

$$|OTP| \geq |HCSP| \quad (12)$$

and

$$|OTP| \leq |VCSP| \quad (13)$$

■

*Proof:* As mentioned before, the $|OTP|$ can be calculated from eqn. 10. For the same reason, the following two

equations hold for corner stitching partition

$$|HCSP| = \{4 + (4 * Nsor) + [2 * (4 * Nsor) - (4 * Nhccpe)]\}/4 \quad (14)$$

$$|VCSP| = \{4 + (4 * Nsor) + [2 * (4 * Nsor) - (4 * Nvccpe)]\}/4 \quad (15)$$

where $Nhccpe$ ($Nvccpe$) represents the total amount of the horizontal (vertical) CCPEs for the HCSP (VCSP).

Therefore, to prove eqns. 12 and 13 we need to prove that

$$Nccpe^* \geqslant \max\{Nhccpe, Nvccpe\} \quad (16)$$

To prove eqn. 16, let us consider the connectivity of the CCPEs:

(i) If all of the CCPEs are disjointed (no crossover and no double corner), then

$$Nccpe^* = Nccpe = Nhccpe + Nvccpe$$

Hence eqn. 16 is true.

(ii) If some of the CCPEs are jointed, according to the elimination algorithm, the number of CCPEs remaining in the OTP must be larger than or equal to $Nhccpe$ ($Nvccpe$). Hence eqn. 16 is true. Thus the theorem. QED

## 5 Concerning the overlapped rectangles

In lemma 3 and theorem 2, the given layout is limited to the nonoverlapped solid rectangles. For a number of overlapped solid rectangles in the 2-D plane, we should find them to get some value. Some items defined below will be useful in proving theorem 3 which is the final and most important theorem of this paper.

For a given layout with overlapped rectangles, we define that:

(i) *Nic:* the number of the inactive corners.

An inactive corner is a corner of solid rectangle such that from which there is no reliable partition edge shot out in the OTP. For example, the inactive corners in Fig. 15*a* are corners of *H*, *I*, *J*, *K*, and *L*.

(ii) *Nac:* the number of the active corners.

If the corner of solid rectangle is not an inactive corner, then it is called an active corner. Hence, in Fig. 15*a*, the corners of *A*, *B*, *C*, *D*, *E*, *F*, and *G* are active corners.

From the definition of *Nic* and *Nac*, if the given layout has no overlapped solid rectangle, then $Nic = 0$ and $Nac = 4 * Nsor$.

(iii) *Nisor:* the total number of intersected (overlapped) solid rectangles in the given layout.

(iv) *Noe:* the number of Manhattan edges located between the SPTs and the boundary of the overlapped solid rectangles. Such a Manhattan edge is also called an outer edge. In other words, an outer edge is the contour edge of the hole (containing in overlapped solid rectangles) and the overlapped rectilinear polygon. For example, in Fig. 15*a* and *b*, we have $Noe = 10$ and $Noe = 20$, respectively. The outer edges in Fig. 15*a* are *AB*, *BM*, *MC*, *CD*, *DN*, *NE*, *EF*, *FL*, *LG*, and *GA*.

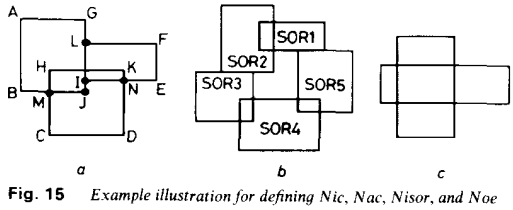



**Fig. 15** *Example illustration for defining Nic, Nac, Nisor, and Noe*

Similarly, from the definition of *Nisor* and *Noe*, if the given layout has no overlapped solid rectangle, then $Nisor = 0$ and $Noe = 0$. However, for some cases, *Noe* may be larger than $4 * Nisor$; for example, in Fig. 15*c*, we have $Noe\ (=12) > 4 * Nisor\ (=8)$.

*Theorem 3:* For a given layout with a number of overlapped solid rectangles, eqn. 17 must hold.

$$|OTP| = \{4 + \{[4 * (Nsor - Nisor)] + Noe\} + \{2 * [(4 * Nsor) - (2 * Nccpe^*) - Nic]\}\}/4 \quad (17)$$

$$= (|Ebr| + |Esor^*| + |Erpe^{****}|)/4 \quad (18)$$

where

$$|Esor^*| = 4 * (Nsor - Nisor) + Noe$$

and

$$|Erpe^{****}| = 2 * [(4 * Nsor) - (2 * Nccpe^*) - Nic] \quad \blacksquare$$

*Proof:* Consider eqn. 10 in lemma 3, if the given layout has no overlapped solid rectangles, then $Nisor = 0$, $Noe = 0$ and $Nic = 0$. Therefore, eqn. 17 can be degenerated into eqn. 10.

Consider eqn. 11 in lemma 3, if the given layout has *Nisor* overlapped solid rectangle, then the $|Espt|$ should decrease $4 * Nisor$; however, the generated outer edges will increase the $|Espt|$ by *Noe*. Therefore, the $|Espt|$ partitioned by edges of solid rectangle is equal to $4 * (Nsor - Nisor) + Noe$; that means

$$|Esor^*| = |Esor| - 4 * Nisor + Noe$$

$$= 4 * (Nsor - Nisor) + Noe$$

For the same reason, the $|Espt|$ partitioned by reliable partition edges will decrease $2 * Nic$; hence, we have

$$|Erpe^{****}| = |Erpe^{***}| - 2 * Nic$$

$$= 2 * [(4 * Nsor) - (2 * Nccpe^*) - Nic]$$

By combining eqns. 10 and 11 and the values of $|Esor^*|$ and $|Erpe^{****}|$ calculated above, eqns. 17 and 18 have been proven. QED

## 6 Plane-sweep OTP algorithm

Consider theorem 3, the elimination algorithm, and the plane-sweep technique described in Reference 19, we have developed the plane-sweep OTP algorithm (shown below). The input of this algorithm is a large number of overlapped or nonoverlapped rectangles represented by any of the VLSI quad trees [8, 9, 20]. After processing by this algorithm, the irregular space regions of the original layout are partitioned into a minimal number space rectangles so that the designed VLSI layout tools such as editor, router, and compactor, will be speeded up and require less memory storage.

*OTP algorithm:*

**Begin** Apply 'plane-sweep algorithm'; (see Reference 19)
  **for** each sweeping-line event **do**
  **Begin** Using the region query technique based on the quad tree data structure to:

Judge whether each solid rectangle is overlapped by the other solid rectangles or not;

(to obtain *Nisor*)

Judge whether each corner is covered by the other solid rectangles or not;

(to obtain *Nac*)

For each uncovered corner determine the possible CCPE;

(to obtain *Nccpe*)

**End**;

**if** (*Nccpe* $\neq$ 0) Apply elimination algorithm to get the maximum set of nonintersected CCPEs;

(to obtain *Ncpe*)

(The elimination algorithm eliminates redundant CCPEs and the remaining CCPEs will not intersect each other)

**if** (*Nisor* $\neq$ 0) Apply the plane-sweep algorithm and region query technique again to decide whether each edge is an outer edge or not;

(to obtain *Noe*)

Apply plane-sweep algorithm to the source layout and the remaining CCPEs, and then output the spacing rectangles by the optimal tile partition;

**End**.

## 7 Result and conclusion

This paper presents an optimal tile partition method which is superior to the partition of corner stitching. Table 1 illustrates some experimental results for the OTP and the corner stitching partition. Besides the theoretical discussion and their rigorous proofs, we also give a practical elimination algorithm and the OTP algorithm in detail. Furthermore, our OTP algorithm can be applied to a given layout with both nonoverlapped and overlapped rectangles. Another experiment result shown in Fig. 16 and the dotted area of Table 1 took 15.32 seconds to obtain our $|OTP| = 270$, which is less than both the original $|HCSP| = 319$ and $|VCSP| = 326$. In this example, the presented OTP has improved the finding-speed and the memory efficiency by about 16.28% with respect to those of the corner stitching data structure. The source code was written in *C*-language under MS DOS and ran on the IBM/386 compatible microcomputer.

**Table 1: Experimental results based on IBM/386PC for the OTP and the corner stitching partition**

| No. of rectangles | Experimental results | | | | |
|---|---|---|---|---|---|
| | \|OTP\| = \|HOTP\| = \|VOTP\| | \|HCSP\| | \|VCSP\| | Runtime for OTP, s. | Improvement of speed and memory |
| 17 | 31 | 36 | 37 | 1.32 | 15.07% |
| 52 | 97 | 107 | 111 | 3.02 | 10.98% |
| 92 | 169 | 186 | 208 | 4.45 | 14.21% |
| 150 (Fig. 16) | 270 | 319 | 326 | 15.32 | 16.28% |
| 200 | 320 | 363 | 410 | 23.63 | 17.21% |

The technique of the OTP not only can be used to improve corner stitching but also can be extended to promote the region query operations for the space rectangles represented by quad trees [8–10] and dynamic bucket data structure [11]. With the assistance of the OTP, the corner stitching and the quad trees are able to be applied to solve the problems of layout editing [22], routing, plowing and compaction [6, 7, 12, 13] more efficiently.

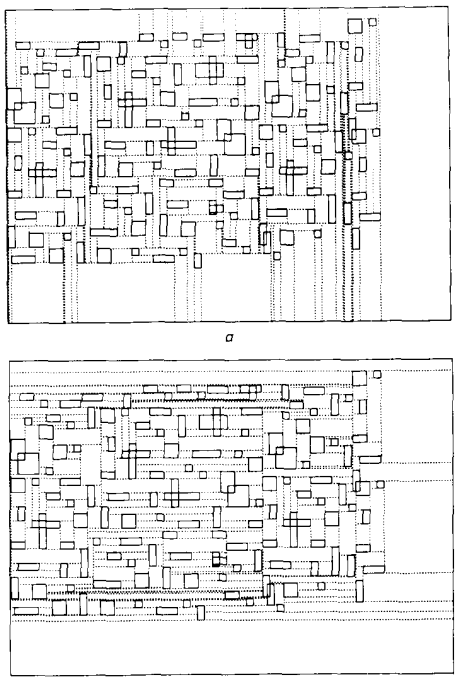


**Fig. 16** *This tested random layout took 15.32 s to obtain our* $|OTP| = |HOTP| = |VOTP| = 270$, *which is less than both of the original* $|HCSP| = 319$ *and* $|VCSP| = 326$

*a* Nsor = 150, Ncpe = 179, |VOTP| = 278
*b* Nsor = 150, Ncpe = 179, |HOTP| = 278

## 8 References


1 OUSTERHOUT, J.K.: 'Corner stitching: a data-structuring technique for VLSI layout tools', *IEEE Trans. Computer-Aided Design*, 1984, **CAD-3**, pp. 87–100
2 SAMET, H.: 'The design and analysis of spatial data structures' (Addison-Wesley, New York, 1990)
3 SAMET, H.: 'Applications of spatial data structures' (Addison-Wesley, New York, 1990)
4 ROSENBERG, J.B.: 'Geographical data structures compared: a study of data structures supporting region queries', *IEEE Trans. Computer-Aided Design*, 1985, **CAD-4**, pp. 53–67
5 OUSTERHOUT, J.K., HAMACHI, G.T., MAYO, R.N., SCOTT, W.S., and TAYLOR, G.S.: 'Magic: A VLSI layout system'. Berkeley EECS Rep. UCS/CSD 83/154, Dec. 1983
6 MARGARINO, A., ROMANO, A., DE GLORIA, A., CURATELLI, F., and ANTOGNETTI, P.: 'A tile-expansion router', *IEEE Trans. Computer-Aided Design*, 1987, **CAD-6**, pp. 507–517
7 SCOTT, W., and OUSTERHOUT, J.K.: 'Plowing: interactive stretching and compaction in Magic'. Proc. 21st IEEE Design Automation Conference, June 1984, pp. 180–187
8 BROWN, R.L.: 'Multiple storage quad trees: A simpler faster alternative to bisector list quad trees', *IEEE Trans. Computer-Aided Design*, 1986, **CAD-5**, pp. 413–419
9 WEYTEN, L., and DE PAUW, W.: 'Quad-list quad tree: a geometrical data structure with improved performance for large region queries', *IEEE Trans. Computer-Aided Design*, 1989, **CAD-8**, pp. 229–233
10 PITAKSANONKUL, A., THANAWASTIEN, S., and LURSINSAP, C.: 'Bisection trees and half-quad trees: memory and time efficient data structures for VLSI layout editors', *INTEGRATION, the VLSI J.*, 1989, No. 8, pp. 285–300
11 KUO, Y.S., HWANG, S.Y., and HU, H.F.: 'A data structure for fast region searches', *IEEE Design & Test Comput.*, October 1989, pp. 20–28

12 HSIAO, P.Y., and FENG, W.S.: 'An edge-oriended compaction scheme based on multiple storage quad tree', *IEEE Proc. ISCAS*, June 1988, pp. 2435–2438
13 HSIAO, P.Y., and FENG, W.S.: 'Using multiple storage quad tree on a hierarchical VLSI compaction scheme', *IEEE Trans. Computer-Aided Design*, 1990, **CAD-9,** pp. 522–536
14 GOURLEY, K.D., and GREEN, D.M.: 'A polygon-to-rectangle conversion algorithm', *IEEE Comput. Graph. Appl.*, 1983, **3,** pp. 31–36
15 FERRARI, L., SANKAR, P.V., and SKLANSKY, J.: 'Minimal rectangular partition of digitized blobs', *Comput. Vision, Graph. Image Proc.*, 1984, **28,** pp. 58–71
16 IMAI, H., and ASANO, T.: 'Efficient algorithms for geometric graph search problems', *SIAM J. Comput.*, 1986, **15,** pp. 478–494
17 LIOU, W.T., TAN, J.J.M., and LEE, R.C.T.: 'Minimum rectangular partition problem for simple rectilinear polygons', *IEEE Trans. Computer-Aided Design*, 1990, **9,** (7), pp. 720–733
18 TSAI, C.C., CHEN, S.J., HSIAO, P.Y., and FENG, W.S.: 'A new iteration construction approach to routing with compacted area', *IEE Proc. E, Computers and Digital Techniques*, 1991, **138,** (1), pp. 57–71
19 HSIAO, P.Y., and TSAI, C.C.: 'A new plane-sweep algorithm based on spatial data structure for overlapped rectangles in 2-D plane'. IEEE 14th Int. Computer Software & Applications Conference, 1990, pp. 347–352
20 HSIAO, P.Y., and JANG, L.D.: 'On VLSI layout systems' spatial data structures: the binary balanced quad list quad trees'. Submitted to *IEEE Trans. Computer-Aided Design*
21 HSIAO, P.Y., and LIN, CHIAO-YI.: 'Minimum partition for the space region of VLSI layout'. Accepted for publication in the 5th International Conference on VLSI Design, Bangalore, India., Jan. 1992
22 HSIAO, P.Y., and YAN, J.T.: 'PC_UNION: a low cost layout tool for consumer IC design'. Int. Symp. on IC Design, Manufacture and Applications Conference, Singapore, Sep. 1991, pp. 452–457