Abstract

An optimal tile partition (OTP) is presented for partitioning the space region of a VLSI layout plane into rectangular spacing tiles (SPTs). It modifies the corner stitching (CST) data structure to optimize the spacing tile partition. We are known there is a serious restriction for the original CST data structure: the solid rectangles can not be overlapped with each other, but our OTP does not limit to it. This paper gives theoretical discussion with respect to some experimental results to obtain the minimal number of the SPTs through the OTP. Moreover, a dynamic plane-sweep algorithm based on region query for the OTP has been developed. By using the OTP, the memory efficiency and the local query operations of the original CST data structure has been enhanced.

1. Introduction

For representing a number of solid rectangles (SORs) in two-dimensional VLSI layout plane, many efficient spatial data structures \[2\]-\[4\] have been presented to achieve the design of various layout tools. Corner Stitching (CST)\[1\] is one of the most famous data structures for designing layout editor \[5\], router \[6\], compactor and plower \[7\]. It provides a variety of powerfully local region query operations, such as neighbor-finding, point-finding, area-searches, directed area enumeration and channel finding. Corner stitching has originally incorporated into the Magic VLSI system \[5\] and is responsible for Magic's success.

The primary idea of corner stitching is to divide the layout plane, which consists of various nonoverlapped SORs, into spacing tiles (SPTs) by using horizontal partition edges (HPEs)\[1\] or by using vertical partition edges (VPEs), respectively.

For example, a source layout with five nonoverlapped SORs shown in Fig.1 can be represented by CST through horizontal partition (HCSP), or vertical partition (VCSP) as illustrated in Fig.2 and Fig.3, respectively. Since the number of SPTs partitioned from HCSP or VCSP (IHCSPI = 13, IVCSPI = 14) is not a minimal value, neither HCSP nor VCSP is an optimal tile partition (OTP). An OTP should partition the layout into a minimum number of SPTs. Fig.4 shows that the number of SPTs from the horizontal oriented OTP (IHOTP) equals 11 which is less than IHCSPI and IVCSPI. Similarly, Fig.5 gives the number of SPTs from the vertical oriented OTP (IVOTP) has to be also equal to 11. The critical partition edges (CPEs) appeared in both HOTP and VOTP are illustrated in Fig.6. In this paper, we present the general rules for creating OTP (HOTP and VOTP) on the basis of theoretic discussion and restricted proofs in detail.

As we have known, the partition of CST is one of the most famous partition methods for partitioning the layout into neighored SPTs. The CST \[1\] describes the VLSI layout as a set of nonoverlapped rectangles consisting separated SORs and neighored SPTs. Each SPT is a horizontal strip whose right edge or left edge never touch the other SPTs. Moreover, for a given layout, the set of partitioned SPTs must be unique.

For the CST data structure described in Ref.\[1\], the number of SPTs may not beyond \(3N+1\) for a given layout with \(N\) nonoverlapped SORs. In the worst case, the memory requirement for CST is three times more than those for linked lists. To obtain a minimum number of SPTs by the presented OTP means to promote the memory efficiency and the local search speed of the CST.
2. Primitive Definitions and Modeling

Before presenting the OTP, we assume the layout plane is large enough so that the SORs are wholly included inside the layout plane. In other words, the edges of SORs are never located outside or exactly at the boundary of the layout plane.

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

(ii) The end points of the line segment must locate exactly at the boundary rectangle (BR) of the layout plane, the edge of the SOR, the corner of the SOR, or the interior point of the other partition edge.

**Definition 2: Reliable Partition Edge (RPE)**
A partition edge satisfying the condition that at least one of its end points should locate exactly at the corner of the SOR is defined as a reliable partition edge.

**Definition 3: Reliable Tile Partition (RTP)**
For a given layout, the reliable tile partition as shown in Fig.8 is a partition which divides the given layout by RPEs and must satisfy the condition that each corner of the SOR should intersect with at least one end point of the RPE except that those corners overlapped with the other SOR should not intersect with any end point of the RPE.

Note that for a given layout, there may have many different kinds of RTPs. From Definition 3, it is obvious that the partition of CST is one kind of RTPs. In other words, HCSP belongs to RTP and VCSP also belongs to RTP. Consequently, we conclude that all the partitions shown in Fig.4.5 and Fig.8 are RTPs. From this point of view, there must have some RTPs which possess a minimum number of partitioned spacing tiles (SPTs). Those RTPs with minimum number of SPTs are called as 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), each of them must be also unique.

**Definition 4: Horizontal Oriented Optimal Tile Partition (HOTP)**
A reliable tile partition which has a minimum number of partitioned spacing tiles (SPTs) and satisfies the condition that to change any one of its vertical RPEs into horizontal RPE reserved with the same end point intersected with the corner of the SOR must be to cause an increase of the number of SPTs (ISPTsl) is defined as the horizontal oriented optimal tile partition (refer to Fig.4).

Note that most of the RPEs of the HOTP are horizontal. If any one of the horizontal RPEs is changed into a vertical one, the ISPTsl must be the same. Moreover, both of the original horizontal RPE and the updated new vertical RPE should have one same end point intersected with the corner of the SOR.

**Definition 5: Vertical Oriented Optimal Tile Partition (VOTP)**
A RTP which has a minimum number of SPTs and satisfies the condition that to change any one of its horizontal RPEs into vertical RPE reserved with the same end point intersected with the corner of the SOR must be to cause an increment of ISPTsl is defined as the VOTP (refer to Fig.5).

**Definition 6: Critical Partition Edge (CPE)**
For a given layout, the vertical RPEs appeared in the HOTP and the horizontal RPEs appeared in the VOTP are defined as critical partition edges (refer to Fig.4-6).

It is obvious that the CPEs play as a key role in finding the general partition rules of the HOTP and the VOTP. However, before describing how to obtain the CPEs step by step, we know that the minimal ISPTsl can be calculated from the minimal total number of edges of SPTs (IEspt1). That means the minimal ISPTsl equals the minimal IEsptl divided by four. As a result, to solve the problem of the HOTP and the VOTP, we should focus our argument on the number of the edges of SPTs (IEsptl) instead of directly counting the ISPTsl. Thus our Theorem 1 gives a classification of the Esptl. It is greatly helpful for the discussion in this paper.

**Theorem 1.**
For a given layout with a number of overlapped SORs, \( L = \{ Ri \mid i = 1, 2, ..., Nsr \} \), the minimal number of the edges of the spacing tiles (IEsptmin) can be calculated from the number of the boundary edges of the whole layout (IEbrl), the number of the edges of the solid rectangles (IEsr), and the number of reliable partition edges for the HOTP or the VOTP (IErепотл) as follows:

\[
IEsptmin = \frac{IEbrl}{4},
\]

\[
IEsptmin = f(IEbrl, IEsr, IErепотл).
\]

---

Fig.5 The vertical oriented optimal tile partition, VOTP.

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

Fig.7 Each dashed line segment representing a partition edge.

Fig.8 Each dashed line segment representing a reliable partition edge. The whole partition in the figure shows an example of RTP.
Theorem 2
For a given layout with a number of nonoverlapped SORs, the number of the SPTs partitioned by the OTP is not larger than that partitioned by the CSP. Hence, we have
\[ IOTP \leq ICSCP, \quad IOTP \leq IVSCP. \]

3. Concerning the Overlapped Rectangles
In Theorem 2, the given layout is limited to the nonoverlapped SORs. For a number of overlapped SORs in the 2D 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:
1. Nic: the number of the inactive corners (ICs).
   An inactive corner is a corner of SOR such that from which there is no RPE shot out in the OTP. For example, the ICs in Fig.9(a) are corners of H, I, J, K, and L.
2. Nac: the number of the active corners (ACs).
   If the corner of SOR is not an IC, then it is called an active corner. Hence, in Fig.9(a), the corners of A, B, C, D, E, F, and G are ACs.
   From the definition of Nic and Nac, if the given layout has no overlapped SOR, then Nic = 0 and Nac = 4 * Nisor.
3. Nisor: the total number of intersected (overlapped) SORs in the given layout.
4. Noe: the number of Manhattan edges located between the SPTs and the boundary of the overlapped SORs. Such a Manhattan edge is also called an outer edge (OE). In other words, an OE is the contour edge of the hole (containing in overlapped SORs) and the overlapped rectilinear polygon. For example, in Fig.9(a) and (b), we have Noe = 10 and Noe = 20, respectively. The OEs in Fig.9(a) are AB, BM, MC, CD, DN, NE, EF, FL, LG, and GA.

Similarly, from the definition of Nisor and Noe, if the given layout has no overlapped SOR, then Nisor = 0 and Noe = 0.
However, for some cases, Noe may be larger than 4 * Nisor; for example, in Fig.9(c), we have Noe (=12) > 4 * Nisor (=8).

4. The Plane-sweep OTP Algorithm
By considering Theorem 3 and the plane-sweep technique described in [14], we have developed the Plane-sweep OTP algorithm shown as follow. The input of this algorithm is a large number of overlapped or nonoverlapped rectangles represented by any of the VLSI quad trees [8-9]. After processing of this algorithm, the space irregular regions of the original layout are partitioned into a minimum number of space rectangles so that the designed VLSI layout tools such as editor, router, and compactor, will be speeded up and require less memory storage.

The OTP Algorithm
Begin Apply The Plane-Sweep Algorithm;

for each Sweeping-line event do
Begin Apply region query technique to quad tree data structure;
Judge each SOR overlapped by the other SORs or not; /* we can obtain Nisor */
Judge each corner covered by the other SORs or not; /* we can obtain Nac */
From each uncovered corner to make CCPE possibly; /* we can obtain Noe */
End;

if (Nisor = 0) Apply The Elimination Algorithm to get the maximum set of nonintersected CCPEs;
/* we can obtain Nce */

/* The Elimination Algorithm eliminates redundant CCPEs and the remaining CCPEs are not intersected */
if (Nisor = 0) Apply Plane-Sweep Algorithm and region query technique again, to decide each edge is outer edge or not;
/* we can obtain Nac */
Apply The Plane-Sweep Algorithm to the source layout and the remaining CCPEs, and then output the spacing rectangle by the optimal tile partition;

End.

5. Result and Conclusion
This paper presents an optimal tile partition method which is prior to the partition of corner stitching. Besides the theoretic discussion, we also give a practical OTP algorithm. Furthermore, our OTP algorithm can be applied to a given layout with both nonoverlapped and overlapped rectangles. The experiment result shown in Fig.10 takes 3.02 seconds to obtain our IOTP = 97, which is less than both of the original...
IHCSPI = 107 and IVCSPI = 111. In this example, the presented OTP has improved the finding-speed and the memory efficiency about 10.98% with respect to those of the CST data structure. The source code was written in C language under UNIX O.S. and run on the SUN4/IPC workstation.

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 spacing rectangles represented by quad trees [8]-[10],[15] and dynamic bucket data structure [11]. With the assistance of the OTP, the CST and the quad trees are able to be applied to solve the problems of layout editing, routing, plowing and compaction [6]-[7],[12]-[13] more efficiently.

Acknowledgements
This work was supported by the National Science Council, Taipei, Taiwan, Republic of China, under Contract NSC 80-0404-E009-73.

References

Fig.10 This tested random layout takes 3.02 seconds to obtain our \( |OTP| = |HOTP| = |VOTP| = 97 \), which is less than both of the original \( |HCSP| = 107 \) and \( |VCSP| = 111 \).