Efficient Polygon Enclosure Algorithms for Device Extraction from VLSI Layouts (Extended Abstract)

Vamsi Krishna Kundeti* Prosenjit Gupta*

Abstract

We consider some problems related to VLSI Layout Analysis and Verification and model them as problems of reporting enclosures of polygons. Algorithms are provided using techniques of computational geometry, which solve these problems in optimal time and space.

1 Introduction

1.1 VLSI Layout Analysis and Verificaton

The typical description of a VLSI layout is the geometrical description of masks. Layout verification [4] is the testing of a layout to see if it satisfies design and layout rules. An important problem in layout verification is layout device extraction, which involves detection of capacitors, resistors, transistors etc. from the geometrical description of masks. The layout extraction process starts with a circuit layout and based on user-defined relationship information and it proceeds to detect electrical components from the geometrical relationships of the layout. In this paper we consider the extraction of BJT (bipolar junction transistor) devices.

1.2 Bipolar Junction Transistors

BJT devices have typically have three terminals, collector, base and emitter. BJTs may be of type npn or pnp where n/p indicates the type of the collector, base or emitter. An npn transistor is formed by forming a p-base in an n-well and an n-diffusion in the p-base. There are four kinds of BJT configurations: (a) vertical, (b) lateral, (c) isoplanar and (d) merged. The fabrication process of BJTs is described in [3]. In the extraction phase, we look at a cross-sectional top view in the layout to identify the devices. For fabricating a vertical npn BJT device, we start with a p-type substrate followed by three diffusion steps: n-type collector diffusion, followed by p-type base diffusion, ending with n-type emitter diffusion. In the cross-sectional view, n-type collector polygons enclose p-type base polygons and latter enclose n-type emitter polygons. When pnp transistors are required in an otherwise npn circuit, instead of constructing the device in the n-type island by three diffusion steps, the pnp transistor is built laterally. In the cross-section view of *lateral* pnp transistors, n-type base polygons enclose p-type emitter polygons and p-type collector polygons. For isoplanar devices, local oxidation is used to isolate devices as well as base and collector contacts. As a result, in the cross-sectional view, n-type collector and p-type base polygons are enclosed by the oxide layer polygon and n-type emitter polygons are enclosed by the base polygons. The merged transistor logic (MTL) or the integratred injection logic (I^2L) type of bipolar device structure has been used in low-power applications. In such a device, a lateral pnp device is merged with a npn device into a integrated unit. Thus in the cross-sectional view, an n-type polygon which serves as the base of the lateral pnp device and the emitter of the (inverted) npn device, encloses a p-type emitter of the pnp device and a p-type polygon which serves as the collector for the pnp device and a base for the npn device. The base polygon for the "inverted" npn device in turn encloses an n-type collector polygon of the same.

1.3 Results and Contributions

In this paper we consider various problems involving enclosures of polygons related to the layout analysis problems discussed in Section 1.2 above. The polygons come in layers, where each layer is a disjoint set of polygons. The assumption that polygons in a single layer do not intersect one another is practical since typically such a layer would correspond to a single material in the VLSI mask and overlapping polygons would be merged. For all the problems considered, we obtain worst-case optimal algorithms which use $O(n)$ space and run in $O(n \log n + k)$ time, where k is the output size.

Traditionally polygonal operations on VLSI layouts are computed as expensive boolean mask operations [2]. Since n input segments can generate $O(n^2)$ intersections in the worst case, such an approach would result in algorithms of complexity $O(n^2)$ or higher. We use variations of the Bentley-Ottmann algorithm for reporting pairwise intersections of segments [1], which reports k pairwise intersections amongst n segments in time $O(n \log n + k \log n)$. Two polygons may intersect in several edges but to know whether they intersect, we need not compute all such intersections. This is where our approach is superior to the boolean mask approach. We avoid detection of all intersections by careful deletions from the dynamic data structures used by the Bentley-Ottmann algorithm to maintain the sweep line status and the event point schedule, which help us in solving the problems optimally. All our algorithms work on simple polygons, without any

*International Institute of Information Technology, Gachibowli, Hyderabad, Andhra Pradesh 500 019, India. Email: {krishna_k@students.iiit.net,pgupta@iiit.net}.

assumptions about restricted orientations of polygon edges, number of sides in each polygon or number of polygons.

2 Extraction Algorithms

2.1 Vertical BJTs

We consider the following problem:

Problem 2.1 *Given three sets of simple polygons, C, B and E, such that no two polygons in the same set intersect, report triplets (c, b, e), $c \in C$, $b \in B$, $e \in E$ such that*
(i) c encloses b and
(ii) b encloses e.

To solve Problem 2.1, we first try to solve a simpler problem:

Problem 2.2 *Given a set of red polygons RP and a set of blue polygons BP, such that no two polygons in the same set intersect, report pairs (r, b), $r \in RP$, $b \in BP$, such that r encloses b.*

Lemma 1 *Given a set of red polygons RP and a set of blue polygons BP, such that no two polygons in the same set intersect, a blue polygon b can be enclosed by at most one red polygon r.*

Lemma 2 *Given a set of red polygons RP and a set of blue polygons BP, such that no two polygons in the same set intersect, if there exists a pair of polygons $r \in RP$ and $b \in BP$ such that r and b intersect, then b cannot be enclosed by any red polygon.*

To solve Problem 2.2, we use a plane-sweep algorithm. We sort the endpoints of the segments by nondecreasing x-coordinates, ties broken arbitrarily. We maintain these points in a balanced binary search tree T_x. This gives us the event-point schedule for the plane sweep. We sweep from left to right maintaining the segments intersected by the sweep line in another balanced binary search tree T_y. We process endpoints in the order defined by T_x. We assume that the red (respectively blue) polygons are numbered from 1 to $|RP|$ (respectively $|BP|$). We maintain an array $ENCLOSED[1 \ldots |BP|]$, initialized to all zeros. Our objective is to assign $ENCLOSED[b]$ to r if the b^{th} blue polygon is enclosed by the r^{th} red polygon and to -1 if it is not enclosed by any red polygon.

If the current endpoint p in the event-point schedule is a left endpoint of some segment s, we find the segments s_u and s_ℓ in T_y above and below s in the ordering defined by T_y and insert s into T_y. If s intersects s_u or s_ℓ properly (i.e. not just at endpoints), the two intersecting segments must come from polygons of different colors, say a red one r and a blue one b. Then by Lemma 2, the polygon b cannot be enclosed by any red polygon. Hence we delete any segments and endpoints of b from both T_x and T_y. We also set $ENCLOSED[b]$ to -1.

If the current endpoint p in the event-point schedule is a right endpoint of some segment s, we find the segments s_u and s_ℓ in T_y above and below s in the ordering defined by T_y and delete s from T_y. If s_u and s_ℓ properly intersect to the right of p, the two intersecting segments must come from polygons of different colors, say a red one r and a blue one b. We again consider the blue polygon b to which the blue intersecting segment belongs, delete any segments and endpoints of b from both T_x and T_y and also set $ENCLOSED[b]$ to -1.

Next we preprocess the polygons in RP into a data structure for planar point location and query with an endpoint for each polygon $b \in BP$ such that $ENCLOSED[b] = 0$ to determine the red polygon if any that encloses b.

Theorem 3 *Given a set of red polygons RP and a set of blue polygons BP, with a total of n vertices such that no two polygons in the same set intersect, all pairs (r, b), $r \in RP$, $b \in BP$, such that r encloses b can be reported in time $O(n \log n)$ using $O(n)$ space.*

To solve Problem 2.1, we apply Theorem 3 twice: first to the sets B and E and then to sets C and B. For any $e \in E$, there is at most one base polygon $b \in B$ that encloses it. Once we know b, we can try to find the at most one collector polygon $c \in C$ that encloses it. The triples (c, b, e) can be reported in total time $O(n \log n + k)$ using the arrays $ENCLOSED$. Since k is $O(n)$, the overall time is $O(n \log n)$. We conclude:

Theorem 4 *Given three sets of simple polygons, C, B and E, with a total of n vertices, such that no two polygons in the same set intersect, the triples (c, b, e), $c \in C$, $b \in B$, $e \in E$ such that*
(i) c encloses b and
(ii) b encloses e,
can be reported in time $O(n \log n)$ using $O(n)$ space.

2.2 Lateral BJTs

We consider the following problem:

Problem 2.3 *Given three sets of simple polygons, C, B and E, such that no two polygons in the same set intersect, report triplets (c, b, e), $c \in C$, $b \in B$, $e \in E$ where:*
(i) b encloses c and e and
(ii) c (respectively e) does not intersect polygons from E (respectively C).

First, we consider a special case of Problem 2.3 when $C = E$. Here collector and emitter polygons come from the same p-type layer.

Theorem 5 *Given two sets of simple polygons, N and P, with a total of n vertices, such that no two polygons in the same set intersect, the k triples (c, b, e), $c \in P$, $b \in N$, $e \in P$ where*

(i) c and e are disjoint and
(ii) b encloses c and e,
can be reported in time $O(n \log n + k)$ using $O(n)$ space.

When the collector and emitter polygons belong to different layers, we are no longer guaranteed that a collector (respectively emitter) polygon does not intersect any emitter (respectively collector) polygon.

To solve Problem 2.3, we first need to eliminate from consideration those collector (respectively emitter) polygons which intersect emitter (respectively collector) polygons.

Problem 2.4 *Given a set of red polygons RP and a set of blue polygons BP, such that no two polygons of the same set intersect, report all red (respectively blue) polygons which do not intersect blue (respectively red) polygons.*

To solve Problem 2.4, we use a plane-sweep algorithm. We sort the endpoints of the segments by nondecreasing x-coordinates, ties broken arbitrarily. We maintain these points in a balanced binary search tree T_x. This gives us the event-point schedule for the plane sweep. We assume that the red (respectively blue) polygons are numbered from 1 to $|RP|$ (respectively $|BP|$). We maintain two arrays $R\_INTERSECTS[1 \ldots |RP|]$ and $B\_INTERSECTS[1 \ldots |BP|]$ initialized to all zeros. Our objective is to assign $R\_INTERSECTS[r]$ (respectively $B\_INTERSECTS[b]$ to 1 if the r^{th} red polygon (respectively b^{th} blue polygon) intersects some blue (respectively red) polygon.

We sweep from left to right maintaining the segments intersected by the sweep line in another balanced binary search tree T_y. We process endpoints in the order defined by T_x. First we try to find each blue polygon which intersects one or more red polygons. If the current endpoint p in the event-point schedule is a left endpoint of some segment s, we find the segments s_u and s_ℓ in T_y above and below s in the ordering defined by T_y and insert s into T_y. If s intersects s_u or s_ℓ properly (i.e. not just at endpoints), the two intersecting segments must come from polygons of different colors, say a red one r and a blue one b. We delete any segments and endpoints of b from both T_x and T_y. We also set $B\_INTERSECTS[b]$ to 1. If the current endpoint p in the event-point schedule is a right endpoint of some segment s, we find the segments s_u and s_ℓ in T_y above and below s in the ordering defined by T_y and delete s from T_y. If s_u and s_ℓ properly intersect to the right of p, the two intersection segments must come from polygons of different colors, say a red one r and a blue one b. We again consider the blue polygon b to which the blue intersecting segment belongs and delete any segments and endpoints of b from both T_x and T_y. We also set $B\_INTERSECTS[b]$ to 1.

Next we try to find each blue polygon which intersects one or more red polygons. We repeat the sweep above with the change that whenever we detect any red-blue intersection between segments from a red polygon r and a blue polygon b, instead of updating $B\_INTERSECTS$ and and deleting segments and endpoints of b from the data structure, we set $R\_INTERSECTS[r]$ to 1 and delete any segments and endpoints of r from both T_x and T_y.

Theorem 6 *Given a set of red polygons RP and a set of blue polygons BP, with a total of n vertices, such that no two polygons of the same set intersect, all red polygons which do not intersect blue polygons and all blue polygons which do not intersect red polygons can be reported in $O(n)$ space and $O(n \log n)$ time.*

Theorem 7 *Given three sets of simple polygons, C, B and E, with a total of n vertices, such that no two polygons in the same set intersect, the k triples (c, b, e), $c \in C$, $b \in B$, $e \in E$ such that*
(i) b encloses c and
(ii) c (respectively e) does not intersect polygons from E (respectively C),
can be reported in $O(n)$ space and $O(n \log n + k)$ time.

2.3 Isoplanar BJTs

We consider the following problem:

Problem 2.5 *Given five sets of simple polygons, N, X, C, B and E, such that no two polygons in the same set intersect, report tuples (n, x, c, b, e), $n \in N$, $x \in X$, $c \in C$, $b \in B$, $e \in E$ such that*
(i) n encloses x,
(ii) x encloses and c and b,
(iii) b encloses e and
(iii) c (respectively b) does not intersect any polygon from B (respectively C).

From Theorem 3, we conclude:

Lemma 8 *Given two sets of polygons B and E, with a total of n vertices such that no two polygons in the same set intersect, all pairs (b, e), $b \in B$, $e \in E$, such that b encloses e can be reported in time $O(n \log n)$ using $O(n)$ space.*

For each $b \in B$, we can define a list $\mathcal{L}(b)$ of all $e \in E$ such that b encloses e.

From Theorem 7, we conclude:

Lemma 9 *Given three sets of simple polygons, X, C and B, with a total of n vertices, such that no two polygons in the same set intersect, the k triples (x, c, b), $x \in X$, $c \in C$, $b \in B$ such that*
(i) x encloses c and b and
(ii) c (respectively b) does not intersect polygons from B (respectively C),
can be reported in $O(n)$ space and $O(n \log n + k)$ time.

Using Lemma 8 and Lemma 9 and the lists $\mathcal{L}(b)$, we can conclude:

Lemma 10 *Given four sets of simple polygons, X, C, B and E, with a total of n vertices, such that no two polygons in the same set intersect, the k tuples tuples (x, c, b, e), $x \in X$, $c \in C$, $b \in B$ and $e \in E$, where*
(i) x encloses c and b and
(ii) c (respectively b) does not intersect polygons from B (respectively C) and
(iii) b encloses e
can be reported in $O(n)$ space and $O(n \log n + k)$ time.

From Theorem 3, we conclude:

Lemma 11 *Given two sets of polygons N and X, with a total of n vertices such that no two polygons in the same set intersect, all pairs (n, x), $n \in N$, $x \in X$, such that n encloses x can be reported in time $O(n \log n)$ using $O(n)$ space.*

Applying Lemma 11, we can determine for each $x \in X$, the polygon $n \in N$ (if any) that encloses it. Applying Lemma 10, we conclude:

Theorem 12 *Given five sets of simple polygons, N, X, C, B and E, with a total of n vertices, such that no two polygons in the same set intersect, the k tuples (n, x, c, b, e), $n \in N$, $x \in X$, $c \in C$, $b \in B$, $e \in E$ such that*
(i) n encloses x,
(ii) x encloses and c and b,
(iii) b encloses e and
(iv) c (respectively b) does not intersect any polygon from B (respectively C)
can be reported in $O(n)$ space and $O(n \log n + k)$ time.

2.4 Merged BJTs

We consider the following problem:

Problem 2.6 *Given four sets of simple polygons, $N1$, $N2$, $P1$, $P2$, such that no two polygons in the same set intersect, report tuples $(n1, p1, p2, n2)$, $n1 \in N1$, $n2 \in N2$, $p1 \in P1$, $p2 \in P2$ such that*
(i) $n1$ encloses $p1$ and $p2$,
(ii) c (respectively b) does not intersect any polygon from B (respectively C) and
(iii) $p2$ encloses $n2$

Applying Lemma 10, from Section 2.3, we conclude:

Theorem 13 *Given four sets of simple polygons, $N1$, $N2$, $P1$, $P2$, with a total of n vertices, such that no two polygons in the same set intersect, report tuples $(n1, p1, p2, n2)$, $n1 \in N1$, $n2 \in N2$, $p1 \in P1$, $p2 \in P2$ such that*
(i) $n1$ encloses $p1$ and $p2$,
(ii) $p1$ (respectively $p2$) does not intersect any polygon from $P1$ (respectively $P2$) and
(iii) $p2$ encloses $n2$
can be reported in $O(n)$ space and $O(n \log n + k)$ time.

3 Conclusion

We have considered some variants of enclosure problems involving sets of simple polygons which arise in VLSI layout processing. We have given an uniform framework for solving these problems based on the line-sweep technique from Computational Geometry. The algorithms presented are all optimal with respect to space and time bounds. It will be interesting to extend these ideas to other problems in VLSI layout analysis and verification.

References

[1] J. L. Bentley and T. A. Ottmann. Algorithms for Reporting and Counting Geometric Intersections. *IEEE Transactions on Computers*, **6–28**, No. 9, pages 643–647, 1979.

[2] T. Li and S. Kang. Layout Extraction and Verification Methodology for CMOS I/O Circuits. In *Proc. ACM/IEEE Design Automation Conference*, pages 291–296, 1998.

[3] B. G. Streetman. Solid State Electronic Devices. *Prentice Hall*, 1995.

[4] T. G. Szymanski and C. J. van Wyk. Layout Analysis and Verification. In *Physical Design Automation of VLSI Systems*, B. Preas and M. Lorenzetti eds., Benjamin/Cummins, pages 347–407, 1988.