Critical Area Computation for Missing Material Defects in VLSI Circuits

Evanthia Papadopoulou
IBM TJ Watson Research Center, Yorktown Heights, NY 10598
evanthia@watson.ibm.com

ABSTRACT

We address the problem of computing critical area for missing material defects in a circuit layout. The extraction of critical area is the main computational problem in VLSI yield prediction. Missing material defects cause open circuits and are classified into *breaks* and *via-blocks*. Our approach is based on the L_∞ *medial axis* of polygons and the weighted L_∞ Voronoi diagram of segments. The critical area problem for both breaks and via-blocks is reduced to a weighted L_∞ Voronoi diagram of segments. This reduction results in a plane sweep algorithm to compute critical area in one pass. The time complexity is $O(n \log n)$ in the case of breaks and $O(n \log n + K)$ in the case of via-blocks, where n is the size of the input and K is bounded by the number of *interacting vias* (in practice K is small). The critical area computation assumes square defects and reflects all possible defect sizes following the $D(r) = r_0^2/r^3$ defect size distribution. The method is presented for rectilinear layouts.

1. INTRODUCTION

VLSI yield prediction is based on the concept of *critical area* which reflects the sensitivity of a design to spot defects occurring during the manufacturing process (see for example [2; 6; 7; 13; 16; 17; 18; 11]). Yield prediction is of growing importance in modern VLSI design due to the need to control the cost of manufacturing. Spot defects are caused by particles such as dust and other contaminants in materials and equipment. They are classified into "extra material" defects causing shorts between different conducting regions and "missing material" defects causing open circuits. This paper addresses the problem of computing critical area for missing material defects in a single layer. In combination with [11] for extra material defects, it provides a unifying approach to critical area extraction via Voronoi diagrams.
The critical area in one layer of a circuit layout C is defined as

$$A_c = \int_0^\infty A(r)D(r)dr$$

where $A(r)$ denotes the area in which the center of a defect of radius r must fall in order to cause a circuit failure and $D(r)$ is the density function of the defect size. The defect density function has been estimated as follows [2; 3; 17; 18]:

$$D(r) = \begin{cases} cr^q/r_0^{q+1}, & 0 \le r \le r_0 \\ cr_0^{p-1}/r^p, & r_0 \le r \le \infty \end{cases} \qquad (1)$$

where p, q are real numbers (typically $p = 3, q = 1$), $c = (q+1)(p-1)/(q+p)$, and r_0 is some minimum optically resolvable size. Using typical values for p, q, and c we derive the widely used defect size distribution $D(r) = r_0^2/r^3$. A circuit failure in this paper represents an open circuit. Defects of size r are modeled as squares of side $2r$ (i.e., squares of radius r). As discussed in [11], modeling defects as squares corresponds to computing critical area in the L_∞ metric[1] instead of the Euclidean geometry. In reality spot defects have any kind of shape thus, the square defect model is good enough for all practical purposes. The worst case bound of [11], $A_c^e \le A_c^\infty \le 2A_c^e$, where A_c^∞ and A_c^e denote the critical area for missing material defects in L_∞ and the Euclidean metric respectively, can be shown similarly.
Missing material defects cause open circuits by breaking intended connections. On a metal interconnect layer an open is created by a defect breaking the continuity of an interconnection or a contact plug; on a via or contact layer an open is a defect destroying a contact. Thus, we have two types of missing material defects: *breaks*, interfering with the continuity of an interconnect, and *via-blocks*, destroying contacts on via layers.
Existing methods of extracting critical area for opens can be summarized as follows: 1) Monte Carlo simulation [19]: Draw a large number of defects with their radii distributed according to $D(r)$, check for each defect if it causes an open. 2) Geometric methods: Compute the area of critical region $A(r)$ for several different values of r independently; use the results to approximate the total critical area. They are usually based on shape manipulation tools providing operations such as *shrink-shape-by-r* and *find-area* (see [8; 10]). The time complexity for each defect radius depends on the underlying shape manipulation algorithms. In [13], $A(r)$ is computed using a more efficient scan-line method. 3)Grid method [18]: A fine grid is assumed over the layout and the critical radius[2] for every grid point is computed. The run-

[1]The L_∞ distance between two points $p = (x_p, y_p)$ and $q = (x_q, y_q)$ is the maximum of the horizontal and the vertical distance between p and q i.e., $d(p,q) = \max\{|x_p - x_q|, |y_p - y_q|\}$.
[2]The critical radius at point t is the radius of the smallest

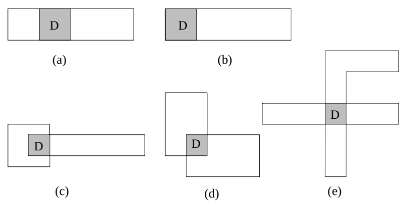

Figure 1: D is a minimal break.

time is $O(I^{1.5})$ time, where I is the number of grid points. 4) A more thorough analysis is given in [15]. $A(r)$, for a given defect radius r, is calculated strictly over each shape (critical regions expanding in the free space are ignored). This method (unlike the geometric ones) considers actual breaks of connectivity and runs in $O(n^2 \log n)$ time.

In this paper we present a geometric modeling of breaks and via-blocks and reduce the problem of computing critical area into variations of (weighted) L_∞ Voronoi diagrams. For a via-block we follow the definition of [9] which is more involved but more realistic than the one in [8]. Once the Voronoi diagrams are computed the total critical area for opens can be computed analytically as a function of Voronoi edges similarly to shorts [11]. The reduction of the critical area computation problem to Voronoi diagrams results in a plane sweep algorithm to compute critical area for opens in one pass. The time complexity is $O(n \log n)$ for the Voronoi diagram of breaks and $O(n \log n + K)$ for via-blocks, where n is the size of the input and K is bounded by the total number of *interacting vias*. (See section 5 for the definition). In practice K is small and should be negligible. In combination with [11] this algorithm forms the first low polynomial algorithm to compute critical area accurately in irregular layouts. Note that all previous methods (except the Monte Carlo simulation and the grid-based method) compute only the critical region $A(r)$ for a given defect size r. The method is presented for rectilinear layouts.

2. GEOMETRIC MODELING OF BREAKS AND VIA BLOCKS

Let's first consider a layer where missing material defects break the continuity of interconnections and contact plugs. A *simple* shape corresponds to a simple polygon and contains no holes. A shape with hole(s) is called *complex*.

For a simple shape, a defect D is a *minimal break*, if D breaks the shape into two or more pieces, and D has minimal size i.e., if D is shrunk by $\epsilon \geq 0$ then D will be entirely contained in the interior of the shape. A piece of a shape may trivially consist of a single edge. Figure 1 shows examples of defects considered to be minimal breaks. A minimal break is called *strictly minimal* if it contains no other minimal break in its interior. A *break* is any defect totally covering a strictly minimal break. For a complex shape, a break is additionally any defect overlaping the outer and inner boundary of the shape or any two distinct inner boundaries. Figure 2 shows examples of defects that are not considered to be breaks.

A limitation to the geometric modeling of breaks is that, by ignoring the actual connections, critical area may be overestimated in case of interconnect shapes that contain redundant or non-conducting regions. Here, similarly to existing

defect centered at t causing an open.

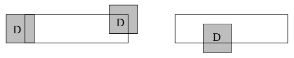

Figure 2: D is not a break.

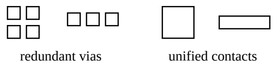

Figure 3: A contact as the minimum enclosing rectangle of redundant vias.

geometric methods, we ignore such redundant regions since their identification can be treated as a separate problem and be removed prior to critical area calculations.

Let's now consider a via or contact layer. Vias between different layers are typically realized by square shapes. To reduce the probability of missing contacts or to achieve a desired resistance designers often use *redundant vias*, a group of multiple vias that connect two shapes on different layers. Redundant vias are usually grouped together side by side and thus they can be regarded as a single via of larger size (see figure 3). Because of redundant vias, contacts can not be assumed to be squares but rectilinear shapes (often rectangles) of any size. A *via-block* (for brevity *block*) is a defect that completely destroys a contact i.e., a defect that completely covers a whole via or a group of redundant vias [9]. A square defect completely covers a rectilinear shape if and only if it totally covers its minimum enclosing rectangle. In this paper we assume that redundant vias have been identified in a preprocessing step and have been grouped together into atomic shapes represented by their minimum enclosing rectangle. (These operations are available in existing shape-processing tools e.g., [10]). In other words, we assume that a via layer has been preprocessed into a collection of disjoint rectangles of various sizes, referred to as *contacts*; A via-block is a defect totally covering a rectangular contact.

3. L_∞ VORONOI DIAGRAMS

The Voronoi diagram of a set of polygonal sites is a partitioning of the plane into regions, called *Voronoi cells*, such that the Voronoi cell of a site s is the locus of points closer to s than to any other site. The boundary that borders two Voronoi cells is called a *Voronoi edge*, and consists of portions of *bisectors* between the owners of the cells. The point where three or more Voronoi edges meet is called a *Voronoi vertex*. In the interior of a simple polygon P the Voronoi diagram is also called *medial axis*[3] ([4]). For more information on Voronoi diagrams see e.g. [14; 1].

The use of the L_∞ metric simplifies the Voronoi diagram of polygonal objects and makes it simple to compute in practice

[3]There is a minor difference in the definition which we ignore in this paper.

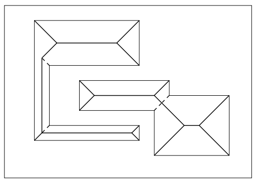
Figure 4: TheL_∞ medial axis of rectilinear polygons.

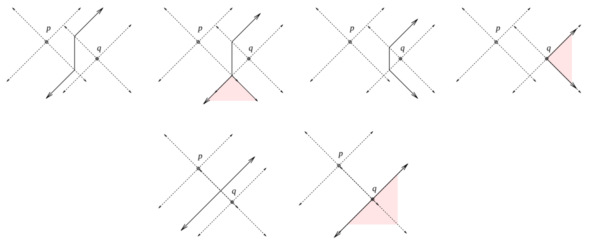

Figure 5: The L_∞ bisector of additively weighted points ($w_p < w_q \leq w_p + d(p,q)$).

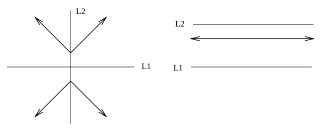

Figure 6: L_∞ bisector of weighted lines.

[12]. Intuitively, the L_∞ distance between two points p and q (denoted $d(p,q)$) is the side of the smallest square touching p and q. The L_∞ distance between a point p and a line l is $d(p,l) = \min\{d(p,q), \forall q \in l\}$. The L_∞ bisector of two polygonal elements (points or lines) is the locus of points at equal L_∞ distance from the two elements.
Let's now assume that points p and q are weighted with additive weights w_p and w_q respectively, such that $0 \leq w_p \leq w_q \leq w_p + d(p,q)$. The (weighted) L_∞ bisector of p, q is the locus of points equidistant from p and q in a weighted sense i.e., $b(p,q) = \{t \mid d(t,p) + w_p = d(t,q) + w_q\}$. Note that if $w_q > w_p + d(p,q)$ there is no bisector between p and q: all points in the plane (including q) are closer to p than q. If $w_q = w_p + d(p,q)$ then the area enclosed by the 45° rays[4] through q is equidistant from both points. Figure 5 shows L_∞ bisectors of additively weighted points as w_q increases ($w_p < w_q$). Without creating any significant difference, when a whole region is equidistant from both points (shaded regions in figure 5) we assign it to one of the points and consider only the outermost boundary of the bisecting region as the bisector (thick rays in figure 5). Thus, the $L\infty$ Voronoi diagram of additively weighted points is similar to the unweighted one. The main difference is that an arbitrarily weighted point may or may not have a Voronoi region.
The Voronoi diagram of additively weighted segments has not been given any attention in the literature (to the best of our knowledge). Figure 7 illustrates the L_∞ Voronoi diagram of arbitrarily weighted axis-parallel segments. The shaded regions depict the Voronoi cell(s) of the horizontal segment. As figure 7 illustrates, the Voronoi cell of a weighted segment need not be connected. Note that the Voronoi region(s) of an additively weighted segment may

[4]A 45° ray is a ray of slope +1 or −1.

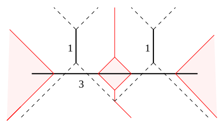

Figure 7: The L_∞ Voronoi diagram of weighted segments may contain disconnected regions.

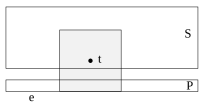

Figure 8: $r_c(t) = d(t,e)$

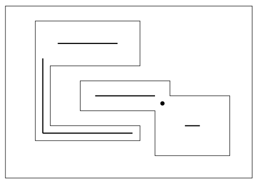
Figure 9: The core of simple shapes.

be induced by only a portion(s) of the segment or have no Voronoi region at all. The portion of a segment inducing a Voronoi region is called *active* and is clearly contained within that region. Voronoi edges consist of portions of bisectors between additively weighted lines and points. A weight along an axis parallel line corresponds to shifting the line in parallel according to the weight. The definition of a bisector remains the same : $B(l_1, l_2) = \{y \mid d(l_1, y) + w(l_1) = d(l_2, y) + w(l_2)\}$ where $w(l_1), w(l_2)$ denote the weights of two lines l_1, l_2. Figure 6 illustrates the bisector of two additively weighted orthogonal lines where $w(l_1) < w(l_2)$.

4. CRITICAL AREA FOR BREAKS

We have a layer in a circuit layout consisting of a collection of rectilinear polygons C. We assume that overlapping polygons have been unified into single shapes and thus all polygons are disjoint. The boundary of the layout is assumed to be a rectangle B. Our goal is to compute the critical area for breaks i.e., to evaluate the integral $A_c = \int_0^\infty A(r)D(r)dr$, where $D(r) = r_0^2/r^3$. Recall that $A(r)$ denotes the area of the *critical region* for square defects of radius r. The critical region for a defect D of radius r is the locus of points where if the center of D is placed it causes a break. The *critical radius* of a point t is the radius of the smallest defect centered at t causing a break. The defect inducing the critical radius of t is called the *critical break* for t. Note that the critical radius of a point over a shape S need not be determined by defects breaking edges of the same shape. In figure 8, the critical radius of $t \in S$ is determined by edge e in P.
Let P be a rectilinear shape (simple or complex). Consider the Voronoi diagram (medial axis) in the interior of P (figure 4). By the definition of the medial axis, any minimal break must be centered along the medial axis of P. Let the Voronoi vertices and Voronoi edges induced by parallel edges form a set of elements called the *core* of P, denoted as $core(P)$. In figure 9, $core(P)$ is shown thickened. Essentially, the core is a *generator* of breaks for shape P since it *generates* all strictly minimal breaks. A defect of radius $w(s)$ centered along $s \in core(P)$ is said to be *generated* by s.

LEMMA 1. *For any break B there is a core element $s \in core(P)$ such that the radius of B is $r \geq d(t,s) + w(s)$.*

The following notation is used throughout the paper: Given a rectangle R, the north, south, east and west edge of R are denoted as R^n, R^s, R^e, and R^w respectively. The north, south, east and west side of the core segment, $s = core(R)$

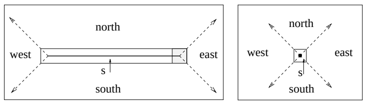

Figure 10: $r_c(t) = d(t,s) + w(s)$.

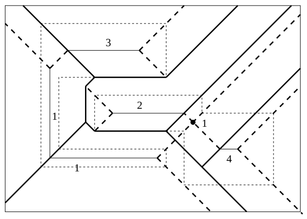

Figure 11: The weighted Voronoi diagram of core elements.

are denoted as s^n, s^s, s^e, s^w respectively. The same notation is used to also denote the y-coordinate of R^s, R^n, s^s, s^n and the x-coordinate of R^e, R^w, s^e, s^w.
Consider a single rectangle R such that $s = core(R)$ is horizontal (see figure 10). The four 45° rays emanating from s^e, s^w partition the plane into four quadrants. For any point t in the north (resp. south) quadrant the critical radius of t is (resp. R^n) i.e., $r_c(t) = d(t, R^s) = d(t,s) + w(s)$. For a point t in the west (resp. east) quadrant the critical break has to overlap with the minimal break centered at s^w (resp. s^e). Thus, $r_c(t) = d(t, s^w) + w(s^w)$ (resp. $r_c(t) = d(t, s^e) + w(s^e)$). In all cases, $r_c(t) = d(t,s) + w(s)$.
Let $G = \cup_P core(P), P \in C$, be the union of the core elements of all polygons on the given layer. The weighted L_∞ Voronoi diagram of G, $\mathcal{V}(G)$, provides a partitioning of the plane into regions where the critical radius within each region is easy to derive (see lemma 2). Figure 11 illustrates the weighted L_∞ Voronoi diagram of a set of core elements where the numbers indicate weights. The 45° dashed lines are bisectors between the endpoints and the open part of a segment.)

Lemma 2. *The critical radius for breaks of any point t in the Voronoi cell of a core element s, $s \in G$, is $r_c(t) = d(t,s) + w(s)$.*

Given $\mathcal{V}(G)$, the critical area integral can be discretized as a summation of Voronoi edges similarly to [11]. Core elements and the 45° rays emanating from their endpoints are considered to be part of $\mathcal{V}(G)$ and are treated as Voronoi edges. Voronoi edges are classified as red and blue. The classification can be summarized as follows: All core segments are colored red. A 45° ray emanating from an endpoint of a core element s is called *converging* if it forms an acute angle with the segment; otherwise it is called *diverging*. Given the Voronoi cell of a core element the incident diverging 45° Voronoi edges are colored red; the remaining Voronoi edges are colored blue. Boundary edges parallel to the owner are colored blue. Note that there may be 45° Voronoi edges that get different coloring with respect to the two adjacent cells. The contribution of these edges to critical area gets cancelled and thus, they are colored neutral and treated as not contributing to critical area. The result is as follows: (see [11] for the derivation)

Theorem 1. *Given the (weighted) L_∞ Voronoi diagram of all core elements of shapes in a layer C of a circuit layout,*

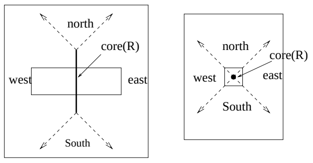

Figure 12: The core of a contact R.

and assuming that defects are squares following the "r_0^2/r^3" defect density distribution, the critical area for breaks in that layer is given by the following formula:
$A_c = r_0^2($

$$\sum_{red\ e} \frac{l}{r_c} - \sum_{blue\ e} \frac{l}{r_c} + \sum_{red\ e_{45}} \ln\frac{r_x}{r_n} - \sum_{blue\ e_{45}} \ln\frac{r_x}{r_n} - \frac{1}{2}\sum_{blue\ b} \frac{l_b}{r_b})$$

where l and r_c denote the length and the critical radius of an orthogonal Voronoi edge, r_x, r_n denote the maximum and the minimum critical radius of a 45° Voronoi edge, and l_b, r_b denote the length and the critical radius of a boundary edge. The first two summations are taken over all red and all blue orthogonal Voronoi edges respectively. The third and forth summations are taken over all red and all blue 45° Voronoi edges respectively. The last summation is taken over all blue boundary edges.

5. CRITICAL AREA FOR VIA-BLOCKS

A via layer, V, is assumed to have been preprocessed into a collection of rectangular contacts of various sizes. A via-block (for brevity block) is a defect (square) totally covering a contact. Clearly, the radius of any block for a contact R is at least $l/2$, where l is the length of R. A block of radius $l/2$ is referred to as a *minimal block*. The width of R is denoted by w ($l \geq w$). We define the *max-distance* of t from contact R to be the maximum L_∞ distance of t from any point of R i.e., $d_{max}(t,R) = \max\{d(t,y), \forall y \in R\}$. The *critical radius* of a point t is the radius of the smallest defect centered at t totally covering a contact. The critical radius of t with respect to a single contact R is $d_{max}(t,R)$. Thus, the critical radius for any point t is $r_c(t) = \min\{d_{max}(t,R), \forall R \in V\}$.
Let the core of a contact R, denoted $core(R)$, be the locus of centers of minimal blocks for R. $core(R)$ has opposite orientation than R i.e., if the longer side of R is horizontal (resp. vertical) $core(R)$ is vertical (resp. horizontal) (see figure 12). Note that if R is a square $core(R)$ is a single point. As in the case of breaks, $core(R)$ can be regarded as a generator of via-blocks for contact R. The segment $s = core(R)$ is weighted by $w(s) = l/2$.

Lemma 3. *The locus of centers of minimal blocks for contact R ($core(R)$) is an axis-parallel segment of length $l - w$ centered at the same point as R, where l, w denote the length and the width of R ($l \geq w$).*

Lemma 4. *The critical radius of t with respect to contact R is $r_c(t) = d_{max}(t,R) = d(t,s) + w(s)$, where $s = core(R)$.*

By lemma 4, the critical radius of any point on a via layer is $r_c(t) = \min\{d(t,s) + w(s), \forall s \in G = \cup core(R)\}$. Thus, the weighted Voronoi diagram of all core elements, $\mathcal{V}(G)$, gives a partitioning of the plane into regions where the critical radius is easy to compute. Due to the nature of the weights

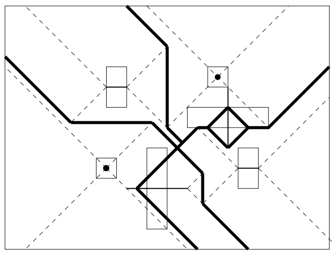

Figure 13: The L_∞ weighted Voronoi diagram for via-blocks.

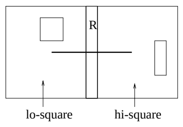

Figure 14: The core-rectangle of R containing interacting contacts.

$\mathcal{V}(G)$ is different than the case of breaks. First, a core segment need not have a Voronoi cell of its own e.g., the core of a large enough R. Furthermore, a core segment s having a Voronoi cell need not be totally contained within its cell. The portion of s contained within its Voronoi cell is called *active*. A contact having no Voronoi cell is called *useless*. As lemma 5 indicates, Voronoi cells remain connected having at most one Voronoi cell per core element. Figure 13 illustrates the L_∞ weighted Voronoi diagram of core elements of a set of contacts.

LEMMA 5. *The weighted Voronoi diagram of via core segments has at most one Voronoi cell per core segment. A core segment with a non-empty cell has exactly one active sub-segment enclosed within its cell.*

Let the *core-rectangle* of a contact R denote the rectangle obtained by two squares of side l centered at the endpoints of $core(R)$ (see figure 14). Let *lo-square* denote the leftmost (resp. bottommost) square and *hi-square* denote the rightmost (resp. topmost) square. The core-rectangle is the union of all minimal blocks for R. Note that $core(R)$ is part of the medial axis of the core-rectangle. If R is a square via then the core-rectangle is identical to R. A contact totally contained within the core-rectangle of R is said to be *interacting* with R. We have the following properties:

LEMMA 6. *A contact R is useless if and only if there are at least two interacting contacts within the core-rectangle of R, at opposite sides of R, within max-distance l from each other, where l is the length of R.*

LEMMA 7. *The core segment $s = core(R)$ is entirely active if and only if the core-rectangle of R contains no interacting contacts.*

The active portion of a core segment $s = core(R)$ can be determined by the interacting vias within the core-rectangle of R. Without loss of generality let's assume that s is horizontal. Let s_1, s_2 denote the endpoints of the active sub-segment where s_1 is to the left of s_2. As the following lemma shows, s_1 (resp. s_2) is determined by the rightmost R_j^w (resp. leftmost R_j^e) where R_j is a rectangle within *lo-square* (resp. *hi-square*) of R. Equivalently for a vertical core segment. Let $s_1.x, s_2.x$ denote the x-coordinates of s_1, s_2 respectively.

LEMMA 8. *Assuming that $s = core(R)$ is horizontal, $s_1.x = \max\{R_j^w \mid R_j \in lo-square(R)\} + w(s)$ and $s_2.x = \min\{R_j^e \mid R_j \in hi-square(R)\} - w(s)$.*

6. COMPUTING THE L_∞ VORONOI DIAGRAM OF CORE-SEGMENTS

A plane sweep algorithm to compute the L_∞ Voronoi diagram of rectangles and arbitrary line segments was given in [11; 12]. We briefly review, here, the main points of the algorithm. Imagine a sweepline sweeping the layout from left to right. Associated with a plane-sweep algorithm there are two major components: a *sweep-line status*, $\mathcal{T}$, maintaining the status of the sweeping process, and an *event list*, Q, containing the events where the sweep-line status changes, ordered in increasing *priority* value. Throughout the sweeping process, a partial Voronoi diagram of all segments to the left of the sweeping line, including the sweepline, is maintained. The *wavefront* is the collection of Voronoi edges (portions of bisectors) bounding the Voronoi cell of the sweepline. The bisectors incident to the wavefront are called *spike bisectors*. As the sweepline moves to the right the wavefront as well as the endpoints of spike bisectors also move to the right. The *sweep-line status* maintains the combinatorial structure of the wavefront. We have two types of events: *site events* and *spike events*. Site events are caused by the endpoints of segments and correspond to new waves entering the wavefront. Spike events correspond to potential Voronoi vertices and are caused by the intersection of two neighboring spike bisectors. The priority of a site event is given by its x-coordinate. The priority of a spike event s is $s.x + d(e, s)$ where e is the owner of s and $s.x$ denotes the abscissa of s. This corresponds to the rightmost abscissa of the square induced by the spike event i.e., the square defined by the tree neighboring sites defining the intersecting spike bisectors.
Computing the weighted Voronoi diagram of core elements for breaks requires only a minor modification to the algorithm described above: the weight of a core element s needs to be added to its priority i.e., the priority of a site event is $s.x + w(s)$ where $s.x$ is the x-coordinate of an endpoint of s. In other words, a core endpoint is not processed as soon as it is reached by the sweep-line but later when the sweep-line reaches position $s.x + w(s)$. The remaining part of the algorithm is identical to [11]. When bisectors are computed the weights of core elements are always added to the equations. Note that here the weights of core elements are such that $w(q) \leq w(p) + d(p, q)$ for any two core elements p, q. This implies that the invariance of the unweighted case holds also here: when a site event s is processed at time $s.x + w(s)$, s has not been covered by the *wavefront* yet. The reader is referred to [11] for the details of the algorithm. The time complexity is $O(n \log n)$ where n is the number of core-segments.
Computing the weighted Voronoi diagram of core segments for via-blocks requires to identify the endpoints of the active portions of core segments (referred to as *active endpoints*). For this purpose additional *active events* need to be generated. Given a core-segment $s = core(R)$ recall that s^n, s^s, s^e, s^w denote the north, south, east, and west side of s respectively. Recall also that the same notation is used

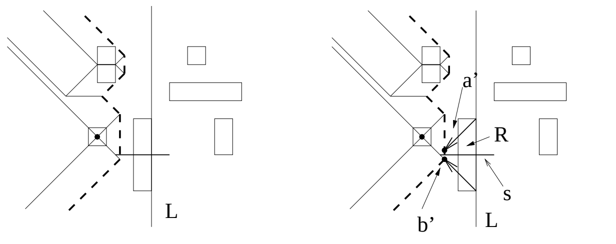

Figure 15: The wavefront at instance $priority(s^w) = s^w + w(s) = R^e$.

to denote the y-coordinate of s^s, s^n and the x-coordinate of s^e, s^w. Note that $s^i + w(s) = R^j$ for $i = s, w$ and $j = n, e$ respectively, and $s^i - w(s) = R^j$ for $i = n, e$ and $j = s, w$ respectively.

In summary we have three types of site events: *left*, *right*, and *active* events. A left event and a right event correspond to s^w and s^e respectively and have priority $priority(s^w) = s^w + w(s) = R^e$ and $priority(s^e) = s^e + w(s) = R^w + 2w(s)$. In other words, the priority of a left event is reached when the sweep-line reaches R^e and the priority of a right event is reached when the sweepline reaches the right edge of *hi-square*. An *active* event corresponds to the leftmost active endpoint, s_a, of a potential active portion of a horizontal core segment and has priority $s_a + w(s)$ (s_a denotes also the x-coordinate of the active endpoint).

Lemma 9. *For a vertical $s = core(R)$, the active subsegment of s (if any) is given by the intersection of s with the wavefront at instance $priority(s^w) = s^w + w(s) = R^e$.*

Lemma 10. *For a horizontal $s = core(R)$, the wavefront overlaps s at instance $priority(s^w) = s^w + w(s) = R^e$, if and only if the lo-square of R contains interacting contacts with R.*

Let a' and b' denote the intersection of the wavefront with $b(R^n, R^e)$ and $b(R^s, R^e)$ at instance $priority(s^w) = s^w + w(s) = R^e$ where a' is above b' (see figure 15). (Recall that $b(R^i, R^j)$ denotes the bisector of $R^i, R^j, i, j = n, e, s, w$ and is a 45° line.) If both a', b' are to the left of s^w we have the ordinary case described in [11; 12]. Otherwise we use a', b' to determine the active subsegment of a vertical core segment and to generate an active event for a horizontal core segment.

Lemma 11. *The contact inducing the leftmost active endpoint (if any) of a horizontal core segment s must have a node in the wavefront between a' and b'.*

Lemma 12. *At instance $priority(s^w) = s^w + w(s) = R^e$, the contacts interacting with R (if any) must have consecutive waves in the wavefront.*

In the following we give pseudo-code describing the handling of site events. Suffix $.x$ denotes the x-coordinate of a point. The reader is referred to [11] for details on the basic plane-sweep algorithm.

```
Procedure Process-Left-Event($s^w$)
begin
 1. $R$ = rectangle s.t. $s = core(R)$ induces the event;
 2. $a'$ = intersection(wavefront, $b(R^s, R^e)$);
 3. $b'$ = intersection(wavefront, $b(R^n, R^e)$);
 4. if ($s$ is horizontal) AND ($a'.x > s^w$ OR $b'.x > s^w$) then
 5.     $q$ = CreateActiveEvent($\mathcal{T}$, $a'$, $b'$, $s$);
 6.     push($q$, $Q$);
 7.     Return;
 8. if ($s$ is vertical) AND ($a'.x > s^w$) then
    $a'$= upper-intersection(wavefront, $s$);
 9. if ($s$ is vertical) AND ($b'.x > s^w$) then
    $b'$= lower-intersection(wavefront,$s$);
10. if no intersection was detected at steps 8,9 Return;
    (* $s$ is totally covered by wavefront; $R$ is useless *)
11. $W = \{w \mid w$ = segment of the wavefront between $a'$ and $b'\}$;
12. Initialize $reg(s)$ to $W$; (* $reg(s)$: Voronoi region of $s$*)
13. Update $reg(s')$ $\forall s' \in W$;
14. Invalidate active, right events for core segments intersected by $W$;
15. Delete the elements of $\mathcal{T}$ between $a'$ and $b'$;
16. if $a' \notin s$ then
17.     $b_1 = b(e_a, s^n) = b(e_a, R^s)$; (*$e_a$ = owner of $a'$*);
18.     Insert $b_1, s^n$ in $\mathcal{T}$; (* $s^n$ corresponds to $R^s$ *)
19.     $s_1$ = intersection($b_1$, $prev(b_1)$);
20.     Insert in $Q$ a spike event corresponding to $s_1$;
21. If $b' \notin s$ then
22.     $b_2 = b(e_a, s^s) = b(e_b, R^n)$, (*$e_b$ = the owner of $b'$*);
23.     Insert $b_2, s^s$ in $\mathcal{T}$; (* $s^s$ corresponds to $R^n$ *)
24.     $s_2$ = intersection($b_2$, $next(b_2)$);
25.     Insert in $Q$ a spike event corresponding to $s_2$;
end
```

```
Procedure Process-Right-Event($s^e$)
begin
 1. if $s_e$ is invalid Return; (* the wavefront covers $s_e$*)
 2. if $s$ is horizontal then
 3.     $a' = b' = s^e$;
 4. else
 5.     $a'$ = upper active endpoint (* $a'$ may be $s^n$ *)
 6.     $b'$ = lower active endpoint (* $b'$ may be $s^s$ *)
 7. $b_1 = b(e_a, s^e)$; (*$e_a$ = the owner of $a'$ *);
 8. $b_2 = b(e_b, s^e)$; (*$e_b$ = the owner of $b'$ *)
 9. Update the nodes of $e_a, e_b$ by inserting $b_1, b_2$ in $\mathcal{T}$;
10. Insert the node of $s^e$ in $\mathcal{T}$ between $e_a, e_b$; (* the weighted $s^e$ is equivalent to $R^w$ *)
11. $s_1$ = intersection($b_1$, $prev(b_1)$);
12. $s_2$ = intersection($b_2$, $next(b_2)$);
13. Insert in $Q$ spike events corresponding to $s_1, s_2$;
end
```

Let $s_a \in s$ denote the potential active endpoint of s corresponding to an active event. If s_a is covered by the wavefront when its priority is reached ($priority(s_a) = s_a.x + w(s)$), it is *invalid.* Note that active events are marked invalid at step 14 of procedure *process-Left-Event.* Alternatively the validity of an event can be determined by checking the wavefront.

```
Procedure Process-Active-Event($s_a$)
begin
 1. if $s_a$ is invalid Return; (*$R$ is useless*)
 2. $b_1 = b(e_a, s^n)$; (* $e_a$ = the owner of $s_a$ *)
```

```
    3. $b_2 = b(e_a, s^s)$;
    4. Initialize $reg(s)$ to $b_1 \cup b_2$;
    5. Split the node of $e_a$ into two nodes bounded by $b_1, b_2$;
    6. Insert nodes for $s^n, s^s$ between the nodes of $e_a$ in $\mathcal{T}$;
    7. $s_1$ = intersection($b_1$, $prev(b_1)$);
    8. $s_2$ = intersection($b_2$, $next(b_2)$);
    9. Insert in $Q$ spike events corresponding to $s_1, s_2$;
end
```

Function *CreateActiveEvent* linearly scans $\mathcal{T}$ between a' and b' to determine the contact inducing the active event i.e., the contact R_i with the rightmost R_i^w. The coordinate of the event is given by lemma 8. By lemmas 11, 12, the number of nodes visited in this step cannot exceed the number of interacting vias with R. Similarly, functions *upper-intersection()* and *lower-intersection()* identify the active endpoint(s) of a vertical core-segment by linearly scanning nodes between a' and b'. Once a non-interacting via is visited the search stops (lemma 12). If no active endpoint is found during the search the contact is useless and processing of the event ends (step 10). Functions *CreateActiveEvent*, *upper-intersection* and *lower-intersection* are responsible for the extra K factor in the time complexity of the algorithm. Any other operation requires $O(log|WF|)$ time, where $|WF|$ denotes the size of the wavefront. In practice the number of nodes between a' and b' must be rather small making K negligible. Spike events are treated as in the unweighted case and we skip the discussion. Thus, the time complexity of the algorithm is $O(n \log n + K)$, where K is upper bounded by the number of interacting contacts

7. EXPERIMENTAL RESULTS

We are currently developing a tool to compute Critical Area for shorts, breaks, and via-blocks based on L_∞ Voronoi diagrams. Preliminary experimental results on computing the Voronoi diagram of axis-parallel layouts have been obtained and verify the almost linear performance of the algorithm. Our tool has the overhead of computing unions of shapes and extracting nets on the fly as the sweeping process proceeds. Running times (including the overhead) for the M1 layer of IBM Macros on a 7043-260 server machine are as follows.

Macros	#Shape-Edges	Time
system 390		
Macro1	11,562	1:96 secs
Macro2	13,090	2.22 secs
Macro3	18,608	3.22 secs
Macro4	28,714	4.99 secs
Macro5	68,210	12.37 secs
Power PC		
fpu	1,095,924	4:15 mins
data-cache	2,721,230	11:40 mins

Acknowledgement. The author wishes to thank Dr. G. Tellez for providing the engine on top of which the Voronoi diagrams are being developed. Thanks are due to prof. D.T. Lee and F. Aurenhammer for helpful comments, and to Dr. A. Efrat and Dr. J. Basch for discussions regarding the size of additively weighted Voronoi diagrams.

8. REFERENCES

[1] F. Aurenhammer, "Voronoi diagrams: A survey of a fundamental geometric data structure," *ACM Comput. Survey,* 23, 345-405, 1991.

[2] A.V. Ferris-Prabhu, "Defect size variations and their effect on the critical area of VLSI devices," *IEEE J. of Solid State Circuits*, SC-20,4, 878-880, Aug. 1985.

[3] I. Koren, "The effect of scaling on the yield of VLSI circuits", *Yield Modeling and defect Tolerance in VLSI circuits* W.R. Moore, W. Maly, and A. Strojwas Eds., Bristol UK: Adam-Hilger Ltd., 91-99, 1988.

[4] D. T. Lee, "Medial Axis Transformation of a Planar Shape", *IEEE Transactions on Pattern Analysis and Machine Intelligence'*, PAMI-4, 4, July 1982, 363-369.

[5] W. Maly, "Modeling of lithography related yield losses for CAD of VLSI circuits," *IEEE Transactions on Computer-Aided Design*, CAD-4,3, 166-177, July 1985.

[6] W. Maly, "Computer Aided Design for VLSI Circuit Manufacturability," *Proc. IEEE*, 356-392, Feb. 90.

[7] W. Maly, and J. Deszczka, "Yield Estimation Model for VLSI Artwork Evaluation," *Electron Lett.* vol. 19, no.6, 226-227, March 1983.

[8] C.H. Ouyang, W.A. Pleskacz, W. Maly, "Extraction of Critical Areas for Opens in Large VLSI Circuits", *IEEE Trans. on Computer-Aided Design*, vol. 18, no 2, 151-162, February 1999.

[9] B. R. Mandava, "Critical Area for Yield Models", IBM Technical Report TR22.2436, 12 Jan 1982.

[10] Niagara: IBM Internal DRC and Shape Processing Tool.

[11] E. Papadopoulou and D.T. Lee, "Critical Area Computation via Voronoi Diagrams", *IEEE Trans. on Computer-Aided Design*, vol. 18, no.4, April 1999.

[12] E. Papadopoulou and D.T. Lee, "The L_∞ Voronoi Diagram of Segments and VLSI Applications", submitted for publication. Preliminary version: E. Papadopoulou, "L_∞ Voronoi Diagrams and Applications to VLSI Layout and Manufacturing" in Proc. *9th International Symposium on Algorithms and Computation, LNCS* 1533, 9-18, 1998.

[13] J. Pineda de Gyvez, C. Di, "IC Defect Sensitivity for Footprint-Type Spot Defects," *IEEE Trans. on Computer-Aided Design*, 11, no5, 638-658, May 1992.

[14] Preparata, F. P. and M. I. Shamos, *Computational Geometry: an Introduction*, Springer-Verlag, New York, NY 1985.

[15] J. S. Rogenski, "Extraction of Breaks in Rectilinear Layouts by Plane Sweeps", Technical Report, University of California, Santa Cruz, UCSC-CRL-94-21, April, 1995.

[16] C. H. Stapper and R. J. Rosner, "Integrated Circuit Yield Management and Yield Analysis: Development and Implementation," *IEEE Trans. on Semiconductor Manufacturing* Vol. 8, No.2, 95-101, 1995

[17] C.H. Stapper, "Modeling of Defects in integrated circuits photolithographic patterns," *IBM J. Research and Development*, vol.28, no.4, 461-475, 1984.

[18] I. A. Wagner and I. Koren, "An Interactive VLSI CAD Tool for Yield Estimation," *IEEE Trans. on Semiconductor Manufacturing* Vol. 8, No.2, 130-138, 1995.

[19] H. Walker and S.W. Director, " VLASIC: A yield simulator for integrated circuits," *IEEE Trans. on Computer-Aided Design*, CAD-5,4, 541-556, 1986.