Geometrical Analysis of Mask Pattern for VLSI and Application of the Technique

Akira Tsukizoe*, Junya Sakemi* and Tokinori Kozawa*

An original definition is presented of four distances between two segments at any angle that are not defined in plane geometry. A definition of a nearby region within a fixed distance between two segments is also presented. These are indispensable for the checking of layout design rules pertaining to mask pattern data including diagonal edges.

Using this geometrical analysis, we have developed a high performance pattern checker, MACH (Mask Artwork CHecking program), that can detect all design rule errors and visualize them on a color graphic terminal. MACH has been applied to more than one hundred LSIs since April 1981.

1. Introduction

The demand for very large scale integrated circuits (VLSIs) is increasing year by year. To reduce the time necessary for errorless design of VLSIs, mask pattern data must be checked for every possible error before the time-consuming and expensive photomask generation process is undertaken. However, when VLSI data reach the volume of 1,000,000 or more, manual checking becomes not only tedious and time-consuming but also unreliable.

Consequently, many kinds of CAD systems have been developed and introduced. Amongst these, a pattern checking program, which detects design rule errors from among dense and complicated mask pattern data, is of especial importance.

Mask pattern data are represented by a simple closed loop of edges on a 2-dimensional plane. During the IC and LSI eras, mask pattern data included only a few diagonal edges. However, diagonal edges are now on the increase because of efforts to increase pattern density on a single LSI chip. This makes it absolutely necessary that mask artwork systems that include a pattern checking program are able to handle diagonal edges within a short run time.

A VLSI designer needs to check design rules for error locations reported by a pattern checking program, and correct real error locations. Therefore, the pattern checking program must not only be able to efficiently handle large amounts of data, but also detect all design rule errors without fail. If this is not possible, the designer can not rely on the program, and real design rule errors may often be overlooked.

In order to make it possible for a pattern checking program to accurately detect all design rule errors from among dense VLSI data including diagonal edges of any angle, a distance between two non-parallel segments and a nearby region that violates a specified tolerance value must now be defined. They have not been defined in plane geometry.

Recently, pattern checking programs providing good performance (2,000 ~ 3,000 figures per second) have been reported [1][2][3][4]. However, previous papers defined neither a distance between two segments at any angle nor a nearby region.

We have geometrically analyzed VLSI mask patterns. We have defined four distances between two segments at any angle and combinatorially analyzed them. We have also defined a nearby region between them. Using this geometrical analysis technique, we have developed a high performance pattern checker.

This paper presents geometrical problems in pattern checking, the definition of four distances and a nearby region between two segments at any angle, and application of the geometrical analysis.

2. Pattern Checking for VLSI Mask Pattern

Mask pattern data are checked in accordance with design rules for the process technology used. Every design rule is generally described such that the minimum distance between two segments must be larger than a tolerance value. Various kinds of design rules are constructed corresponding to the tolerance value and checked points, as shown in Fig. 1.

Let's consider what distance between two segments should be calculated and compared with the tolerance

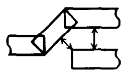
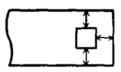
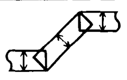
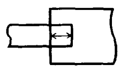
No.	Item	Checked Points	No.	Item	Checked Points
1	Space		3	Enclosure	
2	Width		4	Incursion	

Fig. 1 Design rules.

*Central Research Laboratory, Hitachi, Ltd., Higashi Koigakubo, Kokubunji, Tokyo 185, Japan.

value, and how the detected error locations should be displayed.

In our old pattern checking program, diagonal edges non-parallel to the axes were not handled. The checking method for two segments, P_1P_2 and P_3P_4, parallel to the X (or Y) axis was as follows:

Step 1: The minimum distance is decided by comparison among X (or Y) coordinates of four vertices. In other words, if a linear order of (P_1, P_2, H_3, H_4) on $L(P_1, P_2)$ is $P_iP_jH_mH_n$ or $H_nH_mP_jP_i$, then the minimum distance is the length of P_jP_m. Otherwise, it is the length of perpendicular P_mH_m. Here, $(i, j)=(1, 2)$, $(m, n)=(3, 4)$, and H_m is the foot of a perpendicular from P_m onto $L(P_1, P_2)$. $L(P_1, P_2)$ is a straight line passing through P_1 and P_2.

Step 2: If the minimum distance is less than a specified tolerance value, then the two segments are output and displayed.

There are two problems awaiting in this method. First, if this method is applied to mask pattern data including diagonal edges, two non-parallel segments can not be correctly handled as shown in Fig. 2. Because, even if one or more feet of perpendiculars exist in the interior of segments, their lengths don't always include the minimum distance.

Second, error locations should be represented by shapes rather than by line segments. Because, the designer can notice error locations at a glance in the case of shapes.

3. Geometrical Analysis of Mask Pattern

There are the minimum distance and the maximum one between two segments. Thus, we define four distances between them, which must be measured and checked in pattern checking. Based on this definition, we also define a nearby region within a fixed distance between two segments.

With these definitions,

(1) mask pattern data can be checked in accordance with design rlues, which is more generally described than usual such that the distance between two segments, D, is invalid if $d_1 \leqq D \leqq d_2$, where d_1 and d_2 are tolerance values; and

(2) proper shapes that vividly indicate error locations can be output.

3.1 Definition of Four Distances between Two Segments

We took a look at uniquely-defined distances in terms of plane geometry. In our approach, distances between two points, $d(P, P')$, and between a point and a straight line, $d(P, L)$, are defined as:

$$d(P, P')=\sqrt{(x-x')^2+(y-y')^2} \text{ and}$$
$$d(P, L)=|x* \cos A+y* \sin A-C|,$$

where (x, y) and (x', y') are co-ordinates of P and P', respectively, and the equation for straight line L is

$$X* \cos A+Y* \sin A=C \ (A,\ C\text{: constants}).$$

Based on this elementary knowledge in Euclidean geometry, points at a fixed distance from a segment can be defined as shown in the upper part of Fig. 3. Points on the straight line part are a constant distance from the segment. Points on the semicircular part also maintain a constant distance from one of the vertices.

This yields the following definition of the distance between a point and a segment, as shown in the lower part of Fig. 3. Let H_i be the foot of a perpendicular from P_i onto $L(P_m, P_n)$, where $L(P_m, P_n)$ is a straight line passing through P_m and P_n. We define the distance between point P_i and segment P_mP_n, $d(P_i, P_mP_n)$, as the minimum $d(P_i, P)$ for any P in the interior of P_mP_n. That is, $d(P_i, P_mP_n)=d(P_i, H_i)$ if H_i is in the interior of P_mP_n. Otherwise, $d(P_i, P_mP_n)=\min(d(P_i, P_m), d(P_i, P_n))$. If $d(P_i, P_mP_n)=d(P_i, Q)$, we call P_iQ a distance-edge, where Q is H_i, P_m, or P_n.

Based on above analysis, we define the four distances between two segments P_1P_2 and P_3P_4 as $d(P_1, P_3P_4)$, $d(P_2, P_3P_4)$, $d(P_3, P_1P_2)$, and $d(P_4, P_1P_2)$.

These four distances change depending on relative superposition and lengths of the segments. Thus, we have analyzed four distances for the purpose of correct and efficient processing.

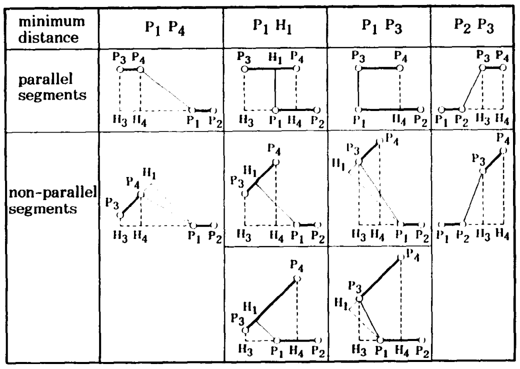


Fig. 2 Minimum distance between two segments.

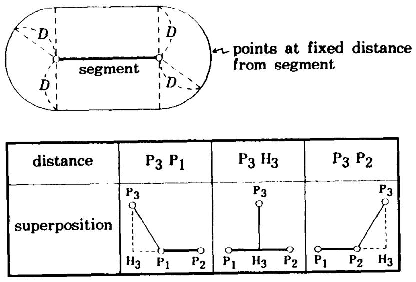


Fig. 3 Distance between point and segment.

3.2 Combinatorial Analysis of Four Distances

Our method for analyzing two segments' superposition is a method whereby a linear order combination is viewed. A combination of two linear orders is examined with regard to vertices and perpendicular feet on straight lines passing through each segment.

Let us consider the distances between two segments, P_1P_2 and P_3P_4.

(P_1, P_2, H_3, H_4) and (H_1, H_2, P_3, P_4) are linearly ordered on $L(P_1, P_2)$ and $L(L_3, P_4)$, respectively. There are 24 linear orders for each (P_i, P_j, H_m, H_n), where $((i, j), (m, n)) = ((1, 2), (3, 4))$ or $((3, 4), (1, 2))$. There are 84 possible combinations of linear orders of (P_1, P_2, H_3, H_4) and (H_1, H_2, P_3, P_4). If an intersection-point exists between two segments, we can deal with them after being cut at the intersecting point.

We assume $d(P_1, H_1) \leqq d(P_2, H_2)$ and $d(P_3, H_3) \leqq d(P_4, H_4)$, without losing generality. By this assumption, the number of linear orders is reduced from 24 to 6, as shown in Fig. 4. The number of combinations is also

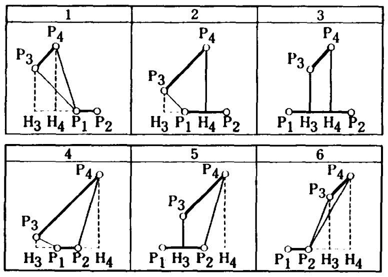


Fig. 4 Linear order (P_1, P_2, H_3, H_4).

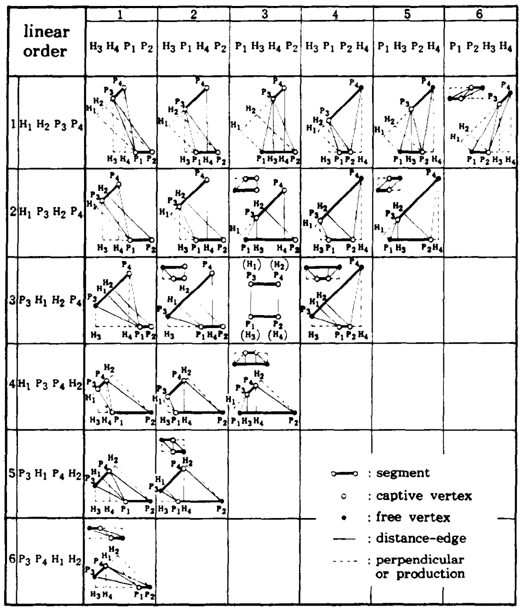


Fig. 5 Combinations of linear orders.

reduced from 84 to 21.

Fig. 5 shows four distances between two segments at any angle, with all cases listed. The distances for each combination include both a minimum and a maximum one. Thus, this definition holds generality and uniqueness. The number of distance-edges for one combination is 2, 3, or 4.

Next, we divide these distances into two classes. If (P_j, H_m, H_n) are on the same side with regard to P_i, and (H_j, P_m, P_n) are on the same side with regard to H_i, then we call P_i a free vertex. Otherwise, we call it a captive vertex.

We divide the four distances between two segments into basic distances and referential distances. If P_i is a captive vertex, we call $d(P_i, P_mP_n)$ a basic distance, and if P_i is a free vertex, we call it a referential distance. A minimum distance is included in basic distances. If P_1 is a free vertex and P_2 is a captive vertex, a half line passing through P_1 with terminal point P_2 has the same basic distances from P_3P_4 as segment P_1P_2 has.

3.3 Definition of Nearby Region between Two Segments

We define a nearby region, R, within a fixed distance, D, between two segments, P_1P_2 and P_3P_4, as:

$$R = \{(x, y) \mid d(P', P'') < D\},$$

where $P(x, y)$ is any point in the interior of segment $P'P''$, and P' and P'' are any points on P_1P_2 and P_3P_4, respectively. Segment $P'P''$, however, does not include any other points on two segments.

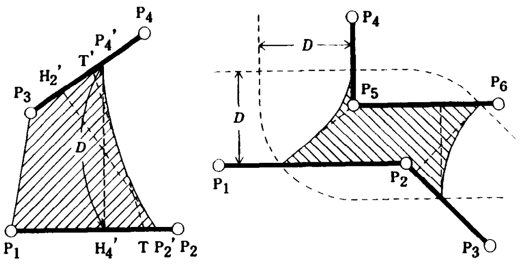
The hatched region shown in Fig. 6(1) indicates a nearby region within a fixed distance, D, between two segments. The lengths $P_2'H_2'$ and $P_4'H_4'$ equal D. Any point on the curved border has tangent TT' of length D. Based on this definition, a nearby region between two segment sequences can be generated as shown in Fig. 6(2).

This nearby region is a strictly-calculated shape, but not a proper output shape for a pattern checker. There are some reasons for this. First, processing of its generation would be very difficult and time-consuming. Second, this shape includes curved lines, so we can't use it as input data for various pattern operations [5], such as decomposing of pattern data concerning width. Thus, transformation of curves into straight lines is necessary.

Fig. 7 shows an example of a proper output shape for a nearby region. This is the maximum polygon covered by the nearby region.

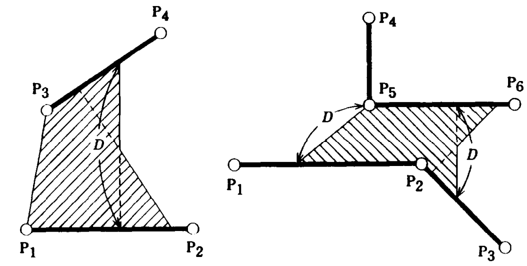
4. Application of Results

We have developed a high performance pattern checker, MACH, that takes this geometrical analysis as a basis. Input data are constructed by mask pattern data, DRL, and items to be checked. DRL is described by pattern operations [5]. Violation data are displayed on a color graphic terminal, electro-static plotter(ESP),



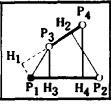
(1) Between two segments (2) Between two segment sequences

Fig. 6 Nearby region within fixed distance.



(1) Between two segments (2) Between two segment sequences

Fig. 7 Example of output shape for nearby region.

combination of linear orders		basic distances	P_3H_3 (minimum) P_4H_4, P_2H_2
		referential distance	P_1P_3

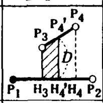
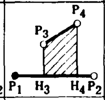
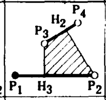
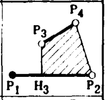
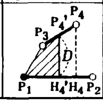
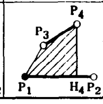
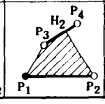
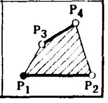
conditions for basic distances	$P_3H_3 \leq D$ $P_4H_4 > D$ $P_2H_2 > D$	$P_3H_3 \leq D$ $P_4H_4 \leq D$ $P_2H_2 > D$	$P_3H_3 \leq D$ $P_4H_4 > D$ $P_2H_2 \leq D$	$P_3H_3 \leq D$ $P_4H_4 \leq D$ $P_2H_2 \leq D$
output shape for $P_1P_3 > D$				
output shape for $P_1P_3 \leq D$				

Fig. 8 Generation method of output shape.

or X-Y plotter.

The program is written basically in PL/I with some assembler language and consists of approximately 75K lines of code. It runs on a HITACHI M-200H with a VOS-3 operating system. In practice, MACH has been applied to more than one hundred LSIs since April 1981.

4.1 Pattern Checking Processing

The pattern checking processing implemented in MACH is as follows:

Step 1: After mask pattern data and tolerance value D are input, four steps from Step 2 to Step 5 are repeated until all input data are checked.

Step 2: A segment pair is selected. It is not necessary to check pairs of orthogonal segments. If $L(P_i, P_j)$ is at right angles to $L(P_m, P_n)$, neither segment P_iP_a connecting P_i nor segment P_jP_b connecting P_j are at right angles to $L(P_m, P_n)$, where a, $b \neq i$ or j. Thus, violation data between P_iP_j and P_mP_n, even if it exists, can be detected by checking between P_iP_a and P_mP_n or between P_jP_b and P_mP_n.

Step 3: Coordinate values for the feet of perpendiculars are calculated.

Step 4: The combination of linear orders is decided. Then, the values of four distances are calculated.

Step 5: A minimum distance is checked. If it is less than D, an output shape for the current segment pair is generated. (This generation method is described below.)

Step 6: After all segment pairs have been checked, output shapes are merged using a Boolean OR operation.

Using the four distances, classified into basic and referential distances, we have derived a generation method of an output shape that vividly indicates invalid distances between two segments.

In our method all distances are checked for the purpose of indicating all invalid distances, not for indicating valid distances. Considering the processing speed, the intersection-point calculation between distance-edges is not executed. An output shape is constructed from the distance-edges or the segments parallel to them that are perpendiculars.

Our generation method for a certain case is shown in Fig. 8. In this case, there are three basic distances and one referential distance. If all basic distances except the minimum distance are valid, a basic distance-edge, that is the perpendicular, is moved parallel to itself. $P_4' H_4'$ is a segment parallel to P_4H_4 such that P_4' is in the interior of P_3P_4, H_4' is in the interior of H_3H_4, and $d(P_4', H_4')=D$. This segment can be easily obtained through proportional allotment of parallel segments. Referential distance P_1P_3 is checked so that fine shapes can be output when adjacent shapes are merged.

Processing for generation of output shapes for other cases is similar.

4.2 Experimental Results

Table 1 shows results from evaluating MACH for three manufactured LSIs. The amount of checked data was about 1,500,000 vectors, on the average. Our old program could not handle diagonal edges because the distances between two non-parallel segments were not defined. MACH can handle diagonal edges and detect all design rule errors without fail. Moreover, the execution time varies in proportion to the number of input data. Thus, MACH is quite effective for large amounts of mask data.

Fig. 9 shows a space error display example. The hatched regions indicate locations that violate a specified tolerance value. The dotted lines are edges selected in generating output shapes for near-by regions between each pair of segments. The thin lines are

Table 1 Performance summary.

Pattern checker	MACH	Old
Diagonal edges	○	×
No. of output shapes	264	210
Detecting rate (%)	100	80
Exec. time complexity (T: CPU time, N: No. of data)	$T \propto N^{1.0}$	$T \propto N^{1.2}$

$$\text{Detecting rate} = \frac{\text{No. of output shapes}}{\text{No. of real design rule errors}} \times 100$$

generated to merge the shapes for all pairs of segments, using an OR operation. By means of this display, the designer can judge to what extent he should modify the mask pattern data.

5. Conclusions

We have defined four distances between two segments at any angle that are to be measured in pattern checking. The distances can be uniquely determined by means of a combination of two linear orders with regard to vertices and perpendicular feet on straight lines passing through each segment.

We have also defined a nearby region between two segments and proposed a proper shape to be output by a pattern checker.

Using our geometrical analysis, a pattern checking program (MACH) has been developed. Owing to our definitions, pattern checking can effectively proceed without any real design rule errors made in an LSI design being missed. Moreover, the definitions yield low-cost programming and high-speed execution of pattern checking. The output shapes vividly indicate all invalid distances. This MACH has been successfully applied to more than one hundred LSIs since April 1981.

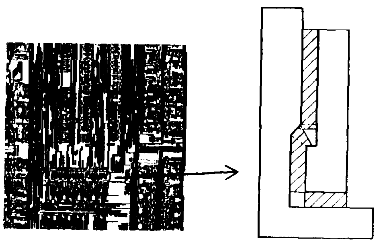


Fig. 9 Example of Space error display.

Acknowledgments

The authors wish to thank Drs. Hisashi Horikoshi and Hideki Fukuda for their invaluable guidance and advice. Also, the authors wish to thank Messrs. Tsuyoshi Takahashi and Hiroki Kohno for their useful discussion and contribution.

References

1. Yoshida, K., Mitsuhashi, T., Nakada, Y., Chiba, T., Ogita, K. and Nakatsuka, S. A Layout Checking System for Large Scale Integrated Circuits, *Proc. of the 14th DA Conf.*, (June 1977), 322-330.
2. Wilcox, P., Rombeek, H. and Caughey, D. M. Design Rule Verification Based on One Dimensional Scans, *Proc. of the 15th DA Conf.*, (June 1978), 285-289.
3. Alexander, D. A Technology Independent Design Rule Checker, *3rd USA-JAPAN Computer Conf.*, (1978), 412-416.
4. McCaw, C. R. Unified Shapes Checker—A Checking Tool for LSI, *Proc. of the 16th DA Conf.*, (June 1979), 81-87.
5. Kozawa, T., Tsukizoe, A., Sakemi, J., Miura, C. and Ishii, T. A Concurrent Pattern Operation Algorithm for VLSI Mask Data, *Proc. of the 18th DA Conf.*, (June 1981), 563-570.

(Received December 27, 1983)