論文 89-26-5-18

A Time and Space Efficient Algorithm for VLSI Geometrical Rule Checking

(시간 및 공간복잡도가 개선된 VLSI설계규칙 검증 알고리듬)

鄭 子 春,* 申 成 勇,** 李 玄 燦,* 李 哲 東,* 柳 瑛 昱*

(Ja Choon Jeong, Sung Yong Shin, Hyun Chan Lee, Chul Dong Lee and Young Uk Yu)

要 約

기하학적인 마스크 설계도면에서 최소폭/간격을 효과적으로 검증하기 위한 새로운 알고리듬을 제안한다. 제안된 알고리듬은 영역탐색문제를 평면소인법을 사용하여 순서적으로 해결하고 있다. 이 알고리듬은 O(n log n)의 시간복잡도를 가지는데, 모든 최소폭/간격의 위반을 보고하는 문제에 있어서 시간복잡도의 lower bound가 Ω(n log n)이기때문에 이론적으로 최적이다. 여기서, n은 마스크 패턴에서의 edge의 총 수이다. 그리고, O($n^{0.5}$)의 공간복잡도를 가지므로 실제적으로 매우 빠르다. 그리고 종래의 알고리듬에 비해 간단하므로 이해하기 용이하고 구현하기가 편리한 장점이 있다. 이 알고리듬은 VAX8650 컴퓨터에서 C-언어로 구현되었으며 직교영역에서 25만개의 정점 데이타에 대해서 116.7초내에 처리했다.

Abstract

A new algorithm is presented which efficiently reports minimum width/space violation in a geometric mask pattern. The proposed algorithm solves a sequence of range search problems by employing a plane sweep method. The algorithm runs in $O(n \log n)$ time, where n is the number of edges in a mask pattern. Since a lower bound in time complexity for reporting all minimum width/space violations is $\Omega(n \log n)$, this algorithm is theoretically optimal within a constant multiplicaive factor. It requires $O(n^{0.5})$ space which is very efficient in practice. Moreover, this algorithm, we believe, is easy to implement and practically fast (116.7 seconds for a rectilinear region with 250000 vertices at VAX 8650.)

*正會員, 韓國電子通信研究所
(Electronics and Telecommunications Research Institute)

**正會員, 韓國科學技術大學 電子電算學部
(Korea Institute of Technology)

接受日字 : 1989年 2月 10日

I. Introduction

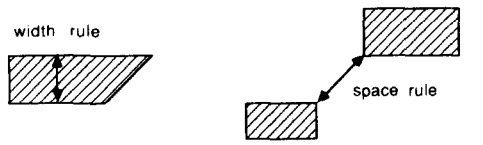
The design rule check for a mask pattern has become an important issue from the early era of using computer for IC design. It is still important nowaday when automatic layout design methods have been established, since manual aids for these methods are needed to increase chip

density and circuit performance [2-11].

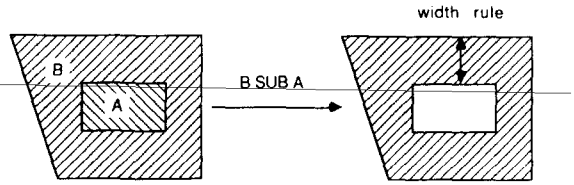
In the last decade, VLSI process technology has matured to the extent that 10^4-10^6 transistors can be integrated in a single chip. This implies that there are 10^6-10^8 vertices in a geometric mask pattern [1]. However, the advancement of computer technology is far behind the growing rate of data volume. Therefore, the demands for fast algorithms with modest memory have ever been increasing. In other words, the time complexity should grow linearly or near linearly in input size, and only a small fraction of the layout data should be held in main memory (sublinear space complexity.) In order to achieve the above requirements, the following conditions are required: (1) whole data is accommodated in a sequential file, (2) only those patterns within a thin slit of the plane are loaded in the main memory. These conditions can be satisfied if a plane sweep method is employed.

The design rule checking procedure at mask level can be done by combining Boolean operations (AND, OR, NOT, SUB etc.) and minimum width/space checking, which is illustrated in Figure 1(a). Optimal algorithms for performing Boolean operations have been developed and used for layout verification [2-7]. These algorithms are also based on the plane sweep method. However, for minimum width/space verification problem, there are several $O(n^2)$ algorithms in the worst case [2, 3, 4, 5, 8], which is not practical any more. The enclosure rule can be checked in two steps, first, region A abutted with layer B is removed using the SUB operation, and then the minimum width checking is performed as shown in Figure 1 (b).

In this paper, we present a minimum width/space verification algorithm which solves a sequence of range search problems by employing a plane sweep method. This algorithm runs in O (n log n) time and requires O ($n^{0.5}$) space for the main memory which is theoretically optimal and practically fast. The minimum width of polygonal regions is equal to the minimum space of their complementary regions with respect to the plane. Therefore, we will concentrate only on the minimum space verification prbolem.



(a) Minimum width/space rule



(b) Enclosure rule

Fig.1. Design rule check procedure.

II. Minimum Space Verification Algorithms

Several techniques have been developed to solve geometric problems in the plane (i.e., 2-dimensional geometric problems) efficiently. Among them, there is the plane sweep method, which reduces a 2-dimentional geometric problem to a sequence of 1-dimensional ones. This method is similar to the work-list [3] method in layout verification. In the work-list method, the plane sweep is exploited in such a way that only geometric objects intersecting the current scan line are fetched from a sequential file which accomodates the whole data. And it is well-known that the data volume corresponding to the intersecting geometric objects is proportional to the square root of whole data volume. Therefore, $O(n)$ space algorithm in theoretical computational geometry becomes $O(n^{0.5})$ space in layout verification. If the line segments in the work list are managed by a balanced tree, e.g. an AVL tree [12], according to their X-coordinates, the unit operation of inserting or deleting a line segment, or searching for the adjacent vertical line segments can be done in O ($log\ m$) time, where m is the number of line segments intersecting the scan line. Therefore, everything desired can be reported in O (n log m) time once the whole plane has been swept from top to bottom.

The minimum space checking algorithms, as far as we known, can be classified as shown in Table 1 according to their computational complexity.

Table 1. Minimum space algorithms.

Algorithm	Time Complexity	Space Complexity
[Tsukizoe 83]	$O(n^2)$	$O(n^{0.5})$
[Sato 85]	O (n log n)	O (n)
Proposed	O (n log n)	$O(n^{0.5})$

[Tsukizoe 83] is one of several $O(n^2)$ time algorithms in the worst case [2, 3, 4, 5, 8] by employing work-list method. $O(n^2)$ time algorithm is not very useful any more. $O(n \log n)$ time and $O(n)$ space [Sato 85] algorithm [10], which needs the whole data in main memory, is not adequate for a recent VLSI layout verification. The sequential range query model proposed in this paper gives $O(n \log n)$ time and $O(n^{0.5})$ space complexity, which is considered as theoretically optimal in layout verification.

Before we present our ideas, we show that how $O(n \log n)$ time algorithm can be obtained for the minimum space checking problem. One general method for solving this problem is using Resizing operation such as EXPAND or SHRINK. The minimum space error of a rectilinear region can be checked as follows:

[Sept 1] Given the minimum space d, expand the regions by d/2 outward from their boundaries.

[Step 2] If such an expansion results in overlapping of regions, they are reported as a violation of the given minimum space rule (Figure 2).

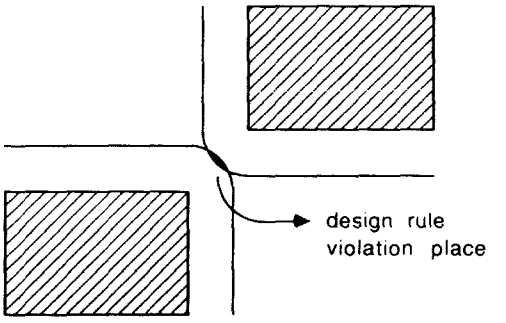


Fig.2. Minimum space check by means of the polygon resizing.

However, this method requires two plane sweeps (horizontally and vertically). This is not desirable when a large amount of data is accommodated in a sequential file. In addition, the expansion may create circular arcs which do not exist in the original mask pattern. This method gives an O (n log n) algorithm, which is theoretically optimal, but not very efficient practically.

An other way to achieve O (n long n) time complexity is to use Voronoi diagram. Consider the Voronoi diagram with respect to the set of convex vertices of the input pattern. Two points p and q are said to be adjacent if their Voronoi polygons V (p) and V(a) have a common boundary. The point q closest to p can be found by checking only those convex vertices whose Voronoi polygon shares common boundary with V(p). General algorithm for constructing Voronoi diagrams [14] are based on the assumption that the whole data is accommodated in main memory. These algorithms can construct a Voronoi diagram of n points in O (n long n) time which is optimal in the worst case. An improved algorithm with $O(n^{0.5})$ space complexity has been proposed by using the plane sweep method [11]. However, this algorithm uses complicated data structures consisting of five balanced trees in addition to a Voronoi diagram. Therefore, it is not practical even though it is theoretically optimal.

III. Sequential Range Query Algorithm

The methods measuring the minimum space as mentioned above use ttwo different algorithms, i.e., plane sweep for finding the minimum space for parallel edges and constructing partial Voronoi diagram for finding convex vertices. It is too expensive to find only the nearest neighbor points using Voronoi diagram. In order to get around these drawbacks, we propose the sequential range query model where only one process is enough for two purposes. This method is based on the range search technique in computational geometry.

Now, we propose a practial algorithm applicable to the minimum space violation checking. The minimum space of a rectilinear region is defined as follows. Let p and q be points on the boundary B(R) of rectilinear regions R. We call the distance between p and q as minimal distance, if the straight line drawn between p and q does

not intersect with R, and for any neighbor p'= B(R) of p and any neighbor q'=B(R) of q, the following inequality holds: d(p, q) less than d(p', q.), where d(p, q) is the Euclidean distance between p and q. (Figure 3.) Then, the minimum space of R is defined as the minimum value among all minimum distance of R. A pair of two objects (points or edges) which constitute the minimum space are called the closest pair of R.

It is clear that the minimum space of rectilinear regions is composed of either

(1) a pair of parallel edges facing each other (Figure 4(a)), or
(2) a pair of convex vertices facing each other (Figure 4(b)).

It is obvious how to check the minimum space error once the minimum space is found. The minimum space in case (1) can be obtained systematically by employing a plane sweep technique. However, the minimum distance in case (2) requires more complicated operations, since Euclidean distance between two convex vertices need to be computed.

The sequential range query search is based on the range search technique in computational geometry. The range search problem is that, given a set S of n points in a plane and a rectangle R with each side parallel with a coordinate axis, find all points in S lying in R as shown in Figure 5. Let d be minimum allowable space. For each horizontal edge of width w, we construct its query region R consisting of the followings: (1) two quad-circles centered at the end points of the edge with a radius d, and (2) a rectangle of height d and width w between the quad-circles (Figure 6). The best algorithm for solving the range search problem with a rectangular query requires O(log n + k) search time and O (n log n) space [15], where k is the number of points reported. For a circular query, the algorithm requires O(log n + k) search time and O(n) space [16].

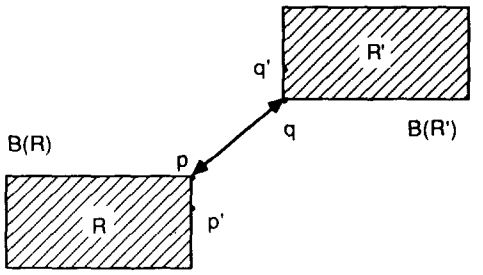
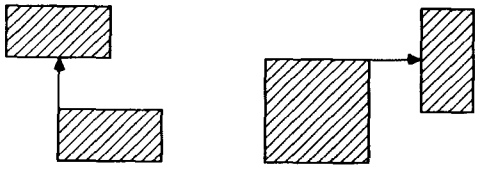
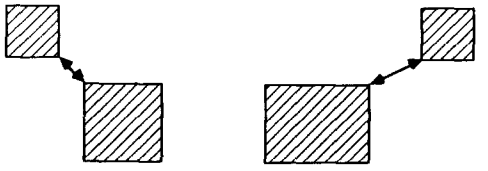


Fig.3. Definition of minimum space.

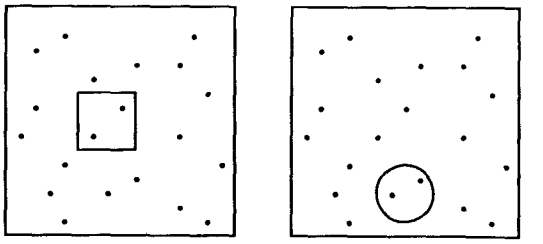


(a) Parallel edge facing to each other



(b) Convex vertices facing to each other

Fig.4. Configuration of the minimum space.



(a) Rectangular query (b) Circular query

Fig.5. Range search problem.

Applying this method directly to design rule check, the total time requirement becomes O(n log n), since a query is needed for each of n horizontal edges in the worst case. However, the space requirement, O(n log n), is larger than other algorithms, it needs to be improved. The height of the query region in this model is always d which is minimum allowable space. Furthermore, the query region is extremely small compared with the entire region. If we sequentially perform queries from top to bottom, employing the plane sweep technique, it can efficiently reduce space

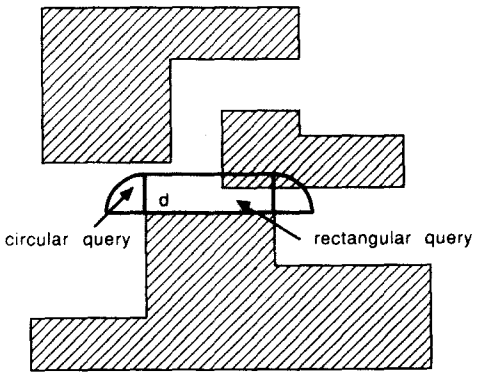


Fig.6. Sequential range query model.

requirement down to O ($n^{0.5}$).

In order to perform queries efficiently, consider two kinds of sweep line, the LSL (Leading Sweep Line) and the CB (Candidate Boundary), as shown in Figure 7. The former extracts only horizontal edges sequentially from the given mask pattern. The horizontal edges are sorted by y-coordinates and stored in E-file. The latter consists of a sequence of edges of polygons, which lie within distance d, and are vertically visible, from the LSL. CB is part of a work-list which stores all edges whithin d from LSL. The work-list is stored in a Queue according to y-coordinates of the edges. The CB is stored in a balanced tree, say 2-3 tree, according to the x-coordinates of two endpoints of the edges in the work-list.

The update operations, INSERT and DELETE, for work-list and CB occur in two cases. The first case is when the LSL hits any horizontal edge Lr of a polygon, where Lr indicates the left (right) endpoint of the edge. Then the edge lr will be inserted into the work-list and CB. If the y-coordinate difference between the LSI and the first element (the edge with the largest y-coordinate) of the Queue is greater than d during the movement of the LSL, the element is deleted from the work-list and CB.

Figure 8 shows an example of how the above algorithm works. When the LSL hits edge cd, the CB and Q will store edge ab and cd. And according to the advancement of the LSL, if the edge lr is hit, the first element, edge ab, of the balanced tree and the edge lr are compared according to their y-coordinate base. Because the y-coordinate difference is greater than d, the corresponding element, edge ab, is deleted from the work-list. But for edge cd is divided into two parts, cc' and c'd, for forming CB. The portion of cc' is covered by the edge lr vertically visbile from it, therefore only the edge c'd is included in CB. Notice that the pseudo vertex c' is generated by this operation. The priority-queue is arranged by (cd, lr). The sequential range search algorithm is described in Figure 9.

For the minimum space violation checking, the range search operation (we call MEMBER) is performed, When the LSL hits the upper side horizontal edge lr of a polygon, the expanded

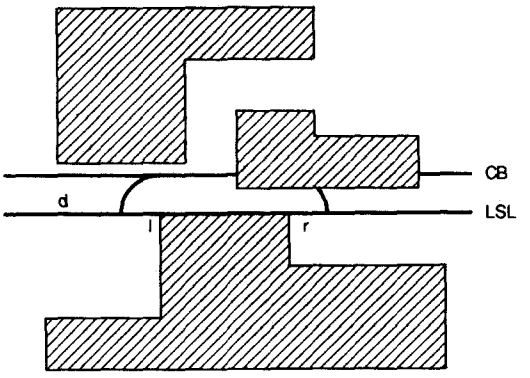


Fig.7. Sequential range query algorithm by the two sweep lines.

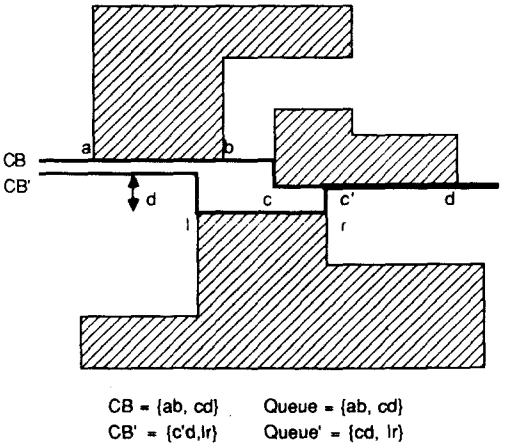


Fig.8. Explanation of the sequential range query algorithm.

```
Min_Space_Violation_Check(R, d)

Input:    a rectilinear region R stored in secondary memory, represented by a sequen-
          tial list of n horizontal edges.
          minimum design rule d.
Output:   all pairs that violate the minimum design rule
Method:   sequential range search algorithm.

begin
  E_file = n/2 horizontal edges of given rectilinear regions;
    /* The edges are sorted by y-coordinates. If there exist edges having endpoints with
       the same y-coordinates, then they are sorted by x-coordinates of the left endpoint
       of edges. */
  CB = ∅;     /* Candidate boundary stored in a balanced tree. */
  Queue = ∅;  /* FIFO structure */
  while (E_file ≠ eof) {
    nedge = EXT(E_file);          /* nedge becomes LSL */
      /* EXT extracts line-segment from E_file. */
    repeat {
          if the difference(dif) of y-coordinates between nedge and the first element of
          Queue is greater than d, then delete the edge from Queue and CB.
      }
    until (dif is less than d)
    delete all edges from CB and Queue which have greater x-coordinates than nedge.l
      has and smaller than nedge.r has;
    insert nedge at the proper places in CB and Queue;
    report = MEMBER(nedge);
  }
end
```

Fig.9. Implementation of sequential range query algorithm.

```
procedure MEMBER(nedge)

Input: nedge is a horizontal edge.
Output:minimum space violation portion from nedge.

begin
  v1 = the left endpoint of nedge;
  v2 = the right endpoint of nedge;

  /* SEGMENT(a, b) generates new line segment whose endpoints are a and b. */
  if (v1 is convex) { e1 = SEGMENT(v1.x-d, v1.x); }
  else { e1 = ∅; }
  if (nedge is upper side of a polygon) { e2 = SEGMENT(v1.x, v2.x); }
  else { e2 = ∅; }
  if (v2 is convex) { e3 = SEGMENT(v2.x, v2.x+d); }
  else { e3 = ∅; }

  /* CB_LINE(a, b) finds all line segments whose left endpoint's x-coordinate is greater
     than or equal to a and right endpoint's is less than or equal to b in CB */
  s1 = CB_LINE(e1);
  s2 = CB_LINE(e2);
  s3 = CB_LINE(e3);

  while (s1 ≠ ∅) {
    e = EXT(s1); /* any one edge in the set s1 */
    if (e violates the circular range query) { report_violation(e); }
  }
  if (s2 ≠ ∅) { report_violation(s2); }
  while (s3 ≠ ∅) {
    e = EXT(s3); /* any one edge in the set s3 */
    if (e violates the circular range query) { report_violation(e); }
  }
end.
```

Fig.11. Implementation of MEMBER operation.

interval by the distance d, [l-d, r + d], is generated, and searches all elements above the edge lr. If it founds any elements by the above step, more complex operations are needed for the regions of two quad-circular queries as shown in Figure 10. The dotted portion in Figure 10, [v,r + $\sqrt{d^2 - c^2}$] will be minimum space violation portion in this example. Figure 11 shows an algorithmic description of MEMBER operation.

The above algorithm can check all minimum space violations by sweeping the plane only once and using two lists. One is the balanced tree [1] for the candidate boundary which can perform INSERT and DELETE in O(log m) time and MEMBER in O(log m + k) time, where m is th number of elements intersected with one sweep line. The other is a 2-3 tree which can also perform INSERT and DELETE in O(log m) time.

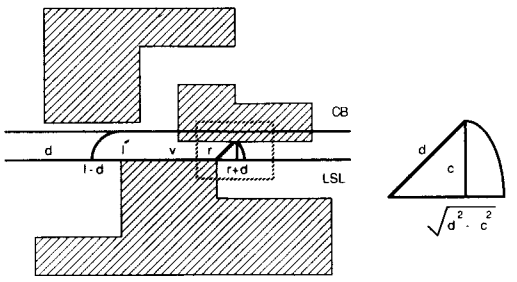


Fig.10. MEMBER operation.

It is clear that the overall time complexity is O(n log n). It is well known that m is proportional to $n^{0.5}$ in the layout verification problem. Therefore, the space complexity for main meory is $O(n^{0.5})$.

IV. Computational Experiences

Our design rule check system consists of the following routines.

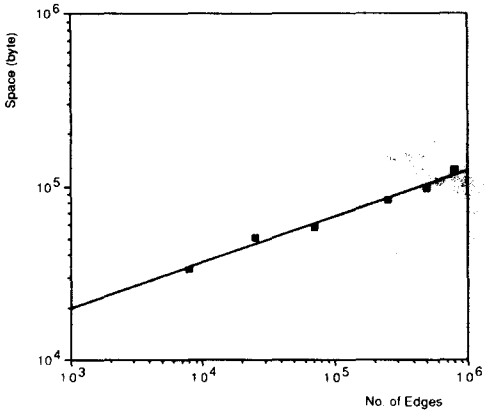
The first one is input transaltion routine. Polygons represented in a layout description language, CIF for instance, are converted into a sequence of horizontal edge. An edge is a quadruple (point, id, sense, slant), where point is the edge's two endpoints. The id is a identification number of membership in a particular geometric region. The sense is an encoded representation of the edge's directionality, i.e. whether it bounds the top of a region (the region is below), the bottom (the region is above), or represents a pair of edges (a rectangle, for example). The slant indicates the edge's angle for non-rectilinear polygons (0 for rectilinear regions). Note that vertical edges are not required since they can easily be inferred. Then these edges are sorted according to y-coordinate in increasing order by file sort method.

Next routine compiles the design rule set provided by the process technology. Boolean

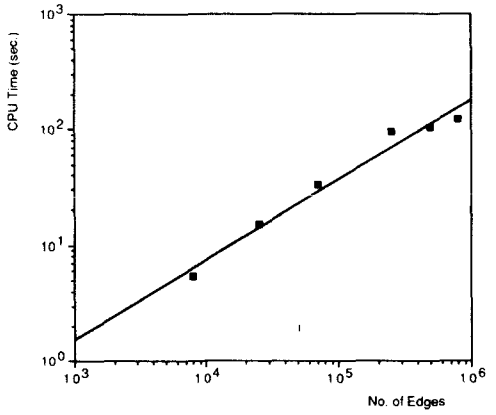
operations are performed according to the complied rule sets. Then, the minimum width/space check operation is performed, and report any violations if it is found.

The sequential range query algorithm were implemented in C programming language on VAX 8650 computer. The graph shown in Figure 12(a) is a long-log plot of the CPU time versus the number of edges. The data does not represent real chip. It is generated by random method. This plot is rather linear with a slope of 1.056, which is slightly less magnitude than O(n log n). This confirms our theoretical result, i.e., O(n log n) time complexity.

The graph shown in Figure 12(b) is a log-log plot of the main memory demand, in bytes, versus the number of input edges. The slope of this plot is about 0.356 which is less than 0.5. Therefore, it also confirms our average case space analysis.



(a) Main memory



(b) CPU time

Fig.12. Experimental results.

V. Conclusions

In this paper, we present an efficient algorithm for minimum width/space verfication. This algorithm is especially efficient when we deal with a large number of designed pattern within a whole chip, since it keeps the theoretically optimal time and space complexity, and runs fast practically. An experimental results show that the average case performance of the sequential range search algorithm runs almost linearly. Although we only dealt with rectilinear regions as input data, it is easy to extend our algorithm to handle regions with tilted edges.

References

[1] K. Yoshida, "Layout verification," in Lyout Design and Verification, ed. T. Ohtsuki, Advances in CAD for VLSI, vol. 4, North-Holland, Amsterdam, pp. 237-265, 1986.

[2] B.W. Lindsay and B.T. Preas, "Design rule checking and analysis of IC mask design," Proc. 13th DAC, pp. 301-308, 1976.

[3] P. Wilcox, H. Rombeek and D.M. Caughey, "Design rule verification based on one dimensional scans," Proc. 15th DAC, pp. 285-289, 1978.

[4] C.R. McCaw, "Unified shapes checker – a checking tool for LSI," Proc. 16th DAC, pp. 81-87, 1979.

[5] E.J. McGrath and T. Whitney, "Design Integrity and immunity checking: a new look at layout verification and design rule checking," Proc. 17th DAC, pp. 263-268, 1980.

[6] U. Lauther, "An O(n log n) algorithm for boolean mask operations," Proc. 18th DAC, pp. 555-562, 1981.

[6] J.A. Wilmore, "Efficient Boolean Operations on IC Masks," Proc. 18ty DAC, pp. 571-579, 1981.

[8] A. Tsukizoe et al., "MACH: a high-hitting pattern checker for VLSI mask data," Proc. 20th DAC, pp. 726-731, 1983.

[9] M. Sato, "A minimum width interval algorithm for polygonal regions (in Japaness)," Institute of Electronics and Communications Engineering of Japan, Report CAS84-116, pp. 5-11, 1984.

[10] M. Sato and T. Ohtsuki, "An O (n log n) algorithm for LSI layout resizing problems," Proc. of Internat. Symp. on Circuits and Systems, pp. 25-28, 1985.

[11] J.C. Jeong, J.B. Kim, M. Sato and T. Ohtsuki, "A fast minimum width/space verification algorithm (in Japaness)," Institute of Electronics, Informations and Communications Engineering of Japan, Report CAS87-16, pp. 39-36, 1987.

[12] A.V. Aho, J.E. Hopcroft and J.D. Ullman, The Design and Analysis of Computer Algorithms, Addison Wesley, Reading, Mass., 1974.

[13] J. Nievergelt and F. Preparata, "Plane-sweep algorithms for intersecting geometric figures," Comm. of ACM, pp. 739-747, vol. 25, no. 10, 1982.

[14] M.I. Shamos and D. Hoey, "Closest point problems," *Proc. 16th Annual IEEE Symp. on Foundations of Computer Science*, pp. 151-162, 1975.

[15] J.L. Bentley and H.A. Maurer, "Efficient Worse case data structures for range searching," acta Informatica 13, pp. 155-168, 1980.

[16] B. Chazelle and H. Edelsbrunner, "Optimal Solutions for a class of point retrieval problems," Brown University, Dept. of Computer Science, Technical Report no. CS-84-16, July 1984.

[17] F.P. Preparata and M.I. Shamos, Computational Geometry: An Introduction, Springer-Verlag, 1975. *

著 者 紹 介

鄭 子 春(正會員)

1960年 2月 8日生. 1983年 고려대학교 전기공학과 졸업 공학사학위 취득. 1986年 한국과학기술원 전기 및 전자공학과 졸업 공학석사학위 취득. 1986年～현재 한국전자통신연구소 근무. 1986年～1987年 Waseda대학 연구원. 주관심분야는 VLSI CAD 알고리듬 개발 및 Computational Geometry 등임.

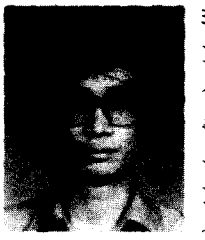
●

申 成 勇(正會員)

1947年 9月 1日生. 1970年 한양대학교 산업공학과 졸업. 1970年～1978年 럭키금성 및 삼성그룹 근무. 1978年～1981年 한국전자기술연구소 선임연구원. 1981年～1986年 미국 미쉬간대학 석사 및 박사학위 취득. 1987年～현재 한국과학기술대학 전자전산학부 조교수. 주관심분야는 알고리듬설계 및 분석, 컴퓨터그래픽스, 계산기하학 및CAD/CAM 등임.

●

李 哲 東 (正會員) 第 26卷 第 4號 參照
현재 전자통신연구소 자동설계기기연구실 실장



李 玄 燦(正會員)

1956年 5月 5日生. 1978年 2月 서울대학교 산업공학과 학사학위 취득. 1980年 2月 한국과학기술원 산업공학과 석사학위 취득. 1980年 3月～1983年 6月 (주)부산파이프 기획실. 1988年 4月 Michigan대 산업공학과 박사학위 취득. 1988年 9月～현재 한국전자통신연구소 자동설계기기연구실 선임연구원. 주관심분야는 Design Automation, CAD/CAM Computer Graphics, Geometric Modeling 등임.

●

柳 瑛 昱 (正會員) 第26卷 第 5號 參照
현재 벨리드 로직 시스템즈 코리아 지사장