F.Dehne, "Computational Geometry and VLSI", 1987 CompEuro Conference, Hamburg, W.-Germany, May 11-15, 1987, pp. 870-875

COMPUTATIONAL GEOMETRY AND VLSI

Frank Dehne
School of Computer Science, Carleton University
Ottawa, Canada K1S 5B6

Abstract

In recent years the design of efficient parallel solutions for Computational Geometry (CG) and closely related problems have been studied on the One-Dimensional Array of Processors, Mesh-of-Processors, Mesh-Connected Computer with Broadcasting, D-Dimensional Hypercube, Cube-Connected Cyrcles, Pyramid Computer, CREW-PRAM, etc. .

From these architectures the One- and Two-Dimensional Array of Processors can be considered those ones which yield the most straight forward VLSI implementation. Since CG methods are powerfull tools for CAD, graphics, image processing, etc. , the design of efficient parallel CG algorithms for these VLSI architectures is of importance for the development of e.g. high performance parallel CAD online applications or graphics processors.

This motivated research in design of efficient algorithms for CG problems on One- and Two-Dimensional Arrays of processors during the last few years.

Research work in this field is far from being finished. As it can be easily seen from e.g [LP84] or [PS85] most of the CG problems which have already been considered under a sequential model of computation have not been studied or implemented on a One- or Two-Dimensional Array of Processors, yet.

This paper surveys results which have been presented so far, gives examples to outline algorithm design methodologies, and points out open problems and directions of further research.

1. Introduction

"Computational Geometry, as it stands nowadays, is concerned with the computational complexity of geometric problems within the framework of analysis of algorithms" [LP84].

This young discipline which was christened by M.I.Shamos in his Ph.D. thesis [Sh78] in 1978 has attracted enourmous research interest in the past decade. A survey of Lee and Preparata in 1984 [LP84] contains already about 350 references to the most important publications in this area, a first introductory textbook on Computational Geometry (CG) written by Preparata and Shamos [PS85] was published in fall 1985, and since summer 1985 an annual Conference on Computational Geometry is organized by ACM SIGGRAPH.

Since CG methods are powerful tools for CAD, graphics, image processing, and robotics the design of efficient parallel CG algorithms for parallel machines and VLSI architectures is of importance for e.g. the development of high performance parallel CAD online applications or graphics processors.

To support the development of such fast parallel systems the design of efficient parallel solutions for CG and closely related problems has been studied in recent years by many researchers on several parallel models of computation:

- One-Dimensional Array of Processors (e.g.[Ch84], [De85b], [De86c], [SSP86]),
- Mesh-of-Processors (e.g.[MS84], [MS85], [De85a], [De86a], [De86c], [DSS86], [Lu86], [LV86]),
- Mesh-Connected Computer with Broadcasting (e.g.[St83], [NS81]),
- Mesh of Trees (e.g.[KE86], [St85]),
- D-Dimensional Hypercube (e.g.[NS80]),
- Cube-Connected Circles (e.g.[PV81]),
- CREW-PRAM (e.g. [ACGDY85], [AG85], [AG86a,b], [El86]),

etc.

From these architectures the One- and Two-Dimensional Array of Processors can be seen as those ones which yield the most straight forward VLSI implementation.

2. One- and Two-Dimensional Processor Arrays

One- and Two-Dimensional Processor Arrays are sets of n syncronized processing elements (PEs) arranged in linear order or on a $\sqrt{n} \times \sqrt{n}$ grid, respectively, with each processor being connect by bidirectional communication links to its direct neighbors (see figure 1).

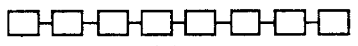
Each processor has a constant number of registers and within one time unit it can simultaneously send an output to and receive an input from each of its communication links.

How are geometric objects stored and manipulated on such a processor array ?

In the recent literature, the following two models have been considered:

(a) object sets : each processor of the array stores a constant number of geometric objects (e.g. points, rectangles, circles) which need O(1) storage space.

(b) digitized pictures : the geometric objects are represented by sets of points located on a finite grid where each grid point (pixel) is represented by one processor.


(a)

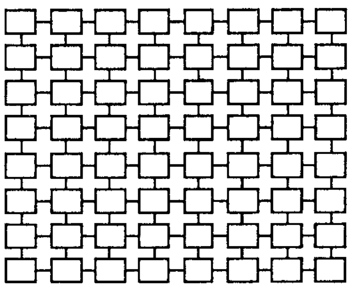

(b)

Figure 1:
(a) One- and (b) Two-Dimensional Processor Array

3. VLSI Algorithms for Sets of Geometric Objects

Several geometric problems have been studied for efficient parallel solutions on One- and Two-Dimensional Processor Arrays.

Most of these algorithms assume that initially each processor stores one (or a constant number of) geometric object(s). Consider, e.g., the problem involveing a set of line segments, then the coordinates of the two endpoints of each line segment are stored in one processor. Hence, storing n line segments involves n processors.

The first part of this section introduces some (selected) geometric problems which have been solved on processor arrays, yet. Subsection 3.2 scetches a parallel solution for a CG problems to outline some algorithm design methodologies.

3.1. Some (Selected) Geometric Problems

We will now review some interesting geometric problems which have been solved on processor arrays, yet. References to publications which introduce new parallel algorithms for several other geometric problems are given in the refence list.

One of the classical problems in Computational Geometry with a wide range of applications (especially in image processing) is the convex hull problem, cf. [PS85].

Given a set of n points in the Euclidean plane, the convex hull of these points is the smallest enclosing convex polygon (see figure 2a). The sequential time complexity of this problem is O(n log n).

The convex hull problem was one of the first to be solved on parallel on processor arrays. Chazelle [Ch84] and Miller/Stout [MS84] introduced O(n) and O(√n) algorithms for solving this problem on one- and two-dimensional processor arrays, respectively.

Since comparing the contents of two arbitrary PEs takes at least time O(n) and O(√n), respectively, in the worst case, these algorithms are asymptotically optimal.

Rectangle Intersection Problems are of special interest in VLSI design; see e.g. [MC80].

In [LV86] O(√n) time (optimal) solutions for solving the following problems for a set of n iso-oriented rectangles on a Mesh of Processors are given:

- Computation of the area of the logic "AND" ("OR"), i.e., the area of the region that is covered by two or more (at least one, respectively) rectangles; see figure 2b.
- Computation of the largest number of rectangles that overlap
- The fixed rectangle placement problem, i.e., determine the placement of a given rectangle such that it includes the maximum number of a given set of points.

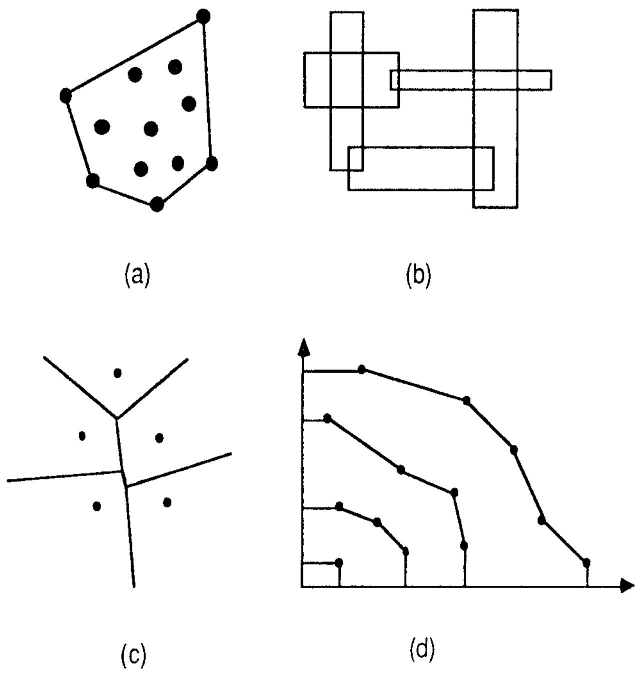


Figure 2 : Some Geometric Problems

The Voronoi Diagram of a set of n sites in the Euclidean plane subdivides the plane into n regions, one for each site, which are the sets of those points which have the respective site as their closest neighbor (see figure 2c).

Voronoi Diagrams are of central role in CG and yield a large number of very interesting and efficient applications.

In e.g. [De86b] efficient O(n log n) [O(n $\log^2 n$), respectively] sequential algorithms for clustering set of n points, line segments, rectangles etc. are presented which are of considerable interest in picture processing and have signifficant advantages with respect to other existing clustering algorithms. These algorithms utilize Voronoi Diagrams for sets of points [and generalized versions for line segments, rectangles , resp.].

To parallelize these algorithms it is important to find efficient parallel algorithms for the computation of Voronoi Diagrams.

In [Lu86] an O(√n log n) algorithm for constructing the Voronoi Diagram of a set of n points (in the Euclidean plane) on an √n x √n mesh is given which makes use of the well known mapping of this problem into computation of the convex hull in three dimensions.

The sequential time complexity of this problem is O(n log n). It is an open problem to find an optimal O(√n) algorithm for the parallel computation of Voronoi Diagrams on a Mesh-of-Procesors of size n.

In a "rectilinear world" (e.g. on a grid) the shape of a set of points is usually determined by their rectilinear convex hull, i.e. the smallest enclosing rectilinear convex polygon (see [PS85]). This problem can be reduced to finding the maximal elements of a set of points, i.e., those points, which are not dominated by any other point (see figure 2d).

An interesting way to represent such a set of points is to compute all its layers in the following sense (see [OL81]) : remove the points on the rectilinear convex hull and compute the hull of the remaining points; iterate this process until all points have been removed - this process is called "peeling". Solving this problem can be easily reduced to the ECDF searching problem : remove the maximal elements, find the maximal elements of the remaining points, etc. (see figure 2d).

Both problems can be solved on a Mesh of Processors in (optimal) time $O(\sqrt{n})$ as presented in [De86a].

Another interesting geometric problem, the largest empty rectangle problem, is covered in more detail in the following section.

3.2. An Example: The Largest Empty Rectangle Problem

Given a rectangle R (with its edges parallel to the coordinate axes) containing a set $S=\{s_1,...,s_n\}$ in the Euclidean plane consider the problem of finding the largest area subrectangle r of R with sides parallel to the coordinate axes that contains no point of S. In this section we will scetch optimal parallel O(n) and $O(\sqrt{n})$ time, respectively, algorithms for solving this problem on a One- and Two- Dimensional Array of Processors which have been presented in [De86c]. Since comparing two arbitrary elements takes at least time O(n) and $O(\sqrt{n})$, respectively, in the worst case, these algorithms are asymptotically optimal.

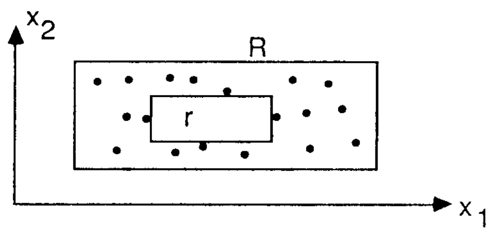


Figure 3: Definition of the Largest Empty Rectangle

An efficient solution for this problem is of considerable interest e.g. in VLSI manufactering. A rectangular silicon wafer (or rectangular part of a circular silicon wafer, respectively) with several points of impurity can be represented by a rectangle R and a point set S. The largest area rectangular area on the wafer, with its sides parallel to those of the wafer, which is free of impurities is the laregst empty rectangle r described above.

Several sequential algorithms have been presented e.g. in [NHL84], [CDL84], [KRS85] to solve this problem in time $O(n^2)$, $O(n \log^3 n)$, and $O(n \log^5 n)$, respectively.

Initially, the coordinates of each point of S are stored in an arbitrary PE. Furthermore, each PE contains the coordinates of R.

Each edge of the largest empty rectangle r is supported by either an edge of the bounding rectangle R or at least one point of S; otherwise it would be contained in a larger empty rectangle (cf. [CDL84]). We shall call these edges or points "supporting elements" with respect to S (and R). To simplify exposition we assume that all points of S have distinct x_1- and distinct x_2-coordinates and, thus, the largest empty rectangle has exactly four supporting elements. The existance of some more supporting elements will not change the algorithms signifficantly.

In order to compute the largest empty rectangle r of a set S with bounding rectangle R, S is first sorted by x_1-coordinate such that S (and R) can be split by a vertical line l_v into two subsets S_{left} and S_{right} of equal size $(| |S_{left}| - |S_{right}| | \leq 1)$, and the subproblems for S_{left} and S_{right} (and the two respective subrectangles of R) are recursively solved in parallel on the left and right, respectively, half of the processor array (see figure 4a).

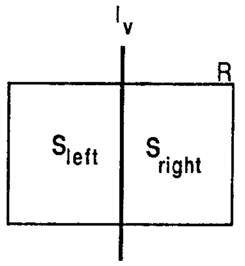


Figure 4a

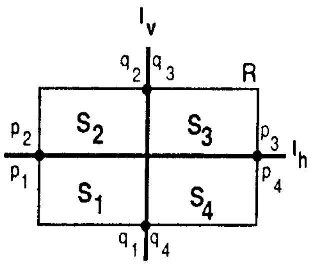


Figure 4b

Given the largest empty rectangles with respect to S_{left} and S_{right}, the maximum area one of these both has to be compared with the largest empty rectangle r^+ which has at least one supporting element with respect to S_{left} and S_{right}, each.

This "merging step" will be done by a second divide and conquer procedure:

S is sorted by x_2-coordinate and then split by an additional horizontal line l_h into four disjoint subsets S_1, S_2, S_3, S_4 as described in figure 4b such that

$S_{left} = S_1 \cup S_2$,

$S_{right} = S_3 \cup S_4$, and

$| |S_2 \cup S_3| - |S_1 \cup S_4| | \leq 1$.

Recursively, the largest empty rectangle having at least one supporting element with respect to S_2 and S_3, each, and none with respect to S_1 or S_4 as well as the largest empty rectangle having at least one supporting element with respect to S_1 and S_4, each, and none with respect to S_2 or S_3 is computed. Both subproblems are solved in parallel on one half of the processor array, each.

To compute r^+ we have to compare these two rectangles with the largest empty rectangle r^* which has the following property :

Let B_1 $[B_2, B_3, B_4]$ be the set of supporting elements of r^* with respect to S_1 $[S_2, S_3, S_4]$ or the respective part of the bounding rectangle R, then

$$|B_1| + |B_2| + |B_3| + |B_4| = 4$$
$$|B_1| + |B_2| > 0$$
$$|B_3| + |B_4| > 0$$
$$|B_2| + |B_3| > 0$$
$$|B_1| + |B_4| > 0 \ .$$

All possible (16) cases that match these requirements are listed in figure 4.

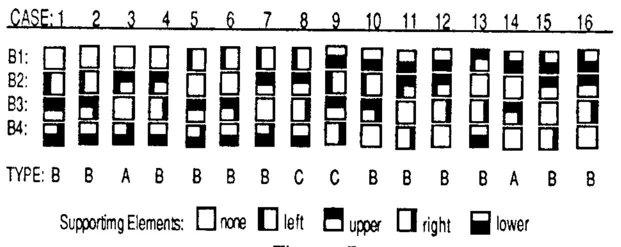

CASE: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
B1:
B2:
B3:
B4:
TYPE: B B A B B B B C C B B B B A B B
Supportimg Elements: none left upper right lower

Figure 5

Furthermore, it is easy to observe that, if r' is an empty rectangle as described above and $\{t_1, t_2\}=B_1$ are two supporting points in the same quadrant then t_1 and t_2 are maximal and close neighbors (i.e., there is no other maximal t' in B_1 with x_1 coordinate between t_1 and t_2) .

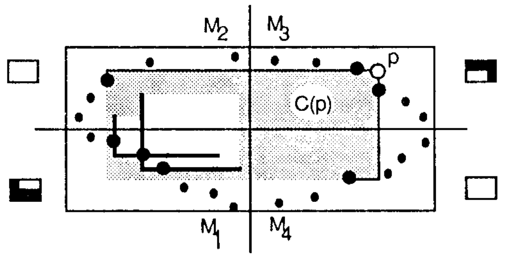
With this the general outline of the remaining part of the algorithm is as follows:

In order to find the rectangle r* the respective Processor Array computes the largest empty rectangle for each case (if it exists) separately, and then finds the maximum area one of these.

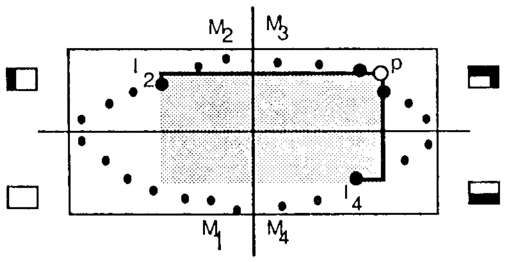
On a one-dimensional array of processors each case can be solved in time O(n) as scetched in figure 6 which yields a total O(n) running time for the entire algorithm, too.

However, on a mesh of processors, for several of these cases there is another divide-and-conquer procedure necessary to obtain an optimal O(√n) time complexity. A describtion of these steps is omitted here (cf.[De86c]).

Type A
(Case 14)



Type B
(Case 2)



Type C
(Case 9)

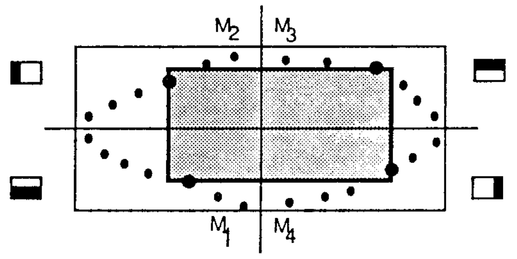


Figure 6

4. VLSI Algorithms for Digitized Pictures on a Mesh-Of-Processors

We will now study another way of representing geometric information on a two-dimensiona array of processors.

Each processor represents the center of a unit-area pixel. We refer to such a configuration as a systolic screen. An image on a systolic screen is a subset of pixels (see figure 7). Pixels that are part of some image are called occupied pixels.

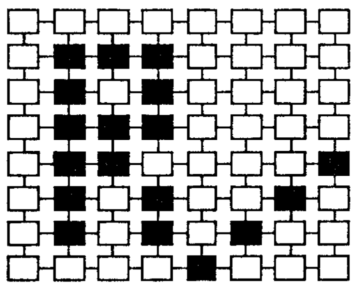


Figure 7

Two-dimensional arrays of processors have long been proposed for image processing [Re84] since they are a very natural way of storing images. The maximum size of such images typically ranges from 256x256 pixels in computer vision in the industrial environment to 4000x4000 pixels and larger for aerial photographs.

The well known MPP designed by NASA for analysing LANDSAT satellite data consists of 16384 PEs organized in a 128x128 matrix with a local memory between 1K and 16K bits for each PE (to represent a subsquare of pixels).

In image processing, however, processor arrays are mostly used for low level local operations such as image restoration, noise removal, edge detection etc. .

4.1. Some Computational Geometry Problems

Recently, processor arrays for digitized pictures have also been proposed as a machine model for Computational Geometry.

Miller and Stout ([MS85]) introduced O(√n) algorithms for computing the distance between two images, extreme points (with respect to the convex hull), diameter, and smallest enclosing circle as well as for convexity and separability testing and related problem.

[DSS86] presented O(√n) algorithms for computing hulls and contours of images as well as peeling images (see section 3.1). These algorithms do also result in a new parallel solution for the longest common subsequence problem.

4.2. An Example: Parallel Visibility

We will now scetch an algorithm on a systolic screen which computes the parallel visibility from a point light source located at infinity to a set of images represented on an $\sqrt{n} \times \sqrt{n}$ mesh-connected computer. The algorithm which is presented in [DHSS87] computes in $O(\sqrt{n})$ time the portion of each pixel illuminated by rays parallel in direction d.

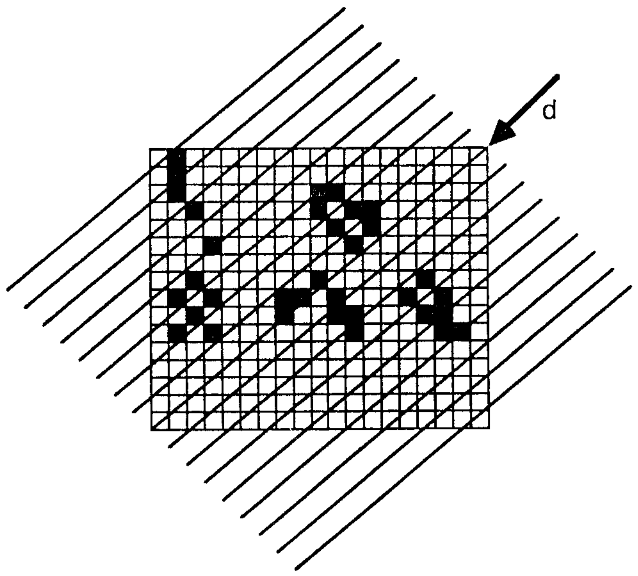


Figure 8

The technique used is to divide the screen into fixed-size strips whose edges are parallel to direction d. The visibility is then computed for each strip independently (in parallel). Inside each strip the visibility is affected only by the occupied pixels that intersect that strip. The visibility information for a strip is represented as an interval which is perpendicular to direction d and is initially set to the width of the strip. The interval indicates which parts of the strip have not yet been hidden from the light source by an occupied pixel. Initially the entire strip is visible. The interval is passed along the strip in direction d, and is updated when an occupied pixel is encountered: the maximum intersection between the pixel and the interval, if the interval were moved across the pixel in direction d, is removed.

5. Conclusion, Open Problems

The aim of this paper is to point out a new field of VLSI algorithm design : Parallel Computational Geometry.

The interaction between parallel Computational Geometry and VLSI is twofold:
On one hand, parallel Computational Geometry utilizes new machine architectures based on VLSI technology; on the other hand, parallel solutions for geometric problems are powerfull tools for VLSI design and manufactoring (consider e.g. the largest empty rectangle or rectangle intersection algorithms) and can speed up these processes signifficantly.

However, research in VLSI algorithms for Computational Geometry has just started solving a handfull of geomtric problems and there is much work ahead.

A large number of problems hasn't even been considered yet (cf. e.g. [LP84]).

Most of the algorithms have not been implemented or simulated and practical issues such as efficient I/O or problem sizes which are larger than the array size are unresolved.

References

[ACGDY85] A.Aggarwal, B.Chazelle, L.Guibas, C.O.Dunlaing, C.Yap, "Parallel Computational Geometry", Proc. IEEE Symp. on Found. of Computer Science, Portland, Oregon, Oct. 1985

[AG85] M.J. Atallah, M.T. Goodrich, "Efficient Parallel Solutions to Geometric Problems" , Report CSD-TR-504, Purdue Univ., March 1985

[AG86a] M.J.Atallah, M.T.Goodrich, "Efficient Plane Sweeping in Parallel (Preliminary Version)", Proceedings of the Second ACM SIGGRAPH Symposium on Computational Geometry, Yorktown Heights, NY, June 2-4, 1986, pp. 216-225

[AG86b] M.J.Atallah, M.T.Goodrich, "Parallel Algorithms for Some Functions of Two Convex Polygons", to appear in Proceedings of the 24th Annual Allerton Conference on Communication, Control and Computing, Monticello, Ill., Oct. 1-3, 1986

[Ch84] B.M. Chazelle, "Computational Geometryu on a Systolic Chip", IEEE Trans. on Computers, Vol. C-33, No. 9, Sept. 1984, pp.774-785

[De85a] F.Dehne, "Solving Geometric Problems on Mesh-Connected and One-Dimensional Processor Arrays", Proceedings of the 11th International Workshop on Graphtheoretic Concepts in Computer Science (WG'85), June 18-21, 1985, Castle Schwanberg, Wuerzburg, W.-Germany, Trauner Verlag 1985, pp. 43-60

[De85b] F.Dehne, "A One Dimensional Systolic Array for the Largest Empty Rectangle Problem", Proceedings of the 23rd Annual Allerton Conference on Communication, Control and Computing, Monticello, Ill., Oct. 2-4, 1985, pp.43-59

[De86a] F.Dehne, "$O(n^{1/2})$ Algorithms for the Maximal Elements and ECDF Searching Problem on a Mesh-Connected Parallel Computer", Information Processing Letters 22 (1986), pp.303-306, May 1986

[De86b] F.Dehne, "Optical Clustering", The Visual Computer (1986) 2, pp. 39-43, Springer 1986

[De86c] F.Dehne, "Parallel Computational Geometry and Clustering Methods", Ph.D. thesis, Univ. of Wuerzburg, W.-Germany, 1986

[DHSS87] F.Dehne, A.Hasenklover, J.-R. Sack, N.Santoro, "Computing Parallel Visibility on a Systolic Screen", Tech. Rep., Carleton Univ., Ottawa, 1987

[DN86] F.Dehne, H.Noltemeier,"Clustering Methods for Geometric Objects and Applications to Design Problems", The Visual Computer (1986) 2, pp. 31-38, Springer 1986

[DS86a] F.Dehne, J.-R.Sack, "Separability of Sets of Polygons", Proceedings of the 12th International Workshop on Graphtheoretic Concepts in Computer Science (WG'86), June 17-19, 1986, Bernried, W.-Germany, to appear in Lecture Notes in Computer Science, Springer 1986

[DS86b] F.Dehne, J.-R.Sack, "Separability of Sets of Polygons", Techn. Report SCS-TR-82, School of Computer Science, Carleton University, Ottawa, August 1986

[DSS86] F.Dehne, J.-R. Sack, N.Santoro, "Computing on a Systolic Screen: Hulls, Contours and Applications", Techn. Report SCS-Tech.Rep. , School of Computer Science, Carleton University, Ottawa, 1986

[El86] H.ElGindy, "A Parallel Algorithm for Triangulating Simplicial Point Sets in Space with Optimal Spee-up", to appear in Proceedings of the 24th Annual Allerton Conference on Communication, Control and Computing, Monticello, Ill., Oct. 1-3, 1986

[Hi85] W.D.Hillis, "The Connection Machine", The MIT Press 1985

[HMS86] J.P.Hayes, T.N.Mudge, Q.F.Stout, "Architecture of a Hypecube Supercomputer", Proceedings of the 1986 International Conference on Parallel Processing, St.Charles, Ill., Aug. 19-22, 1986, pp.

[KE86] V.Kumar, M.Eshaghian, "Parallel Geometric Algorithms for Digitized Pictures on Mesh of Trees", Proceedings of the 1986 International Conference on Parallel Processing, St.Charles, Ill., Aug. 19-22, 1986, pp. 270-273

[LP84] D.T.Lee, F.P.Preparata, "Computational Geometry - A Survey", IEEE Trans. on Computers, Vol. C-33, No. 12, Dec. 1984, pp. 1072-1101

[Lu86] M.Lu, "Constructing the Voronoi Diagram on a Mesh-Connected Computer", Proceedings of the 1986 International Conference on Parallel Processing, St.Charles, Ill., Aug. 19-22, 1986, pp. 806-811

[LV85] M.Lu, P.Varman, "Solving Geometric Proximity Problems on Mesh-Connected Computers", Proc. of the 1985 IEEE Computer Society Workshop on Computer Architecture for Pattern Analysis and Image Database Management, pp.249-255, Nov. 1985

[LV86] M.Lu, P.Varman, "Mesh-Connected Computer Algorithms for Rectangle Intersection Problems", Proceedings of the 1986 International Conference on Parallel Processing, St.Charles, Ill., Aug. 19-22, 1986, pp. 301-307

[MB86] S.Manohar, G.Baudet, "VLSI Supercomputing", to appear in Proceedings of the 24th Annual Allerton Conference on Communication, Control and Computing, Monticello, Ill., Oct. 1-3, 1986

[MC80] C.Mead, L.Conway, "Introduction to VLSI-Systems", Addison-Wesley, Reading, MA, 1980

[MS84] R. Miller, Q.F. Stout, "Computational Geometry on a Mesh-Connected Computer", Proc. Int. Conf. on Parallel Processing, 1984

[MS85] R. Miller, Q.F. Stout, "Geomtric Algorithms for Digitized Pictures on a Mesh-Connected Computer", IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. PAMI-7, No. 2, March 1985, pp 216-228

[NS80] D.Nassimi, S.Sahni, "Finding Connected Components and Connected Ones on a Mesh-Connected Parallel Computer, SIAM J. COMPUT, Vol.9, No.4, Nov. 1980, pp.744-767

[NS81] D. Nassimi, S. Sahni, "Data Broadcasting in SIMD Computers", IEEE Trans. on Computers, Vol. C-30, No. 2, Feb. 1981, pp. 101-106

[OL81] M.H.Overmars, J.v.Leeuven, "Maintenance of Configurations in the Plane", Report RUU-CS-81-3, Dept. of Computer Science, Univ. of Utrecht, 1981

[PS85] F.P.Preparata, M.I.Shamos, "Computational Geometry, An Introduction", Springer 1985

[PV81] F.P.Preparata, J.Vuillemain, "The Cube-Connected Circles: A Vertasile Network For Parallel Computation", Comm. ACM, Vol.24, No.4, May 1981

[Re84] A.P.Reeves, "Survey, Parallel Computer Architectures for Image Processing", Computer Vision, Graphics, and Image Processing 25, 1984

[SSP86] C.D.Savage, M.Stallmann, J.E.Perry, "Solving Compinatorial Problems on Arrays with One-Way Dataflow", to appear in Proceedings of the 24th Annual Allerton Conference on Communication, Control and Computing, Monticello, Ill., Oct. 1-3, 1986

[St83] Q.F. Stout,Mesh-Connected Computers wit Broadcasting , IEEE Trans. on Computers, Vol. C-32, No. 9, Sept. 1983, pp826-830

[St85] Q.F.Stout, "Pyramid Computer Solutions for the Closest Pair Problem", Journal of Algorithms, No.6, 1985, pp. 200-212

PROCEEDINGS

Edited by Walter E. Proebster and Hans Reiner

VLSI and Computers

First International Conference on Computer Technology, Systems and Applications

Hamburg · May 11–15, 1987

Topic 1: The Impact of VLSI on Computers
Co-editor: Robert Latin

Topic 2: The Influence of Computers on VLSI
Co-editor: Joachim Mucha

Topic 3: Microelectronics and VLSI: Status and Trends
Co-editor: Hugo Rüchardt

Topic 4: Computer-Systems: Status and Trends
Co-editor: D. C. J. Poortvliet

Topic 5: Designing Systems with VLSI Today
Co-editor: Ernst Rothauser

COMP EURO 87

SAN 264-620X
ISBN 0-8186-0773-4
IEEE Catalog Number 87CH2417-4
Computer Society Order Number 773
Library of Congress Number 86-83415

Sponsored by THE INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS INC · IEEE COMPUTER SOCIETY · GESELLSCHAFT FUR INFORMATIK · VERBAND DEUTSCHER ELEKTROTECHNIKER