J. Indian Inst. Sci., Nov.–Dec. 1987, 67, pp. 423–428.

VLSI architecture for the computation of the diameter and the closest points of a set*

GOPAL PANNEERSELVAM
Department of Electrical Engineering, The University of Manitoba, Winnipeg, Canada R3T 2N2.

Abstract

VLSI architecture using systolic arrays for the computation of the diameter (maximum distance between two points) and the closest points (minimum distance between two points) for a set of points is developed and presented. An optimal time complexity for these computations irrespective of the number of points in the set and their dimensionality has been achieved.

Key words: Diameter and closest pair of points of the set, pseudo-matrix multiplication, systolic architecture.

1. Introduction

The diameter (maximum distance between any two points) and the closest points (minimum distance between any two points) in a set of points have many applications in pattern recognition and computational geometry[1,2]. The diameter and the closest points of a set of N points are defined as follows:

$$\text{Diameter } (S) = \max_{i,j} \{d(P^i, P^j)\}$$

$$\text{Closest points } (S) = \min_{k,l} \{d(P^k, P^l)\}$$

where d denotes the distance measure in L_P metric and $1 \leq i, j, l, k \leq N$.

The problem is to select two pairs of points in the set of N points with a pair having a maximum distance between them and the other a minimum. The straight forward way of solving this problem is to compute the distances between all the pair of points in the set and then select the maximum and minimum distances which in turn give a pair of points corresponding to the diameter and the closest points respectively. The distance computation increases rapidly with the number of points in the set (N) and their dimensionality (D). The complexity of this problem is $O(N^2D)$ for distance computations and $O(N^2)$ for comparisons. Since the computation is time consuming, many researchers have developed[1,2] more efficient algorithms only for $D = 2$. There is no faster algorithm available to date for $D > 2$. To achieve an optimal time complexity for $D \geq 2$, VLSI architecture is developed and implemented.

*First presented at the Platinum Jubilee Conference on Systems and Signal Processing held at the Indian Institute of Science, Bangalore, India, during December 11–13, 1986.

2. Systolic arrays implementation

Consider a set $S = \{P^1, P^2, \ldots P^i, \ldots P^N\}$ consisting of N points. Each point in the D dimensional space can be represented as a vector, $P^i = \{a_1^i, a_2^i, \ldots a_D^i\}$ for $1 \le i \le N$. To select the maximum and minimum distances between any two points, one has to calculate the distance between each point and rest of the points in the set, *i.e.*, P^1 to P^2, P^3, $\ldots$, P^N, P^2 to P^3, P^4, $\ldots P^N$, $\ldots$ P^i to P^{i+1}, P^{i+2}, $\ldots$ P^N, $\ldots$ and then the maximum and minimum values of the distances are identified with respect to their pair of points.

These distance computations can be formulated as a pseudo-matrix multiplication as described by Liu and Fu[3], and Panneerselvam[4], given as $T = S \otimes S^T$. This pseudo-matrix multiplication is expounded in fig. 1, where S contains all the N points and forms as a matrix of dimension $N \times D$, and T is the resultant matrix of order $N \times N$ due to the pseudo-matrix multiplication. The element b in T represents the distance between the points i and j or j and i. The diagonal elements in the matrix represent the distance between the same points which is zero; therefore, the dimension can be reduced as $N-1$ instead of N. It is also evident that T is a symmetric matrix and there are $(N-1)$, $(N-2)$, $\ldots$ 1 elements in T.

The distance between any two points in L_p metric is given as follows:

$$d_p(P^i, P^j) = \{|a_1^i - a_1^j|^p + |a_2^i - a_2^j|^p + \ldots |a_D^i - a_D^j|^p\}^{1/p}$$

for $p = 1, 2, \ldots$.

$$d_\infty(P^i, P^j) = \max\{|a_1^i - a_1^j|, |a_2^i - a_2^j|, \ldots |a_D^i - a_D^j|\}.$$

Once the distances between every point and rest of the points are computed, one has to identify the largest and the smallest (> 0) distances with respect to their pair of points. To select the pair of points in the set, the distances have to be labelled, as shown in fig. 1. For example, the ith row of the matrix T, *i.e.*, $b_i^1, b_i^2, \ldots b_i^j, \ldots b_i^N$ can be labelled as $c_i^1, c_i^2, \ldots c_i^j, \ldots c_i^j, \ldots c_i^N$, where c_i^j represents the points i, j having distance b_i^j, between them and $1 \le i, j \le N$.

This compute-bound problem can be solved as follows. First compute all the distances with respect to every point to rest of the points in the set, then label them. These operations require 'compute' processors. Next, compare all distances and select a subset T_1 which contains only $(n-1)$ maximum distances from T such that every element in the subset represents the largest element of $N-1$ rows. Simultaneously select another subset T_2 which contains only $(n-1)$ minimum distances. These comparisons require 'compare_max' and 'compare_min' processors for respective operations. Finally, choose a single element of the subsets, T_1 and T_2, which possess the maximum and the minimum values, respectively. To hold the maximum and minimum values, the 'compare_max_hold' and 'compare_min_hold' processors are used. Since the distances are identified along with their labels, one can easily identify the pair of points.

To increase the degree of concurrency in computations, the systolic system is designed. Each processor is square-shaped, that contains four inputs and four outputs. The data

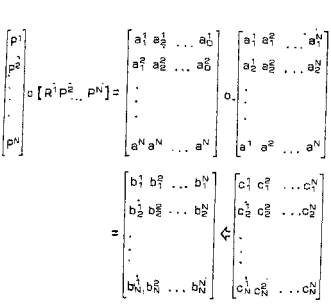

FIG. 1. The pseudo-matrix multiplication.

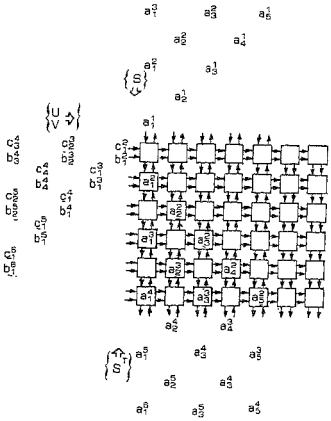

FIG. 2. The data movement in systolic arrays.

flow diagram for the systolic architecture is shown in fig. 2, for a set of points $N = 7$ and $D = 5$. As the diagonal elements in the matrix T are zeros, a two-dimensional network of $(N-1) \times D$ compute processors is sufficient to compute all the distances. The systolic architecture uses $N-2$ compare__max and compare__min processors. To hold the maximum and minimum distances, the last compare processors' design is modified with hold registers. As soon as the distance computations and comparisons are over, the compute__max__hold processor releases the maximum distance along its label. In a similar manner, the compare__min__hold processor releases the closest distance of the set along its label.

The data stream of S, *i.e.*, the data of the set $S = \{P^1, P^2, \ldots P^N\}$ enters from the top and moves down the network. While a copy of the same data stream in a different order, *i.e.*, $S_1 = \{P^2, P^3, \ldots P^N\}$, enters from the bottom of the network and flows up, at the same time, the distance components stream $U = \{b_1^1, b_2^2, \ldots b_N^1, \ldots b_N^1, b_N^2, \ldots b_N^N\}$ and the index components stream $V = \{c_1^1, c_2^1, \ldots c_1^N, \ldots c_N^1, c_N^2, \ldots C_N^N\}$ enter from the left side of the network and move towards right. The data circulation is controlled by the system clock and the processors' algorithms. The data streams S and S_1 move in directions opposite to each other. The distances and index components (U and V) move parallel to each other but orthogonal to the data streams of S and S_1. The four data streams are skewed as shown in fig. 2. This ensures that proper data reach the appropriate processors at the correct time. One can reduce the distance computations considerably, by utilizing the symmetric property of matrix T.

3. Design concept of processors

The systolic square array network contains $(N-1)*D$ compute processors, $(N-1)$ compare_max/compare_min processors, a compare_max_hold processor and a compare_min_hold processor. Each processor contains an ALU, a buffer register and four pipelines. Each pipeline is assigned for a particular data. The algorithmic principles of the processors are as follows.

The recurrence relation of the compute processors are given below.

$$
\begin{aligned}
b_i^j(1) &:= 0 \\
b_i^j(d+1) &:= b_i^j(d) + |a_{d+1}^j - a_{d+1}^j|^p \\
b_i^j &:= b_i^j(D+1) \\
a_d^l &:= a_d^j(t) \\
a_d^l &:= a_d^j(t) \\
c_i^j &:= c_i^j(t)
\end{aligned}
$$

where t represents the active time of the processor. The internal structure of the compute processor is exhibited in fig. 3.

The algorithmic principle of the compare_max processor, which compares two distances and selects the larger one, is as follows.

If $a \geq c$ then
$e := a;\ f := b;\ g := c;\ h := d.$
Else
$e := c;\ f := d;\ g := a;\ h := b.$

The block diagram of the compare_max processor is shown in fig. 4.

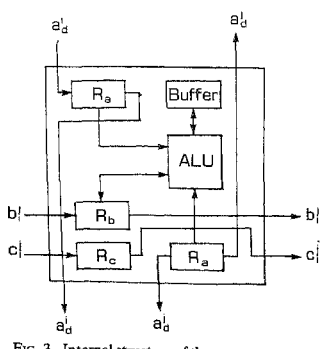

FIG. 3. Internal structure of the compute processor.

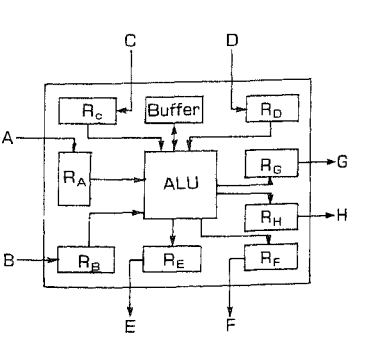

FIG. 4. Internal structure of the compare processor.

In a similar manner, the compare__min processor is designed to select the smallest distance from two distances. The compare__max__hold and compare__min__hold processors are modified designs of the compare__max and compare__min processor, respectively. These processors hold and release the maximum and the minimum distances by setting the end of execution flags.

The important micro operations are as follows:

1. Transfer the data serially into the registers either from memory or from the previous processors.
2. Do the following operations sequentially (as per the algorithms).

Compute processor	*Compare__max processor*
$Bu \longleftarrow a_d^l - a_d^l$	$Bu \longleftarrow a - c$
$Bu \longleftarrow \lvert Bu \rvert$	if $Bu \geq 0$ then
	$e \longleftarrow a;\ f \longleftarrow b;$
	$g \longleftarrow c;\ h \longleftarrow d;$
$b \longleftarrow b + Bu$	else
	$e \longleftarrow c;\ f \longleftarrow d;$
	$g \longleftarrow a;\ h \longleftarrow b;$

where Bu represents buffer register. The total time required for one operation includes n clock cycles to transfer one word of n bits length for step (1) and 3 clock cycles for step (2) *i.e.*, considering the longer time requirement of these operations. The processors are synchronized on the basis of time requirement of the compute processor. This $n+3$ clock cycles are assumed as unit time for single operation, for the case of $p = 1$. The design of compute processor will vary depending upon the metric chosen.

4. Conclusion

VLSI architecture for the computation of the diameter and the closest points of a set of points has been proposed. The overall computations require only $(3N+D-1)$ time units. In contrast, a uniprocessor requires $O(N^2D)$ computations and $O(N^2)$ comparisons for the minimum distance and $O(N^2)$ comparisons for the maximum distance. For example,if $N = 100$, $D = 5$ then a uniprocessor requires 50,000 computations and 20,000 comparisons as compared to 204 time units for the VLSI architecture. Also, by computing AT and AT^2 (area and time bounds) one can easily show that, the bounds are optimal. The concept of these problems can also be extended to the polygon problems, all nearest neighbors of the set, and problems of similar nature.

Acknowledgment

This research work is supported by the National Overseas Scholarship, Ministry of Home Affairs, Government of India, New Delhi, India.

References

1. Toussaint G. T. Pattern recognition and geometrical complexity. *Proc. 5th Int. Conf. Pattern Recognition*, 1980, pp. 1324–1347.

2. SHAMOS, M. I. *Computational geometry*, Ph.D. Thesis, Computer Science Department, Yale University, 1978.

3. LIU H. H. AND FU, K. S. VLSI architectures for minimum-distance classification, *Proc. IEEE Int. Conf. on Computer Design: VLSI in Computers*, Oct. 31–Nov. 3, 1983, pp. 549–552.

4. PANNEERSELVAM, G. VLSI architectures for the k-nearest neighbor problem, *Proc. Int. Conf. on Computers, System & Signal Processing*, Bangalore, India, December 10–12, 1984, pp. 1487–1490.

5. KUNG, H. T. AND LEISERSON, C. E. *Systolic arrays for VLSI: Introduction to VLSI system*, by C. A. Mead and L. A. Conway, Addison-Wesley, Reading, Mass. 1980.