M. Nussbaum*

RESUMEN

Se propone una arquitectura sistólica para el problema de ordenamiento. El circuito ordenador propuesto se basa en un algoritmo de enumeración, consistente en dos fases: 'ranking' y reordenamiento. 'Ranking' se realiza utilizando un arreglo sistólico, cuyo comportamiento AT^2 es $O(n^3)$. El reordenamiento es posible de realizar en tiempo constante ocupando una red $O(n^2)$, dando así un comportamiento area tiempo de $O(n^3)$.

ABSTRACT

A sistolic architecture for sorting is proposed. The sorting circuit is based on an enumeration algorithm, which consists of two phases: ranking and rearanging. Ranking is done using a sistolic array whose AT^2 is $O(n^3)$. Rearanging is possible to do in constant time using an $O(n^2)$ network. In this form the sorter area time behaviour is $O(n^3)$.

* Ingeniero Civil Electricista (PUC 1980); M.Sc. Information and Computer Science (GA TECH, 1984). Profesor Escuela de Ingeniería, Departamento Ciencia de la Computación (143), P. Universidad Católica de Chile, Casilla 6177, Santiago, Chile. Investigación financiada parcialmente por DIUC.

Sorting is one of the main activities in data processing. Any application that manages a file system entails sorting. Therefore, it is important to design a fast and cost effective system to handle sorting applications. Such a system could be used as a standalone, or as a special purpose processor in connection with a central processing unit.

In order to implement such a device, it is necessary to achieve a high degree of integration. A new generation of logic design has evolved with VLSI. This new technology makes possible the construction of a collection of processors organized in a regular topology in a single chip. What before was a set of chips with one processor per chip now includes many processors per chip [1].

Since a mapping of high level computations into hardware structures is possible [2], the VLSI approach could present a powerful yet cost effective solution to the sorting problem. However, a number of constraints must be considered. These constraints include limitations in the I/O bandwith, interprocessor communication, and data routing overhead.

A high degree of throughput can be achieved in sorting by introducing systolic architecture. As the name implies, in a systolic architecture input data streams traverse the processor array in a synchronous fashion, passing through different processing elements before reaching their output. A systolic architecture is easy to implement

not only because it has a regular computing pattern, but also because it is easy to reconfigure. A variety of outside constraints can be met, balancing the available I/O bandwith with the host computer. This permits a sustained continuous activity of the host and the coprocessor [3],[4].

The underlying algorithm used will decide the VLSI processor structure and the degree of parallelism and pipelinig. The amount of processing power one can put on a chip will depend (besides the technology) on the size of the processor and the amount of communication required between processors. Therefore, one must employ a simple algorithm that supports a high degree of concurency with minimum amounts of communication between processors.

From the study of the existing sorting algorithms, one finds that most algorithms have one or more of the following problems, making them unsuitable for VLSI implementation:

1. Difficulty in processor-memory comunication (memory conflicts) [8], [9].

2. High amount of computation required in each processor [5], [10], [11], [12]

3. High degree of communication between processors [13], [14].

There is an additional point in which most algorithms fail. The stable condition (records with equal keys retain their original relative order [15]) is difficult to achieve, and most algorithms do not treat this point or have to analyze it as a special case [5],[6],[16].

We present a VLSI sorting network that overcomes the problems discussed in the above lines. The proposed network is based on an enumeration sorting algorithm (records with equal keys retain their original relative order [15]), [1], [20], [21]. It consists of two phases: ranking and rearanging. Ranking consists of finding the rank of the elements in the group of numbers to be sorted. Rearanging places each key in its final position according to its rank.

Ranking is done using a systolic array. The area time complexity (AT^2) of the systolic array is $O(n^3)$ [18] using Thompson's model [5]. But Thompsons's modell does not take into account the complexity of the processing unit. In comparing two circuits, both having the same number of nodes, same number of steps to sort, and same degree of concurrency, the model will give the same area time complexity, even if the processing units in one circuit are P times more complex than those used in the other one. Different processing unit complexities yield different areas per node and different times per step (more processing time and more communication between processors), which in practical terms, lead to different area time behaviors. Therefore, further considerations are required to compare various sorting circuits at implementation level.

Rearanging may be done in constant time at the systolic array output, using an $O(n^2)$ network (e.g. crossbar switch). So the AT^2 of the sorting configuration will be $O(n^3)$.

Figure 1 shows the general scheme used for sorting. As a pipeline machine, series of sequences of numbers may be pumped at the input of the systolic ranking machine, while at its output the sequences appear with its sorted indexes. This information goes to the rearranger, i.e. an alignment network [19], which takes the indexes asociated to each number of the sequence, and put the number to its sorted place.

SYSTOLIC SORTER DESIGN

Consider a matrix of NxN processors for sorting N records. The records to be sorted are presented to the columns and the rows of processors. Each processor has a comparator performing the operation:

$$C_{ij} = (X_i \geq X_j)$$

So the rank of the i^{th} record, R_i, is defined as:

$$R_i = \sum_{j=1}^{N} C_{ij} \qquad (1)$$

Figure 2 shows the rank computation using the above definitios for N=4 and the sequence (6,2,4,3). A similar approach is presented by a

N^2-processor-Rank sort [5]. Both approaches, however, are not stable sorting algorithms. In this case records with identical key (i.e., identical rank) present problems in the rearanging process, because the final position of the record, uniquely determined by the rank of that record, will be shared by other records with the same rank (see Figure 3).

One simple solution to this problem is to use the fact that the lower triangle is the transpose of the upper triangle, in the way the operations are defined, i.e. $C_{ij}=C'_{ji}$. Also $C_{ii}=0$ for all i, so:

$$R_i = \sum_{j=1}^{i-1} C_{ij} + \sum_{j=i+1}^{N} C'_{ji} \qquad (2)$$

Hence stability is mantained without using special conditions over the ones defined for sorting, which is very difficult to achieve [6], [10]. The approach in which all processors are not performing the same operation presents difficulties, especially in VLSI implementation. For example [6] has to use control signals to tell the processors which one of the two operations to perform.

So it will be sufficient to use only the lower triangular part of the matrix of processors to calculate the rank (Figure 4). This approach may be easily implemented in VLSI. Based on the lower triangular matrix, a systolic array with triangular topology can be constructed. From eq.(2), the rank of the i^{th} record is the sum of two terms: one is the sum of the comparison results on the i^{th} row; and the other is the sum of the comparison results on the i^{th} column of the lower triangular matrix. Figure 5 shows a sorter of size 4. The inputs are records to be sorted

while the outputs are records with their corresponding rank. The inputs, with skewed pattern (Figure 5a), are fed into the sorter both horizontally and vertically. Furthermore, each input record has a counter associated with it. The counter is used to calculate the rank of the record. A counter, with initial value zero called horizontal counter H_i or vertical counter V_i, depending upon the corresponding record (X_i) is fed to the sorter horizontally or vertically.

From the above description it is easy to visualize that the hardware required to implement each processor is extremly simple.

There are two basic processing units. Units represented by the square (Figure 5a) have two inputs; one horizontal input record (X_i) with a horizontal counter (H_i) and a vertical input record (X_j) with a vertical counter (V_j). The processor performs the operation $C_{ij}=(X_i \geq X_j)$, which updates the horizontal counter. The complement of C_{ij}, C'_{ij}, is used to update the vertical counter (V_j). The horizontal output of a square unit is the horizontal record and its updated counter (H_{ij}). Similarly, the vertical output is the vertical record (X_j) and its updated counter (V_j). Figure 5c represents the logic design of the processor.

Units represented by the circle in Figure 5d perform the addition of the horizontal and vertical counter i, whose output is the rank of the record.

From the figures, it can be seen that each processor is independent of the other and that only a synchronization signal must go through all of them. this allows a systolic approach to the problem, which offers a high-degree of parallelism and pipelining. Figure 6 shows a pipelining aplication, where three sequences are sorted.

FINAL REMARKS

There are cases in which the sequence to be sorted is bigger than the capacity of the sorting chip, i.e. when m, the size of the capacity of the sorting circuit is less than n, the length of the sequence to be sorted. There are several approaches to this problem [18]. One of the ways to solve this problem is using the inherent modularity of the systolic architectures, i.e. extending the capacity of the sorter adding more sorting modules.

Figure 7a shows that any systolic sorter may be decomposed in a triangular and a square sorter. A triangular sorter is the one defined in the previous section, while a square sorter (Figure 7b) is the same one with the only difference that no additions are made between rows and columns. So the two basic construction blocks which may be rearranged as in Figure 7, may sort any sequence of numbers bigger than the basic triangular sorter size.

In order to sort sequences of n=k*m elements, it will be necessary k triangular sorting modules and k(k+1) square sorting modules. When only one sequence is to be sorted, less hardware is required if we want to

maximize the hardware utilization. In this case, only k-1 square sorters are required. This idea can be extended to a series of sequences p, which is smaller than n, the maximum concurency of the configuration. The number of square sorters required in this case is:

$$\begin{cases} k-1 & \text{when } \lceil p/m \rceil < k-1 \\ \lceil p/m \rceil * (\lceil p/m \rceil - 1)/2 & \text{when } \lceil p/m \rceil \geq k-1 \end{cases}$$

Besides the modularity, the significant features of the proposed sorting network are pipelining, a high degree of concurrency, simple processing units and a minimum amount of communication. This last point is of main importance in designing VLSI, since the numbers of layers achievable with current technology is limited.

The architectural approach for sorting presented here, i.e. a triangular systolic architecture has also been used for transitive closure [1], matrix triangularization [17], and two dimensional convolution [3]. These aplications suggest that a triangular topology may be used in a wide range of complex functions. The architecture we proposed, as being an extension of a pipeline and a parallel machine, has the following advantages [3]:

1. The design makes multiple use of each data element.
2. The design maximizes concurrency.
3. Information flows in a simple way through the machine, and control is simple.
4. There are only few basic simple processing cells.

12. D.S. Hirschberg, "Fast Parallel Sorting Algorithms", Communications of the ACM, Vol 21, No. 8, Aug 1978, pp 657-661.

13. I.S. Charnaya, "Sorting by Merging in a Parallel Computer", Automatika i Telemekhanika, No. 9, Sep 1981, pp 186-191.

14. C.D. Thompson and H.T. Kung, "Sorting on a Mesh Connected Parallel Computer", Communications of the ACM, Vol 20, No.4, April 1977.

15. D.E. Knuth, "The Art of Computer Programming, Vol 3: Sorting and Searching", Addison Wesley, Reading, Mass. ,1973.

16. K.E. Batcher, "Sorting Networks and their applications", Proc. AFIPS SJCC., Vol 32, 1968, pp 307-314.

17. W.M. Gentleman and H.T. Kung, "Matrix Triangularization by Systolic Arrays", SPIE, Real-Time Signal Processing IV, Vol 298, 1981, pp 19-26.

18. P.Y. Chen and M. Nussbaum, "Triangular Sorters: A VLSI Systolic Architecture for Sorting", Georgia Institute of Technology, Technical Report No. GIT-ICS 84/13.

19. D.J. Kuck, "The Structure of Computers and Computations", John Wiley & Sons, 1978.

20. D.E. Muller, F.P. Preparata, "Bounds to complexities of networks for sorting and for Switching", JACM, pp 195-201, 1975.

21. F.T. Leighton, "New lower bound techniques for VLSI", Proc. of IEEE FOCS Conf., pp 1-12, 1981.

1. J.D. Ullman, "Computational Aspects of VLSI", Computer Science Press, 1984.

2. Y.P. Chan and K.S. Fu, "Matching Parallel Algorithms and Architecture", 1983 International Conference on Parallel Processing, Aug 1983, pp 335-337.

3. H.T.Kung, "Why Systolic Architectures", IEEE Computer, Jan 1982 pp 37-46.

4. P.S. Liu and T.Y. Young "VLSI Arrray Design Under Constraint of Limited I/O Bandwith", IEEE Transactions on Computers, Vol c-32, No.12, Dec 1983, pp 1160-1170.

5. C.D. Thopson, "The VLSI Complexity of Sorting", IEEE Transactions on Computers, Vol. 32, No.12, Dec 1983, pp 1171-1184.

6. H. Yasura, N.Takagi and S. Yajima, "The Parallel Enumeration Sorting Scheme for VLSI", IEEE Transactions on Computers, Vol C-31, No.12, Dec. 1982, pp 1192-1201.

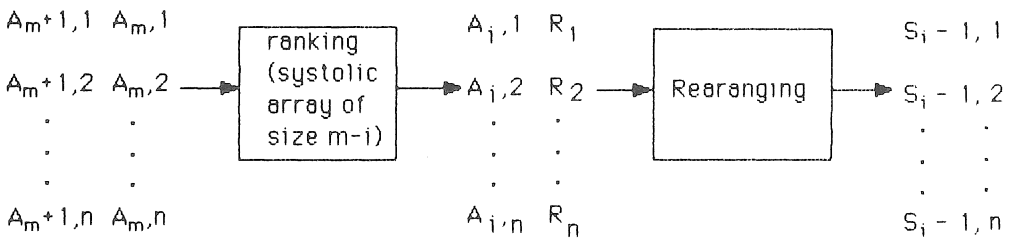
7. P.N. Armstrong and M. Rem, "A serial Sorting Machine", Comput and Elect. Eng. Vol 9, No.1, 1982, pp 53-58.

8. Y. Shiloach and U. Vishkin, "Finding the Maximum, Merging, and Sorting in a Parallel Computation Model", Journal of Algorithms, 2, 1981, pp 88-102.

9. C.P. Kruskal, "Searching, Merging and Sorting in Parallel Computations ", IEEE Transactions on Computers, Vol C-32, No.10, Oct. 1983, pp 942-946.

10. F.P. Preparata, "New Parallel-Sorting Schemes", IEEE Transactions on Computers, Vol C-27, No.7, July 1978, pp 669-673.

11. M. Ajtai, J. Komlos, E. Szemeredi, "An O(nlogn) Sorting Network", Proc 15th ACM Symp. on Theory of Computing, April 1983, pp 1-9.



Used Nomenclature:

$A_{i,j}$: Element j of secuence i to be sorted.
R_i : Rank of record i.
$S_{i,j}$: Element of jth position of the sorted sequence i.

Fig. 1 - General scheme used for sorting.

					Rank
6	1	1	1	1	4
2	0	1	0	0	1
4	0	1	1	1	3
3	0	1	0	1	2
	6	2	4	3	

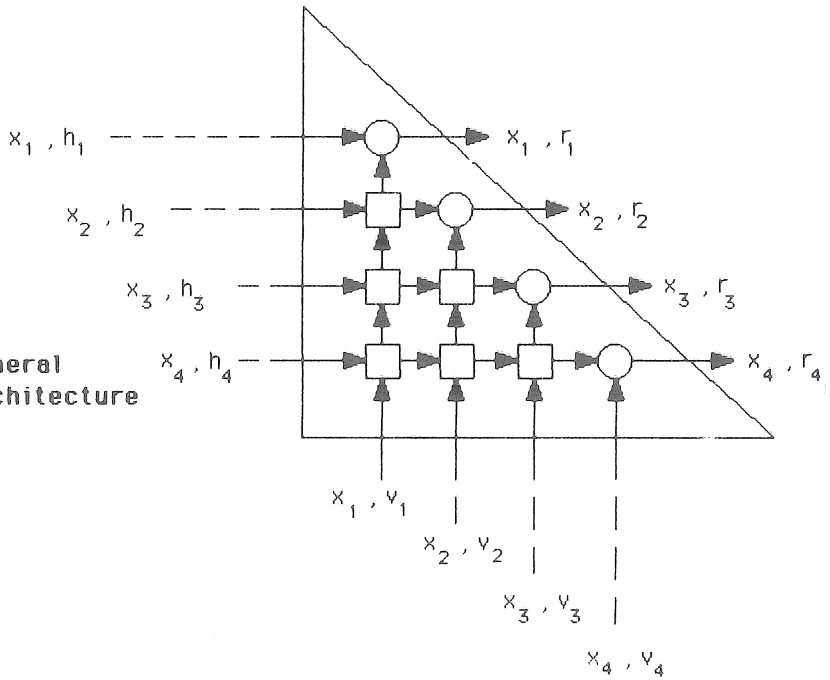
Fig. 2 - Rank Computation

					Rank
6	1	1	1	1	4
2	0	1	0	1	2
4	0	1	1	1	3
3	0	1	0	1	2
	6	3	4	3	

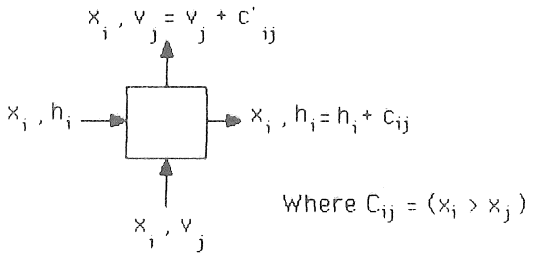
Fig. 3 - Rank Computation With Equal Keys.

6	—	—	—	—
3	0	—	—	—
4	0	1	—	—
3	0	0	0	—
	6	3	4	3

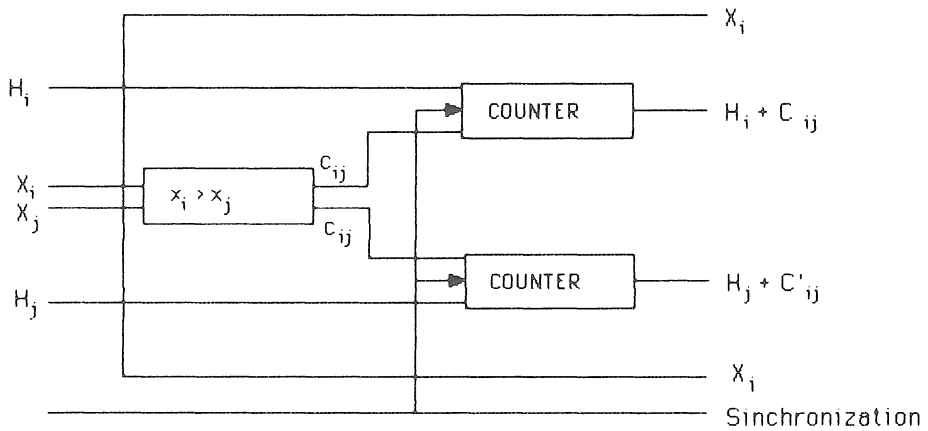
Fig. 4 - Modified Rank Computation.



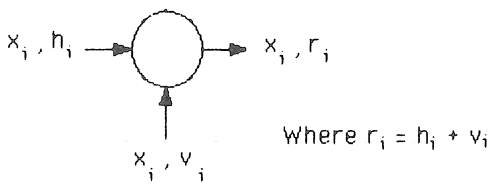
(a) General Architecture



(b) Comparator and Adder



(c) Logic dessign for Comparator and Adder



(d) Adder

Fig. 5 Systolic Architecture for Sorting.

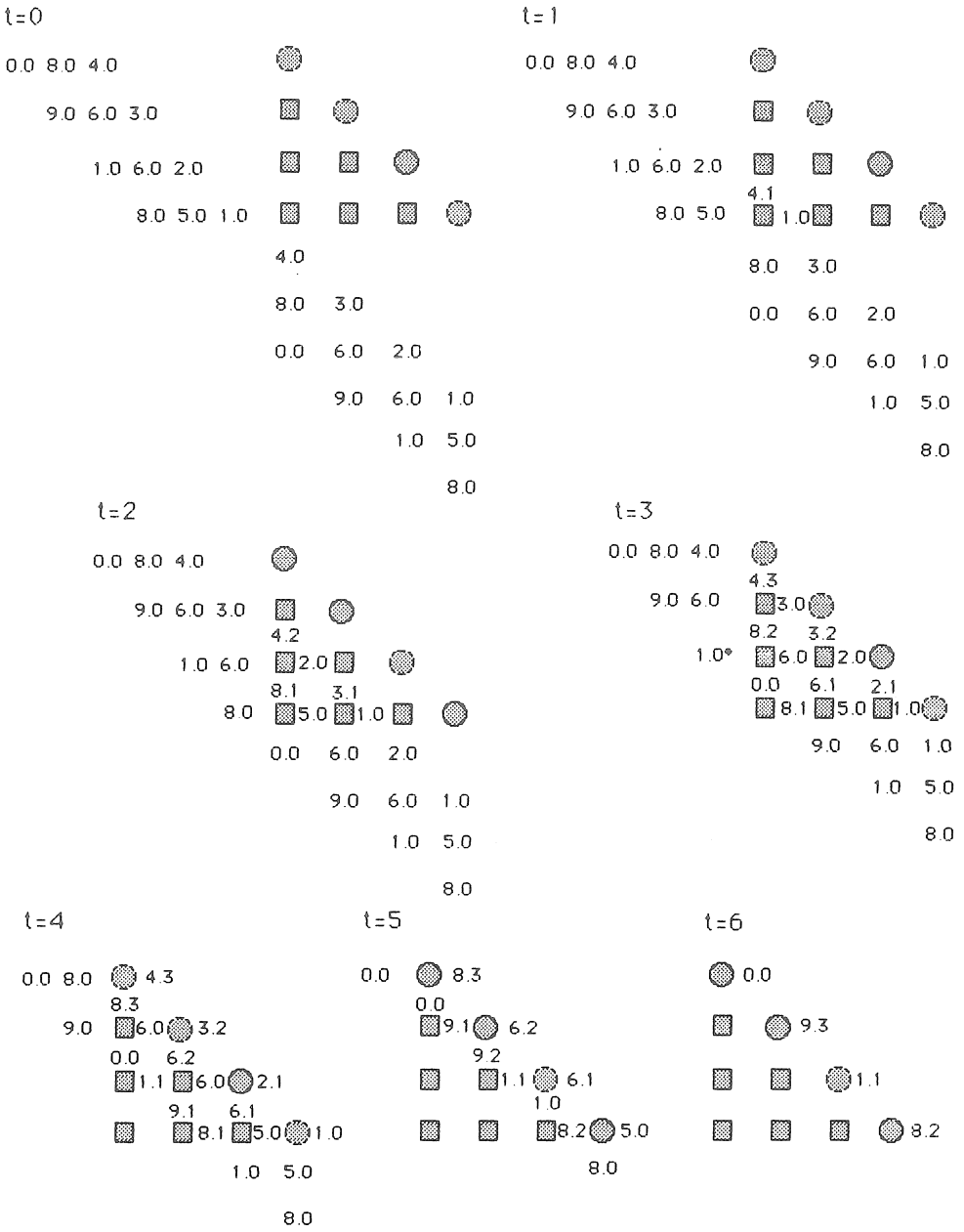


Fig. 6 - Timing secuence for Pipeline Sorting.