AN IMPROVED NEW EMBEDDING FOR VLSI DICTIONARY MACHINES ON MESHES*

FRANK DEHNE and NICOLA SANTORO

Center for Parallel and Distributed Computing
School of Computer Science, Carleton University, Ottawa, K1S 5B6, Canada

Abstract

In [DS87] we have studied how to implement dictionaries on a systolic mesh. The VLSI dictionary machines proposed in [DS87] consist of two structures, a snake and a broadcast net , which are both embedded in and operate simultaneously on the same mesh. Insert, Delete, Search, Extract Min, and Find Min can be performed with O(1) period, and the response time for Search and Find Min operations is $O(\sqrt{n})$ and O(1), respectively. Furthermore, the proposed solutions are capable of handling duplicate insertions and redundant deletions. The difference in performance between the proposed machines rests solely in the size of the constant, which depend on the simultaneous embedding of the two structures in the mesh. The best performance could be achieved by using a disjoint embedding. This embedding, however, had a major drawback: it made a special additional type of processors called 'relayers' necessary. These processors are wasted in that they can not store data. Furthermore, they make the implementation rather complicated.

In this paper we present a new disjoint embedding which yields optimal performance while avoiding any additional type of (wasted) processors. The new structure is much more regular and much easier to implement.

1. INTRODUCTION

A dictionary is a basic data type which allows for update and retrieval operations. Because of its general and fundamental capabilities, several researchers have studied the problem of designing special-purpose VLSI chips implementing dictionaries. Almost all proposed implementations are based on the complete binary tree structure [AK85], [CCIR86], [L79], [ORS82], [SA85]. From a theoretical viewpoint, the time to perform most of the dictionary operations when using such a structure is O(log n). Unfortunately, the practical layout of a binary tree leads to the presence of "long" wires [PRS81]; depending on conditions, models ranging from O(1) to O(lenght2) may be appropiate for the delay to traverse such a wire [MC80], [CM81a]. Furthermore, if the tree is embedded in a systolic mesh [BC86], [GKS82], [BPP82], [G87], [YS87] or in a programmable grid [S82] of size $\sqrt{n} \times \sqrt{n}$, the maximum delay is at least $\Omega(\sqrt{n})$.

Unlike trees, theoretical bounds on systolic meshes always correspond directly to the actual performance of the VLSI implementation. In [DS87] it is shown that the systolic mesh is an efficient architecture for implementing dictionaries (even with added priority queue operations) which can handle the problem of duplicate insertions and redundant deletions. Insert, Delete, Search, Extract Min, Find Min instructions can be sent to the dictionay in a pipelined fashion and the answers (if necessary) are reported in the same order in which the respective queries arrive.

The proposed VLSI dictionary machines consist of two structures, a snake and a broadcast net , which are both embedded in the same mesh and operate on it simultaneously. Several implementations where described, all having the following (asymptotically optimal) performance: operations Insert, Delete, Search, Extract Min, Find Min can be performed with O(1) period, and the response time tor Search and Find Min is $O(\sqrt{n})$ and O(1), respectively. The difference between the introduced implementation lies in the constant factors, in aparticular a trade-off between performance (i.e., response time and period) and link complexity (i.e., linear, quadrilateral, hexagonal, octagonal).

The major problem encountered in designing an architecture with optimal performance, i.e. maximum thoughput, (which needs an octagonal mesh) is that a disjoint embedding of a snake and a broadcast net in a mesh has to be found which also satisfies several additional restrictions.

In [DS87] such an embedding was introduced. However, this embedding included wasted processing elements which considerably reduced hardware efficiency and , furthermore, made implementation of the algorithm very complicated.

In this paper we present a new disjoint embedding which yields optimal performance while avoiding any additional type of (wasted) processors. The new structure is much more regular and much easier to implement.

In the following section 2 we will describe the topologies of the snake and broadcast net introduced in [DS87] and outline the additional restrictions on the disjoint ombedding of both structures

* Research supported by the Natural Sciences and Engineering Research Council of Canada.

F. Dehne and N. Santoro, "An improved new embedding for VLSI dictionary machines on meshes," in Proc. *International Symposium on Computer Applications in Design, Simulation and Analysis*, Reno, NA, 1989, pp. 113-116.

(in the mesh) which have imposed when both are simultaneously operated.

In section 3 we will then introduce a new improved disjoint embedding which yields a dictionary machine with optimal throughput while avoiding the drawbacks encountered by the embedding described in [DS87].

2. OVERVIEW OF THE DICTIONARY MACHINE PROPOSED IN [DS87]

The VLSI dictionary machine described in [DS87] consists of two known logical structures, a snake and a broadcast net, which are both embedded in the same physical mesh and operate on it simultaneously. All incoming search instructions are handed over to the broadcast net, whereas all other instructions are executed by the snake. We will now breifly outline these two structures and how they are simultaneously executed on the same mesh (for more details consult [DS87]).

2.1. The Snake

The snake is basically the well known linear array implementation of a systolic priority queue (e.g., [L79], [KL84]). The records are stored in increasing sorted order and without gaps, starting from the I/O port. The embedding of the snake in the mesh is such that (i) every processor is contained in the snake exactly once, and (ii) the leftmost I/O processor (which contains the minimum element) is coincident with the upper left I/O processor of the mesh; a possible embedding is shown in figure 1.

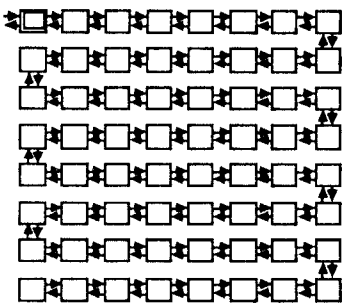

Figure 1: A Snake Embedding in the Mesh

If only the snake is operated on the mesh, the latency for FindMin queries is one time unit and two time units for ExtractMin and Delete operations.

2.2. The Broadcast Net

The broadcast net is a systolic structure whose function will be the handling of Search instructions. Its topology consists of two acyclic directed graphs, G1 and G2, where:

- G1 is a spanning graph of the mesh with only one source.
- G2 is a subgraph of the mesh with only one sink, and whose sources coincide with the sinks of G1.
- The I/O processor of the mesh is the source of G1 and the sink of G2.

A Search(k) instruction is "broadcasted" through G1; the respective message contains a Boolean value denoting whether the record has been found, and a field containing the record if found. The sinks of G1 will then start a "reverse broadcast" process which has the final effect of collecting at the I/O port (the sole sink of G2) either the record (if it is in the dictionary) or a negative acknowledgment.

The only additional constraint on the structure of the broadcast net is that, if more than one message is received on the same graph by the same processor at the same time, they must all contain the same search key k.

A special class of broadcast nets is the one of broadcast trees : in a broadcast tree, G1 is a directed binary tree rooted at the I/O port where all leaves have the same height, and G2 is coincident with G1 except for the direction of the edges which is reversed. In a broadcast tree, each query is broadcasted down the tree (broadcast) and then, starting from the leaves, the partial results move upwards towards the I/O port where the final result is computed (reverse broadcast). For broadcast trees, the above additional constraint is equivalent to the requirement that all leaves of G1 have the same height.

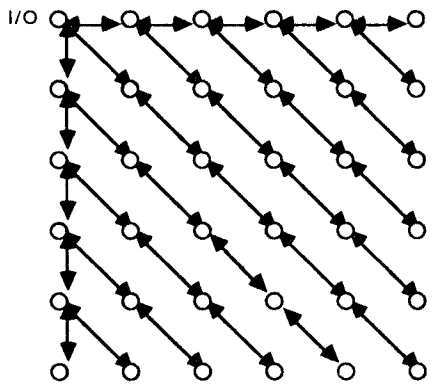

Figure 2: A Broadcast Tree

2.3. Interference between Snake and Broadcast Net

When embedding and operating both structures on the mesh, several factors must be taken into consideration; in fact, most embeddings would not archieve the desired performance and, even worse, would not correctly perform the desired operations. In [DS87], these factors have been identified and conditions established for a correct operation of the machine; these conditions will in turn express requirements for the embedding.

The problems encountered, when the snake and broadcast net are executed simultaneously are the following:

- Snake and broadcast net may both need the same processor at the same time.

- Snake and broadcast net may both need the commincation line processor at the same time.
- Insert or delete operations handled by the snake may take several steps until the data is actually inserted or deleted, respectively. Meanwhile, a subsequent search operation handled by the broadcast net may report an incorrect result.
- Delete operations cause the snake to shift all subsequent data to fill the gap. This shifting process may result in broadcast messages not finding the required data.

The first two problems can be easily solved by either splitting each time step on the mesh into two phases, one for the snake and the broadcast net, each. This solution, however, slows down the throughput and response time of the dictionary by a factor of two. A better solution is to find a disjoint embedding of the snake and broadcast net in the mesh (this solves actually both problems as desribed in [DS87]). Given such a disjoint embedding, the snake and broadcast net can be run in parallel yielding maximum throughput and response time.

The latter two problems can be solved by storing additional information at each processing element P and slightly modfying the snake and proadcats process. The amount of information has been identified to be proportional do $\Delta(P)$ where $\Delta(P)$ is defined as follows:

For each processing element P let *DIST(P)* denote the number time steps necessary for a search message to to travel from the I/O port to P, then

$\Delta(P) := \max\{ |t(P)-t(P')| : P \text{ and } P' \text{ are directly connected by an edge in the snake}\}$.

Furthermore, let Δ be the maximum $\Delta(P)$ for all PEs P in the mesh.

The values $\Delta(P)$ as well as Δ are strictly dependent on the interconnection between snake and broadcast net when embedded in the mesh. In fact, given an embedding , these values and whether the embedding is edge disjoint or shared completely characterize the interference between snake and broadcast net.

Note that the constraint on PE's having a constant number of registers implies that the only feasible embeddings (for our technique) are the ones with $\Delta \in O(1)$; see [DS87] for more details.

3. THE L-EMBEDDING: A NEW IMPROVED DISJOINT EMBEDDING

Since non-disjoint embeddings slow down the time performance of the dictionary machine by a factor of two it is highly desirable to find disjoint embeddings of the broadcast net and snake. Let us summarize, which requirements such an embedding must meet :

The embedding consist of two structures

- a snake, i.e. a linear ordering of PEs connected by edges of the mesh which contains each processor exactly once, and
- a broadcast tree, i.e. a spanning tree of the mesh rooted at the I/O port such that all leaves have the same hight

with the additional properties that

- the snake and the broadcast net are edge disjoint, and
- for each pair P, P' of PEs connected by an edge in the snake $|Dist(P) - Dist(P')| \leq D$ for some constant D.

Disjoint embeddings are not as simple to derive and do not exhibit a regular pattern easily scaled to meshes of arbitrary size (see [DS87]). In [DS87] such an embedding was introduced. However, this embedding included wasted processing elements which considerably reduced hardware efficiency and , furthermore, made implementation of the algorithm very complicated.

We now present a new disjoint embedding, refereed to as l-embedding, which yields optimal performance while avoiding any additional type of (wasted) processors. The new structure is much more regular and much easier to implement.

The broadcast net of the l-embedding is shown in figure 3; figure 4 shows the snake embedding (and broadcast superimposed). As it can be easily see from both figures, the snake and broadcast are edge disjoint. Furthermore, it follows that for each pair P, P' of PEs connected by an edge in the snake $|Dist(P) - Dist(P')| \leq 3$, i.e., $\Delta=3$; see figure 5.

The l-embedding described here is also much simpler and much more regular that the one described in [DS87].

Hence, it follows that the l-embedding yields a dictionary machine with optimal throughput while avoiding the drawbacks encountered by the embedding described in [DS87]. Figure 6 summarizes our results.

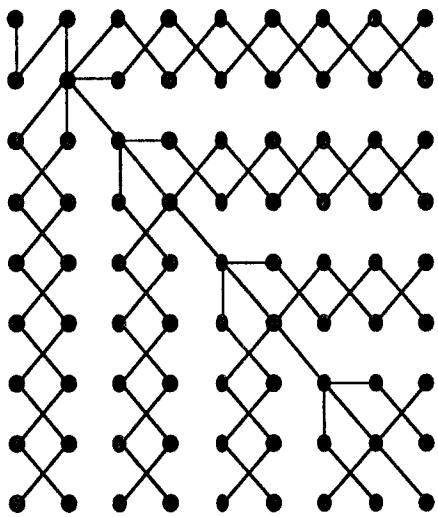

Figure 3: Broadcast Tree of the L-Embedding (on a Mesh of Size 9x9)

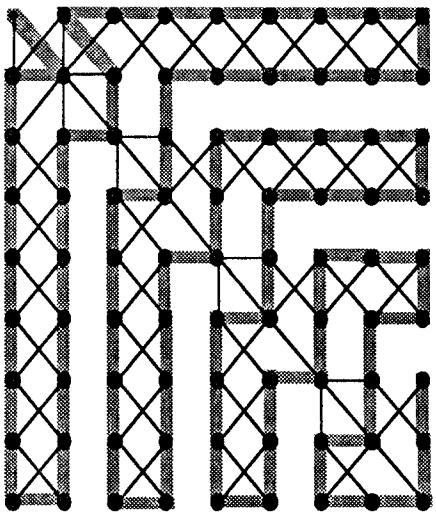

Figure 4: Broadcast Tree and Snake of the L-Embedding (on a Mesh of Size 9x9)

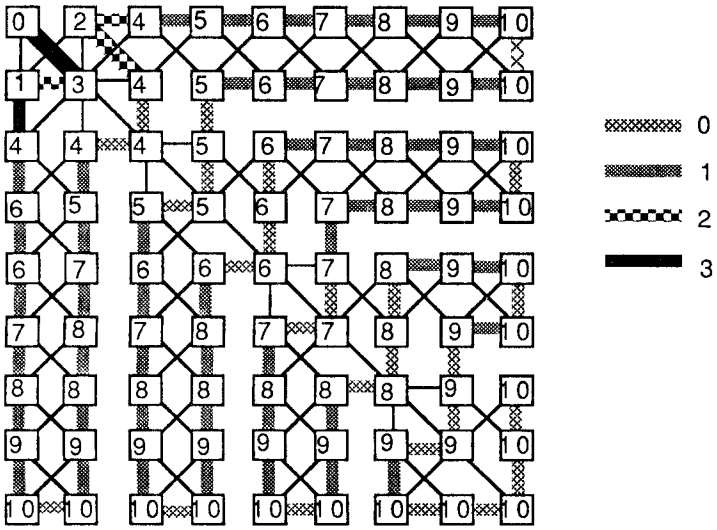

Figure 5: Distances of all Processing Elements from the I/O Port in an L-Embedding

REFERENCES

[AK85] M.J. Atallah and S.R. Kosaraju, "A generalized dictionary machine for VLSI", IEEE Trans. on Computers C-34, 2 (Feb. 1985), 151-155.

[BC86] D.A. Bailey and J.A. Cuny, "An efficient embedding of large trees in processor grids", Proc. 1986 Int. Conf. on Parallel Processing, St. Charles, Aug. 1986, 819-822.

[BPP82] G. Bilardi, M. Pracchi, F.P. Preparata, "A Critique Of Network Speed in VLSI Models of Computation", IEEE J. Solid-State Circuits, Vol. CS-17, Aug. 1982, pp. 696-702

[CCIR86] J.H. Chang, M.J. Chung, O.H. Ibarra, K.K. Rao, "Systolic tree implementation of data structures", Proc. 1986 Int. Conf. on Parallel Processing, St. Charles, Aug. 1986, 669-671.

[CM81a] B.M. Chazelle, L.M. Monier, "A model of computation for VLSI with related complexity results" Proc. 13th ACM Conf. on Theory of Computing, May 1981.

[DS87] F. Dehne, N. Santoro, "Optimal VLSI dictionary machines on meshes", Proceedings of the 1987 Conference on Parallel Processing, St.Carles, IL, August 17-21, 1987, pp. 832-840

[G87] D. Gordon, "Efficient Embeddings of Binary Trees in VLSI Arrays", IEEE Transactions on Computers, Vol. C-36, No. 9, Sept. 1987, pp.1009-1018

[GKS82] D. Gordon, I. Koren, G. Silberman, "Embedding tree structures in VLSI hexagonal arrays", IEEE Trans. on Computers C-31, 9 (sept. 1982), 892,897.

[KL84] M.R.Kramer, J. v.Leeuwen, "Systolische Berechnungen und VLSI", Informatik Spektrum 7, 1984, pp.154-165

[L79] C.E. Leiserson, "Systolic priority queues", Report CMU-CS-79-115, Carnagie-Mellon University, April 1979.

[MC80] C.A. Mead and L.A. Conway, Introduction to VLSI Systems, Addison-Wesley, 1980

[ORS82] T.A. Ottman, A.L. Rosenberg, and L.J. Stockmeyer, "A dictionary machine for VLSI", IEEE Trans. on Computers C-31, 9 (Sept. 1982), 892-897.

[PRS81] M.S. Paterson, W.L. Ruzzo, and L. Snyder, "Bounds on minimax edge lenght for complete binary trees", Proc. 13th ACM Symp. on Theory of Computing, May 1981, 293-299.

[S82] L. Snyder, "Introduction to the Configurable Highly Parallel Computer", Computer Journal, Jan. 1982.

[SA85] A.K. Somani, and V.K. Agarwal, "An efficient unsorted VLSI dictionary machine", IEEE Trans. on Computers C-34, 10 (Sept. 1985), 841-852.

[YS87] H.Y. Youn, A.D. Singh, "On Area Efficient and Fault Tolerant Tree Embedding in VLSI", Proc. 1987 Int. Conf. on Parallel Processing, St.Charles, Ill., Aug. 1987, pp. 170-177

performance (time units) \ layout	non-disjoint		disjoint	
(maximum) period	4	4	2	2
latency: find min	2	2	1	1
latency: search	$8\sqrt{n}$	$4\sqrt{n}$	$2\sqrt{n}$	$2\sqrt{n}$
hardware	c-linear mesh	quadrilateral mesh	octagonal mesh with $(\sqrt{n}-1)/2$ wasted relayers	octagonal mesh without relayers
			[DS87]	new

Figure 5: Summary of Results: Hardware vs. Performance