ELSEVIER

Information Processing Letters 58 (1996) 65–69

Information Processing Letters

Multiple search problem on reconfigurable meshes

Chia-Chiang Chao [a], Wen-Tsuen Chen [a], Gen-Huey Chen [b,*]

[a] *Department of Computer Science, National Tsing Hua University, Hsinchu, Taiwan 30043*

[b] *Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan 10764*

Received 15 August 1993; revised 1 February 1996

Communicated by S.G. Akl

Abstract

In this paper, a constant time algorithm on an $n^{1/2} \times n^{1/2} \times n^{1/2}$ reconfigurable mesh is presented for solving the multiple search problem.

Keywords: Multiple search problem; Parallel algorithms; reconfigurable mesh

1. Introduction

Recent advances in VLSI technology have made it possible to construct various massively parallel computers. Among them, the mesh-connected computers are one of the most important because of their simple structure and regular interconnection. However, their large diameters impose a strict time lower bound on many applications whose execution involves long-distance data transfers. To overcome this problem, mesh-connected computers augmented with various bus systems have been proposed [3,8,11,13]. The reconfigurable meshes [7,8] are one of them. A reconfigurable mesh is a mesh-connected computer augmented with a reconfigurable bus system whose bus configurations can be changed dynamically in run time. Recently, the tremendous power of the reconfigurable bus system has been discussed in the literature [5,10,14,15].

In this paper, we further show the power of the reconfigurable bus system by solving the multiple search problem on the reconfigurable meshes. The *multiple search problem* is defined as follows. Given a nondecreasing sequence $A = \{a_1, a_2, \ldots, a_n\}$ of items and a nondecreasing sequence $Q = \{q_1, q_2, \ldots, q_m\}$ of queries, where $1 \leqslant m \leqslant n$, determine for each query q_j the item a_i such that $a_{i-1} \leqslant q_j < a_i$. If no such a_i is found, then ∞ is taken as the solution. The multiple search problem has applications in image processing, computational geometry, computer graphics, etc. Akl and Meijer [1] first presented a parallel solution to this problem. Their algorithm runs in $O(\log m \log n/\log\log n)$ time using m processors on an EREW PRAM (Exclusive-Read Exclusive-Write Parallel Random Access Machine). Then, Wen [16] presented a faster algorithm, which takes only $O(\log m + \log n)$ time, using m processors on the same computation model. In this paper, we show that the multiple search problem can be solved in $O(1)$ time on an $n^{1/2} \times n^{1/2} \times n^{1/2}$ reconfigurable mesh.

* Corresponding author. Email: ghchen@csie.ntu.edu.tw.

PII S0020-0190(96)00033-6

2. Reconfigurable meshes

A reconfigurable mesh has the same underlying topology as a mesh-connected computer. But, unlike the latter, the former supplies each processor with ports, two for each dimension, which can be dynamically connected to form various bus configurations. Fig. 1 shows a two-dimensional reconfigurable mesh of size 5×5, where each processor is equipped with four ports, denoted by N, S, E, and W. The ports W, E are built along dimension i, and N, S are built along dimension j. A three-dimensional reconfigurable mesh of size $3 \times 2 \times 2$ is depicted in Fig. 2, where F, B are two ports built along dimension k.

Many bus configurations may result if ports of processors are connected in different ways. For example, if each processor of Fig. 1 connects ports W, E together, then five straight buses along the i direction are built. As usual, it is assumed that at any time, at most one processor is allowed to broadcast a value on a bus. It is also assumed that port connection and broadcast all take O(1) time.

Throughout this paper, we use $P(c_1, c_2)$, where c_1 and c_2 are constants, to denote the processor of a two-dimensional reconfigurable mesh whose coordinates with respect to dimensions i and j are c_1 and c_2, respectively. Similarly, each processor of a three-dimensional reconfigurable mesh is denoted by $P(c_1, c_2, c_3)$. Besides, we use "$*$" to denote "all coordinates". So, $P(c_1, c_2, *)$ represents a linear processor array (or a one-dimensional reconfigurable mesh), and $P(c_1, *, *)$ represents a two-dimensional reconfigurable mesh. We assume the coordinate begins with 1.

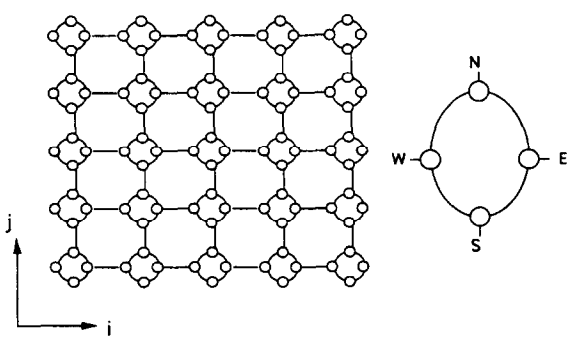

Fig. 1. A two-dimensional reconfigurable mesh of size 5×5.

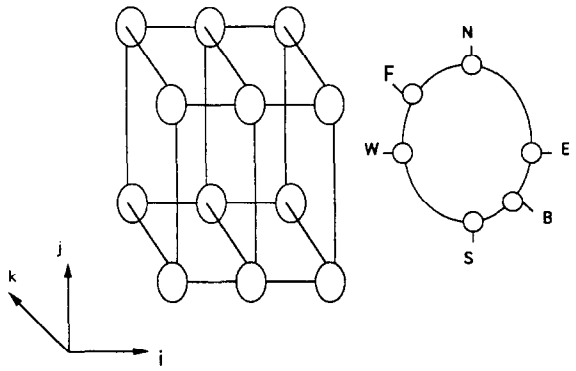

Fig. 2. A three-dimensional reconfigurable mesh of size $3 \times 2 \times 2$.

The following three facts are easy to understand, and thus no further explanation is provided.

Fact 2.1 [10]. On a reconfigurable mesh, broadcast along any dimension can be performed in O(1) time.

On a square reconfigurable mesh, the operation of permuting a row or column of values, each held by a processor, into another (may be the same) row or column according to a predefined sequence is called *permutation routing*.

Fact 2.2 [2]. On a square reconfigurable mesh, permutation routing can be performed in O(1) time.

Fact 2.3 [9]. Suppose S is a subset of processors in a two-dimensional reconfigurable mesh. Of each row or column, the leftmost or rightmost processor belonging to S can be determined in O(1) time.

3. The multiple search problem

Initially, each item a_i is stored in $P(c_1, c_2, 1)$, where $i = (c_1 - 1)n^{1/2} + c_2$, and each query q_j is stored in $P(1, c_2, c_3)$, where $j = (c_3 - 1)n^{1/2} + c_2$. That is, the sequence A is stored in the submesh $P(*, *, 1)$ with the first dimension being the major order, and the sequence Q is stored in the submesh $P(1, *, *)$ with the third dimension being the major order. For each query q_j, we define its *rank* as the coordinate y_j such that the solution of q_j can be found in $P(*, y_j, 1)$.

Lemma 3.1. *The ranks of all queries can be determined in* O(1) *time on an* $n^{1/2} \times n^{1/2} \times mn^{-1/2}$ *reconfigurable mesh.*

Proof. We let each $P(*, *, z)$, where $1 \leqslant z \leqslant mn^{-1/2}$, be responsible for processing $n^{1/2}$ queries that are stored in $P(1, *, z)$. The computations are detailed as follows. First, copy the sequence A to each $P(*, *, z)$. Secondly, move the items of A stored in $P(n^{1/2}, *, z)$ to $P(*, 1, z)$ without changing their order, and then broadcast the items stored in $P(*, 1, z)$ along the j dimension. Thirdly, broadcast queries stored in $P(1, *, z)$ along the i dimension, and then mark each processor of $P(*, *, z)$ owning a larger item than the query. Finally, for each $P(*, y, z)$, where $1 \leqslant y \leqslant n^{1/2}$, determine the smallest x such that $P(x, y, z)$ is marked, and then return x to $P(1, y, z)$ as the rank of its containing query. If no processor of $P(*, y, z)$ is marked, then $n^{1/2} + 1$ is returned. By the facts of Section 2, the computations take O(1) time. □

In the rest of this section, each $P(1, *, z)$, where $1 \leqslant z \leqslant mn^{-1/2}$, is referred to as *unified* or *mixed* depending on whether or not the ranks of all its containing queries are the same.

Lemma 3.2. *On an* $n^{1/2} \times n^{1/2} \times mn^{-1/2}$ *reconfigurable mesh, the queries of a unified* $P(1, *, c_3)$, *where* c_3 *is a constant, can be solved in* O(1) *time.*

Proof. Without loss of generality, we assume the queries of $P(1, *, c_3)$ have rank r. If the sequence A is replicated at the beginning such that each $P(*, *, z)$, where $1 \leqslant z \leqslant mn^{-1/2}$, holds one copy, then the queries of $P(1, *, c_3)$ can be solved on $P(*, *, c_3)$ as follows. First, broadcast the items of $P(*, r, c_3)$ along the j dimension. Then, broadcast the queries of $P(1, *, c_3)$ along the i dimension. Now, since each $P(*, y, c_3)$, where $1 \leqslant y \leqslant n^{1/2}$, holds a copy of the items of $P(*, r, 1)$ and each of its processor also holds a copy of the query of $P(1, y, c_3)$, all queries can be solved simultaneously by letting each $P(*, y, c_3)$ compute the solution of a unique query. The necessary computation for $P(*, y, c_3)$ is to uniquely determine two adjacent processors, say $P(c_1 - 1, y, c_3)$ and $P(c_1, y, c_3)$, where $1 < c_1 \leqslant n^{1/2}$, such that the query of $P(1, y, c_3)$ is smaller than the item of $P(c_1, y, c_3)$ but is greater than or equal to the item of $P(c_1 - 1, y, c_3)$. The item of $P(c_1, y, c_3)$ is then returned to $P(1, y, c_3)$ as the solution of its containing query. By the facts of Section 2, the total time complexity is O(1). □

From the proof of Lemma 3.2, it is not difficult to understand that Lemma 3.2 can be extended to all unified $P(1, *, z)$, where $1 \leqslant z \leqslant mn^{-1/2}$. We formally state this fact in the following lemma.

Lemma 3.3. *On an* $n^{1/2} \times n^{1/2} \times mn^{-1/2}$ *reconfigurable mesh, the queries of all unified* $P(1, *, z)$, *where* $1 \leqslant z \leqslant mn^{-1/2}$, *can be solved in* O(1) *time.*

Theorem 3.4. *The multiple search problem can be solved in* O(1) *time on an* $n^{1/2} \times n^{1/2} \times n^{1/2}$ *reconfigurable mesh.*

Proof. The following procedure provides a solution to the multiple search problem.

Step 1. Replicate the sequence A such that each $P(*, *, z)$, where $1 \leqslant z \leqslant n^{1/2}$, holds one copy.

Step 2. Compute the ranks of all queries. A query has a solution ∞ if its rank is $n^{1/2} + 1$.

Step 3. Determine for each $P(1, *, z)$, where $1 \leqslant z \leqslant mn^{-1/2}$, whether it is mixed or unified.

Step 4. Solve the queries of all unified $P(1, *, z)$, where $1 \leqslant z \leqslant mn^{-1/2}$.

The queries of all mixed $P(1, *, z)$ are solved in succeeding steps. Without loss of generality, let us consider a mixed $P(1, *, c_3)$ and assume its ranks are $r_1 < r_2 \cdots < r_t$.

Step 5. Move the query of each $P(1, y, c_3)$ to $P(y, r, c_3)$, where $1 \leqslant y \leqslant n^{1/2}$ and $r \in \{r_1, r_2, \ldots, r_t\}$ is the rank held by $P(1, y, c_3)$. Fig. 3 shows the situations before and after the data movement.

The next two steps, which are executed for each $P(*, *, z)$ holding queries, move all queries of $P(*, *, z)$ down to $P(*, *, 1)$, where $z \geqslant 2$. Without loss of generality, we assume the ranks of queries of $P(*, *, z)$ are $r_1 < r_2 < \cdots < r_t$.

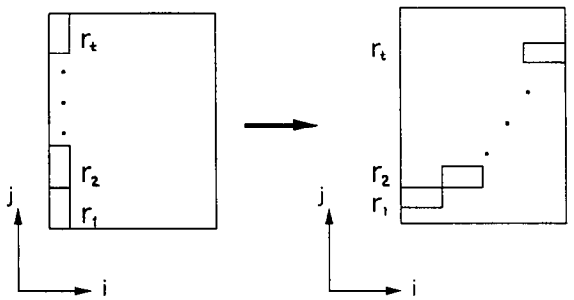

Fig. 3. Data movement for mixed $P(1, *, c_3)$.

Step 6. Broadcast queries of $P(*, *, z)$ whose ranks are in $\{r_1, r_2, \ldots, r_{t-1}\}$ along the k dimension.

Step 7. Broadcast queries of $P(*, *, z)$ whose rank is r_t the along k dimension.

After Steps 6 and 7, the queries of all mixed $P(1, *, z)$ are collected in $P(*, *, 1)$, and the queries whose rank is $r \in \{r_1, r_2, \ldots, r_t\}$ appear in $P(*, r, 1)$. Note that the queries must be moved with two time steps so as to avoid bus conflicts, because the greatest rank of $P(*, *, z)$ may be equal to the least rank of $P(*, *, z+1)$ (each processor of $P(*, *, 1)$ receives at most two queries). After Step 7, each $P(*, r, 1)$ becomes unified.

Step 8. Move the queries of each $P(*, r, 1)$ to $P(*, r, r)$.

Step 9. Solve the queries of each $P(*, r, r)$.

Step 10. Return the solutions obtained in Step 9 to the processors of $P(1, *, *)$ where the queries are initially stored.

The time complexity of the procedure is O(1), which we explain as follows. By the facts of Section 2, Steps 1, 5, 6, 7 and 8 require O(1) time. By Lemma 3.1, Step 2 requires O(1) time. Step 3 can be completed in O(1) time by checking whether or not the rank in $P(1, 1, z)$ is equal to the rank in $P(1, n^{1/2}, z)$. Step 4 takes O(1) time by Lemma 3.3. Step 9 needs only O(1) time because the queries of each $P(*, r, r)$ are solved on a unique $P(*, *, r)$, and the execution is very similar to that shown in the proof of Lemma 3.2. Step 10 can be completed in O(1) time by reversing the data movement performed in Step 5 to Step 9. □

4. Remarks

Recently, the reconfigurable meshes have received a lot of attention and various models have been proposed [6]. A major difference between these models lies in the allowed connection patterns. Besides, each model can be further divided into a word model and a bit model depending on the width of the buses and the size of the registers in the processors. In the word model both the bus width and the register size are $O(w)$ bits for a sufficiently large positive integer w. Besides each arithmetic or logic operation involving $O(w)$-bit operands takes O(1) time. On the other hand, the bit model allows the bus width and the register size to be O(1) bits. In this paper the word model is adopted.

We have shown in this paper that the multiple search problem can be solved in O(1) time on an $n^{1/2} \times n^{1/2} \times n^{1/2}$ reconfigurable mesh. The multiple search problem has assumed sorted sequences A and Q. However, due to Chen and Chen's recent work [5], our result is valid even if A and Q are not sorted. In [5], Chen and Chen show that sorting n numbers can be completed in O(1) time on an $n^{1/2} \times n^{1/2} \times n^{1/2}$ reconfigurable mesh.

Also note that we have assumed constant broadcast delay which is somewhat controversial. Certainly linear or logarithmic propagation delay may be more realistic for coaxial cables. However, since the speed of propagation is much smaller than the clock period, if we can essentially reach the speed of light for transmission, then the broadcast delay is reasonably small and the O(1) assumption is considered true over the relevant size range. In this setting, Schuster and Ben-Asher [12] have shown that the O(1) assumption can be made true if the reconfigurable bus is manufactured using optical fibers.

It can be understood that the proposed algorithm gains efficiency mainly from the reconfigurable bus system. If the reconfigurable bus system is not provided, more execution time is needed. Chao, one of the authors, has been interested in how fast the multiple search problem can be solved if a mesh-connected computer is adopted. His final answer is

$O(n^{1/2})$ time on a mesh-connected computer of size $n^{1/2} \times n^{1/2}$ (see [4]).

References

[1] S.G. Akl and H. Meijer, Parallel binary search, *IEEE Trans. Parallel Distributed Systems* **1** (1990) 247–250.

[2] Y. Ben-Asher, D. Peleg, R. Ramaswami and A. Schuster, The power of reconfiguration, *J. Parallel Distributed Comput.* **13** (1991) 139–153.

[3] S.H. Bokhari, Finding maximum on an array processor with a global bus, *IEEE Trans. Comput.* **33** (1990) 133–139.

[4] C.C. Chao, Some parallel algorithms on mesh-connected computers, Master Thesis, Dept. of Computer Science, National Tsing Hua University, Hsinchu, Taiwan, 1993.

[5] Y.C. Chen and W.T. Chen, Constant time sorting on reconfigurable meshes, *IEEE Trans. Comput.* **43** (1994) 749–751.

[6] J. Jang, H. Park and V.K. Prasanna, A bit model of reconfigurable mesh, Manuscript.

[7] H. Li and Q.F. Stout, Reconfigurable massively parallel computers: An introduction, Manuscript.

[8] R. Miller, V.K. Prasanna Kumar, D.I. Reisis and Q.F. Stout, Meshes with reconfigurable buses, in: *Proc. 5th MIT Conf. on Advanced Research in VLSI* (1988) 163–178.

[9] R. Miller, V.K. Prasanna Kumar, D.I. Reisis and Q.F. Stout, Parallel computations on reconfigurable meshes, *IEEE Trans. Comput.* **42** (1993) 678–692.

[10] S. Olariu, J.L. Schwing and J. Zhang, Fundamental data movement for reconfigurable meshes, in: *Proc. Internat. IEEE Phoenix Conf. on Computers and Communications* (1992) 472–479.

[11] V.K. Prasanna Kumar and C.S. Raghavendra, Array processor with multiple broadcasting, *J. Parallel Distributed Comput.* **2** (1987) 173–190.

[12] A. Schuster and Y. Ben-Asher, Algorithms and optic implementation for reconfigurable networks, in: *Proc. 5th Jerusalem Conf. on Information Technology* (1990).

[13] Q.F. Stout, Meshes with multiple buses, in: *Proc. 27th IEEE Symp. on Foundations of Computer Science* (1986) 264–273.

[14] B.F. Wang and G.H. Chen, Two-dimensional processor array with a reconfigurable bus system is at least as powerful as CRCW model, *Inform. Process. Lett.* **36** (1990) 31–36.

[15] B.F. Wang and G.H. Chen, Constant time algorithms for the transitive closure and some related graph problems on processor arrays with reconfigurable bus systems, *IEEE Trans. Parallel Distributed Systems* **1** (1990) 500–507.

[16] Z. Wen, Parallel multiple search, *Inform. Process. Lett.* **37** (1991) 181–186.