A Fast Parallel Sorting Algorithm on the k-dimensional Reconfigurable Mesh

Ju-wook Jang
Department of Electronic Eng.
Sogang University
Seoul, Korea 121-742
E-mail: jjang@ccs.sogang.ac.kr

Kichul Kim
Department of Electrical Eng.
University of Seoul
Seoul, Korea
E-mail: kkim@scucc.scu.ac.kr

We presents a new parallel sorting algorithm on the k-dimensional reconfigurable mesh which is a generalized version of the well-studied (two dimensional) reconfigurable mesh. We introduce a new mapping technique which combines the enlarged bandwidth of the multidimensional mesh and the feature of the reconfiguable mesh. Using our mapping technique, we show that N^k numbers can be sorted in $O(4^k)$ (constant time for small k) time on a $k+1$ dimensional reconfigurable mesh of size $\overbrace{N \times N \times \ldots \times N}^{k+1 \text{ times}}$. In addition, it is shown that the number of 1's in a 0/1 array of size $\overbrace{N \times N \times \ldots \times N}^{k \text{ times}}$ can be computed in $O(\log^* N + \log k)$ time on reconfigurable mesh of size $\overbrace{N \times N \times \ldots \times N}^{k \text{ times}}$.

1 Introduction

The reconfigurable mesh model has been introduced in [13]. This parallel model of computation captures the fundamental properties of CHiP [17], mesh computers augmented with broadcast buses, the bus automaton [4], the polymorphic-torus network [12], and corterie network in the latest version of the Content Addressable Array Parallel Processor (CAAPP). We use the term mesh with reconfigurable bus or reconfigurable mesh to describe this model. Several algorithms on the reconfigurable mesh are known [9,10,13,19,20]. In particular, some graph algorithms are implemented on a 3-dimensional reconfigurable mesh of size $N \times N \times N$[20].

In this paper, we consider a k-dimensional reconfigurable mesh and its applications for sorting and counting of 1's in a 0/1 array when the input is given in multidimensional format. We show that the sorting of N^k numbers can

(0-7803-4229-1/97/$10.00 1997 IEEE)

be performed in $O(1)$ time on a $(k+1)$-dimensional reconfigurable mesh of size $\overbrace{N \times N \times \ldots \times N}^{k+1 \text{ times}}$. The $(k+1)$-dimensional reconfigurable mesh is a natural extension of the 2-dimensional reconfigurable mesh defined in [13]. Wang, Chen and Lin[19] have shown that N numbers can be sorted in $O(1)$ time on a 3-dimensional reconfigurable mesh of size $N \times N \times N$. Our sorting algorithm improves their result by a factor of N in the number of PEs employed without asymptotic increase in time complexity. Our result would have wide application for many computational geometry problems involving points in a k-dimensional space, since the time complexity of numerous geometry problems are closely related to that sorting.

In addition, we show that the number of 1's in a 0/1 array of size $\overbrace{N \times N \times \ldots \times N}^{k \text{ times}}$ can be obtained in $O(\log^* N + \log k)$ time on a k-dimensional reconfigurable mesh of size $\overbrace{N \times N \times \ldots \times N}^{k \text{ times}}$.

This paper is organized as follows. In section 2, we will briefly review two dimensional reconfigurable mesh architecture and its extension to k dimension. In Section 3, the problems of sorting N^k numbers and the counting of 1's in a 0/1 array of size $\overbrace{N \times N \times \ldots \times N}^{k \text{ times}}$ are considered. Concluding remarks are made in Section 4.

2 The k-dimensional Reconfigurable Mesh

First, we briefly review the 2-dimensional reconfigurable mesh of size $N \times N$ defined in [13] and then show its extension to k-dimensions. The 2-dimensional $N \times N$ reconfigurable mesh consists of $N \times N$ array of processors connected to a grid-shaped reconfigurable broadcast bus, where each processor can locally control the interconnection between four(E,W,N and S) I/O ports. This allows the broadcast bus to be divided into *subbuses*, providing smaller reconfigurable meshes. For a given set of switch settings, a *subbus* is a maximal connected subset of processors. Other than the buses and switches the reconfigurable mesh is similar to the standard mesh in that it has $\Theta(N^2)$ area, under the assumption that processors, switches, and a link between adjacent switches occupy unit area in the *word-model* of VLSI. We consider the *exclusive write* model which allows only one processor to broadcast to a *subbus* shared by multiple processors at any given time. We assume that the value broadcast consists of $O(\log N)$ bits and takes $\Theta(1)$ time, as is the assumption in [18] for

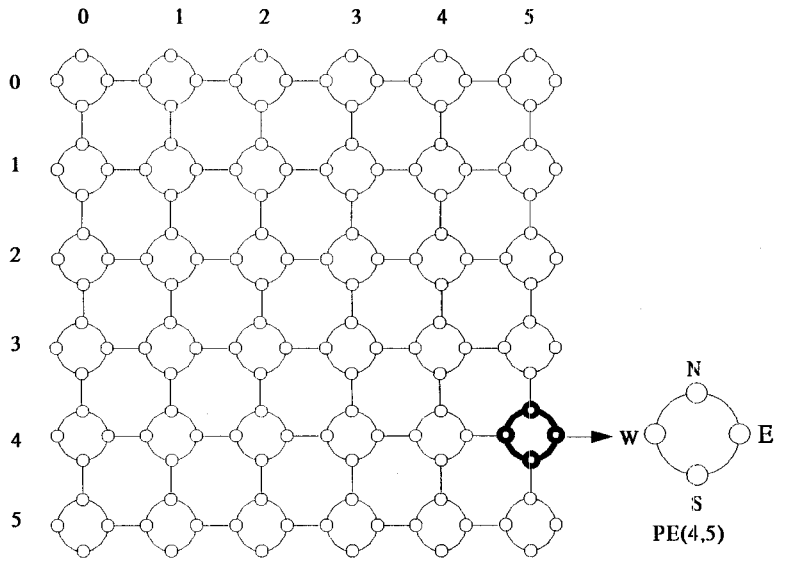


Figure 1: A 6 × 6 Reconfigurable Mesh

models that assume various broadcasting strategies. Also, each PE can perform arithmetic and logic operations on $O(1)$ words in a time unit. The size of the local storage is $O(1)$ words, where each word is $\Theta(\log N)$ bits. A 6 × 6 reconfigurable mesh is shown in Figure 1.

Each PE has four I/O ports(E,W,N,S). Internal connection between the 4 ports of a PE can be configured during the execution of algorithms. Figure 2 shows some possible configurations. For example, {SW, NE} represents the configuration in which S(South) port is connected to W(West) port while N(North) port is connected to E(East) port. Each bit of the bus can carry

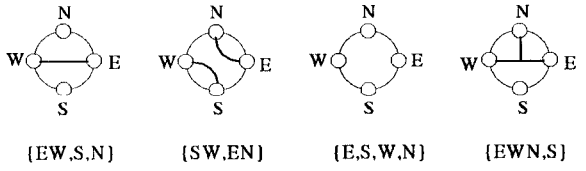


Figure 2: Some patterns of internal connection between the four I/O ports of a PE

one of two signals: *1-signal* or *0-signal.* The presence of *1-signal* at a port is represented by a dark circle.

In the *bit-model* of the reconfigurable mesh, buses are capable of carrying $O(1)$ bit data. Each PE has $O(1)$ bits of local storage and has constant area in the bit model of VLSI. Other than the width of the buses and the wordlength, the *bit-model* is same as the *word-model.* Throughout this paper, reconfigurable mesh is used to denote the word model unless specified as bit model.

The 2-dimensional reconfigurable mesh can be naturally extended to k-dimensional reconfigurable mesh. $PE(i_0, i_1, \ldots, i_{k-1})$ denotes a PE in a k-dimensional reconfigurable mesh, where $0 \leq i_r \leq N-1, 0 \leq r \leq k-1$. Each PE has two I/O ports along each dimension(a total of $2k$ I/O ports/PE). The connection between the I/O ports is controlled by the PE. Any such connection can be realized in $O(1)$ time. Also, broadcast of data along any dimension can be performed in $O(1)$ time.

Any permutation on N data in any row(dimension 1) or column(dimension 2) of $N \times N$ reconfigurable mesh can be performed in $O(1)$ time. Given a 0/1 sequence of length N, the number of 1's can be counted in $O(1)$ time using $N \times N$ reconfigurable mesh[5]. Given N k-bit binary numbers, $1 \leq k \leq N$, the *addition problem* is to add these numbers into a $k + \log N$ bit binary number. In [5], it is shown,

Lemma 1 *Given N k-bit binary numbers $1 \leq k \leq N$, these numbers can be added in $O(1)$ time on $N \times Nk$ bit-model of reconfigurable mesh.*

3 Applications

We develop efficient parallel implementation of two applications: sort of N^k numbers and count of 1's in a 0/1 array of size $\overbrace{N \times N \times \ldots \times N}^{k \text{ times}}$.

3.1 Sort of N^k numbers

Given N^k numbers stored as a k-dimensional block in a $(k+1)$-dimensional reconfigurable mesh, the problem is to permute the numbers so that the resulting data forms a nondecreasing sequence in the lexicographic ordering of the PEs. We start with a subproblem.

Lemma 2 *Sort of $N^{\frac{3}{4}}$ numbers can be performed in $O(1)$ time on $N \times N^{\frac{3}{4}}$ Reconfigurable Mesh.*

Proof: Let $x(i), 0 \leq i \leq N^{3/4} - 1$ be the input $N^{\frac{3}{4}}$ numbers in the top row. Let $SM(l), 0 \leq l \leq N^{3/4} - 1$ be a submesh of size $N^{1/4} \times N^{3/4}$ which consists of $PE(i_0, i_1), lN^{1/4} \leq i_0 \leq (l+1)N^{1/4} - 1, 0 \leq i_1 \leq N^{3/4} - 1$. Broadcast

$x(i), 0 \leq i \leq N^{3/4} - 1$ along column buses(dimension 1). $SM(l)$ compares $x(l)$ against $x(i), 0 \leq i \leq N^{3/4} - 1$ and the rank of $x(l)$ can be represented as a 0/1 sequence of length $N^{3/4}$. First, divide $SM(l)$ into blocks of size of $N^{1/4} \times N^{1/4}$. Count the number of 1's in each 0/1 sequence of size $N^{1/4}$ using a block. Partial sums in the blocks can be merged using divide-and-conquer and Lemma 1 in $O(1)$ time. Let $rank(l)$ be the rank of $x(l)$. Then, route $x(l)$ to $PE(0, rank(l))$. This can be performed in $O(1)$ time. □

Theorem 1 *For $k \geq 1$, sort of N^k numbers can be performed in $O(4^k)$ time on $\overbrace{N \times N \times \ldots \times N}^{k+1 \text{ times}}$ reconfigurable mesh.*

Proof: Leighton[11] has shown that if $n = rs$, $r \bmod s \equiv 0$ and $r \geq 2(s-1)^2$ then n numbers can be sorted by constant number of iterations with each iteration involving one of shifting, shuffling of n numbers or s parallel sort of r numbers. Leighton's "8-step column sort"(see[11] for details) of N numbers can be alternatively represented by the following six stages. Stages 1,3, 5 and 6 are identical or similar and each of them consists of $N^{\frac{1}{4}}$ $N^{\frac{3}{4}}$-sorters. Stage 2 performs $N^{\frac{1}{4}}$-shuffle on N numbers while stage 4 performs inverse $N^{\frac{1}{4}}$-shuffle on N numbers.

Assume $x(i)$, $i = \sum_{r=0}^{k-1} i_r N^r$ is the i-th input number stored in $PE(i_0, i_1, \ldots, i_{k-2}, i_{k-1}, 0)$, where $0 \leq i_r \leq N-1$. Sort of N^k numbers can be performed as follows:

case 1: ($k < 4$) stages 1,3,5,6: $N^{k/4}$ $N^{3k/4}$-sorters, stage 2: $N^{k/4}$-shuffle and stage 4: inverse $N^{k/4}$-shuffle.

For $k = 1$, use Lemma 2 and column sort technique to sort N numbers in $O(1)$ time on $N \times N$ reconfigurable mesh[5]. Here, we will illustrate the technique for $k = 2$. From the above discussion, sort of N^2 numbers can be performed by the six stages, where each stage consists of either $N^{1/2}$ $N^{3/2}$-sorters or (inverse) $N^{1/2}$-shuffle. Input is assumed to be as shown in Figure 3(a). Let $SM(l)$ be the l-th submesh which consists of $PE(i_0, i_1, i_2)$, $0 \leq i_0, i_1 \leq N-1$ and $lN^{1/2} \leq i_2 \leq (l+1)N^{1/2} - 1$. Broadcast $x(i)$ along dimension 3. $SM(l)$ receives $x(i)$ such that $lN^{3/2} \leq i \leq (l+1)N^{3/2} - 1$(see Figure 3(b)). $SM(l)$ acts as a $N^{3/2}$-sorter. Note that each $N^{3/2}$-sorter can be decomposed into $N^{3/8}$ $N^{9/8}$-sorters and $N^{3/8}$-shuffle or inverse shuffle. The $N^{9/8}$-sorter, in turn, can be decomposed into $N^{1/8}$ N-sorters and $N^{1/8}$-shuffle or inverse shuffle. From the above, $N \times N$ reconfigurable mesh can act as a N-sorter. $N^{1/8}$-shuffle (or inverse shuffle) on $N^{9/8}$ numbers can be performed in $O(1)$ time on a submesh of size $N \times N \times N^{1/8}$. Also, $N^{3/8}$-shuffle (or inverse shuffle) on $N^{3/2}$ numbers can be performed in $O(1)$ time on a submesh of size $N \times N \times N^{1/2}$. Results from

$SM(l), 0 \leq l \leq N^{1/2} - 1$ are collected into $PE(i_0, i_1, 0), 0 \leq i_0, i_1 \leq N - 1$. Let $x'(i)$ be the resulting data in the i-th PE. $N^{1/2}$-shuffle is performed as follows. Broadcast $x'(i)$ along dimension 3. $SM(l)$ receives $x'(i)$ such that $i \bmod N^{1/2} \equiv l$(see Figure 3(c). The $N^{3/2}$ numbers can be collected in $O(1)$ time as shown in Figure 3(d). Collecting data from $SM(l), 0 \leq l \leq N^{1/2} - 1$ into $PE(i_0, i_1, 0), 0 \leq i_0, i_1 \leq N - 1$ completes the shuffle. Similar technique can be employed for $k = 3$.

case 2: ($k \geq 4$) stages 1,3,5,6: N N^{k-1}-sorters, stage 2: N-shuffle and stage 4: inverse N-shuffle.

Let $SMESH_k(l)$ be a k-dimensional submesh which consists of PEs having $i_k = l$. Initially, the N^k input numbers are in $SMESH_k(0)$. Broadcast $x(i)$ along dimension k. $SMESH_k(l)$ receives $x(i)$ such that $lN^{k-1} \leq i \leq (l+1)N^{k-1} - 1$. Now, $SMESH_k(l)$ performs as a N^{k-1}-sorter. Results from $SMESH_k(l)$, $0 \leq l \leq N - 1$ are collected into $SMESH_k(0)$. Let $x'(i)$ be the resulting data in the i-th PE. N-shuffle on $x'(i)$ is performed as follows. Broadcast $x'(i)$ along the dimension k. $SMESH_k(l)$ receives $x'(i)$ such that $i \bmod N \equiv l$.

If $T(N^k)$ is the time to sort N^k numbers, the above discussion implies $T(N^k) = 4T(N^{k-1}) + O(1)$, $k \geq 4$ and $T(N^3) = O(1)$. Thus, $T(N^k) = O(4^k)$. □

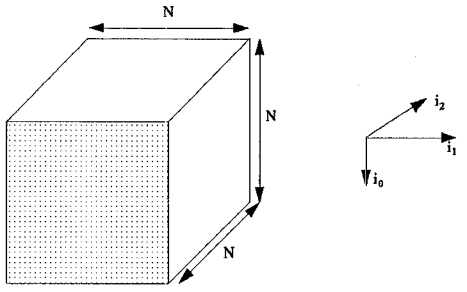
3.2 Counting the number of 1's in a k-dimensional array

In many applications, counting the number of 1's in a 0/1 array arises as a subproblem. The problem can be defined as follows: given an input array $I(i_0, i_1, \ldots, i_{k-1})$ of size $\overbrace{N \times N \times \ldots \times N}^{k \text{ times}}$ $I(i_0, i_1, \ldots, i_{k-1}) \in \{0, 1\}$, where $0 \leq i_r \leq N-1, 0 \leq r \leq k-1$, find $| \{(i_0, i_1, \ldots, i_{k-1}) \mid I(i_0, i_1, \ldots, i_{k-1}) = 1\} |$. We will show that the problem can be solved in $O(\log^* N + \log k)$ time on a k-dimensional reconfigurable mesh of size $\overbrace{N \times N \times \ldots \times N}^{k \text{ times}}$. We will start with *prefix modular k computation.*

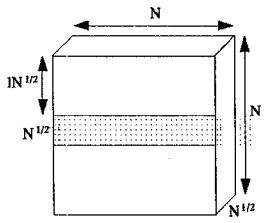
Given a 0/1 sequence of size N, $b_j, 0 \leq j \leq N - 1$, *prefix modular k computation* is to compute, for each j, $(\sum_{w=0}^{j} b_w) \bmod k$.

Lemma 3 *Prefix modular k computation of a* 0/1 *sequence of length N can be performed in* $O(1)$ *time on* $(k+1) \times 2N$ *reconfigurable mesh.*

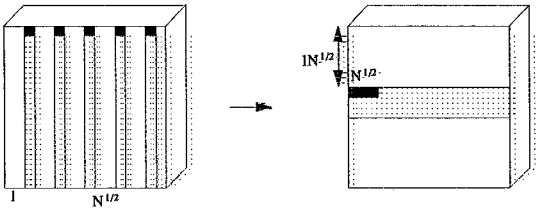
Proof: Let $b_j, 0 \leq j \leq N-1$ be the input sequence, where $b_j = 0$ or 1. Initially, b_j is stored in $PE(0, 2j)$. For $j = 0$ to $j = N - 1$, do the following in parallel.



(a) Initial Input Distribution



(b) SM(l) performs as a $N^{3/2}$-sorter



(c) SM(l) receives x'(i) such that i mod $N^{1/2} = l$ (d) After shuffle

Figure 3: Sort of N^2 numbers

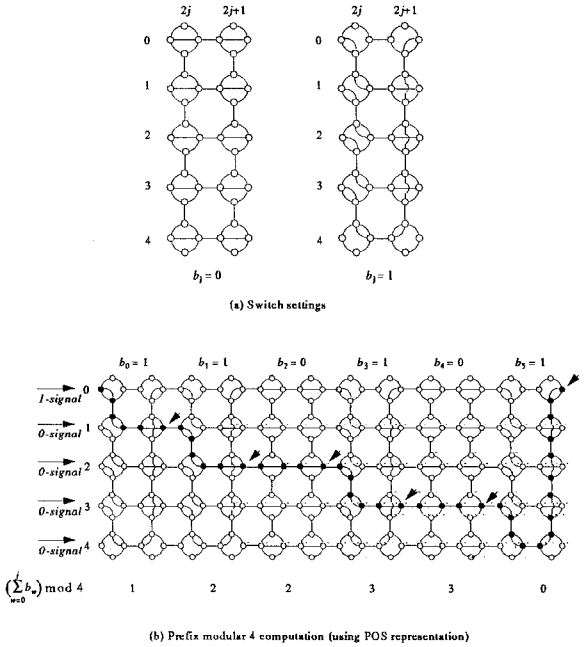


Figure 4: Prefix modular k computation for $k = 4$

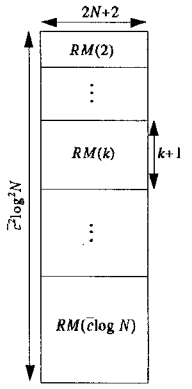


Figure 5: $RM(i)$ computes for each j, $0 \leq j \leq N$, $j \bmod i$

The configuration of $PE(i,2j), PE(i,2j+1), 0 \leq i \leq k$ is determined by input b_j as illustrated in Figure 4(a) for $k = 4$. Apply a *1-signal* at the W port of $PE(0,0)$. PEs in column $(2j+1)$ check their E port. If a *1-signal* is received at the E port of $PE(r, 2j+1)$, $\sum_{w=0}^{j} b_w \bmod k = r$ (see Figure 4 (b)). □

It follows from the Chinese Remainder Theorem[2] that if $A \equiv B \bmod k$ for $2 \leq k \leq n$, then $A \equiv B \bmod \text{LCM}(2 \times 3 \times \ldots \times n)$. Since there exists constant c, such that for all n, $n \geq 2$, $\text{LCM}(2 \times 3 \times \ldots \times cn) > 2^n$, if $A, B \in [0, 2^n - 1]$ and $A \equiv B \bmod k$, for $2 \leq k \leq cn$, then, $A = B$. Let $\bar{c}$ be the constant. By combining Lemma 3 and the above fact, we have:

Corollary 1 *Counting the number of 1's in a 0/1 sequence of length N can be performed in $O(1)$ time on $\log^2 N \times N$ reconfigurable mesh.*

Proof: Let $b_j, 0 \leq j \leq N-1$ be the input 0/1 sequence. The proof is based on a similar technique in[16]. For ease of explanation, we use $\bar{c}^2 \log^2 N \times (2N+2)$ reconfigurable mesh, where $\bar{c}$ is the constant defined above. Divide the reconfigurable mesh into $RM(i), 2 \leq i \leq \bar{c} \log N$, where $RM(i)$ is a submesh of size $(i+1) \times (2N+2)$(see Figure 5). From Lemma 3, $RM(i)$ is capable of computing for each j, $0 \leq j \leq N$, $j \bmod i$(by prefix modular i computation on sequence $0111\ldots111$ of length $N+1$). $j \bmod i$ is stored in $(2j+1)$-th column of $RM(i)$, $0 \leq j \leq N-1$. Using Lemma 3, $(\sum_{j=0}^{N-1} b_j) \bmod i$ can be computed by $RM(i)$. $(\sum_{j=0}^{N-1} b_j) \bmod i$ is broadcast to all the columns of $RM(i)$. For $0 \leq j \leq N$ in parallel, $(2j+1)$-th column checks if $(\sum_{j=0}^{N-1} b_j) \bmod i$ is equal to $j \bmod i$ or not. If so, let all the PEs in the $(2j+1)$-th column to have a *1-signal*. Now, $RM(i), 2 \leq i \leq \bar{c} \log N$ are vertically cascaded into a reconfigurable mesh. There is a unique column r such that all PEs in the $(2r+1)$-th column of the mesh have *1-signal*. Then, $\sum_{j=0}^{N-1} b_j = r$. □

Combining Lemma 1 and Corollary 1, we have:

Theorem 2 *Counting number of 1's in a 0/1 array of size $N \times N$ can be performed in $O(\log^* N)$ time on an $N \times N$ reconfigurable mesh.*

Proof: Assume the number of 1's in each subarray of size $l^4 \times l^4$ has been computed in $T(l^4)$ time. Consider submeshes of size $2^l \times 2^l$. Divide each submesh into blocks of size $l^4 \times l^4$. Let $N(i,j)$ be the number of 1's in the (i,j)-th block of size $l^4 \times l^4$, $0 \leq i,j \leq 2^l/l^4 - 1$. We will show how to merge $(2^l/l^4 \times 2^l/l^4)$ intermediate results on $2^l \times 2^l$ reconfigurable mesh. Since $N(i,j) \leq l^8$, each $N(i,j)$ can be represented by $8 \log l$ bits. First, we show how to compute $\sum_{j=0}^{2^l/l^4-1} N(i,j)$ on a $l^4 \times 2^l$ reconfigurable mesh. Thus, we are adding $2^l/l^4$ numbers with each number represented using $8 \log l$ bits. For $0 \leq j \leq 2^l/l^4 - 1$, collect the bits having the same significance from the $8 \log l$-bit representation of $N(i,j)$ to form $8 \log l$ 0/1 sequences, each of length $2^l/l^4$.

Count the number of 1's in the p-th sequence and multiply the count by 2^p, $0 \leq p \leq 8\log l - 1$. This can be performed in $O(1)$ time on submesh of size $l^2 \times 2^l$ using Corollary 1. Since $8\log l \times l^2 \leq l^4$, this can be performed in parallel for all the $8\log l$ bit positions. As a result, $\sum_{j=0}^{2^l/l^4-1} N(i,j)$ is reduced to $8\log l$ partial sums with each partial sum represented using $l+3\log l$ bits. These can be added into a $l+3\log l$-bit number on a $l^4 \times 2^l$ reconfigurable mesh using Lemma 1 in $O(1)$ time. Now, we compute $\sum_{i=0}^{2^l/l^4-1}\sum_{j=0}^{2^l/l^4-1} N(i,j)$ in $O(1)$ time on $2^l \times 2^l$ reconfigurable mesh. This reduces to adding $2^l/l^4$ numbers in $l+4\log l$-bit representation. $\sum_{i=0}^{2^l/l^4-1}\sum_{j=0}^{2^l/l^4-1} N(i,j)$ can be reduced to $l+4\log l$ $2l$-bit numbers using the above technique for counting the number of 1's having the same weight. Recall that a 0/1 sequence of length $2^l/l^4$ can be summed up on a submesh of size $2^l \times l^2$ in $O(1)$ time. The resulting $l+4\log l$ $2l$-bit numbers can be added into a number in $O(1)$ time on a $2^l \times 2^l$ reconfigurable mesh using Lemma 1. Thus, we have $T(2^l) = T(l^4) + O(1)$, which implies $T(N) = O(\log^* N)$.□

The above result can be extended to the k-dimension. For this purpose, we define a new number representation for a r-dimensional reconfigurable mesh of size $\overbrace{N \times N \times \ldots \times N}^{k \text{ times}}$ (called r-mesh hereafter). The PEs are numbered in a snake-like lexicographic ordering. For 2-mesh, it is same as the snake-like row major ordering in which PEs are numbered from left to right(right to left) in even(odd) numbered rows. For 3-mesh, the numbering of PEs in odd numbered planes is exactly opposite to that of the PEs in the even numbered planes. The PEs in a r-mesh are numbered, in a snake-like manner, from 0 to $N^r - 1$. A number x in $[0, N^r]$ is represented by the *1-signals* in the PEs numbered 0 to x. This is defined as the 1UN representation of x using a r-mesh. Let $G(i,-)$ or $G(i,+)$ be the I/O ports located in the negative or positive direction along dimension i. Note that the N ports of the PEs in a 2-mesh are $G(2,-)$ ports.

Lemma 4 *Given an integer x, $0 \leq x \leq N^r - 1$, in 1UN representation, an r-mesh contained in an $(r+1)$-mesh can be configured as "increment 1" to obtain $x+1$ in 1UN, "decrement 1" to obtain $x-1$ in 1UN or "no change" to keep x in 1UN in $O(1)$ time.*

Proof: Assume that a specific I/O port (for example $G(r+1,-)$) of the k-th PE has the k-th bit (representing existence or absence of *1-signal*) of the 1UN representation of x. For "increment 1", connect the port to the I/O port on the way to the next (in the snake-like ordering) PE. As an example, consider a row (1-mesh) in a 2-mesh. The 1UN representation of x is applied to $G(2,-)$ ports of the N PEs. Connect the $G(2,-)$ and the $G(1,+)$, the $G(1,-)$ and the $G(2,+)$ of each PE. Note that the output $x+1$ in 1UN represenation will

be available at the $G(2,-)$ of the N PEs.

The "decrement 1" can be similarly implemented. The "no change" can be implemented by simply connecting the designated input ports to the designated output ports in each PE. □

Lemma 5 *Given an integer x, $0 \leq x \leq N^r - 1$ in 1UN representation, the $x + y$ and the $\lfloor x/2 \rfloor$, where y is the count of 1's in a 0/1 sequence of length N^r, can be obtained in $O(1)$ time using a $2r$-mesh.*

Proof: For simplicity of proof, assume $x, y \leq N^{r/2}$. Note that a $2r$-mesh can be regarded as a casacde of N^r r-meshes. This is due to the fact that there exist N^r connections between two r-meshes. To compute $x+y$, assign the j-th bit of the 0/1 sequence to the j-th r-mesh, $0 \leq j \leq N^r - 1$. If the bit is 1, configure it as "increment 1". or if the bit is 0, configure the r-mesh as "no change" using Lemma 4. The configuration of each r-mesh can be determined in $O(1)$ time once the 0/1 sequence is given. In $O(1)$ time after the x in 1UN representation is applied to the first r-mesh, the output $x+y$ will be available in the last r-mesh. The $\lfloor x/2 \rfloor$ can be obtained as follows. The configuration is arranged in such a way that the $2l$-th PE of the first r-mesh has a disjoint and unbroken path to the l-th PE of the last r-mesh. This can be systematically performed by combining "decrement 1" and 'no change" in Lemma 4. The case of $r = 1$ is shown in[5]. □

Lemmas 4 and 5 can be combined to result in the following theorem:

Theorem 3 *Addition of N^r $r \log N$-bit numbers, $r \leq N$, can be performed in $O(1)$ time using a $(2r+1)$-dimensional reconfigurable mesh of size $\overbrace{N \times N \times \ldots \times N}^{2r \text{ times}} \times r \log N$, $1 \leq r \leq N$.*

Proof: Consider the serial paper-and-pencil method for the addition of the N^r $r \log N$-bit numbers. For each bit position, the carry-in (no greater than N^r) is added to the sum (the number of 1's among the N^r bits in that bit position). The result is divided by 2 to provide the carry-out to the next (more significant) bit position and the remainder (0 or 1) is left as the output bit of that bit position. To achieve $O(1)$ time complexity, all the carries should be computed in $O(1)$ time. Distribute the input data so that the j-th $2r$-mesh has N^r bits from the j-th bit positions (thus having the same weight). Use Lemma 5 to configure each $2r$-mesh as " $\lfloor (x+y)/2 \rfloor$ ". In fact, two $2r$-meshes would be needed but a $2r$-mesh can simulate the two $2r$-meshes for this purpose. Now apply 0 as the carry-in to the first $2r$-mesh. Then carry-out for the j-th bit position will be available on the j-th $2r$-mesh in $O(1)$ time, $0 \leq j \leq r \log N - 1$. Carry-outs for the next $r \log N$ bit positions can be obtained in $O(1)$ time once the carry-out of the $(r \log N - 1)$-th bit position is known. □

Divide-and-conquer technique can be combined with Theorem 3 for reduction of the needed mesh size. For example, addition of N^r $r \log N$-bit numbers can be performed in two phases: divide the N^r numbers into $N^{r/2}$ groups of $N^{r/2}$ numbers each and perform addition within each group. From Theorem 3, each group needs only $r \log N$ r-meshes. Thus, $rN^{r/2} \log N$ r-meshes would be sufficient for this phase. After this we have no more than $N^{r/2}$ partial sums with less than $2r \log N$ bits to represent each sum. The $N^{r/2}$ partial sums can be merged using $2r \log N$ r-meshes. Thus we can have the following:

Corollary 2 *N^r $r \log N$-bit numbers can be added in $O(1)$ time using a $2r$-mesh.*

From Theorem 2 and Corollary 2, we have the following:

Theorem 4 *Given a 0/1 array of size $\overbrace{N \times N \times \ldots \times N}^{k \text{ times}}$, the count of 1's in the array can be obtained in $O(\log^* N + \log k)$ time on a k-dimensional reconfigurable mesh of size $\overbrace{N \times N \times \ldots \times N}^{k \text{ times}}$.*

Proof: From Theorem 2, the number of 1's in each subarray of size $N \times N$ can be obtained in $O(\log^* N)$ time using a 2-mesh. Each result is represented by $2 \log N$ bits. Thus, in a 4-mesh we have N^2 $2 \log N$-bit numbers. From Corollary 2, the numbers can be added in $O(1)$ time using a 4-mesh. Similarly, 8-mesh can merge N^4 partial results in $O(1)$ time. Continuing in this manner, the final result can be obtained after $\log k$ merging steps. Thus, overall time complexity is $O(\log^* N + \log k)$. □

4 Conclusion

We have shown that N^k numbers can be sorted in $O(4^k)$ time on a $(k+1)$-dimensional reconfigurable mesh of size $\overbrace{N \times N \times \ldots \times N}^{k+1 \text{ times}}$. Compared with [19], our sorting algorithm reduces the number of PEs by a factor of N when $k = 1$. Since the time complexities of many computational geometry problems are lower bounded by the complexity of the sorting, the result would have wide application for geometry problems involving k-dimensional points.

In addition, it is shown that the number of 1's in a 0/1 array of size $\overbrace{N \times N \times \ldots \times N}^{k \text{ times}}$ can be computed in $O(\log^* N + \log k)$ time on a reconfigurable mesh of size $\overbrace{N \times N \times \ldots \times N}^{k \text{ times}}$.

Acknowledgments

This research was supported in part by KOSEF under contract 961-0909-052-1.

References

1. A. Aggarwal, "Optimal bounds for finding maximum on array of processors with k global buses," *IEEE Trans. on Computers,* pp. 62-64, 1986.
2. A. Baker, "A Concise Introduction to the Theory of Numbers", *Cambridge University Press*, 1984.
3. Y. Ben-Asher, D. Peleg, R. Ramaswami, A. Schuster, "The Power of Reconfiguration", *Journal of Parallel and Distributed Computing*, V. 13, No. 2, pp. 139-153, 1991.
4. D. M. Champion and J. Rothstein, "Immediate parallel solution of the longest common subsequence problem," *Proc. of Inter. Conf. on Parallel Processing,* pp. 70-77, 1987.
5. J. Jang and V. K. Prasanna, "An optimal sorting algorithm on reconfigurable mesh," *Journal of Parallel and Distributed Computing*, Vol. 25, pp. 31-41, 1995.
6. J. Jang, H. Park, V. K. Prasanna, "A fast algorithm for computing histogram on reconfigurable mesh," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 17, No. 2, 1995.
7. J. Jang, M. Nigam, S. Sahni, V. K. Prasanna, "Constant time algorithms for computational geometry on the reconfigurable mesh," *IEEE Transactions on Parallel and Distributed Systems,* Vol. 8, No. 1, pp. 1-12, 1997.
8. J. Jang, H. Park, V. K. Prasanna, "An optimal multiplication algorithm on reconfigurable mesh," *To appear IEEE Transactions on Parallel and Distributed Systems,*
9. J. Jenq and S. Sahni, "Rreconfigurable Mesh Algorithms for the Area and Perimeter of Image Components", *Proc. ICPP*, The Pennsylvania State University Press, 280-281, 1991.
10. J. Jenq and S. Sahni, "Reconfigurable mesh algorithms for image shrinking, expanding, clustering, and template matching," *Proc. Inter. Parallel Processing Symp.,* pp. 208-215, 1991.
11. F. T. Leighton, "Tight bounds on the complexity of parallel sorting," *IEEE Trans. on Computers,* pp. 344-354, 1985.
12. H. Li and M. Maresca, "Polymorphic-Torus network," *IEEE Transactions on Computers,* pp. 1345-1351, September 1989.

13. R. Miller, V. K. Prasanna Kumar, D. I. Reisis and Q. F. Stout, "Meshes with reconfigurable buses," *MIT Conf. on Advanced Research in VLSI,* pp. 163-178, 1988.
14. R. Miller, V. K. Prasanna Kumar, D. Reisis and Q. F. Stout, " Image Computations on Reconfigurable Mesh", *Proc. Computer Vision and Pattern Recognition*, pp. 925-930, 1988
15. R. Miller and Q. F. Stout, "Mesh Computer Algorithms for Computational Geometry", *IEEE Transactions on Computers*, V.38, no. 3, pp. 321- 340, March 1989
16. K. Nakano, T. Masuzawa, and N. Tokura, "A sub-logarithmic time sorting algorithm on a reconfigurable mesh," *IEICE Transactions,* Vol. E 74, No. 11, 3894-3901, Nov. 1991.
17. L. Snyder, "Introduction to the configurable highly parallel computer," *Computer,* 15(1), pp. 47-56, 1982.
18. Q. F. Stout, "Meshes with multiple buses," *Proc. FOCS,* pp. 264-272, 1986.
19. B. F. Wang, G. H. Chen and F. C. Lin, "Constant time sorting on a processor array with a reconfigurable bus systems," *Info. Processing Letters,* pp. 187-192, 1990.
20. B. F. Wang and G. H. Chen, "Constant time algorithms for the transitive closure problem and some related graph problems with reconfigurable bus systems," *IEEE Trans. on Parallel and Distributed Systems,* pp. 500-507, 1991.