A Fast Algorithm for Computing a Histogram on Reconfigurable Mesh

Ju-wook Jang, Heonchul Park and Viktor K. Prasanna

Abstract — **The reconfigurable mesh captures salient features from a variety of sources, including the CAAPP, the CHiP, the polymorphic-torus network and the bus automaton. It consists of an array of processors interconnected by a reconfigurable bus system. The bus system can be used to dynamically obtain various interconnection patterns between the processors. In this paper, we present a fast algorithm for computing the histogram of an $N \times N$ image with h grey levels in**

$$O\left(\min\left\{\sqrt{h} + \log^*(N/h), N\right\}\right)$$

time on an $N \times N$ reconfigurable mesh assuming each PE has a constant amount of local memory. This algorithm runs on the PARBUS and MRN/LRN models. In addition, histogram modification can be performed in $O(\sqrt{h})$ time on the same model.

A variant of our algorithm runs in

$$O\left(\min\left\{\sqrt{h} + \log\log(N/h), N\right\}\right)$$

time on an $N \times N$ RMESH in which each PE has constant storage. This result improves the known time and memory bounds for histogramming on the RMESH model.

Index Items — **Histogram, reconfigurable mesh, mapping, parallel algorithm**

I. Introduction

The reconfigurable mesh has been introduced in [20]. This parallel model of computation captures the fundamental properties of the CHiP computer [31], mesh connected computers augmented with broadcast buses [28], the bus automaton [30], the polymorphic-torus network [17], and the coterie network in the latest version of the Content Addressable Array Parallel Processor (CAAPP) [39]. Such an architecture has been realized using Field Programmable Gate Arrays(FPGAs) [35]. Many algorithms on the reconfigurable mesh are known [1], [3], [6], [9], [17], [20], [21], [24], [27], [29], [36], [38]. The proceedings of the Reconfigurable Architectures Workshop [40] held at the International Parallel Processing Symposium (IPPS) summarizes recent work in this area.

Computing a histogram is a basic operation in image processing and computer vision. It can provide useful information for segmentation and measuring the textual properties of a digitized image. Given an $N \times N$ image with h grey level values, the histogramming problem is to compute the number of occurrences of these values.

In [15] computing a histogram and histogram modification are performed in

$$O\left(\min\left\{\sqrt{h}\log\left(N/\sqrt{h}\right), N\right\}\right)$$

time[1] and in $O(\sqrt{h})$ time, respectively, on a restricted model of the reconfigurable mesh (denoted RMESH) of size $N \times N$ (see Section II for details). This result assumes that each Processing Element (PE) in the array has $O(\sqrt{h})$ memory. Olariu, Schwing and Zhang solve this problem in $O(\log\log N)$ time on a stronger model of the reconfigurable mesh (denoted PARBUS) of size $N \times N$, assuming h is a constant [26]. Several architectures can compute the histogram in $O(\log N)$ time [5], [19], assuming h is a constant. In [32] it is shown that computing the histogram on a pyramid computer having a base of size $N \times N$ can be performed in $O(h + \log N)$ time.

In this paper, we present an

$$O\left(\min\left\{\sqrt{h} + \log^*(N/h), N\right\}\right)$$

time algorithm[2] for computing the histogram of an $N \times N$ image using the reconfigurable mesh of size $N \times N$, assuming each PE has a constant amount of memory. The reconfigurable mesh algorithm runs on the PARBUS as well as on MRN/LRN models. If we assume h to be a constant, then our algorithm runs in $O(\log^* N)$ time. This improves the best known result [26] by a factor of $O(\log\log N / \log^* N)$. Our algorithm can be modified to run in

$$O\left(\min\left\{\sqrt{h} + \log\log(N/h), N\right\}\right)$$

(RMESH) of size $N \times N$. Compared with the result in [15], our algorithm improves the time complexity while using a constant amount of storage in each PE. Wang and Chen [37] have shown that an $N \times M$ reconfigurable mesh can simulate the CRCW PRAM model having N PEs and M memory modules

Manuscript received April 13, 1992; revised September 2, 1994. Recommended for acceptance by Dr. J. Little.

J. Jang is with Samsung Electronics Co. Ltd., APT 2-506, Okeumdong 160, Kongpa-Ku, Seoul, Korea 138-130.

H. Park is with DSP Team, Samsung Electronics Co. Ltd., Suwon PO Box 105, SAIT-Bldg. 3, Nongseo-Ri, Kiheung-Yup, Youngin-Kun Kyoungki-Do, Korea.

V.K. Prasanna is with the Department of Electrical Engineering-Systems, EEB-244, University of Southern California, Los Angeles, CA 90089-2562, e-mail prasanna@halcyon.usc.edu.

IEEECS Log Number P95017.

[1] All logarithms in this paper are to base 2.

[2] $\log^* n = min\ \{i \geq 0 : \log^{(i)} n \leq 1\}$, where $\log^{(i)} n$ is the logarithm functic applied i times in succession.

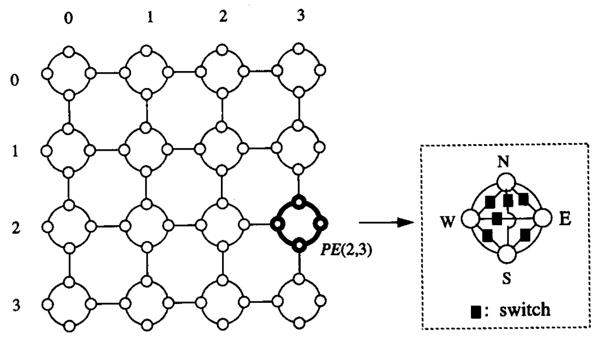


Fig. 1. A 4 x 4 reconfigurable mesh.

without asymptotic loss in time. Note that computing the histogram of an $N \times N$ image cannot be performed faster than $\Omega(\log N/\log\log N)$ time on a CRCW PRAM having N^2 PEs, since the addition of N numbers takes $\Omega(\log N/\log\log N)$ time on this model having N PEs [8]. Our histogramming algorithm is an example for which the reconfigurable mesh is more powerful than the CRCW PRAM model. In addition, we show that the histogram modification can be performed in $O(\sqrt{h})$ time on RMESH as well as on MRN/LRN and on PARBUS models.

This paper is organized as follows. In Section II, we briefly introduce the reconfigurable mesh architecture. In Section III, for the sake of completeness as well as for the sake of illustration, some basic operations on the reconfigurable mesh are discussed. Main results are contained in Section IV and concluding remarks are made in Section V.

II. The Reconfigurable Mesh Model

For the sake of completeness, we briefly define the reconfigurable mesh model and some variants of it.

A. The Reconfigurable Mesh

The reconfigurable mesh architecture used in this paper is based on the architecture defined in [20]. The $N \times N$ reconfigurable mesh consists of an $N \times N$ array of PEs connected to a grid-shaped reconfigurable broadcast bus. A 4×4 reconfigurable mesh is shown in Fig. 1. Each PE has locally controllable bus switches. Internal connection among the four ports (N, E, W, and S) of a PE can be configured during the execution of algorithms. Note that, there are 15 possible connection patterns. For example, {SW, EN} 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 one of *1-signal* or *0-signal* at any time. The switches allow 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 the PEs. Other than the buses and the switches, the reconfigurable mesh is similar to the standard 2-dimensional mesh in that has $\Theta(N^2)$ area, under the assumption that PEs, switches, and a link between adjacent PEs occupy unit area in the word model of VLSI. In this paper, we use the exclusive write model which allows only one PE to broadcast to a *subbus* shared by multiple PEs at any given time. We assume that the value broadcast consists of $O(\log N + \log h)$ bits and takes $\Theta(1)$ time. Also, we assume that each PE can perform an arithmetic and logic operation on $O(1)$ words in unit time. The size of the local storage in each PE is $O(1)$ words, where each word is $\Theta(\log N + \log h)$ bits. Note that this model is a variant of the standard word model of reconfigurable mesh in the literature.

B. Related Models

After the definition of the reconfigurable mesh in [20], other models have been defined [3], [24], [36]. These models restrict the allowed connection patterns. The most general and the powerful model among these is the PARBUS model [36] which allows any combination of four-port connections in each PE. Notice that the PARBUS model is same as our model in Section II.A. In [3] the Reconfigurable Network (RN) model is introduced and several algorithms on this model are derived under the mesh restriction. This model has been denoted as MRN in [24] and LRN in [4]. The connection patterns allowed in the MRN/LRN model are shown in Fig. 2.

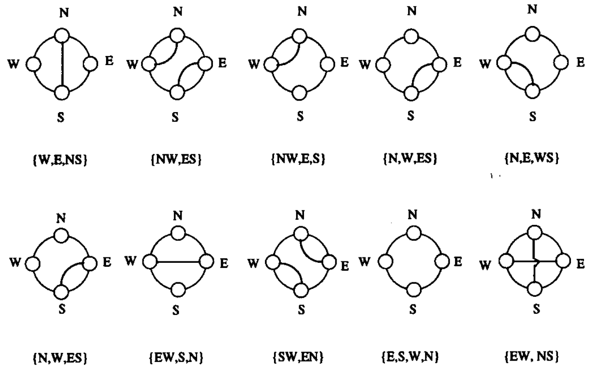


Fig. 2. Connection patterns allowed in MRN/LRN.

In the MRN model, the number of possible connection patterns in each PE is 10. In [15] the RMESH model is introduced, which does not allow {NS, EW}, {NE, SW}, and {NW, SE} connections allowed in the MRN model. However, the RMESH model allows {NEWS}, {NEW, S}, {NES, W}, {NWS, E}, and {N, EWS} connections. Thus, the total number of possible connection patterns in each PE of the RMESH model is 12. This corresponds to having switches on the mesh links only. The set of allowed connection patterns in the RMESH models is shown in Fig. 3. Notice that the set of all connection patterns allowed in a PE in the reconfigurable mesh is the union of the connection patterns allowed in a PE in the MRN and RMESH models.

The reconfigurable mesh algorithms derived in this paper (Theorem 1) can be simulated on the MRN model of same size without slowdown, since we only employ the connection patterns shown in Fig. 2. Our reconfigurable mesh algorithm can

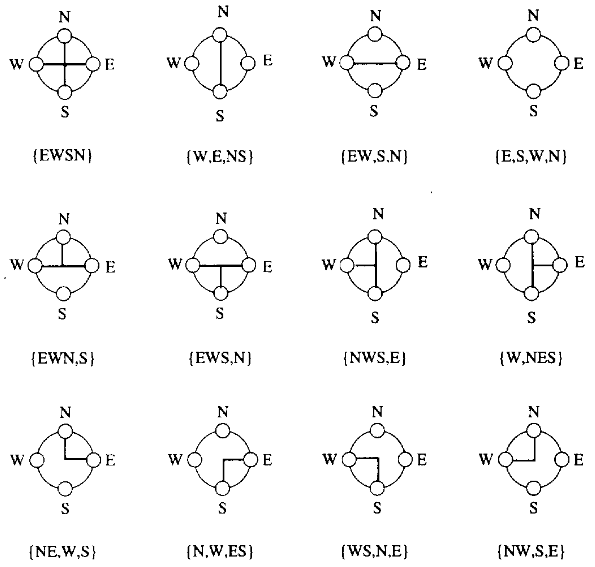


Fig. 3. Connection patterns allowed in RMESH.

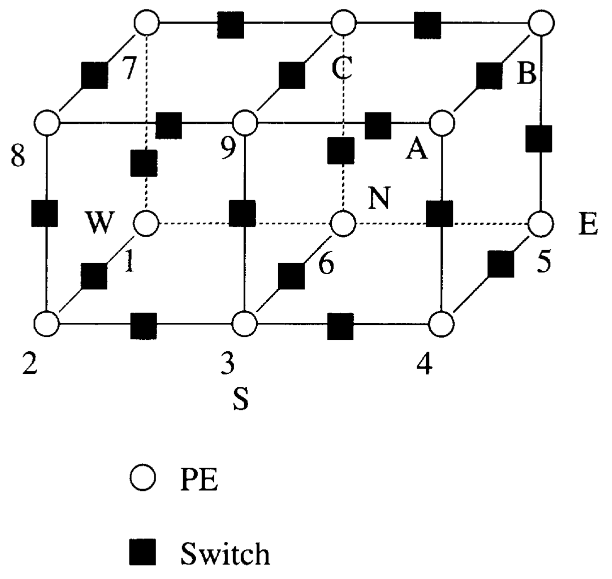


Fig. 4. Two-layer RMESH model.

also be simulated by a two-layer RMESH without asymptotic loss in time. In the two-layer RMESH, each PE has at most five local links and the switches are located on these links. By assigning two sets of six PEs in the two-layer RMESH to simulate a PE in the reconfigurable mesh, as shown in Fig. 4, the two-layer RMESH can simulate any reconfigurable mesh algorithm without asymptotic loss in time. The PEs are numbered 1, 2, ⋯, *A*, *B*, *C*. For example, {NS, EW} connection pattern in the reconfigurable mesh (as well as in the MRN) can be simulated by {36,17CB5} connection pattern by assigning PEs 1, 3, 5, and 6 as N, E, W, and S ports, respectively. The {NE, SW} connection in the reconfigurable mesh can be simulated by the {56,123} connection in the two-layer RMESH. Thus, the two-layer RMESH can simulate any algorithm on the reconfigurable mesh (and MRN) of corresponding size without loss in time. Throughout this paper, reconfigurable mesh refers to the model defined in Section II.A. Lin and Olariu define a variant of the reconfigurable mesh model denoted REBSIS [18]. Another variant of the model can be found in [34].

C. Histogram Problem

The histogram problem can be defined as follows: given an input image $I(u,v)$ of size $N \times N$, $0 \leq I(u,v) \leq h-1$, where $0 \leq u$, $v \leq N-1$, for each grey level, find the number (count) of occurrences of the grey level in the image (i. e., for each t, $0 \leq t \leq h-1$, compute $F(t) = |\{I(u,v)\colon I(u,v) = t\}|$).

One of the common operations after computing the histogram of an image is histogram modification. The histogram modification problem is to map the original grey level value of each pixel onto a new grey level value using a given function. The histogram modification provides a desired distribution by approximating the histogram to improve the utilization of the available range of grey levels and enhance the contrast of the image.

III. Number Representations and Addition

In this section, we introduce several number representations on the reconfigurable mesh and show a fast algorithm for adding numbers on the reconfigurable mesh. These notations and results appear in our work on sorting. These have been included for the sake of completeness and to illustrate the features of the reconfigurable mesh model.

A. Number Representations on the Reconfigurable Mesh

Besides the usual binary representation of numbers (using $\Theta(\log N)$ bits) within a PE, the input as well as the intermediate results can be represented using the I/O ports of a group of PEs in the reconfigurable mesh. For the sake of simplicity, assume N is a power of 2. In the following four representations, a particular I/O port of each PE (E, W, N, S) is designated to carry a *0-signal* or a *1-signal* to represent a number. Integer i, $0 \leq i \leq N-1$ can be represented as follows.

BIN representation: uses $\log n$ PEs to represent an integer i in $[0, n-1]$. The designated port of PE k is set to a *1-signal* iff the k-th bit of the $\log n$ bit representation of i is equal to 1, $0 \leq k \leq \log n - 1$.

1) POS representation: uses n PEs to represent an integer i in $[0, n-1]$. The designated port of PE i carries a *1-signal* and the designated ports of the rest of the PEs carry a *0-signal*.
2) 1UN representation: uses n PEs to represent an integer i

in [0, $n - 1$]. The designated port of *PE k*, $0 \le k \le i$ carries a *1-signal* and the designated ports of the rest of the PEs carry a *0-signal.*

3) 2UN representation: uses n PEs to represent an integer i in [0, $n - 1$]. The designated ports of some subset of i PEs carry a *1-signal* and the designated ports of the rest of the PEs carry a *0-signal.* Note that the 2UN representation of an integer is not unique.

Fig. 5 shows the above representations for number 3. The presence of a *1-signal* at a port is represented by a dark circle.

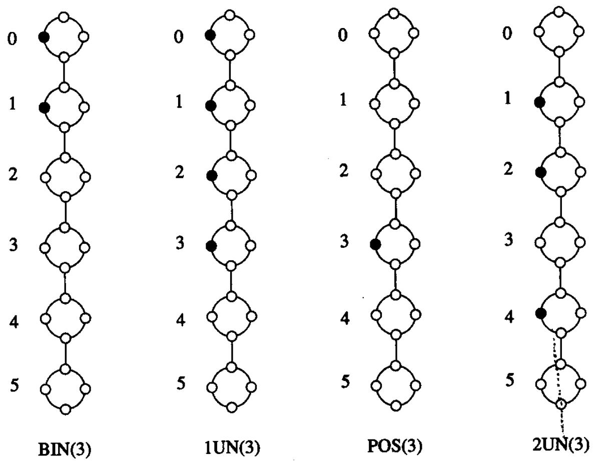


Fig. 5. Four different representations of number 3.

B. Conversion between Number Representations

We will show how a given representation of an integer x in [0,$N - 1$] can be converted in $O(1)$ time into another representation:

1) $\rightarrow$ BIN(x): This can be performed on a reconfigurable mesh of size $1 \times \log N$. The input x is in a register in a PE. This PE broadcasts the $\log N$ bit binary representation of x. A bitwise AND operation of x with 2^i is performed in *PE*(0, i), $0 \le i \le \log N - 1$. If the result is nonzero, then *PE*(0, i) generates a *1-signal* at its designated port, otherwise it generates a *0-signal.*
2) BIN(x) $\rightarrow x$: The mesh size is $\log N \times N$. Assume BIN(x) is available in the leftmost column. For $0 \le i \le N - 1$, in parallel, the ith column generates BIN(i). The leftmost column broadcasts BIN(x) to all the columns. If BIN(x) = BIN(i), then *PE*(0, i) outputs i.
3) POS(x) $\rightarrow$ 1UN(x): The mesh size is $1 \times N$. *PE*(0, x) sends a *1-signal* to all the PEs to its left and a *0-signal* to all the PEs to its right. The PEs set their designated port to the received signal.
4) 1UN(x) $\rightarrow$ POS(x): The mesh size is $1 \times N$. For $0 \le i \le N - 2$, in parallel, *PE*(0, i) checks if its right neighbor PE has a *1-signal.* If *PE*(0, $I + 1$) has a *1-signal*, then *PE*(0, i) resets its *1-signal.*
5) POS(x) $\rightarrow x$: The mesh size is $1 \times N$. There is only one PE (i.e., *PE*(0, x)) having a *1-signal* in the POS representation. It outputs x.
6) $x \rightarrow$ POS(x): The mesh size is $1 \times N$. The PE having x broadcasts it to all the PEs. If $i = x$, then *PE*(0, i) outputs a *1-signal* at the designated port, otherwise the PE outputs a *0-signal.*
7) 2UN(x) $\rightarrow$ 1UN(x): The mesh size is $\log^2 N \times N$. See Lemma 5.

By combining the above conversion techniques, we can perform conversion between any given pair of representations in $O(1)$ time. For example, 1UN(x) $\rightarrow$ BIN(x) can be performed on a reconfigurable mesh of size $1 \times N$ by a sequence of conversions, 1UN(x) $\rightarrow$ POS(x); POS(x) $\rightarrow x$; $x \rightarrow$ BIN(x).

All the above conversion techniques can be employed on the MRN/LRN models of corresponding size.

C. Addition of N Numbers

Given N k-bit binary numbers, $1 \le k \le N$, the *addition problem* is to add these numbers into a ($k + \log N$)-bit binary number. Let the N k-bit numbers be X_i, $0 \le i \le N - 1$, such that

$$X_i = \sum_{j=0}^{k-1} x_{i,j} 2^j$$

and the sum of these numbers be represented by

$$\sum_{i=0}^{N-1} X_i = \sum_{j=0}^{k-1+\log N} z_j 2^j .$$

In the following, we show that this problem can be solved in $O(1)$ time using a mesh of size $N \times Nk$. Initially, $x_{i,j}$ is stored in PE(i, $2jN$) as shown in Fig. 6, $0 \le i \le N - 1$, $0 \le j \le k - 1$.

Lemma 1. Given N k-bit numbers in the BIN representation, $1 \le k \le N$, these numbers can be added in O(1) time on an $N \times Nk$ reconfigurable mesh.

Proof: For the ease of explanation, we use a reconfigurable mesh of size $2N \times 2Nk$. Our algorithm is based on a sequential paper-and-pencil method shown in Fig. 7. While the (sequential) method computes the carries sequentially, all the k least significant carries are computed simultaneously in ou algorithm. The computation of the remaining $\log N$ most sig nificant carries is not necessary since these bits can be ob tained directly once (all the bits of) the kth least significar carry is known. A carry is computed in a block of size $2N \times 2$. by configuring the buses and using the number representatioı introduced in Section III.A. In each block, the buses are co figured based on the input bits. Also, the blocks are configur such that a cascade of k blocks can compute all the k carri simultaneously. Each carry is output in 1UN representatic Let 1UN(C_j), $1 \le j \le k$ be the resulting carry-in at the jth le significant bit position. The k least significant output bits (z_j $\le j \le k - 1$) can be obtained in $O(1)$ time using the inputs a the computed carries. Note that BIN(C_k) is nothing but {z_j} formal proof including an illustration of the configuration the buses is shown below.

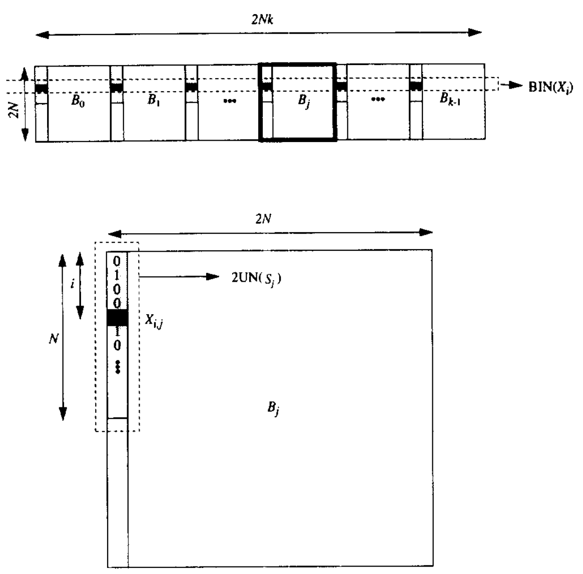


Fig. 6. Initial distribution of the input.

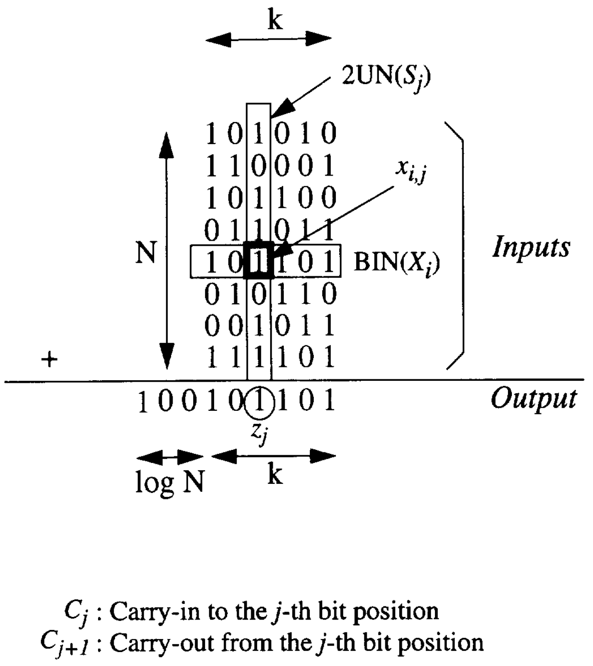

k
2UN(S_j)
$x_{i,j}$
BIN(X_i)
N
Inputs
+
Output
z_j
log N
k
C_j : Carry-in to the j-th bit position
C_{j+1} : Carry-out from the j-th bit position

Fig. 7. Addition of N k-bit numbers for $N = 8$ and $k = 6$.

Without loss of generality, assume N is a power of 2. Output z_j, $0 \le j \le k - 1 + \log N$, can be computed as follows: $\le j \le k + \log N - 1$. BIN(C_k) can be obtained from 1UN(C_k) using the conversion techniques shown in Section III.B. A

$$\text{Let, } S_j = \sum_{i=0}^{N-1} x_{i,j} \quad 0 \le j \le k-1,$$
$$= 0, \quad k \le j \le k - 1 + \log N,$$
$$C_{j+1} = \left(S_j + C_j\right) div 2, \ 0 \le j \le k - 2 + \log N, \text{ and } C_0 = 0$$
$$\text{Then, } z_j = \left(S_j + C_j\right) mod 2, \quad 0 \le j \le k - 1 + \log N$$

The *mod* 2 and *div* 2 functions give the remainder and the quotient, respectively, when the input number is divided by 2. If we can compute carries, C_j, for $1 \le j \le k$, in $O(1)$ time, the (k + log N) output bits can be obtained in $O(1)$ time. Note that, $0 \le C_j \le N - 1$, $0 \le j \le k$.

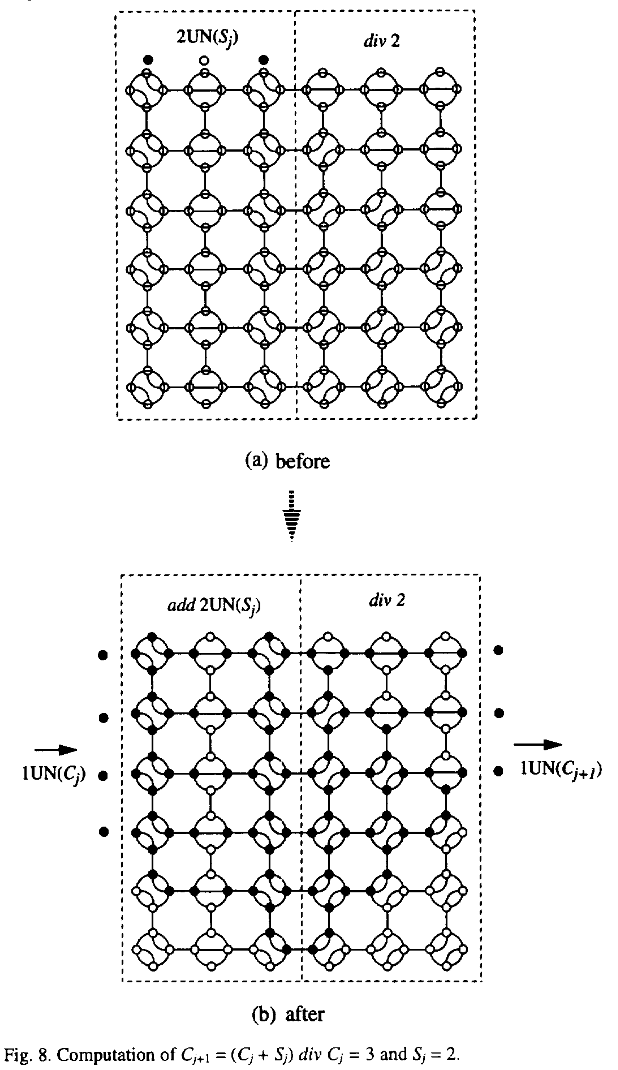


Fig. 8. Computation of $C_{j+1} = (C_j + S_j)$ *div* $C_j = 3$ and $S_j = 2$.

In our solution, blocks of size $2N \times 2N$ compute 1UN(C_{j+1}) = (2UN(S_j) + 1UN(C_j)) *div* 2, for $0 \le j \le k - 1$. S_j is not computed but is available as 2UN(S_j) in the $2jN$-th column. Initially, $x_{i,j}$ is stored in $PE(i,2jN)$, $0 \le i \le N - 1$, $0 \le j \le k - 1$.

Note that the 0/1 sequence, $\{x_{i,j}\}$, $0 \le i \le N-1$ stored in $PE(i,2jN)$ represents $2\mathrm{UN}(S_j)$. Fig. 8 shows an example where $1\mathrm{UN}(C_j) = 1\mathrm{UN}(3)$, $2\mathrm{UN}(S_j) = 2\mathrm{UN}(3)$ and $N = 3$. Initially, $2\mathrm{UN}(S_j)$ (= 1 0 1 in this example) is available in $PE(0, 0)$, $PE(1, 0)$, and $PE(2, 0)$. The bit in $PE(i, 0)$ is routed to $PE(0, i)$ in $O(1)$ time, $0 \le i \le 2$. The configuration of the PEs for the *add* S_j operation depends on $2\mathrm{UN}(S_j)$; however, the configuration is used to control the configuration of the PEs in a column. If the bit is a '1', then all the PEs in its column set their configuration to {SW, NE}. If the bit is a '0', then the PEs set their configuration to {EW, N, S}. Now $1\mathrm{UN}(C_j)$ is applied at the W port of the leftmost column of the the block. It is easy to verify that, $1\mathrm{UN}(C_{j+1})$ is available in the rightmost column of the block. Cascading k such blocks, $1\mathrm{UN}(C_{j+1})$, $0 \le j \le k-1$, can be computed in $O(1)$ time. The *mod* 2 function can be computed in $O(1)$ time by configuring the mesh as in the *div* 2 computation. Using the carries, z_j, $0 \le j \le k + \log N - 1$ can be computed in $O(1)$ time and routed to $PE(0,j)$, $0 \le j \le k + \log N - 1$, in $O(1)$ time. □

In [3] it was shown that the addition of N N-bit numbers can be performed in $O(\log^* N)$ time using an $N \times N \times N$ (3D) reconfigurable mesh. Compared with the design in [3], our design employs an $N \times N^2$ 2-dimensional reconfigurable mesh and also improves the time complexity to $O(1)$. Note that the above algorithm can be implemented on the MRN/LRN and RMESH models as well. Also, the algorithm can be implemented on a bit model of the reconfigurable mesh, in which the word length and the bus width is $O(1)$.

IV. An Algorithm for Computing the Histogram

For computing the histogram of an $N \times N$ image with h grey levels, we employ an $N \times N$ reconfigurable mesh. Throughout this section, we assume that $h \le N$. If $h > N$, the pixels can be simply sorted in $O(N)$ time.

Lemma 2. Computing the histogram of a $h \times h$ image with h grey levels can be performed in $O(\sqrt{h})$ time on a $h \times h$ reconfigurable mesh.

Proof: Partition the reconfigurable mesh into h blocks of size $\sqrt{h} \times \sqrt{h}$. Using the reconfigurable mesh as a 2-MCC, sort the grey levels within each block in $O(\sqrt{h})$ time into a snake-like row major order using any well known 2-MCC algorithm (see, for example, [16]). The sort is performed simultaneously within all the h blocks. Partition each block into at most h connected regions corresponding to h grey level values. This can be performed in $O(1)$ time by each PE checking the grey level values in its immediate neighbors in the snake-like row major ordering. Using the indices of the beginning and the ending PEs in each region, compute the number of PEs in each region. This number is the histogram count of the grey level stored in the PEs in the region. This can be done in $O(1)$ time simultaneously for all the regions using the reconfiguration feature. Move the histogram counts such that the ith PE in the row major order has the ith histogram value, $0 \le i \le h-1$. This can be done in $O(\sqrt{h})$ time using a standard 2-MCC algorithm. Merge the h blocks in $O(\sqrt{h})$ time as follows. During the ith iteration of the merge operation, $0 \le i \le \log\sqrt{h} - 1$, 4 blocks of size $2^i\sqrt{h} \times 2^i\sqrt{h}$ are merged into a block of size $2^{i+1}\sqrt{h} \times 2^{i+1}\sqrt{h}$. At the beginning of the ith iteration, the histogram data resides in $\sqrt{h}/2^i$ leftmost columns of the block. For $0 \le j \le \frac{\sqrt{h}}{2^i} - 1$, the jth column of the right block is moved and aligned over the corresponding column of the left block. Then, the corresponding values in the left block are added. It takes $O(\sqrt{h}/2^i)$ time to merge the two right blocks with the corresponding left blocks. (see Fig. 9(a)). Next, route $\sqrt{h}/2^i$ columns in the lower left block into the upper left block and add the corresponding counts (see Fig. 9(b)). Distribute the h histogram values residing in the leftmost $\sqrt{h}/2^i$ columns in the upper left block into the leftmost $\sqrt{h}/2^{i+1}$ columns in the resulting block of size $2^{i+1}\sqrt{h} \times 2^{i+1}\sqrt{h}$ (see Fig. 9(c)).The total time complexity of the merge operation is:

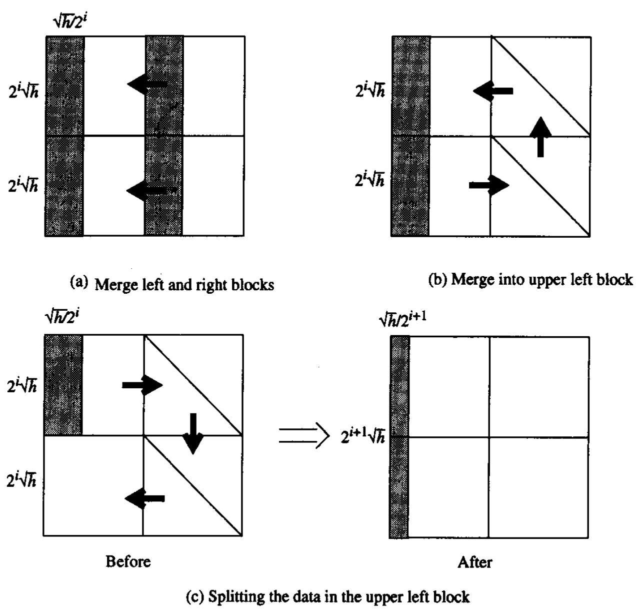


Fig. 9. The i-th merge operation.

In the above equation, the first term denotes the time for routing the histogram values, the second term denotes the time for addition, and the third term denotes the time for distribution of the histogram values. □

The above algorithm can be implemented on the MRN/LRN model as well as on the RMESH model of size $h \times h$. The h histogram counts in a block of size $h \times h$ can be stored in a particular column (say the leftmost column) of the block. By copying the histograms into a column and adding the corresponding entries, we can merge the histogram counts in the adjacent four blocks into a block of size $2h \times 2h$ in $O(1)$ time. This implies that we can go from blocks of size $h \times h$ to blocks of size $h^2 \times h^2$ in $O(\log (h^2/h)) = O(\log h)$ time. Repeating this idea, we can merge the blocks to compute the histogram of a block of size $h^4 \times h^4$ in $O(\log h)$ time. If $N < h^4$, it takes $O(\log (h^4/h^2)) = O(\log h)$ time to merge the results to obtain the histogram of the image of size $N \times N$. In this case, the total time to obtain the histogram of an $N \times N$ image is $O(\sqrt{h} + \log h) = O(\sqrt{h})$. The above discussion leads to:

Lemma 3. Given an $h^4 \times h^4$ image with h grey levels, the histogram of the image can be computed in $O(\sqrt{h})$ time on an $h^4 \times h^4$ reconfigurable mesh.

Note that repeatedly merging the blocks leads to an

$$O(\sqrt{h} + \log(N/h))$$

time solution to the histogram problem. In the following, we further improve this bound.

Given a 0/1 sequence, b_j, $0 \le j \le N-1$, the *prefix modular k computation* is to compute, for each j,

$$\left(\sum_{w=0}^{j} b_w\right) \mathbf{mod}\, k.$$

We represent the outputs using the POS notation.

Lemma 4. Prefix modular k computation of a 0/1 sequence of length N can be performed in O(1) time on a (k + 1) × 2N reconfigurable mesh.

Proof: An input bit (say b_j) is assigned to a submesh of size $(k + 1) \times 2$. The submesh is configured to output POS($(x + b_j)$ mod k), given POS(x mod k) as its input. By cascading N such submeshes, we can perform prefix modular k operation in $O(1)$ time.

Initially, b_j is stored in $PE(0, 2j)$, $0 \le j \le N-1$. The configuration of $PE(i, 2j)$, $PE(i, 2j + 1)$, $0 \le i \le k$ is determined by the input b_j. This is illustrated in Fig. 10(a) for $k = 4$. Now, apply a *1-signal* at the *W* port of $PE(0,0)$ and a *0-signal* at the W port of $PE(i, 0)$, $1 \le i \le k$. The signals at the E port of the PEs in column $(2j + 1)$, $0 \le j \le N-1$, represent

$$\left(\sum_{w=0}^{j} b_w\right) \mathbf{mod}\, k$$

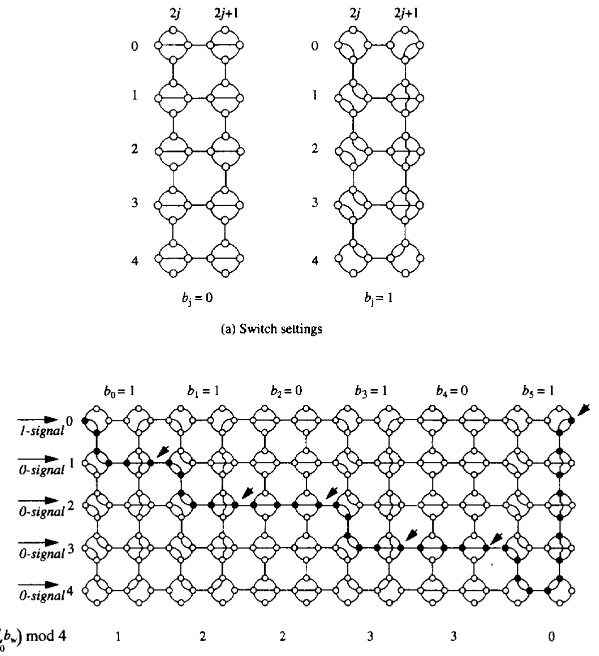
in the POS representation (see Fig. 10(b)). □

Note that the algorithm employed in the proof of Lemma 4 is not applicable to RMESH. However, it can be used on MRN/LRN of corresponding size.

It follows from the Chinese Remainder Theorem [2], that if $A \equiv B$ mod k for $2 \le k \le n$, then $A \equiv B$ mod LCM $(2 \times 3 \times \ldots \times n)$. Since there exists a constant c, such that, for all n, $n \ge 2$, LCM $(2 \times 3 \times \ldots \times cn) > 2^n$, if $A,B \in [0, 2^n - 1]$ and $A \equiv B$ *mod k*, for $2 \le k \le cn$, then $A = B$. Let $\bar{c}$ be the constant. By combining Lemma 4 and the above fact, we have:

Lemma 5. Given a 0/1 sequence of length N in a row, counting the number of 1's (i.e., converting a 2UN representation of a number to its 1UN representation) can be performed in O(1) time on a $\log^2 N \times N$ reconfigurable mesh.

Proof: Let x denote the number of 1's in the sequence. The input is 2UN(x). We determine x by computing x *mod* 2,



(a) Switch settings

(b) Prefix modular 4 computation (using POS representation)

Fig. 10. Prefix modular k computation for $k = 4$.

x m0d 3, ..., x *mod* $\bar{c}\log N$. To do this, we generate i *mod* 2, i *mod* 3, ..., i *mod* $\bar{c}\log N$, for $0 \le i \le N$. x *mod* 2, x *mod* 3, ..., x *mod* $\bar{c}\log N$ are simultaneously compared with i *mod* 2, i *mod* 3, ..., i *mod* $\bar{c}$ $\log N$, for $0 \le i \le N$, to find an i such that $x \equiv i$ *mod* k, $2 \le k \le \bar{c}\log N$.

Let b_j, $0 \le j \le N-1$ be the input sequence. To simplify the explanation, we use a $\bar{c}^2\log^2 N \times (2N + 2)$ reconfigurable mesh. Partition the reconfigurable mesh into $RM(k)$, $2 \le k \le \bar{c}$ $\log N$, where $RM(k)$ is a submesh of size $(k + 1) \times (2N + 2)$ (see Fig. 11). From Lemma 4, $RM(k)$ can compute i *mod* k, $0 \le i \le N$ (by performing prefix modular k computation on the sequence 0111...11 of length $N + 1$). POS(i mod k) is available in the PEs in the $(2i + 1)$ -th column of $RM(k)$, $0 \le i \le N-1$. Using Lemma 4,

$$\text{POS}\left(\left(\sum_{j=0}^{N-1} b_j\right) \mathbf{mod}\, k\right)$$

can be computed by $RM(k)$.

$$\text{POS}\left(\left(\sum_{j=0}^{N-1} b_j\right) \mathbf{mod}\, k\right)$$

is broadcast to all the columns of *RM*(k). For $0 \le i \le N$, in parallel, the PEs in column ($2i$+1) check if

$$\mathrm{POS}\left(\left(\sum_{j=0}^{N-1} b_j\right) \mathbf{mod}\, k\right)$$

is equal to POS(i mod k). If so, all the PEs in the ($2i$ + 1)-th column generate a *1-signal.* Now, *RM*(k), $2 \le k \le \bar{c}\log N$ are vertically cascaded into a reconfigurable mesh. There exists a unique i such that all the PEs in column ($2i$ + 1) of the mesh have a *1-signal.* This column can be easily identified in $O(1)$ time. Thus, we have computed

$$x = \sum_{j=1}^{N-1} b_j .$$

The result can be easily converted into its 1UN representation. □

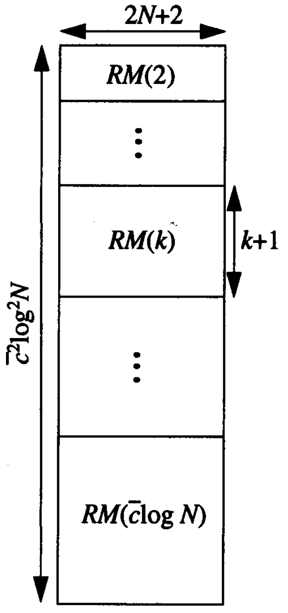


Fig. 11. Organization of the mesh to compute I mod k, for $0 \le i \le N$.

The following Lemma is used in the proof of the main Theorem.

Lemma 6. Addition of N log N-bit binary numbers stored in the PEs in a row can be performed in O(1) time on a reconfigurable mesh of size $\log^3 N \times N$.

Proof: The idea is as follows. Collect the jth bit from all the input numbers and count the number of 1's in the collection and multiply the count by 2^j, $0 \le j \le \log N - 1$. Pad the resulting sequence with j 0's to the right and with ($\log N - j$) 0's to the left to result in a 2log N bit number. Now, the addition of N log N-bit numbers in $[0, N-1]$ reduces to addition of log N 2 log$N$-bit partial sums. The counting is performed using Lemma 5 and the addition of the log N 2log N-bit partial sums is performed using Lemma 1.

Assume that each PE in the topmost row has a log$N$-bit number. Represent the number x in *PE*(0, i) in its BIN representation using the PEs in column i such that *PE*($j\log^2 N$, i) has the jth bit of the representation, $0 \le j \le \log N - 1$. Partition the mesh into submeshes of size $\log^2 N \times N$. Then, each submesh has a 0/1 sequence of length N in the PEs in its top row. Let *RM*(j) denote the jth submesh, $0 \le j \le \log N - 1$. For $0 \le j \le \log N - 1$, in parallel, using Lemma 5, *RM*(j) converts the 2UN representation of the number in its top row into its 1UN representation. Using the conversion techniques discussed in Section III.B, the number is converted to its BIN representation. In *RM*(j), multiply the number by 2^j (by shifting it). Let *BIN*(*S*(j)) denote the resulting 2log N-bit number in *RM*(j). Now, we have to add the log$N$ 2log N-bit numbers to obtain the final output. *RM*(j) stores BIN(*S*(j)) in the PEs in its topmost row such that the i-bit of BIN(*S*(j)) is in *PE*($j\log^2 N$, $2i\log N$), $0 \le i \le 2\log N - 1$, $0 \le j \le \log N - 1$. Addition of BIN(*S*(j)), for $0 \le j \le \log N - 1$, can be performed in $O(1)$ time using Lemma 1. □

Now, we are ready for the main theorem.

Theorem 1:The histogram of an $N \times N$ *image with h grey levels can be computed in*

$$O\left(\min\left\{\sqrt{h} + \log^*(N/h), N\right\}\right)$$

time on an $N \times N$ *reconfigurable mesh.*

Proof: Partition the mesh into disjoint blocks of size $h^4 \times h^4$. Using Lemma 3, the histograms of blocks of size $h^4 \times h^4$ are obtained in $O(\sqrt{h})$ time simultaneously for all the blocks. These histograms are merged in $O(1)$ time to compute the histogram of blocks of size $2^h \times 2^h$. To perform the merge, we use the addition technique in Lemma 6. After $\log^*(N/h^4) = O(\log^*(N/h))$ such merge steps, the histogram of the complete image is available.

The merge is performed as follows. First, the histograms of blocks of size $h^4 \times h^4$ are merged to compute the histogram of a block of size $h^4 \times 2^h$. Note that each block of size $h^4 \times h^4$ has h histogram values. These values are stored such that the ith PE in the leftmost column of the block has the ith histogram count, $0 \le i \le h-1$. The merge is performed in parallel, simultaneously for all the grey levels. Since there are $2^h/h^4$ histograms, we need to sum up $2^h/h^4$ values for each grey level. Note that, for each grey level, in each block of size $h^4 \times 2^h$, the count does not exceed $h^4 2^h$. Using Lemma 6, the addition of the histogram counts corresponding to a particular grey level can be performed in $O(1)$ time on a submesh of size $h^3 \times 2^h$. The merge of all the h grey level counts is performed in parallel using a submesh of size $h^4 \times 2^h$. Now we have computed the histogram of each image of size $h^4 \times 2^h$. Using a similar technique, the histograms of $2^h/h^4$ blocks of size $h^4 \times 2^h$ are merged in $O(1)$ time to compute the histogram of blocks of size $2^h \times 2^h$. Let $T(N)$ be the time to compute the histogram of an $N \times N$ image. The above discussion leads to:

$$T(h^4) = O(\sqrt{h})$$

and $T(N) = T(\log^4 N) + O(1)$, $\log N \ge h$. This leads to

$$T(N) = O\left(\sqrt{h} + \mathbf{log*}(N/h)\right).$$

f $\sqrt{h} > N$, then using a 2-MCC sorting algorithm, sort the ›ixel values to compute the histogram. This can be performed n $O(N)$ time [16]. Thus, we have

$$T(N) = O\left(\mathbf{min}\left\{\sqrt{h} + \mathbf{log*}(N/h), N\right\}\right). \qquad \square$$

Theorem 1 is also applicable to the MRN/LRN model. However, the theorem does not apply to the RMESH model since Lemma 5 does not apply to the RMESH model. Lemma 5 plays a crucial role in achieving the $\log^*(N/h)$ term in the time complexity. Our algorithm can be modified to run in

$$O\left(\mathbf{min}\left\{\sqrt{h} + \mathbf{log\,log}(N/h), N\right\}\right)$$

time on an $N \times N$ RMESH. The histograms in l^2 blocks of size $l \times l$ can be merged in $O(1)$ time using Lemma 1 and the conversion techniques in Section III.B, for $l \geq h^4$. Recursive application of the merge technique results in the $O(\log\log(N/h))$ term in the time complexity.

Corollary 1: Given a $N \times N$ image with h grey levels, the histogram of the image can be computed in

$$O\left(\mathbf{min}\left\{\sqrt{h} + \mathbf{log\,log}\,N, N\right\}\right)$$

time on a $N \times N$ RMESH.

Lemma 7: Given a histogram modification table in a block of size $\sqrt{h} \times \sqrt{h}$, the histogram modification of an $N \times N$ image can be performed in $O(\sqrt{h})$ time on an $N \times N$ reconfigurable mesh.

Proof: Using a sequence of row and column broadcasts, copy the histogram modification table into each block of size $\sqrt{h} \times \sqrt{h}$. This can be completed in $O(\sqrt{h})$ time. Sort the pixel data in each block into a snake-like row major order. Partition each block into at most h connected regions, such that within each region all the PEs have the same pixel value. Route the data in the table such that the beginning PE of the ith region receives the ith table entry, $0 \leq i \leq h - 1$. This can be performed in $O(N)$ time using standard 2-MCC techniques. Each beginning PE broadcasts the new value of the pixel to all the PEs in its region. This can be performed in constant time by configuring a broadcast bus within each region. Within each block, the modified pixel values are moved back to the original PEs in $O(\sqrt{h})$ time using a standard 2-MCC technique [23]. $\square$

Note that Lemma 7 also applies to the MRN/LRN and the RMESH models.

V. Conclusion

We have improved the time bound in [26] for computing the histogram of an $N \times N$ image with h grey levels on the reconfigurable mesh to

$$O\left(\mathbf{min}\left\{\sqrt{h} + \mathbf{log*}(N/h), N\right\}\right)$$

time on the PARBUS and MRN/LRN models of size $N \times N$. A variant of our algorithm runs in

$$O\left(\mathbf{min}\left\{\sqrt{h} + \mathbf{log\,log}(N/h), N\right\}\right)$$

time on the RMESH model. This improves the time bound in [15]. We have also reduced the amount of local storage from $O(\sqrt{h})$ in [15] to $O(1)$. The histogram modification algorithm improves the memory requirements by a factor of $O(\sqrt{h})$ in each PE.

Our result implies that the number of 1's in a $N \times N$ 0/1 table can be computed in $O(\log^* N)$ time on an $N \times N$ reconfigurable mesh.

Throughout this paper we have employed the word model of reconfigurable mesh. The bit model of reconfigurable mesh has been introduced in [13]. In this model, the bus width and the storage in each PE are constant. The PE and the wire connecting adjacent PEs occupy unit area. Thus, the bit model of reconfigurable mesh of size $N \log N \times N \log N$ occupies the same area as the word model of reconfigurable mesh of size $N \times N$ in which the word size is $\Theta(\log N)$. Among its other features, the bit model allows bit level switch control which is not possible in the word model. In [13] it has been shown that, given an $N \times N$ image with h grey levels, $h \leq N$, such that each pixel is mapped to a $\log N \times \log N$ block in a natural fashion, the histogram of the image can be computed in $O(\sqrt{h})$ time on the bit model of reconfigurable mesh of size $N \log N \times N \log N$.

Acknowldgements

This research was supported in part by the U. S. National Science Foundation under grant IRI-9217528 and in part by the (U. S. Defense) Advanced Research Projects Agency under contract F-49620-89-C-0126.

References

[1] H.M. Alnuweiri, "Fast algorithms for image labeling on a reconfigurable network of processors," *Proc. Int'l Parallel Processing Symp.*, pp. 569-575, Apr. 1993.

[2] A. Baker, *A Concise Introduction to the Theory of Numbers*, Cambridge Univ. Press, 1984.

[3] Y. Ben-Asher, D. Peleg, R. Ramaswami, and A. Schuster, "The power of reconfiguration," *J. of Parallel and Distributed Computing*, vol. 13, no. 2, pp. 139-153, Oct. 1991.

[4] Y. Ben-Asher, D. Gorden, and A. Schuster, "Optimal simulations in reconfigurable arrays," Technical Report No. 716, Dept. of Computer Science, Technion- Israel Inst. Technology, 1992.

[5] T. Bestul and L.S. Davis, "On computing histograms of images in log *n* time using fat pyramids," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 11, no. 2, pp. 212-213, 1989.

[6] H. Elgindy and P. Wegrowicz, "Selection on the Reconfigurable Mesh," *Proc. Int'l Conf. on Parallel Processing*, pp. III.26-III.33, Aug. 1991.

[7] K.S. Hedlund and L. Snyder, "Wafer scale integration of configurable, highly parallel processors," *Proc. Int'l Conf. Parallel Processing*, pp. 262-264, 1982.

[8] J. Já Já, *An Introduction to Parallel Algorithms*, page 556, Addison-Wesley Publishing Company, 1992.

[9] J. Jang and V.K. Prasanna, "An optimal sorting algorithm on reconfigurable mesh," *Proc. Int'l Parallel Processing Symp.*, pp. 130-137, Mar. 1992.

[10] J. Jang and V.K. Prasanna, "Efficient parallel algorithms for some geometric problems on reconfigurable mesh," *Proc. Int'l Conf. Parallel Processing*, pp. III.127-130, 1992.

[11] J. Jang, H. Park, and V.K. Prasanna, "A fast algorithm for computing histogram on reconfigurable mesh," *Proc. Frontiers of Massively Parallel Computation*, pp. 244-251, 1992.

[12] J. Jang, H. Park, and V. K. Prasanna, "An optimal multiplication algorithm on reconfigurable mesh," *Proc. Symp. Parallel and Distributed Processing*, pp. 384-391, 1992.

[13] J. Jang, H. Park, and V. K. Prasanna, "A bit model of reconfigurable mesh," *Proc. of Reconfigurable Architectures Workshop* held at *Int'l Parallel Processing Symp.(IPPS)*, April 1992.

[14] J. Jenq and S. Sahni, "Reconfigurable mesh algorithms for image shrinking, expanding, clustering, and template matching," *Proc. Int'l Parallel Processing Symp.*, pp. 208-215, 1991.

[15] J. Jenq and S. Sahni, "Histogramming on a reconfigurable mesh computer," *Proc. Int'l Parallel Processing Symp.*, pp. 425-432, 1992.

[16] V. Kumar, A. Grama, A. Gupta, and G. Karypis, *Introduction to Parallel Computing, Design, and Analysis of Algorithms*, Benjamin/Cummings Publishing Co., 1993.

[17] H. Li and M. Maresca, "Polymorphic-torus network," *IEEE Trans. on Computers*, vol. 38, no. 9, pp. 1345-1351, Sept. 1989.

[18] R. Lin and S. Olariu, "Reconfigurable meshes with shift switching-architecture and applications," *Proc. of Int'l Phoenix Conf. on Computers and Communications*, 1993.

[19] W. Lin and V.K. Prasanna Kumar, "Efficient histogramming on hypercube SIMD machines," *Computer Vision, Graphics, and Image Processing*, vol. 49, pp. 104-120, 1990.

[20] 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.

[21] R. Miller, V.K. Prasanna Kumar, D.I. Reisis and Q.F. Stout, "Parallel computations on reconfigurable meshes," *IEEE Trans. on Computers*, pp. 678-692, June 1993.

[22] 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.

[23] D. Nassimi and S. Sahni, "Data broadcasting in SIMD computers," *IEEE Trans. on Computers*, vol. 30, pp. 101-107, Feb. 1981.

[24] M. Nigam and S. Sahni, "Sorting n numbers on $n \times n$ reconfigurable meshes with buses," *Proc. Int'l Parallel Processing Symp.*, pp. 73-78, Apr. 1993.

[25] K. Nakano, T. Masuzawa, and N. Tokura, "A sub-logarithmic time sorting algorithm on a reconfigurable mesh," *Inst. Electronics, Information and Communication Engineers Trans.*, vol. E74, no. 11, Japan, Nov. 1991.

[26] S. Olariu, J.L. Schwing, and J. Zhang, "Fast computer vision algorithms for reconfigurable meshes," *Image and Vision Computing*, pp. 610-616, 1992.

[27] H. Park, V.K. Prasanna, and J. Jang, "Fast arithmetic on reconfigurable meshes," *Proc. Int'l Conf. on Parallel Processing*, 1993.

[28] V.K. Prasanna Kumar, and C.S. Raghavendra, "Array processor with multiple broadcasting," *J. of Parallel and Distributed Computing*, vol. 4, pp. 173-190, 1987.

[29] S. Rajashekaran, "Mesh Connected Computers with Fixed and Reconfigurable Buses: Packet Routing, Sorting and Selection," *Manuscript*, Dept. of CIS, Univ. of Pennsylvania, Oct. 1993. Also, appears at the First Ann. European Symp. on Algorithms, 1993.

[30] J. Rothstein, "Bus Automata, Brains, and Mental Models," *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 18, no. 4, pp. 522-531, Apr. 1988.

[31] L. Snyder, "Introduction to the configurable highly parallel computer," *Computer*, vol. 15, no. 1, pp. 47-56, 1982.

[32] S.L. Tanimoto, "Sorting, histogramming, and other statistical operations on a pyramid machine," in *Multiresolution Image Processing and Analysis*, A. Rosenfeld, ed., pp. 136-145, Springer-Verlag, New York, 1984.

[33] C.D. Thompson and H.T. Kung, "Sorting on a mesh connected parallel computer," *Comm. ACM*, vol. 20, no. 4, pp. 263-271, 1977.

[34] R.K. Thiruchelvan, J.L. Trahan, and R. Vaidyanathan, "On the Power of Segmenting and Fusing Buses," *Proc. of Int'l Parallel Processing Symp.*, pp. 79-83, Apr. 1993.

[35] M. Thornburg and S. Casselman, "Transformable Computers," *Proc. of Int'l Parallel Processing Symp. (IPPS)*, Apr. 1994.

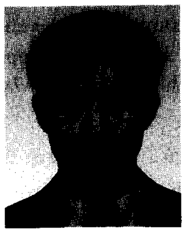
[36] B.F. Wang, G.H. Chen, and F.C. Lin, "Constant time sorting on a processor array with a reconfigurable bus systems," *Information Processing Letters*, vol. 34, pp. 187-192, 1990.

[37] B.F. Wang and G.H. Chen, "Two dimensional processor array with reconfigurable bus system is at least as powerful as CRCW model," *Information Processing Letters*, vol. 36, pp. 31-36, 1990.

[38] B. F. Wang and G. H. Chen, "Constant time algorithms for the transitive closure problem and some related graph problems on processor arrays with reconfigurable bus systems," *IEEE Trans. on Parallel and Distributed Systems*, vol. 1, no. 4, pp. 500-507, 1991.

[39] C.C. Weems, S.P. Levitan, A.R. Hanson, E.M. Riseman, J.G. Nash, and D.B. Sheu, "The image understanding architecture," *Int'l J. of Computer Vision*, vol. 2, pp. 251-282, 1989.

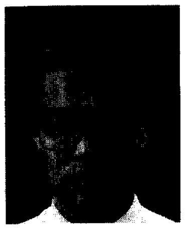
[40] *Proc. of Reconfigurable Architectures Workshop*, Apr. 1994.



Ju-wook Jang received the B.S. in electronics engineering from Seoul National University in 1983 and the M.S. in electrical engineering from the Korea Advanced Institute of Science and Technology in 1985. He received his Ph.D. in electrical engineering from the University of Southern California, Los Angeles, Calif., in 1993.

Dr. Jang joined Samsung Electronics Co. Ltd. in 1984. He is currently working on the development of multiprocessors.

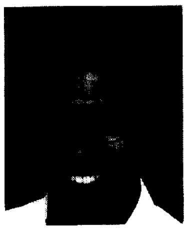
Dr. Jang's research interests include digital image processing and parallel architectures and algorithms.



Heonchul Park received the B.S. and M.S. degrees from the Department of Control and Instrumentation Engineering, Seoul National University, Korea, in 1985 and 1987, respectively. He received the Ph.D. degree from the Department of Electrical Engineering-Systems, the University of Southern California, Los Angeles, Calif., in 1993.

Dr. Park joined the DSP team of Semiconductor Business, Samsung Electronics Co. Ltd., Korea, in 1993, where he is involved in VLSI architecture definitions for multi-media.

Dr. Park's research interests include VLSI computations, parallel architectures and algorithms, computational aspects of image processing, and multimedia.



Viktor K. Prasanna (V.K. Prasanna Kumar) received his B.S. in electronics engineering from the Bangalore University and his M.S. from the School of Automation, Indian Institute of Science. He obtained his Ph.D. in computer science from Pennsylvania State University in 1983.

Dr. Prasanna is currently an associate professor in the Department of Electrical Engineering-Systems, University of Southern California, Los Angeles, Calif., and serves as the director of the Computer Engineering Division. His research interests include parallel computation, computer architecture, VLSI computations, and computational aspects of image processing, vision, robotics, and neural networks.

Dr. Prasanna has published extensively and consulted for industries in his areas of interest. He is the editor of the book, *Parallel Architectures and Algorithms for Image Understanding*, published by Academic Press. He is the symposium chair of the IEEE International Parallel Processing Symposium, 1995, and serves on the editorial boards of the *Journal of Parallel and Distributed Computing* and *IEEE Transactions on Computers*. He is the founding chair of the IEEE Computer Society Technical Committee on Parallel Processing.