Median 필터를 위한 RMESH 병렬 알고리즘의 설계

전 병 문† · 정 창 성††

요 약

Median 필터는 임계치 분할, 스택킹 특성, 그리고 선형 분리성에 기반하여 이진 영역에서 구현이 가능하다. 본 논문에서는 VLSI 구현에 적합한 변형 가능한 메쉬(RMESH) 구조에서 median 필터링을 위한 1차원 및 2차원 병렬 알고리즘을 개발하고, 고정된 토폴로지를 갖는 메쉬에서의 시간 복잡도와 비교함에 의해 본 논문에서 제안하는 병렬 알고리즘의 성능을 평가한다. 실제로 M 레벨의 1차원 시그널 길이가 N이고 윈도우 폭이 w일 때, 메쉬 구조에서는 $O(Mw^2)$의 시간 복잡도를 갖는 반면 RMESH 구조에서의 알고리즘은 $O(Mw)$ 시간 복잡도를 갖는다. 또한 M 레벨의 2차원 영상의 크기가 $N\times N$이고 윈도우 크기가 $w\times w$라고 가정하면, 본 논문에서 제안한 $N\times N$ RMESH 상에서의 median 필터링 알고리즘은 $N\times N$ 메쉬의 $O(Mw^2)$ 시간 보다 더욱 향상된 $O(Mw)$ 시간에 계산되어진다.

Design of RMESH Parallel Algorithms for Median Filters

Byeong-Moon Jeon† · Chang-Sung Jeong††

ABSTRACT

Median filter can be implemented in the binary domain based on threshold decomposition, stacking property, and linear separability. In this paper, we develop one-dimensional and two-dimensional parallel algorithms for the median filter on a reconfigurable mesh with buses(RMESH) which is suitable for VLSI implementation. And we evaluate their performance by comparing the time complexities of RMESH algorithms with those of algorithms on mesh-connected computer. When the length of M-valued 1-D signal is N and w is the window width, the RMESH algorithm is done in $O(Mw)$ time and mesh algorithm is done in $O(Mw^2)$ time. Besides, when the size of M-valued 2-D image is $N\times N$ and the window size is $w\times w$, our algorithm on $N\times N$ RMESH can be computed in $O(Mw)$ time which is a significant improvement over the $O(Mw^2)$ complexity on $N\times N$ mesh.

1. Introduction

A regular mesh of size $N\times N$ has a communication diameter which equals to $2(N-1)$. As a result, a lower bound for the time complexity for problems that involve comparing or combining data that reside in different processors is $O(N)$. To improve the time complexity for these problems, researchers have studied special architectures whose bus system can be dynamically changed. Reconfigurable mesh architectures set an excellent example for such machines. These include the polymorphic-torus network of Li and Maresca[1, 2], the reconfigurable mesh with

† 준 회 원 : 고려대학교 대학원 전자공학과
†† 정 회 원 : 고려대학교 전자공학과 교수
논문접수 : 1998년 5월 14일, 심사완료 : 1998년 8월 20일

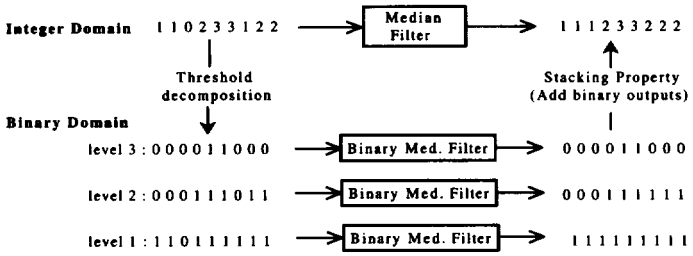
buses(RMESH) of Miller et al.[3, 4], the processor array with a reconfigurable bus system(PARBUS) of Wang and Chen[5], and the reconfigurable network(RN) of Ben-Asher et al.[6]. Conceptually, these reconfigurable architectures are functionally equivalent and the regular structure of the reconfigurable architectures makes it suitable for VLSI implementation such as the YUPPIE(Yorktown Ultra-Parallel Polymorphic Image Engine) chip[2] and the PPA(Polymorphic Processor Array)[18].

On the other hand, a median filter[7] is a simple nonlinear smoothing operation that takes a median value of the data inside a moving window with finite length. Fitch et al.[8] introduced powerful tools for the analysis of the median filter, namely, threshold decomposition and stacking property, which establish a relationship between integer(multilevel) and binary domain filtering. And it has been shown that the median filter belongs to the class of stack filters which correspond uniquely to positive Boolean functions(PBFs)[9]. This means that the principle of threshold decomposition and binary processing with the unique PBF can be used to implement the median filter in the binary domain. The binary processing approach to realizing a multilevel signal processing is widely used in VLSI implementations because of its simplicity. Based on this advantage, several VLSI-implementable algorithms for the median filter have been proposed[19-21]. However, they all employed the bit-serial design at the input stage so they are not practically the parallel algorithms[10]. In addition, it is very difficult to define the PBF(window operator) of the median filter according to its size. Fortunately, the median filter has efficient realization because it has linearly separable PBF in the binary domain. In this paper, we restrict ourselves to the development of RMESH algorithms for one-dimensional and two-dimensional median filtering, which are helpful to VLSI implementation, by using the properties of the median filter such as threshold decomposition, stacking property, and linear separability. This is the first time the design of RMESH algorithm for the median filter is reported in the literature.

The rest of this paper is organized as follows. In Section 2, we review the basic concepts needed in the binary processing of the median filter. Section 3 describes the RMESH model that we adopt and shows some algorithms proposed in other literatures. The efficient RMESH algorithms for the median filtering of 1-D signal and 2-D image are proposed in Section 4. Section 5 proves the RMESH architecture is superior in performance as compared to the conventional mesh architecture with fixed network topology and finally some conclusions are given in Section 6.

2. The Properties of Median Filters

In this paper, we use integer- and binary-valued signals/images which are referred to as the integer domain and the binary domain, respectively. Uppercase letters are used for integer domain filtering and lowercase letters for binary domain filtering, unless



(Fig. 1) Illustration of threshold decomposition and the stacking property of median filter with window width 3

otherwise specified. The key for the binary processing of the median filter is the use of the threshold decomposition and the stacking property(see Fig. 1).

A sequence of the window width w among signal length N will be denoted $X=[X_1, X_2, \cdots, X_w]$, in which X_i is a M-valued signal sample and w is generally odd.

Definition 1. The threshold decomposition of an M-valued signal X_i is the set of M-1 binary signals, called threshold signals, $x_i^1, x_i^2, \cdots, x_i^{M-1}$, which are defined by

$$x_i^m = T^m(X_i) = \begin{cases} 1, & \text{if } X_i \geq m \\ 0, & \text{else.} \end{cases} \quad \blacksquare$$

Note that the sum of the threshold signals x_i^m is X_i, i.e., $\sum_{m=1}^{M-1} x_i^m = X_i$.

Definition 2. The binary filter $f(\cdot)$ with the window width w is said to possess the stacking property if and only if

$$f(x) \geq f(y), \quad \textit{whenever} \quad x \geq y. \quad \blacksquare$$

The stack filter S_f in the integer domain is defined by a binary filter $f(x)$ as follows

$$S_f(X) = \sum_{m=1}^{M-1} S_f(x^m) = \sum_{m=1}^{M-1} f(x^m).$$

Fig. 1 illustrates that applying a median filter to M-valued signal is equivalent to decomposing the signal to M-1 binary signals, filtering each binary signal with the binary median filter or with the PBF $f_{med}(x)$, and then adding the binary output signals together. At this time, instead of adding the binary filter outputs, we can easily obtain the multilevel output by noting that the binary outputs possess the stacking property[9]. So the multilevel output is the level just before the transition from 1 to 0 takes place. However, we have a difficult problem that is how we define the PBF of the median filter according to its size. Fortunately, this problem can be solved by the linear separability[11, 12].

Definition 3. A positive Boolean function $f(x)$ is said to be linearly separable if and only if there exist real number $W_1, W_2, \cdots, W_w$ and T such that

$$f(x_1, \cdots, x_w) = \begin{cases} 1, & \text{if } \sum_{i=1}^{w} W_i x_i \geq T \\ 0, & \textit{otherwise.} \end{cases}$$

The weights W_i and threshold T can be restricted to be positive integers. ■

Definition 4. A stack filter defined by a PBF $f(x)$ is a median filter if and only if $f(x)$ can be expressed as

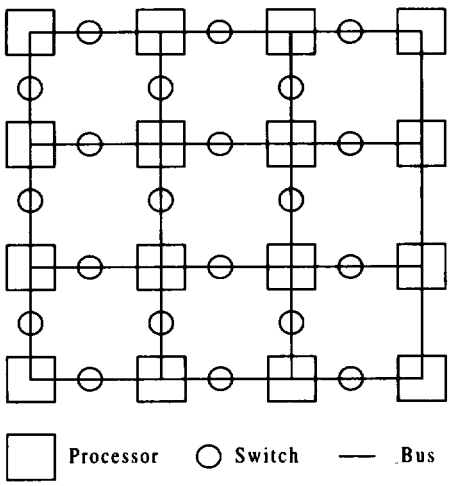
$$f(x_1, \cdots, x_w) = \begin{cases} 1, & \text{if } \sum_{i=1}^{w} x_i \geq (w+1)/2 \\ 0, & \textit{otherwise.} \end{cases}$$

This means that the median filter can be realized by the linear separability in the binary domain. ■

3. RMESH Model

The particular reconfigurable mesh architecture that we use in this paper is due to Miller et al.[3, 4]. The RMESH consists of $N \times N$ processors connected to a grid-shaped reconfigurable broadcast bus, where each processing element(PE) has four locally controllable bus switches labeled E(east), W(west), S(south) and N(north), as shown Fig. 2. The switches allow the broadcast bus to be divided into subbuses, providing smaller reconfigurable meshes. Furthermore, in one unit time, each PE can set any of its four switches and can send and receive a piece of data from the bus. To read the content of the broadcast bus into a register R, the statement R:=read(bus) is used. And the RMESH model of this paper allows several PEs to read the same bus component; however, it does not allow more than one PE to write on the same bus component at the same time(i.e., CREW model).

During the last decade, a large number of algorithms have been designed to run on reconfigurable architectures. As mentioned earlier, conceptually, the reconfigurable architectures are functionally equivalent. These include algorithms for sorting[6, 17], fundamental data manipulation operations[1, 3, 4, 16], and image processing[13-15]. Li and Maresca designed three algorithms based on the Polymorphic-torus



(Fig. 2) 4×4 RMESH

such as an $O(1)$ time boolean algorithm, an $O(1)$ time maximum(minimum) algorithm, and an $O(\log n)$ time sum algorithm[1]. Miiler et al. designed $O(1)$ time algorithms for performing various logic operations and finding maximum(minimum), and an $O(\log N)$ time algorithm for finding connected components of a graph(N is the size of the mesh)[3, 4]. Jenq and Sahni presented a number of algorithms for fundamental data manipulation operations on reconfigurable mesh architectures[16]. These include an $O(w)$ window broadcast algorithm, $O(\log n)$ time algorithms for prefix sum and data sum, and $O(1)$ time algorithms for ranking, shift, random access read (RAR), and random access write(RAW). They also designed an $O(p\log(n/p))$ time algorithm for the p angle Hough transform[13], an $O(1)$ time algorithm for the q-step shrinking and expansion of a binary image, and an $O(M^2)$ time algorithm for template matching using an $M\times M$ template and an $N\times N$ image[14]. Wang and Chen designed a number of constant-time algorithms on the PARBUS[5]. Ben-Asher et al. examined the power of reconfiguration and presented a number of algorithms, such as an $O(\log n)$ time algorithm for addition and an $O(1)$ time algorithm for sorting on a reconfigurable mesh of size $n\times n\times n$ [6].

4. Parallel Algorithms

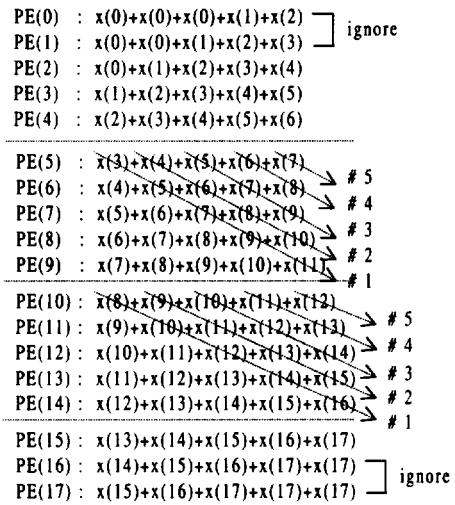
4.1 One-Dimensional RMESH Algorithm

For convenience, we assume that the 1-D signal length is N and it is mapped onto the one dimensional processor array on row 0 of $N\times N$ RMESH such that PE(i) contains M-valued signal $X(i)$. Each processor PE(i) has the array $x[1, \cdots, M-1](i)$, which will be obtained by threshold decomposition of $X(i)$. That is, $x[m](i)$ is the binary value related to mth level of $X(i)$. Also, each processor has the array $med[1, \cdots, M-1](i)$ to contain the value obtained by binary median operation at each level. This operation can be easily achieved by using the linear separability(see Definition 4). That is, in order to obtain the median output in the binary domain, the sum of all binary values in a window is thresholded by $(w+1)/2$. Finally, PE(i) can determine the multilevel output $MED(i)$ by examining the level whose output is 1 and the next greater output is 0 from $med[1, \cdots, M-1](i)$ array by stacking property. When the window width is w, the 1-D signal parallel algorithm on RMESH is as follows.

Algorithm 1 1-D Signal Algorithm for Median Filter
Input $X(0), X(1), \cdots, X(N-1)$
Output $MED(0), MED(1), \cdots, MED(N-1)$
Step 1. PE(i) gets $x[1](i), x[2](i), \cdots, x[M-1](i)$ from M-level signal $X(i)$ through threshold decomposition.
Step 2.
Repeat for m := 1 **to** M-1
PE(i) gets $med[m](i)$ by applying the Algorithm 2 at level m.
Step 3. PE(i) obtains $MED(i)$ in the integer domain by searching the level whose output is 1 and the next greater output is 0 from $med[1](i), med[2](i), \cdots, med[M-1](i)$.

Since each PE(i) has only the binary value $x[m](i)$ at level m, we need to communicate the its

binary data in a fashion that each PE has binary values who contribute to obtain the sum in the window. Fig. 3 represents the terms in sum required by each processor at each level for w=5 and N=18. Generally, the unknown signal samples in the window are obtained by repeating the samples x_0 or x_{N-1} when the window is positioned at the boundary of the signal sequence. However, we consider only the remaining signal sequence except for front and rear parts of signal having $\lfloor w/2 \rfloor$ length respectively, since the boundary of the signal sequence is relatively unimportant.



(Fig. 3) Terms in each PE's sum at level m, when w=5 and N=18

Let's number the diagonals in this figure by dividing all PEs at intervals of w and assigning the main diagonal the number one at each unit. The diagonal just above this is numbered two, and next is numbered three and so on. The diagonal just below the main diagonal is numbered 5, and the one below it is numbered 4 and so on. Notice that the numbers 2~5 are assigned to exactly two diagonals each. Also note that the number of elements on the at most two diagonals is 5 for every assigned number. After assigning the numbers to all diagonals, we examine the terms in each number. Since we ignore the front and rear parts of signal, the pattern of x values is found out by examining from PE(5) to PE(9) or from PE(10) to PE(14). The pattern of x values required by each processor at each number can be achieved by pairing the present value needed for current step and next value needed for next step. When we fix that present value of PE(i) is $p(i)$ and next value is $n(i)$, the $p(i)$ of first step can be obtained by $i-\lfloor w/2 \rfloor+(i \bmod w)$ and $n(i)$ of first step can be obtained by $i+(i \bmod w)-(\lfloor w/2 \rfloor-1)$. Table 1 gives $(p(i), n(i))$ pair in each PE for the case w=5 and N=18. Once the $(p(i), n(i))$ pair has been obtained, $p(i)$ is used to obtain the sum value for median operation and the pair may be changed by left shifting $p(i)$ and exchanging $p(i)$ and $n(i)$. Initially, $n(i)$ is true for all processors except those with $i \bmod w = w-1$. It is updated by reconfiguring the RMESH and receiving the correct value in those processors. Following the left shift of $p(i)$ and exchange of $p(i)$ and $n(i)$, we get the new pair $(p(i), n(i))$, and then $n(i)$ in processors with $i \bmod w = w-2$ is changed. With this insight, we can design Algorithm 2. As far as the range of i is not mentioned in Algorithm 2, we regard its range as $0 \le i < N$.

〈Table 1〉 $(p(i), n(i))$ pair in each PE at level m, when w=5 and N=18

PE	$(p(i), n(i))$	$(p(i), n(i))$	$(p(i), n(i))$	$(p(i), n(i))$	$(p(i), n(i))$
0	-	-	-	-	-
1	-	-	-	-	-
2	x(2), x(3)	x(3), x(4)	x(4), x(0)	x(0), x(1)	x(1), x(2)
3	x(4), x(5)	x(5), x(1)	x(1), x(2)	x(2), x(3)	x(3), x(4)
4	x(6), x(2)	x(2), x(3)	x(3), x(4)	x(4), x(5)	x(5), x(6)
5	x(3), x(4)	x(4), x(5)	x(5), x(6)	x(6), x(7)	x(7), x(3)
6	x(5), x(6)	x(6), x(7)	x(7), x(8)	x(8), x(4)	x(4), x(5)
7	x(7), x(8)	x(8), x(9)	x(9), x(5)	x(5), x(6)	x(6), x(7)
8	x(9), x(10)	x(10), x(6)	x(6), x(7)	x(7), x(8)	x(8), x(9)
9	x(11), x(7)	x(7), x(8)	x(8), x(9)	x(9), x(10)	x(10), x(11)
10	x(8), x(9)	x(9), x(10)	x(10), x(11)	x(11), x(12)	x(12), x(8)
11	x(10), x(11)	x(11), x(12)	x(12), x(13)	x(13), x(9)	x(9), x(10)
12	x(12), x(13)	x(13), x(14)	x(14), x(10)	x(10), x(11)	x(11), x(12)
13	x(14), x(15)	x(15), x(11)	x(11), x(12)	x(12), x(13)	x(13), x(14)
14	x(16), x(12)	x(12), x(13)	x(13), x(14)	x(14), x(15)	x(15), x(16)
15	x(13), x(14)	x(14), x(15)	x(15), x(16)	x(16), x(17)	x(17), x(13)
16	-	-	-	-	-
17	-	-	-	-	-

Algorithm 2 Binary Median Operation at Level m

Step 1 {initialize}
All PEs connect their W and E switches;
sum[m](i):=0;

Step 2 {present value, $p(i)$}
for j := 0 **to** w-1 **do**
begin
if $j < \lfloor w/2 \rfloor$ **then**
begin
PE(i) disconnects its W switch and broadcasts $x[m](i)$ if $i = (wn+j) - \lfloor w/2 \rfloor + j$ where $n = 0, 1, 2, \cdots$;
$p(i)$:=read(bus) if $i \bmod w = j$;
PE(i) connects its W switch if $i = (wn+j) - \lfloor w/2 \rfloor + j$ where $n = 0, 1, 2, \cdots$;
end;
else if $j = \lfloor w/2 \rfloor$ **then**
$p(i) := x[m](i)$ if $i \bmod w = j$;
else begin
PE(i) disconnects its E switch and broadcasts $x[m](i)$ if $i = (wn+j) - \lfloor w/2 \rfloor + j$ where $n = 0, 1, 2, \cdots$;
$p(i)$:=read(bus) if $i \bmod w = j$;
PE(i) connects its E switch if $i = (wn+j) - \lfloor w/2 \rfloor + j$ where $n = 0, 1, 2, \cdots$;
end;
end;

Step 3 {next value, $n(i)$}
for j := 0 **to** w-1 **do**
begin
if $j < \lfloor w/2 \rfloor - 1$ **then**
begin
PE(i) disconnects its W switch and broadcasts $x[m](i)$ if $i = (wn+j)+j-(\lfloor w/2 \rfloor - 1)$ where $n = 0, 1, 2, \cdots$;
$n(i)$:=read(bus) if $i \bmod w = j$;
PE(i) connects its W switch if $i = (wn+j)+j-(\lfloor w/2 \rfloor - 1)$ where $n = 0, 1, 2, \cdots$;
end;
else if $j = \lfloor w/2 \rfloor - 1$ **then**
$n(i) := x[m](i)$ if $i \bmod w = j$;
else begin
PE(i) disconnects its E switch and broadcasts $x[m](i)$ if $i = (wn+j)+j-(\lfloor w/2 \rfloor - 1)$ where $n = 0, 1, 2, \cdots$;
$n(i)$:=read(bus) if $i \bmod w = j$;
PE(i) connects its E switch if $i = (wn+j)+j-(\lfloor w/2 \rfloor - 1)$ where $n = 0, 1, 2, \cdots$;
end;
end;

Step 4 sum[m](i) := sum[m](i)+$p(i)$;

Step 5 {update $n(i)$}
PE(i) disconnects its W switch and broadcast $x[m](i)$ if $j \bmod w = w-1$ and $i = j - \lfloor w/2 \rfloor$, where $0 \le j < N$;
$n(i)$:=read(bus) if $i \bmod w = w-1$;
PE(i) connects its W switch if $j \bmod w = w-1$ and $i = j - \lfloor w/2 \rfloor$, where $0 \le j < N$;

Step 6
for j := 1 **to** w-1 **do**
begin
{left shift $p(i)$ by 1}
for k := 0 **to** 1 **do**
begin
PE(i) disconnects its E switch and broadcasts $p(i)$ if $(i-1) \bmod 2 = k$;
$temp(i)$:=read(bus) if $i \bmod 2 = k$;
PE(i) connects its E switch if $(i-1) \bmod 2 = k$;
end;
$p(i)$:=$temp(i)$;

{exchange $p(i)$ and $n(i)$}
$temp(i)$:=$n(i)$;
$n(i)$:=$p(i)$;
$p(i)$:=$temp(i)$;

sum[m](i):= sum[m](i)+$p(i)$;

{update $n(i)$}
PE(i) disconnects its W switch and broadcasts $x[m](i)$ if $k \bmod w = w-1-j$ and $i = k - \lfloor w/2 \rfloor$, where $0 \le k < N$;
$n(i)$:=read(bus) if $i \bmod w = w-1-j$ and $\lfloor w/2 \rfloor \le i < N$;
PE(i) connects its W switch if $k \bmod$

$w = w-1-j$ and $i = k-\lfloor w/2 \rfloor$, where $0 \le k < N$;
end;

Step 7 {linear separability}
if sum[m](i) $\ge (w+1)/2$ **then**
med[m](i):=1;
else
med[m](i):=0;

4.2 Two-Dimensional RMESH Algorithm

We assume that the processors in $N \times N$ RMESH are indexed such that PE(i,j) corresponds to the image pixel $X(i,j)$. Two-dimensional parallel algorithm for the median filter is similar to the one-dimensional algorithm. The difference between them is only the dimension. That is, since the median operation with the property of linear separability requires the sum of all the samples in $w \times w$ window, we need to merge the sum values in each row obtained by Algorithm 2 and threshold it by $(w^2+1)/2$.

Algorithm 3 2-D Image Algorithm for Median Filter

Input $X(0,0)$, ⋯, $X(N-1, N-1)$
Output $MED(0,0)$, ⋯, $MED(N-1, N-1)$
Step 1 PE(i,j) gets $x[1](i,j)$, $x[2](i,j)$, ⋯, $x[M-1](i,j)$ from M-level signal $X(i,j)$ through threshold decomposition.
Step 2 PE(i,j) disconnects its N and S switches.
Step 3
Repeat for m := 1 **to** M-1
PE(i,j) gets sum[m](i,j) by applying the Algorithm 2 with an exception of Step 7 at level m.
Step 4 PE(i,j) connects its N and S switches and disconnects its E and W switches.
Step 5
Repeat for m := 1 **to** M-1
(1) $\text{sum}_{2D}(i,j)$:=0;
(2) **for** k := 1 **to** $\lfloor w/2 \rfloor$ **do**
begin
{downward shift sum[m](i,j) by 1}
for l := 0 **to** 1 **do**
begin
PE(i,j) disconnects its N switch and broadcasts sum[m](i,j) if $j \bmod 2 = l$;
temp(i,j):=read(bus) if $(j-1) \bmod 2 = l$;
PE(i,j) connects its N switch if $j \bmod 2 = l$;
end;
$\text{sum}_{2D}(i,j) := \text{sum}_{2D}(i,j) + temp(i,j)$;
end;
(3) **for** k := 1 **to** $\lfloor w/2 \rfloor$ **do**
begin
{upward shift sum[m](i,j) by 1}
for l := 0 **to** 1 **do**
begin
PE(i,j) disconnects its S switch and broadcasts sum[m](i,j) if $j \bmod 2 = l$;
temp(i,j):=read(bus) if $(j-1) \bmod 2 = l$;
PE(i,j) connects its S switch if $j \bmod 2 = l$;
end;
$\text{sum}_{2D}(i,j) := \text{sum}_{2D}(i,j) + temp(i,j)$;
end;
(4) **if** $\text{sum}_{2D}(i,j) \ge \frac{w^2+1}{2}$ **then**
med[m](i,j) :=1;
else
med[m](i,j) :=0;
Step 6 PE(i,j) obtains $MED(i,j)$ in the integer domain by searching the level whose output is 1 and the next greater output is 0 from *med*[1](i,j), *med*[2](i,j), ⋯, *med*[$M-1$](i,j).

In above algorithm, Step 3 shows that each processor gets sum value in 1-D window with the width w by applying Algorithm 2 with an exception of Step 7 at each level. Step 5 explains the process of obtaining the binary median output at each level. Because each processor knows sum value in 1-D window of width w centered at itself from Step 3, PE(i,j) positioned at the center of 2-D window must collect the sum values of every 1-D window within the limits of 2-D window and combine them to obtain the median output in the binary domain by using linear separability. In Step 5, the sum value of

2-D window $sum_{2D}(i,j)$ is initialized in (1) and 1-D sum value in each PE is shifted and combined into $sum_{2D}(i,j)$ $\lfloor w/2 \rfloor$ times in (2) and (3). And then $sum_{2D}(i,j)$ is thresholded by $(w^2+1)/2$ to obtain 2-D median output in the binary domain. In Step 6, PE(i,j) examines the level whose output is 1 and the next greater output is 0 from $med[1](i,j)$, $med[2](i,j)$, $\cdots$, $med[M-1](i,j)$ to get 2-D median output in the integer domain.

5. Performance Analysis

In this section, we evaluate the performance of our RMESH algorithms by comparing their time complexities with those of algorithms on mesh-connected computer. It is possible to compare time complexity on RMESH with that on mesh because the reconfigurable mesh architecture just injects switches with circuit-switching capability into every PE of mesh. This is reasonable in the light of experiments with the YUPPIE[2] and the PPA[18].

Before computing time complexities of two parallel models, let us consider the median filtering on a single processor. In 1-D signal, the median filter in the integer domain is computed in $O(w\log w)$ time within a window, which is obtained by sorting the window samples and selecting the median value. So the total time complexity is $O(Nw\log w)$, where N is the input signal length. In 2-D image, the time complexity of the median filter is computed equally and it is $O(N^2w^2\log w^2)$, where image size is $N\times N$ and window size is $w\times w$.

Now, let us consider the time complexities of our RMESH algorithms. In 1-D signal, Step 1 and Step 3 of Algorithm 1 can be done in $O(M)$ time for M-level signal and Step 2 applies Algorithm 2 at each level. So we need to check the time of Algorithm 2. In Algorithm 2, Step 2 and Step 3 are easily obtained in $O(w)$ time respectively and also Step 6 can be obtained in same time, so that the time complexity of Algorithm 2 is $O(w)$. Consequently, Step 2 of Algorithm 1 is done in $O(Mw)$ and the time complexity of 1-D RMESH algorithm is $O(M)+O(Mw)+O(M)\approx O(Mw)$. In 2-D image, Step 1 of Algorithm 3 is done in $O(M)$ time and Step 3 is in $O(Mw)$ time because the time complexity of Algorithm 2 is $O(w)$. And Step 5 is computed in $O(M\lfloor w/2 \rfloor)$ time and Step 6 is in $O(M)$ time. So the time complexity of 2-D RMESH algorithm is $O(M)+O(Mw)+O(M\lfloor w/2 \rfloor)+O(M)\approx O(Mw)$.

Finally, let us compute the time complexity on mesh. 1-D signal is mapped onto the row 0 of mesh such that PE(i) contains M-valued signal $X(i)$ and 2-D image is mapped on $N\times N$ mesh such that PE(i,j) corresponds to the image pixel $X(i,j)$. The parallel algorithm on mesh is similar to our RMESH algorithm except for data routing technique. In 1-D signal, Step 2, Step 3 and Step 5 of Algorithm 2 is done in $O(\lfloor w/2 \rfloor)$ time respectively because maximum routing distance between PEs is $\lfloor w/2 \rfloor$. But, in Step 6, $O(\lfloor w/2 \rfloor)$ time is needed to update next value $n(i)$ at each loop as compared with $O(1)$ time on RMESH. Therefore, Algorithm 2 is done in $O(w^2)$ time and the parallel algorithm on mesh is computed in $O(M)+O(Mw^2)+O(M)\approx O(Mw^2)$ time. In 2-D image, the time complexity is $O(M)+O(Mw^2)+O(M\lfloor w/2 \rfloor)+O(M)\approx O(Mw^2)$.

The time complexities of various models are summarized in Table 2. Although the time complexities on mesh architecture are considerably improved over the complexities on a single processor because N is even larger than both M and w, it is clear that our RMESH algorithms significantly outperform mesh algorithms.

〈Table 2〉 Time complexities

Model	1-D signal	2-D image
Single processor	$O(Nw\log w)$	$O(N^2w^2\log w^2)$
mesh architecture	$O(Mw^2)$	$O(Mw^2)$
RMESH architecture	$O(Mw)$	$O(Mw)$

6. Conclusion

Median filter is included in the class of stack filters which have the principle of threshold decomposition, stacking property, and binary processing based on PBF. So the median filter can be implemented in the binary domain by using the properties of the stack filter. And the binary domain filtering makes it possible to implement in VLSI. In this paper, we have developed the RMESH algorithms for median filtering of 1-D signal and 2-D image, which are suitable for VLSI implementation. In addition, we have proved that the algorithmic time complexities on RMESH were far better than those on mesh. The fact shows that RMESH architecture has an excellent performance in comparison with mesh architecture.

References

[1] H. Li and M. Maresca, "Polymorphic-torus Network," IEEE Trans. Comput., Vol.38, No.9, pp. 1345-1351, 1989.

[2] M. Maresca and H. Li, "Connection Autonomy in SIMD Computers : A VLSI Implementation," J. Parallel Distrib. Computing, Vol.7, pp.302-320, 1989.

[3] R. Miller, V. K. Prasanna-Kumar, D. I. Reisis, and Q. F. Stout, "Parallel Computations on Reconfigurable Meshes," IEEE Trans. Computers, Vol.42, No.6, pp.678-692, 1993.

[4] R. Miller, V. K. Prasanna-Kumar, D. I. Reisis, and Q. F. Stout, "Data Movement Operations and Applications on Reconfigurable VLSI Arrays," Proc. Int. Conf. Parallel Processing, pp.205-208, 1988.

[5] 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 Distrib. Syst., Vol.1, No.4, pp.500-507, 1990.

[6] Y. Ben-Asher, D. Peleg, and A. Schuster, "The Power of Reconfiguration," J. Parallel Distrib. Computing, Vol.13, pp.139-153, 1991.

[7] J. W. Tukey, "Nonlinear (nonsuperposal) methods for smoothing data," Conf. Rec. EASCON'74, pp. 673, 1974.

[8] J. P. Fitch, E. J. Coyle, and N. C. Gallagher, Jr., "Median Filtering by Threshold Decomposition," IEEE Trans. Acoust., Speech, Signal Processing, Vol.32, pp.1183-1189, 1984.

[9] P. D. Wendt, E. T. Coyle, and N. C. Gallagher, Jr., "Stack Filters," IEEE Trans. Acoust., Speech, Signal Processing, Vol.34, No.4, pp.898-911, 1986.

[10] B. M. Jeon, K. Y. Nam, and C. S. Jeong, "A Parallel Algorithm for Optimal Stack Filter," Proc. ISCA 10th Parallel and Distributed Computing Systems, pp.368-371, 1997.

[11] O. Y. Harja, J. Astola, and Y. Neuvo, "Analysis of the Properties of Median and Weighted Median Filters using Threshold Logic and Stack Filter Representation," IEEE Trans. Signal Processing, Vol.39, No.2, pp.395-410, 1991.

[12] P. T. Yu and W. H. Liao, "Weighted Order Statistics Filter - Their Classification, Some Properties, and Conversion Algorithm," IEEE Trans. Signal Processing, Vol.42, No.10, pp.2678-2691, 1994.

[13] J. Jenq and S. Sahni, "Reconfigurable Mesh Algorithms for The Hough Transform," Proc. Int. Conf. Parallel Processing, Vol.3, pp.34-41, 1991.

[14] J. Jenq and S. Sahni, "Reconfigurable Mesh Algorithms for Image Shrinking, Expanding, Clustering, and Template Matching," Proc. 5th Int. Parallel Processing Symposium, pp.208-215, 1991.

[15] J. Jenq and S. Sahni, "Histogramming on A Reconfigurable Mesh Computers," Proc. 6th Int. Parallel Processing Symposium, pp.425-432, 1992.

[16] J. Jenq and S. Sahni, "Reconfigurable Mesh Algorithms for Fundamental Data Manipulation Operations," Tech-Report 91-031(University of

Florida), 1993.

[17] M. Nigam and S. Sahni, "Sorting n Numbers on $n \times n$ Reconfigurable Mesh with Buses," Tech-Report 92-5(University of Florida), 1992.

[18] M. Maresca, H. Li, and P. Baglietto, "Hardware Support for Fast Reconfigurability in Processor Arrays," Proc. Int. Conf. Parallel Processing, Vol.1, pp.282-289, 1993.

[19] K. Chen, "Bit-Serial Realizations of a Class of Nonlinear Filters Based on Positive Boolean Functions," IEEE Trans. Circuit. Syst., Vol.36, No.6, pp.785-794, 1989.

[20] D. S. Richards, "VLSI Median Filters," IEEE Trans. Acoust., Speech, Signal Processing, Vol. 38, No.1, pp.145-153, 1990.

[21] L. A. Christopher, W. T. Mayweather, and S. S. Perlman, "A VLSI Median Filter for Impulse Noise Elimination in Composite or Component TV Signals," IEEE Trans. Consumer Electronics, Vol.34, No.1, pp.262-267, 1988.

전 병 문

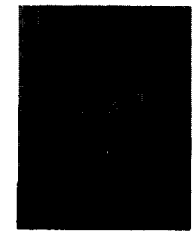
jbm@snoopy.korea.ac.kr

1991년 고려대학교 전자공학과 졸업(학사)

1994년 고려대학교 대학원 전자공학과(공학석사)

1994년~현재 고려대학교 대학원 전자공학과 박사과정

관심분야 : 영상 처리, 병렬 알고리즘, 병렬구조 설계



정 창 성

csjeong@charlie.korea.ac.kr

1981년 서울대학교 전기공학과 졸업(학사)

1985년 Northwestern University Dept. of Computer Science (공학석사)

1987년 Northwestern University Dept. of Computer Science (공학박사)

1987년~1992년 포항공대 전산학과 조교수

1992년~1997년 고려대학교 전자공학과 부교수

1998년~현재 고려대학교 전자공학과 정교수

관심분야 : 병렬/분산 알고리즘, 병렬구조, 병렬/분산 simulation, Visualization, DIS