비트 레벨 정렬 알고리즘을 이용한 3×3 윈도우 가중 메디언 필터의 하드웨어 구현에 관한 연구

論 文
53D-3-9

A Study on the Hardware Implementation of A 3×3 Window Weighted Median Filter Using Bit-Level Sorting Algorithm

李 泰 旭* · 趙 相 福**
(Tae-Wook Lee · Sang-Bock Cho)

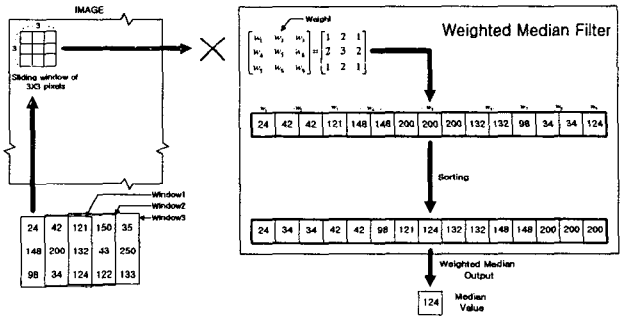
Abstract - In this paper, we studied on the hardware implementation of a 3×3 window weighted median filter using bit-level sorting algorithm. The weighted median filter is a generalization of the median filter that is able to preserve sharp changes in signal and is very effective in removing impulse noise. It has been successfully applied in various areas such as digital signal and video/image processing. The weighted median filters are, for the most part, based on word-level sorting methods, which have more hardware and time complexity. However, the proposed bit-serial sorting algorithm uses weighted adder tree to overcome those disadvantages. It also offers a simple pipelined filter architecture that is highly regular with repeated modules and is very suitable for weighted median filtering. The algorithm was implemented by VHDL and graphical environment in MAX+PlusII of ALTERA. The simulation results indicate that the proposed design method is more efficient than the traditional ones.

Key Words : 메디언 필터(Median filter), 가중 메디언 필터(Weighted median filter), 정렬 알고리즘(Sorting algorithm)

1. 서 론

1971년 Tukey에 의해 제안된 메디언 필터(median filter)는 비선형 필터의 일종으로 디지털 신호 처리 및 영상 신호 처리 등에 많이 사용되고 있다[1]. 메디언 필터는 일정한 크기의 윈도우를 통해 영상의 픽셀 값을 받아들여 크기 순으로 정렬한다. 이때 임펄스 잡음(impulse noise)의 경우 보통 최대값 또는 최소값으로 정렬의 양끝 단에 위치하게 되므로 정렬값의 중앙값을 윈도우의 중심 픽셀의 값으로 대체하는 방법으로 임펄스 잡음을 효과적으로 제거한다. 또한 영상의 급격한 변화인 에지(edge)를 잘 보존하는 성질이 있어 현재까지 영상의 전처리 단계에서 광범위하게 사용되고 있다[2-5]. 메디언 필터의 변형인 가중 메디언 필터는 메디언 필터와 같은 특성을 가지고 있으면서, 잡음의 제거에 더 좋은 성능을 나타내며 가중치의 조절로 주파수 선택이 가능한 장점이 있어 그 응용이 점점 늘어가는 추세이다[6-7]. 가중 메디언 필터는 각 윈도우에 가중치를 두고 윈도우의 픽셀을 가중치 만큼 반복시킨 후 크기 순으로 정렬하여 중앙값을 윈도우의 중심값으로 사용한다. 그림 1은 3×3 윈도우와 일반적으로 많이 쓰이는 가중치 w_i=[1 2 1 2 3 2 1 2 1]을 가진 가중 메디언 필터의 기본 동작을 설명한 것이다[8].

이상과 같이 메디언 및 가중 메디언 필터는 정렬 알고리즘을 바탕으로 수행되며 하드웨어 구현시 많은 시간과 복잡도를 필요로 하므로 이를 개선 하기위한 연구가 활발히 진행되어 왔다.



Each pixel value is duplicated according to the value of the corresponding weight value and the median value is chosen from this modified sequence

그림 1 가중 메디언 필터의 기본 동작
Fig. 1 Basic operation of the weighted median filter

가장 널리 알려진 정렬 방법은 홀/짝 교차 정렬 네트워크(Odd/Even transposition sort network)[8-9]를 이용하는 것과 Batcher가 제안한 홀/짝 합병 정렬 네트워크 (Odd/Even merge sort network)를 이용하는 것이다[10-11]. 그러나 위의 두 알고리즘은 워드 레벨에서의 데이터 정렬에 바탕을 두어 윈도우의 크기가 커짐에 따라 면적 및 시간 복잡도가 비선형적으로 증가하여 필터의 성능을 저하시키는 단점을 가진다. 최근의 연구 방향은 워드 레벨 대신 비트 레벨로 정렬을 수행하는데 맞추어지고 있다[1-4],[12-14]. 비트 레벨 정렬은 입력의 개수 증가에 따라 면적 복잡도는 선형적으로 증가하나 시간 복잡도는 증가하지 않는 특징을 가져, 많은 수의 입력 데이터를 가지는 정렬기 설계에 적합하다. 이러한 비트

* 準 會 員 : 蔚山 大學校 博士課程
** 正 會 員 : 蔚山 大學校 敎授
接受日字 : 2003年 11月 18日
最終完了 : 2004年 1月 20日

레벨 정렬 방식은 대부분 메디언 비트 값을 결정하는데 다수결 함수를 사용한다. 다수결 함수는 구현 방법에 따라 크게 3가지로 분류 된다. 이중 본 논문에서는 다수결 함수를 효과적으로 구현하기 위해 기존의 가산기 트리 방식을 변형한 가중 가산기 트리 방식을 제안하며, 이를 가중 메디언 필터에 적용하였다. 제안된 방식을 이용한 비트 레벨 정렬 알고리즘은 하드웨어로 구현시 규칙적이고 반복적인 구조를 가져 회로의 설계가 용이한 장점을 가진다.

본 논문의 구성은 다음과 같다. 2장에는 기존의 워드 레벨 정렬 알고리즘에 대해 논하고 3장에서 비트 레벨 정렬 알고리즘에 대해 살펴본다. 그리고 4장에서는 비트 레벨 정렬 알고리즘의 하드웨어 구현에 대해 살펴보며 5장에서 성능 분석 및 시뮬레이션을 수행한다. 끝으로 6장에서는 성능 분석 결과를 바탕으로 결론을 맺는다.

2. 워드 레벨 정렬 알고리즘

2.1 홀/짝 교차 정렬 네트워크[8-9]

그림 2는 5입력에 대한 홀/짝 교차 정렬 과정을 나타낸 것이다. 그림에서 보듯이 5개의 입력은 홀수 정렬기를 통해 1, 2입력과 3, 4입력이 정렬되고 짝수 정렬기를 통해서는 2, 3입력과 4, 5입력이 정렬되어 5단계의 과정을 반복하면 전체 입력에 대한 정렬결과가 나온다.

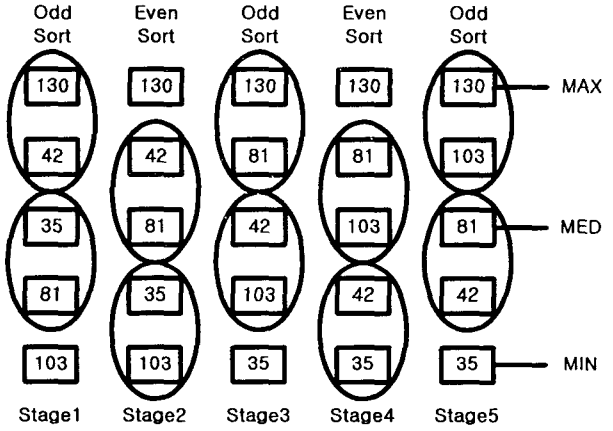


그림 2 5입력 홀/짝 교차 정렬 과정

Fig. 2 Procedure of the 5 input odd/even transposition sort network

홀수 정렬기와 짝수 정렬기는 2입력 비교&교환기(C&S : Compare &Swap module)와 지연소자로 구성할 수 있으며 파이프라인 기법을 이용한 5입력 정렬기를 설계하면 그림 3과 같다. 홀/짝 교차 정렬 네트워크는 일반적으로 입력이 N일 때 파이프라인 스테이지가 N이고 각 파이프라인 스테이지 마다 $(N-1)/2$ 개의 2입력 비교&교환기를 필요로 하므로 입력이 5인 경우 파이프라인 스테이지가 5개 필요하고 각 파이프라인 스테이지 마다 2개의 2입력 비교&교환기를 필요로 한다.

그림 4는 그림 3을 확장하여 가중치 w_i=[1 2 1 2 3 2 1 2 1]를 두었을 경우 가중 메디언 필터의 하드웨어 구조이다. 그림에서 점선으로 표시된 부분은 가중 메디언 필터 구현에서 필요 없는 부분을 나타낸다. 홀/짝 교차 정렬 네트워크는 하드웨어 구현이 비교적 간단한 장점이 있으나 워드레벨로 정렬을 하는데 바탕을 두고 있어 윈도우의 사이즈가 커질 경우 회로가 복잡해지고 전달 지연시간이 커진다는 단점이 있다.

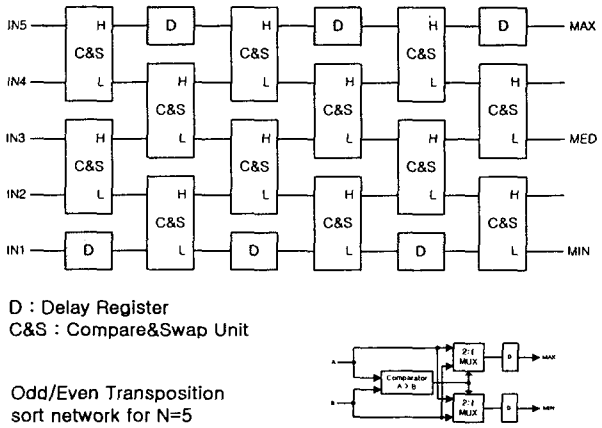


그림 3 5입력 홀/짝 교차 정렬 네트워크

Fig. 3 5 input odd/even transposition sort network

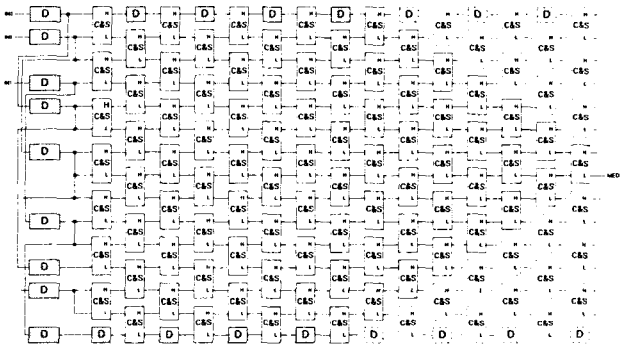


그림 4 홀/짝 교차 정렬 네트워크를 이용한 가중 메디언 필터 구조

Fig. 4 Architecture of the weighted median filter using odd/even transposition network

2.2 Batcher의 홀/짝 합병 정렬 네트워크[10,11]

내림차순으로 정렬된 입력 $A=(a_{n-1}, a_{n-2}, ..., a_0$ 단, $a_{n-1} > a_{n-2} > , ..., > a_0)$, $B=(b_{n-1}, b_{n-2}, ..., b_0$ 단, $b_{n-1} > b_{n-2} > , ..., > b_0)$가 있다고 가정할 때 Batcher가 제안한 홀/짝 합병 정렬 네트워크 생성 알고리즘은 다음과 같다.

1) 만일 n = 1이라면 홀/짝 합병 정렬 네트워크는 단순히 2입력 비교&교환기로 생각할 수 있다.
2) 만일 n > 1면 그림 5의 홀/짝 합병 정렬 네트워크 반복 규칙 구조를 사용하여 구성한다. 입력 A의 홀수 번째 값$(a_{n-1}, a_{n-3}, ..., a_1)$과 입력 B의 홀수 번째 값$(b_{n-1}, b_{n-3}, ..., b_1)$은 $N/2\times N/2$ 홀수 합병 정렬 네트워크를 통해 정렬되어 $(O_{n-1}, O_{n-2}, ..., O_0)$의 값을 생성하며, 같은 방식으로 입력 A의 짝수 번째 값$(a_{n-2}, a_{n-4}, ..., a_0)$과 입력 B의 짝수 번째 값$(b_{n-2}, b_{n-4}, ..., b_0)$은 $N/2\times N/2$ 짝수 합병 정렬 네트워크를 통해 정

렬되어 (e_{n-1}, e_{n-2}, ..., e_0)의 값을 생성한다.
위의 과정을 정리하여 식(1)에 나타내었다.

$$
\begin{aligned}
&R_{2n-1} = o_{n-1} \\
&R_{2m} = max(o_{m-1}, e_m),\ 1 \le m \le n-1 \\
&R_{2m-1} = min(o_{m-1}, e_m),\ 1 \le m \le n-1 \\
&R_o = e_o
\end{aligned}
\tag{1}
$$

이때, R_{2n-1}는 가장 큰 값을 R_0는 가장 작은 값을 나타낸다.

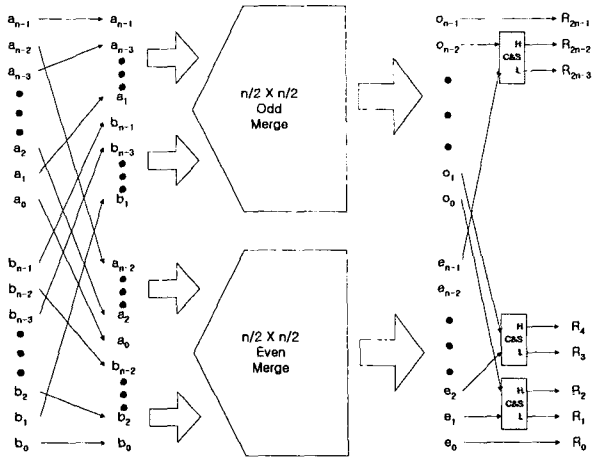


그림 5 홀/짝 합병 정렬 네트워크 반복 규칙 구조
Fig. 5 Circuit recursive definition of the odd/even merge sort network

임의의 $2N$비트의 입력에 대한 정렬은 홀/짝 합병 정렬 네트워크를 반복적으로 사용하여 구현할 수 있다. $2N$비트의 입력을 정렬하기 위해 N개의 홀수 내림차순 입력과 N개의 짝수 내림차순 입력이 필요하므로 2개의 $N/2 \times N/2$ 합병기가 필요하게 된다. 마찬가지로 N개의 입력을 정렬하기 위해서는 $N/2$개의 홀수 내림차순 입력과 $N/2$개의 짝수 내림차순 입력이 필요하므로 4개의 $N/4 \times N/4$합병기가 필요하게 된다. 이런 식으로 최종 1×1 합병기가 될 때까지 반복하여 구성하면 그림 6과 같은 $2N$ 비트에 대한 정렬기 구조가 된다.

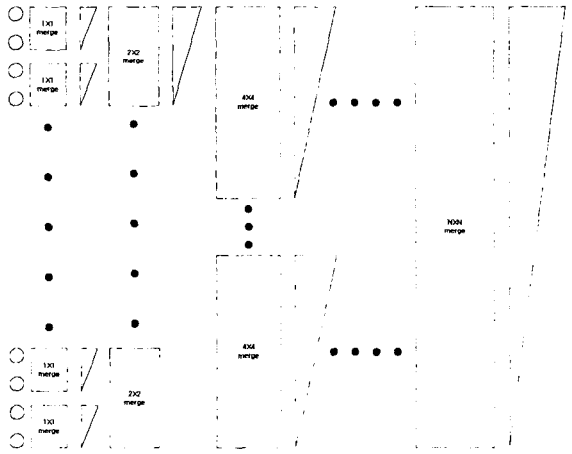


그림 6 2N 비트 정렬을 위한 홀/짝 합병 정렬 네트워크
Fig. 6 Odd/even merge sort network for 2N bit sort

그림 6의 정렬기 구조를 바탕으로 가중치 w_i=[1 2 1 2 3 2 1 2 1]를 두었을 경우 가중 메디언 필터의 하드웨어 구조는 그림 7과 같다.

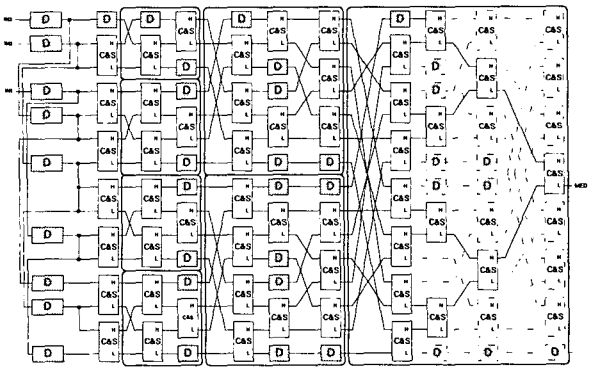


그림 7 홀/짝 합병 정렬 네트워크를 이용한 가중 메디언 필터 구조
Fig. 7 Weighted median filter using odd/even merge sort network

3. 비트 레벨 정렬 알고리즘

s개의 데이터가 부호 없는 정수(unsigned integer)값을 가지며, n비트로 구성된다고 가정하면 입력 값은 식(2)와 같이 표현될 수 있다.

$$
\begin{aligned}
&X = \{x_i\},\ where\ 1 \le i \le s \\
&x_i = (b_i^n b_i^{(n-1)} \cdots b_i^1),
\end{aligned}
\tag{2}
$$

where b^k_i is the k^{th} bit of x_i with a weight of 2^{k-1}

위 식을 행렬 형태로 바꾸어 나타내면 식(3)과 같다.

$$
X = [\ x_1\ \ x_2\ \cdots\ x_s\ ] = \begin{bmatrix} b_1^n & b_2^n & \cdots & b_s^n \\ b_1^{(n-1)} & b_2^{(n-1)} & \cdots & b_s^{(n-1)} \\ \vdots & \vdots & & \vdots \\ b_1^1 & b_2^1 & \cdots & b_s^1 \end{bmatrix} \tag{3}
$$

식(3)을 입력으로 한 비트 레벨 정렬 알고리즘은 다음과 같다.

비트레벨정렬알고리즘:

① Start with $j = n$.
② m_j = majority(b^j_1, b^j_2, ..., b^j_s).
③ Do, in parallel, for all i, where $1 \le i \le s$.
- If $b^j_i = m^j$, then the less significant bits of x_i do not change : ($b^{j-1}_i = b^{j-1}_i$).
- If $b^j_i \neq m^j$, then all is propagated down by one position, replacing the less significant bits of x_i : ($b^{j-1}_i = b^j_i$).

④ Decrement j. If $j \neq 0$, go to step ②.
⑤ The lth ranked number would be : $M = M^n M^{n-1} \ldots m^1$.

여기서, 입력의 l번째 값을 찾기 위한 다수결 함수

(majority function)는 알고리즘 1 또는 알고리즘 2로 정의된다.

다수결함수알고리즘1[12]:

$majority(b_1^j b_2^j \cdots, b_s^j) = lth\ ranked\ bit\ sort(b_1^j b_2^j \cdots, b_s^j)$

다수결함수알고리즘2[13]:

```
For i=1 to s
  sum[j] = sum[j] + b^j_i
end for;
If sum[j] ≤ s - l then  -- Selecting the jth bit
  majority(b^j_1, b^j_2, ..., b^j_s) = 0 -- of the lth ranked number
else
  majority(b^j_1, b^j_2, ..., b^j_s) = 1
end if;
```

비트 레벨 정렬 알고리즘을 가중 메디언 필터에 적용하기 위해서는 입력 비트의 합을 구할 때 가중치를 고려해야 하고, 가중치에 따른 임계 값을 수정하여야 한다. 따라서 본 논문에서는 알고리즘 2를 다음과 같이 수정하여 사용하였다.

가중메디언필터에적합한수정된다수결함수알고리즘:

```
For i = 1 to s
  sum[j] = sum[j] + w_i • b^j_i -- w_i = [w1 w2 ... ws]
end for;
If sum[j] ≤ (t-1)/2 then -- t = (w1 + w2 + ... + ws)
  majority(b^j_1, b^j_2, ..., b^j_s) = 0  -- Selecting the jth bit
else                                      -- of the median number
  majority(b^j_1, b^j_2, ..., b^j_s) = 1
end if;
```

수정된 알고리즘에 따라 가중치 w_i=[1 2 1 2 3 2 1 2 1]를 가진 필터를 설계한다면, 슬라이딩 윈도우 내의 영상 데이터 개수 s는 9가 되고 입력 영상이 그레이 레벨의 값을 가진다면 n은 8이 된다. 또한 가중치에 따른 임계 값을 구하면 t는 7이 된다. 그림 8은 수정된 비트 레벨 정렬 알고리즘을 이용하여 24, 42, 121, 148, 200, 132, 98, 34, 124의 9가지 그레이 레벨 영상과 가중치를 적용하여 가중 메디안 필터링하는 예를 나타낸다.

입력에 대한 최상위 비트 워드인 B^8 = 0 0 0 1 1 1 0 0 0 이 들어오면 가중치와 승산 하여 합한 후, 임계 값 7과 비교하여 '0'을 선택하게 되고 가중 메디안의 최상위 비트(m^8)가 된다. 이 값은 입력의 최상위 비트들과 비교되어 값이 같으면 다음 최상위 비트인 B^7를 받아들이고 값이 다를 경우 B^8의 값에 따라 '0'또는 '1'의 값이 다음 단으로 전파된다. 따라서 B^7의 148과 200 그리고 132의 입력은 전단에서의 B^8과 메디언 비트 값(m^8)이 다르므로 B^7 대신 B^8의 값 '1'이 입력으로 들어온다. 이상의 과정 그림 8과 같이 최하위 비트가 될 때까지 반복하면 9입력에 대한 가중 메디언 값으로 124(011111002)가 출력됨을 확인할 수 있다. 그림의 진하게 표시된 부분은 다수결 함수의 출력과 비트 워드가 다를 경우 전단의 입력 비트가 다음 단으로 전파됨을 나타낸다.

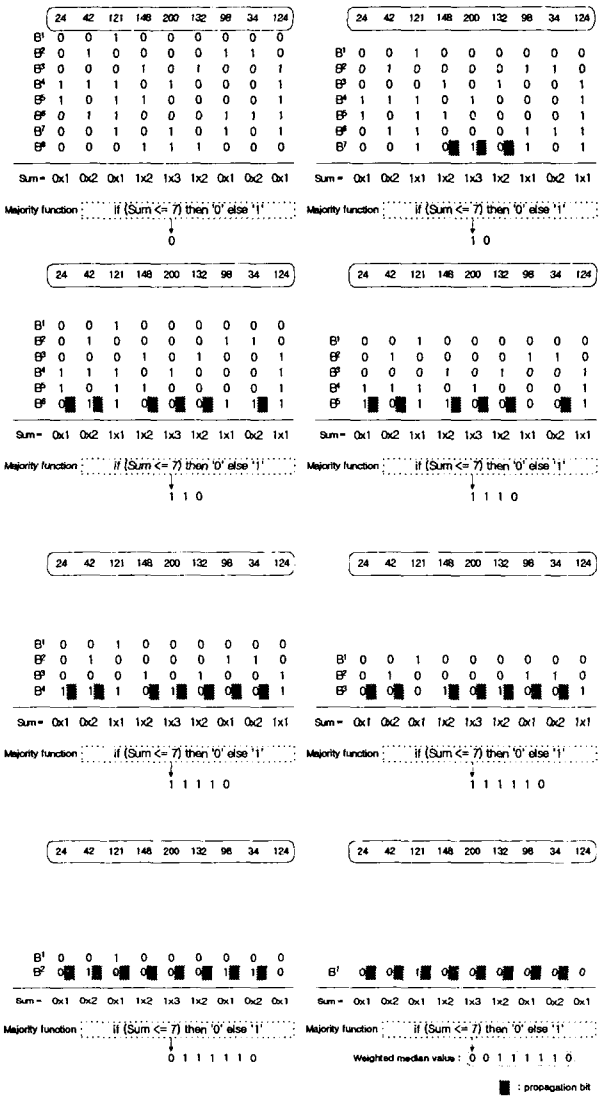


그림 8 9가지 그레이 레벨 영상에 대한 가중 메디언 필터링 예
Fig. 8 The example of the weighted median filter for 9 input grey images

4. 알고리즘의 하드웨어 구현

가중 메디언 값을 찾는 코어 블록은 크게 MS 블록과 MD 블록으로 나누어 설계된다. MS 블록은 규칙적이고 반복적인 구조를 가져 설계가 용이한 장점을 가지며 코어 블록 전체는 입력 데이터를 비트 레벨로 처리하도록 파이프라인 구조로 설계된다.

1. Modifier/Selector 블록 : 임의의 j번째 비트의 입력 값(b^j_i)과 이전의 입력 값(b^{j*}_i)을 선택하여 실질적인 입력 값(b^{j**}_i)을 선택하고 j-1번째 비트에 쓰일 입력 값($b^{j\ 1*}_i$) 및 선택 값($sel^{j\ 1}_i$)을 결정하는 블록
2. Majority or Median Decision 블록 : 9 비트 입력 값(b^j_1, b^j_2, ..., b^j_9)의 다수결 함수를 결정하는 블록

4.1 MS블록(Modifier/SelectorBlock)

MS 블록은 그림 9와 같다. F/F1은 j번째 비트의 입력(b^j_i)을 받아 저장한 후 매 클럭마다 시프트 시켜 다음 단(b^j_{i+1})의 입력으로 사용되도록 한다. 현재의 입력 값(b^j_i)과 이전 입력 값(b^{j*}_i)은 현재 선택 신호(sel^j_i)에 의해 실질적인 입력 값(b^{j**}_i)으로 선택되며 to_md 신호를 통해 MD 블록의 입력으로 보내짐과 동시에 F/F2에 저장되어 다음 비트에 사용될 입력 값(b^{j-1*}_i)으로 쓰인다.

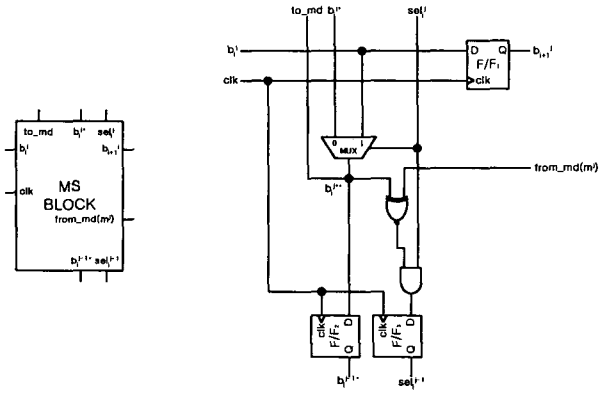


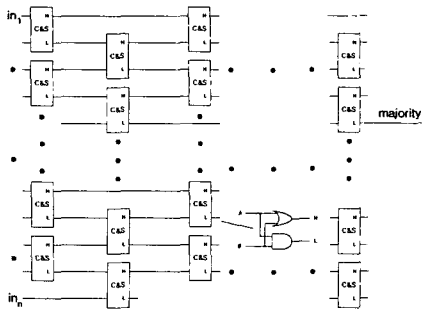
그림 9 MS 블록 및 게이트 레벨
Fig. 9 MS block and gate level

XNOR및 AND 게이트는 b^{j**}_i와 from_md 신호의 비교 결과를 F/F3에 저장하여 다음 비트에 사용될 선택 신호(sel^{j-1}_i)를 생성한다. 여기서 AND 게이트는 선택 신호(sel^j_i)가 한번 '0'이 되면 다음 비트의 선택 신호(sel^{j-1}_i)가 모두 '0'이 되므로 이전의 비트(b^{j**}_i)가 최하위 비트까지 계속 전파될 수 있도록 한다.

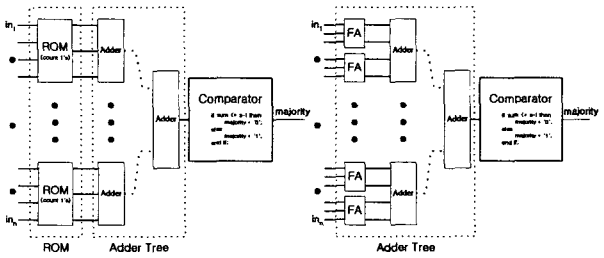
4.2 MD블록(MajorityorMedianDecisionBlock)

MD 블록은 다수결 함수(majority function)를 사용하여 메디언 비트를 결정한다. 다수결 함수는 하드웨어 구현시 복잡도 및 지연시간이 크고 MD 블록의 출력이 다시 MS 블록으로 피드백(feed back)되어 사용되므로 가중 메디언 필터 설계에서 가장 중요한 요소가 된다. 기존의 다수결 함수 구현 방법은 그림 10과 같이 크게 3가지로 나눌 수 있다.

1) 1 비트 홀/짝 교차 정렬 네트워크 이용 [12]: 다수결 함수 알고리즘 1을 이용한 것으로 1비트 홀/짝 교차 정렬 네트워크를 이용하여 정렬을 수행한 후 중앙에 위치한 비트 값을 다수결 함수의 출력으로 사용한다.
2) 롬, 가산기 트리 그리고 비교기 이용 [8]: 다수결 함수 알고리즘 2를 이용한 것으로 4 입력 롬을 사용하여 '1'의 개수를 세고 가산기 트리를 사용하여 전체 입력의 '1'의 개수를 구한다. 그리고 비교기를 통해 가산된 '1'의 개수와 임계 값을 비교하여 다수결 함수의 출력을 결정한다.
3) 가산기 트리와 비교기 이용 [14]: 2)의 방법과 같이 다수결 함수 알고리즘 2를 이용하나 4 입력 롬을 사용하지 않고 가산기 트리 만을 사용하여 전체 입력의 '1'의 개수를 구한다. 다수결 함수의 출력은 비교기를 통해 구해진다.



(a) 1-bit odd/even transposition sort network



(b) Rom, adder tree and comparator　　(c) Adder tree and comparator

그림 10 다수결 함수 구현을 위한 하드웨어 구조
Fig. 10 Hardware architecture of majority function

3가지 구현 방법 중 1비트 홀/짝 교차 정렬 네트워크를 이용한 방법은 입력의 개수 증가에 따라 비교&선택기 개수가 비선형 적으로 증가하는 단점을 가지며, 롬, 가산기 트리 그리고 비교기를 이용한 방법 역시 4입력 롬 구현시 많은 수의 로직 셀을 필요로 한다. 가산기 트리와 비교기를 이용한 방법은 기존 방식들 보다 입력 개수에 따른 하드웨어 복잡도가 적고 구현이 간단한 장점이 있어 다수결 함수 구현에 가장 적합하다. 그림 11은 가중치 w_i=[1 2 1 2 3 2 1 2 1]를 가진 MD 블록을 가산기 트리를 사용하여 구현한 것이다. 각 입력 비트는 가중치만큼 반복되므로 가중치를 고려한 MD 블록 입력은 15가 된다. 15 입력시 다수결 함수가 '1'이 되는 범위는 가산 결과가 8(1000) ~ 15(1111) 이므로 가산 결과의 최상위 비트에 따라 다수결 함수의 출력이 정의된다. 따라서 15 입력의 MD 블록은 비교기 없이 가산기 트리만으로 구현 가능하다.

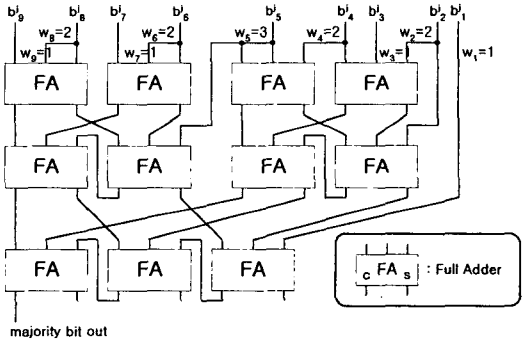


그림 11 가산기 트리를 이용한 MD 블록 구조
Fig. 11 MD block architecture using adder tree

본 논문에서는 MD 블록에 3)의 방식을 변형한 가중 가산기 트리 구조를 사용하여 다수결 함수를 구현한다. 가중 가산기 트리는 입력의 각 비트와 가중치를 승산한 9개의 입력을 가지며 입력 비트의 합은 가산기 트리 구조를 이용하여 구한다. 만일 입력 비트와 가중치가 식(4)와 같다면,

$$\begin{aligned} b_i^j &= (b_1^j b_2^j b_3^j b_4^j b_5^j b_6^j b_7^j b_8^j b_9^j) \\ w_i &= (w_1 w_2 w_3 w_4 w_5 w_6 w_7 w_8 w_9) \\ &= (1\,2\,1\,2\,3\,2\,1\,2\,1) \end{aligned} \tag{4}$$

가중치를 고려한 입력 비트의 합은 식(5)와 같이 나타난다.

$$\begin{aligned} sum[j] &= b_i^j \cdot w_i \\ &= (2^0 b_1^j + 2^1 b_2^j + 2^0 b_3^j + 2^1 b_4^j + 3b_5^j + 2^1 b_6^j + 2^0 b_7^j + 2^1 b_8^j + 2^0 b_9^j) \\ &= 2^0 (b_1^j + b_3^j + b_7^j + b_9^j) + 2^1 (b_2^j + b_4^j + b_6^j + b_8^j) + 3b_5^j \end{aligned} \tag{5}$$

이 중 $3b^j_5$는 $2^0b^j_5 + 2^1b^j_5$로 나타낼 수 있으므로 식(5)를 다시 표현 하면 식(6)과 같다.

$$sum[j] = 2^0 (b_1^j + b_3^j + b_5^j + b_7^j + b_9^j) + 2^1 (b_2^j + b_4^j + b_5^j + b_6^j + b_8^j) \tag{6}$$

식(6)을 이용한 가중 가산기 트리 구조의 MD 블록은 그림 12와 같다.

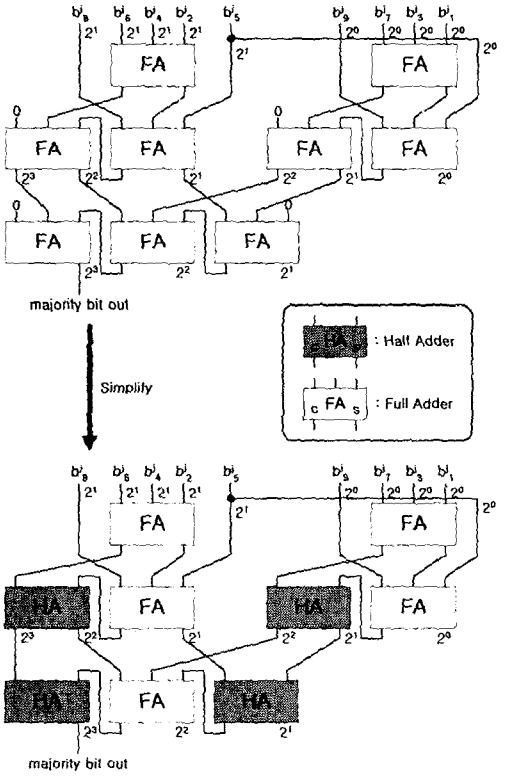


그림 12 가중 가산기 트리를 이용한 MD 블록 구조
Fig. 12 MD block architecture using weighted adder tree

4.3 가중 메디언 필터 전체 구조

가중 메디언 필터는 크게 3개의 블록으로 구성된다. 첫째, 매 클럭마다 3개의 입력을 받아들이는 입력 쉬프트 레지스터 어레이(input shift register array). 둘째, 가중 메디언 필터링을 수행하는 가중 메디언 정렬 코어(weighted median sorting core). 셋째, 메디언 비트 값을 저장하는 출력 쉬프트 레지스터 어레이(output shift register array)이다. 가중 메디언 정렬 코어는 그림 13과 같이 9개의 MS 블록과 1개의 MD 블록으로 구성된 비트 슬라이스(bit-slice)구조를 가진다. 비트 슬라이스의 MS 블록은 매 클럭마다 3개의 입력을 받아 들일 수 있도록 3개의 블록이 동시에 동작하며 MD 블록의 입력 신호를 출력한다. MD의 출력은 MS 블록에 피드백 됨과 동시 출력 시프트 레지스터 어레이에 저장된다. MS 블록에 피드백 된 값은 다음 비트의 선택 값(sel^{j-1}_i)을 결정하는데 사용된다.

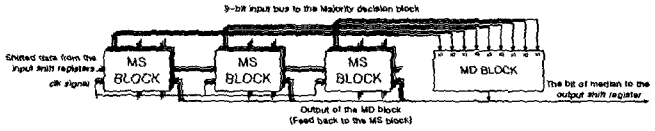


그림 13 가중 메디언 필터 코어의 비트 슬라이스 구조
Fig. 13 Bit slice architecture of the weighted median filter core

그림 14는 w_i=[1 2 1 2 3 2 1 2 1]의 가중치를 가지는 3×3 윈도우 가중 메디언 필터의 하드웨어 구조를 나타낸 것이다. 설계된 메디언 필터는 파이프라인 구조를 가지고 있기 때문에 초기 입력이 들어가 출력이 나오는데 10클럭이 필요하나 그 다음 클럭부터는 매 클럭마다 출력이 나온다.

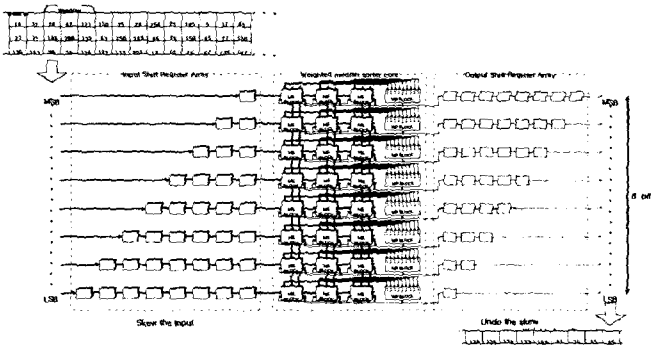


그림 14 3×3 윈도우 가중 메디언 필터의 하드웨어 구조 (w_i=[1 2 1 2 3 2 1 2 1])
Fig. 14 Hardware architecture of the 3×3 window weighted median filter(w_i=[1 2 1 2 3 2 1 2 1])

5. 성능비교 및 시뮬레이션

가중 메디언 필터 설계는 하드웨어 모델링 언어의 하나인 VHDL을 사용하였으며 회로의 합성 및 시뮬레이션은 ALTERA사의 MAX+PlusII를 사용하였다. 성능 비교는 다음의 3가지 방식으로 수행하였다. 먼저 워드 레벨과 비트 레벨

정렬 알고리즘의 성능을 비교하여 비트 레벨 알고리즘이 면적 및 시간 복잡도에서 뛰어남을 확인하였다. 다음으로 비트 레벨 정렬기에서 가장 중요시 되는 다수결 함수를 기존 방식 및 제안된 방식으로 구현하여 성능 비교하였다. 마지막으로 제안된 방식을 이용한 가중 메디언 필터를 구현한 후 기존 방식들과 로직 셀 및 동작 주파수를 비교하여 제안된 방식의 우수성을 확인하였다.

5.1 워드 레벨 정렬기와 비트 레벨 정렬기의 성능 비교

표 1은 워드 레벨과 비트 레벨 정렬 알고리즘을 면적 및 시간 복잡도로 나타낸 것이다.

표 1 워드 레벨과 비트 레벨 정렬 알고리즘 비교

Table 1 Comparison of the word-level and the bit-level sort algorithm

Algorithm	Area complexity	Time complexity
Odd/Even Transposition sort	$\theta = N(N+j)$	$\theta = Nj$
Odd/Even Merge sort	$\theta = (N \cdot \log^2 N \cdot j)$	$\theta = (\log^2 N \cdot j)$
Bit level sort	$\theta = N \cdot j$	$\theta = j$

입력 영상 데이터의 비트 수를 j라 할 때 홀/짝 교차 정렬 알고리즘은 N개의 입력 데이터에 대해 $N(N-1)/2$개의 비교&교환기와 N개의 파이프라인 스테이지를 필요로 하여 면적 및 시간 복잡도가 각각 $\theta = N(N+)j$ 및 $\theta = Nj$가 된다. 홀/짝 합병 정렬 알고리즘은 같은 입력 조건에 대해 $N(1 - 1/N + (\log N \cdot (\log N-1))/4)$개의 비교&교환기와 $\log N \cdot (\log N+1)/2$의 파이프라인 스테이지를 필요로 하여 면적 및 시간 복잡도가 각각 $\theta = (N \cdot \log^2 N \cdot j)$, $\theta = (\log^2 N \cdot j)$이 된다. 위의 두 알고리즘을 비교한다면, 홀/짝 합병 정렬 알고리즘이 홀/짝 교차 정렬 알고리즘에 비해 면적 및 시간 복잡도가 우수함을 알 수 있다. 그러나 워드 레벨 정렬 알고리즘은 입력의 개수 증가에 따라 면적 및 시간 복잡도가 비선형적으로 증가하는 단점을 가지므로 입력 데이터 개수가 많은 정렬기 설계에는 비효율적임을 알 수 있다. 반면, 비트 레벨 정렬 알고리즘은 입력의 개수 증가에 따라 MS 블록 개수만 증가하면 되므로 면적 및 시간 복잡도가 각각 $\theta = N \cdot j$ 및 $\theta = j$이 되어 면적 복잡도는 선형적으로 증가하고 시간 복잡도는 증가하지 않는 장점을 가진다. 따라서 비트 레벨 정렬 알고리즘은 면적 복잡도의 선형적 증가 특성이 있기 때문에 입력 개수가 많은 정렬기를 효율적으로 구현할 수 있다.

5.2 비트 레벨 정렬기의 MD 블록 성능 비교

비트 레벨 정렬 알고리즘은 3절에 설명 하였듯이 다수결 함수 구현 방식의 차이만 가지므로 MS 블록을 제외한 MD 블록의 성능만 고려하였다. 표 2는 제안된 가중 가산기 트리 방식이 기존의 가산기 트리 방식 보다 유용함을 증명하기위해 나타낸 것으로, MD 블록을 2 가지 방식으로 구현시 사용되는 하드웨어를 비교한 것이다.

표 2 MD 블록 구현시 사용되는 하드웨어 비교(가산기 트리 및 가중 가산기 트리 사용)

Table 2 Comparison of required hardware to implement the MD block(using adder tree and weighted adder tree)

Design Method	Adder Tree	Weighted Adder Tree	Logic Savings
HA	-	4	-
FA	11	5	-
Logic	55	33	22

표 2에 나타낸 바와 같이 기존 가산기 트리 방식은 11개의 전가산기(FA: Full Adder)로 구성되나 제안된 가중 가산기 트리 방식은 4개의 반 가산기(HA: Half Adder)와 5개의 전 가산기(FA: Full Adder)로 구성된다. 반 가산기와 전 가산기에 사용되는 XOR, AND 및 OR게이트를 하나의 로직으로 가정한다면, MD블록 구현시 필요한 로직 수는 가산기 트리 방식이 55개 이고 가중 가산기 트리 방식은 33개이므로 22개의 로직이 절약된다.

표 3은 가산기 트리를 제외한 기존 방식들과 가중 가산기 트리 방식을 사용하여 MD 블록을 구현하였을 때 사용되는 하드웨어 및 로직 셀을 비교한 것이다.

표 3 MD 블록 구현시 사용되는 하드웨어 및 로직 셀 비교(기존 방식 및 가중 가산기 트리)

Table 3 Comparison of required hardware and logic cells to implement MD block(using conventional methods and weighted adder tree)

Design Method	C&S	ROM	HA	FA	Logic Cell Used
1 bit odd/even transport sort	84	-	-	-	83
Rom, adder tree and comparator	-	4	3	-	67
Weithted adder tree and comparator	-	-	4	5	22

표 3을 살펴보면 1 비트 홀/짝 교차 정렬 방식이 가장 많은 로직 셀을 필요로 하고 제안된 가중 가산기 트리 방식이 가장 적은 로직 셀을 필요로 함을 알 수 있다. 표 2와 표 3에서 살펴 보았듯이 가중 가산기 트리 방식은 기존 방식들에 비해 MD블록 구현에 있어 가장 효율적이다.

5.3 가중 메디언 필터 성능 비교 및 시뮬레이션

표 4는 5가지 가중 메디언 필터의 합성 결과 비교를 나타낸 것이다. 워드 레벨의 홀/짝 교차 정렬 네트워크를 이용하여 가중 메디언 필터를 구현했을 때 동작 주파수는 26.04MHz이고 로직 셀은 1895개가 사용되었다. 그리고 홀/짝 합병 네트워크를 적용한 경우 동작 주파수는 28.57MHz, 로직 셀은 1101개가 사용되었다. 비트 레벨의 다수결 함수에 1 비트 홀/짝 교차 정렬기를 적용한 경우 동작 주파수는 22.93MHz로 워드 레벨 보다 떨어지나 로직 셀은 931로 줄어듦을 알 수 있다. 이는 MD 블록의 출력이 다시 MS 블록의

표 4 5가지 가중 메디언 필터의 합성 결과 비교

Table 4 Comparison of the synthesis results for 5 types weighted median filter

Design Method	*Design Level*	*Device*	*Logic Cell Used*	*Operating Frequency*
Odd/Even Transport sort network based	Word-level	EPF10K40RC208-3	1895	26.04MHz
Odd/Even merge sort network based	Word-level	EPF10K40RC208-3	1101	28.57MHz
1 bit odd/even transport sort network based in MD blocks	Bit-level	EPF10K20TC144-3	931	22.93MHz
Rom and adder tree based in MD blocks	Bit-level	EPF10K10LC84-3	573	29.76MHz
Weighted adder tree based in MD blocks	Bit-level	EPF10K10LC84-3	453	38.31MHz

입력으로 피드백 되는데 따른 지연 시간 증가로 해석 할 수 있다. MD 블록에 롬, 가산기 트리를 이용한 경우 동작 주파수 및 로직 셀 사용도가 각각 29.76MHz와 573개로 나왔다. 반면 본 논문에서 제안한 가중 가산기 트리를 이용한 경우, MD 블록의 로직과 MS 블록에 피드백 되는 지연 시간을 동시에 줄일 수 있기 때문에 동작 주파수와 로직 셀 사용도가 각각 38.31MHz와 453개로 기존 방식들에 비해 가장 우수한 성능을 나타냄을 알 수 있다.

그림 15는 제안된 방식을 이용한 비트 레벨 가중 메디언 필터의 시뮬레이션 파형을 나타낸 것이다. 그림에서 보듯이 슬라이딩 윈도우는 매 클럭마다 한 칸씩 이동하며 한번 이동 시 마다 3개의 입력이 가중 메디언 필터에 들어간다. 그리고 입력이 들어가서 10클럭 후 첫 슬라이딩 윈도우의 가중 메디언 값이 출력되며 그 다음 클럭부터는 매 클럭마다 가중 메디언 값이 출력 된다.

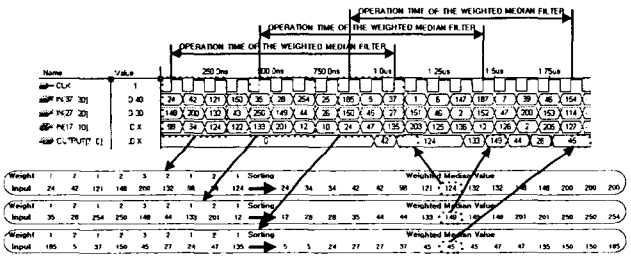


그림 15 제안된 가중 메디언 필터의 시뮬레이션 파형

Fig. 15 Simulation waveform of the proposed weighted median filter

6. 결 론

본 논문에서는 3×3윈도우 가중 메디언 필터를 효율적으로 구현하기 위해 가중 가산기 트리를 이용한 비트 레벨 정렬 알고리즘을 사용하였다. 비트 레벨 정렬 알고리즘은 워드 레벨 정렬 알고리즘보다 입력 개수가 많은 정렬기를 효율적으로 구현할 수 있는 장점이 있으나 다수결 함수 구현에 있어 비효율적인 하드웨어 구조를 가진다. 가중 가산기 트리는 다수결 함수 구현에 있어 기존 방식 보다 로직 셀 사용도 및 동작 속도가 뛰어나 효율적인 하드웨어 구현이 가능하다. 기존 방식과 제안된 방식의 가중 메디언 필터를 VHDL을 사용하여 설계한 후 ALTERA의 MAX+PlusII를 통해 합성 및 시뮬레이션 한 결과, 제안된 방식으로 구현된 가중 메디언 필터가 동작 속도 및 로직 셀 사용도 면에서 기존 방식 보다 우수한 성능을 보임을 확인 하였다. 설계된 가중 메디언 필터는 규칙성과 반복성을 가지며 다수결 함수 구현이 효율적이므로 영상 처리의 잡음 제거나 에지 검출 그리고 디지털 신호 처리 분야의 필터 설계에 유용하게 사용될 수 있다.

감사의 글

본 연구는 한국과학재단 지정 울산대학교 네트워크 기반 자동화연구센터의 지원(NARC)과 산업자원부 반도체 설계 교육센터(IDEC)의 지원으로 수행되었음.

참 고 문 헌

[1] L.W. Chang and J.H. Lin, "Bit level systolic arrays for real time median filters", *in Proc ICASSP*, Vol. 3, pp. 1791 1794, Apr. 1990.

[2] İ. Hatırnaz, F.K. Gürkaynak and Y. Leblebici, "A modular and scalable architecture for the realization of high-speed programmable rank-order filters", *ASIC/SOC '99 Proceedings*, pp. 382-386, 1999.

[3] İ. Hatırnaz, Y. Leblebici, "Scalable binary sorting architecture based on rank ordering with linear area-time complexity", *13th Annual IEEE International ASIC/SOC Conference*, p.p 369 373, 2000.

[4] İ. Hatırnaz, F.K. Gürkaynak and Y. Leblebici, "A compact modular architecture for high-speed binary sorting", *in Proc ICASSP*, Vol. 6, pp. 3339 3342, 2000.

[5] D.S. Richards, "VLSI Median Filters", *IEEE Trans. Acoust., Speech, Signal Processing*, Vol. 38, pp. 145-153, Jan. 1990.

[6] C. Henning and T.G. Noll, "Architecture and implementation of a bitserial sorter for weighted median filtering", *IEEE Custom Integrated Circuits Conf.*, pp. 189 192, May, 1998.

[7] Odeh, S., Shdaifat, I., Kahhaleh, B., Zabalawi, I., "A fast weighted median filter architecture for image processing", *'98 IEEE Proceedings Southeastcon.*, 24-26 Apr, 1998, p.p. 114 117.

[8] K. Benkrid, D. Crookes, A. Benkrid, "Design and Implementation of A Novel Algorithm for General

Purpose Median Filtering On FPGAs", *IEEE International Symposium on Circuits and Systems*, Vol. 4, p.p 425428, 2002.

[9] K. Oflazar, "Design and implementation of a single chip 1-D median filter", *IEEE Trans. Acoust., Speech, Signal Processing*, Vol. ASSP-31, pp. 1164 1168, Oct. 1983.

[10] D.L. Lee and K.E. Batcher, "A Multiway Merge Sorting Network", *IEEE Trans. on Parallel and Distributed Systems*, Vol. 6, No. 2, pp. 211215, Feb. 1995.

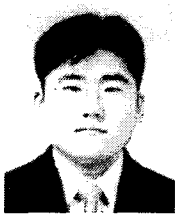
[11] R. Kannan, D.S. Lee, K.Y. Lee, and H.F. Jordan, "Optical TDM sorting networks for high-speed switching", *IEEE Trans. Commun.*, Vol. 45, No. 6, pp. 723736, Jun 1997.

[12] A.A. Hiasat, "Semi-custom VLSI chip implementation of a new two-dimensional separable median filtering algorithm", *in Proc. ICECS 2001*, Vol. 2, pp. 841-844, 2001.

[13] B.K. Kar and D.K. Pradhan, "A new algorithm for order statistic and sorting", *IEEE Trans. on Signal Processing*, Vol. 41, pp. 2688-2694, aug. 1993.

[14] K. Chen, "An integrated bit-serial 9-point median chip", *European Conf. on Circuit Theory and Design*, pp. 339-343, Sep. 1989.

저 자 소 개



이 태 욱 (李 泰 旭)

1973년 5월 28일생. 1998년 울산대학교 전자공학과 졸업. 2000년 동 대학원 전자공학과 졸업(공학석사). 2004 동 대학원 전자공학과 졸업(공학박사)

Tel : 052-259-1629

Fax : 052-259-1686

E-mail : twlee00@korea.com

조 상 복 (趙 相 福)

1955년 6월 10일생. 1979년 한양대학교 전자공학과 졸업. 1981년 동 대학원 전자공학과 졸업(공학석사). 1985년 동 대학원 전자공학과 졸업(공학박사). 1986년~현재 울산대학교 전기전자정보시스템 공학부 교수

Tel : 052-259-2202, Fax : 052-259-1686

E-mail : sbcho@mail.ulsan.ac.kr