論文 96-21-12-3

이진 영상화 방법을 이용한 그레이 준위 영상의 형태학적 연산자의 설계

正會員 신 진 욱*, 박 동 선**

The Design of Gray-Scale Morphological Operators Using Binarization Technique

Jin Wook Shin*, Dong Sun Park** *Regular Members*

요 약

본 논문에서는 영상 처리에 많이 응용되는 그레이 준위 형태학적 연산자의 하드웨어를 제안하며 제안될 하드웨어는 처리속도를 빠르게 하기 위하여 그레이 준위 영상을 이진 영상으로 변환시키는 방법을 이용하며 또한 형태소의 최대값에 영향을 받지 않는 일반화된 하드웨어를 제안한다. 그리고 형태소의 최대값을 제한시킨 기존의 하드웨어와 비교 분석한다. 그리고 단위 하드웨어를 이용하여 형태소가 가변으로 주어지는 응용에서도 개선된 하드웨어를 사용하여 연산 처리가 가능하도록 대단위 병렬 하드웨어 구현 방법을 제시한다. 끝으로 구현된 단위 하드웨어는 하드웨어 시뮬레이션 언어인 VHDL를 이용하여 검증을 하여 유용성을 증명하였다.

ABSTRACT

In this paper we propose fast real-time hardwares for morphological operators using the method of decomposing gray-scale values into binary. The designed hardware is generalized not to be affected by the maximum values of structuring elements. We also propose a method of designing parallel hardware using the unit morphological hardware to deal with situations of using various sizes of structuring elements. The VHDL simulation results of the unit morphological hardwares are identical to those of theorectical operators.

*삼보 정보통신 연구원
**전북대학교 공과 대학 정보 통신공학과 조교수
論文番號:96104-0328
接受日字:1996年 3月 28日

Ⅰ. 서 론

정보통신 분야의 발전에 따라 문자 정보, 음성 정보, 영상 정보 등이 크게 활용되고 있으며 그중 영상 정보는 공장 자동화를 위한 산업 현장에서 자동차의 표면 검사, 직물 검사, 물체의 자동 분류등 여러 응용 분야에서 사용된다[1][2]. 그러나 영상 정보는 획득하는 과정과 전송되는 과정에서 불필요한 잡음의 요소가 첨가되며 영상을 인식하는 과정에서 불필요한 정보가 포함된다. 이러한 불필요한 잡음 등의 영향으로 인하여 영상 정보의 인식 성능은 커다란 영향을 받게 되며 심할 경우 오 인식되는 경우가 발생한다. 그러나 잡음의 영향을 받은 영상에 필터링(Filtering)등과 같은 선처리(Preprocessing)과정을 적용하면 잡음이 제거된 양질의 영상 정보를 얻을 수 있으며 또한 영상 인식에 필요한 정보만을 선택적으로 추출할 수 있다.

필터링을 이용하여 영상을 처리하는 방식에는 선형 필터 방식과 비선형 필터 방식으로 나눌 수 있다. 선형 필터 방식은 적은 계산량으로 처리 속도를 빠르게 할 수 있으나 영상 인식의 필수 요소인 에지 영역 부분 등을 흐리게 만드는 효과를 나타낸다[3]. 반면에 비선형 필터 방식은 영상 인식의 중요한 부분 등을 유지시키면서 불필요한 잡음 등을 제거하는 장점[4][5]을 가지고 있으나 선형 필터와는 달리 많은 연산 처리 시간을 요구한다. 수리 형태학(Mathematical Morphology)은 비선형적인 특성을 가지며 필요한 정보를 선택적으로 유지하고 불필요한 부분을 효과적으로 제거시키는 특성을 가지고 있다[4][6]. 이러한 기본 특성을 가진 수리 형태학은 특징 추출, 잡음 제거, 이미지 고양과 같은 간단한 영상처리로부터 부호화, 의료 사진의 인식, 산업 현장에서의 오류 검출, 항공 사진의 판독 등에 이르는 복잡한 컴퓨터 시각 문제 등의 다양한 분야에서 폭넓게 응용하고있다[1][4].

수리 형태학은 형태소(Structuring Element)라는 영상을 이용하여 화소 단위로 동일한 연산을 반복적으로 수행하며 소프트웨어로 구현할 경우 처리 시간이 매우 길어지게 된다. 최근 초고집적회로의 발전으로 하드웨어 개발비용이 급속히 하락하고 집적도가 높아짐에 따라 복잡한 영상처리에 적용되는 전용 하드웨어의 개발이 많아지고 있다. 특히 알고리즘의 특성이 뛰어나지만 소프트웨어로 구현할 경우 시간이 많이 걸리는 알고리즘들을 위하여 빠른 연산 처리가 가능하도록 전용 하드웨어화가 이루어지고 있다[12]. 따라서 동일한 연산 처리를 반복 수행하는 수리 형태학의 연산자를 위하여 빠른 연산 처리가 가능하도록 병렬적인 수리 형태학의 전용 하드웨어를 설계하여 실시간 처리에 적용할 수 있다.

기존의 형태학적 연산자 하드웨어에는 그레이 준위 영상을 이진 영상의 형태로 변환시켜 연산 처리를 한 방법[7]과 가산기를 이용하여 최대값 또는 최소값을 구하는 형태[8]의 하드웨어가 있다. 이진 영상의 형태로 변환시킨 하드웨어는 형태소의 값을 일정한 값으로 고정시켜 하드웨어의 감소 효과를 얻을 수 있으나 새로운 형태소의 값을 이용하여 연산 처리할 경우에는 하드웨어의 구조상 연산이 불가능하다. 반면에 가산기를 이용한 하드웨어는 형태소의 값에는 영향을 받지 않으나 최대값 또는 최소값을 구하는데 전자의 하드웨어에 비해 많은 연산 시간이 소요된다. 그리고 두 방법 모두 형태소의 크기가 고정되어 있으므로 형태소의 크기를 변화시키면서 연산 처리를 수행하는 다해상도(Multiresolution)응용 분야 등에서는 제약을 받게 된다[2].

본 논문에서는 형태소의 값에 제한을 받지 않고 동일한 입력값을 반복하지 않는 방법을 이용하여 전체 연산 시간을 크게 단축할 수 있는 그레이 준위 영상 처리를 위한 개선된 형태학적 연산자 하드웨어를 제안한다. 그리고 개선된 단위 하드웨어는 가산기를 이용한 방법보다 하드웨어의 요구가 비교적 많지만 연산 처리 시간을 단축하기 위하여 이진 영상으로 표현하는 방법을 선택한다. 그리고 형태소가 가변으로 주어지는 응용에서도 개선된 하드웨어를 사용하여 연산 처리가 가능하도록 대단위 병렬 하드웨어 구현 방법을 제시한다. 개선된 형태학적 연산자의 단위 하드웨어는 하드웨어 시뮬레이션 언어인 VHDL(VHSIC Hardware Description Language)[9][10][11]을 이용하여 모의 실험하였으며 유용성을 증명하였다.

본 논문의 구성은 Ⅱ장에서 이진영상과 그레이 준위 영상의 수리 형태학 기본 연산자들에 대하여 설명하며, Ⅲ장에서는 형태소의 값에 제한 받지 않는 단위 하드웨어를 구성하기 위한 방법을 제안한다. 그리고 Ⅳ장에서는 단위 하드웨어를 병렬로 구성하여 형태소의 크기가 변할 때 사용 가능한 대 단위 하드웨

어를 구현할 수 있는 제안된 방법에 대하여 기술한다. Ⅴ장에서는 제안된 방법을 사용한 하드웨어의 동작을 증명하기 위하여 VHDL를 이용하여 모의 실험을 하였으며 Ⅵ장에서 본 논문의 결론을 기술하였다.

Ⅱ. 수리 형태학의 기본 개념

영역 형태의 표현과 묘사에 있어서 유용한 영상 요소를 추출해내는 도구로서의 수리 형태학은 형태소(Structuring Element)라는 영상을 이용하여 원 영상에 대하여 다양한 연산 처리를 수행한다. 기본 연산에는 영상의 모든 부분을 확장 시켜주는 불림(Dilation) 연산과 이와 반대로 축소 시켜주는 작용을 하는 녹임(Erosion) 연산이 있다. 그리고 이 두 가지 기본 연산 작용을 반복 적용함으로써 수행되는 영상의 윤곽선을 완만히 해주고 길게 돌출된 부분과 좁은 지협을 제거시켜 주는 열림(Opening) 연산과 형태소의 크기보다 작은 구멍을 제거하고 윤곽부분에 있어서의 틈새를 채워주는 닫힘(Closing) 연산이 존재한다[4][6].

2.1과 2.2에서는 이진 영상과 그레이 준위 영상에 적용되는 수리 형태학의 기본 연산자들에 대하여 설명하였다.

2.1 이진 영상의 수리 형태학 연산자

이진 영상의 수리 형태학 연산자는 집합의 개념을 도입하여 표현하며 N-차원 유클리디언 공간 즉, E^N에서 이동(Translation)은 식 (1)과 같이 정의된다[7].

$$(A)_b = \{c \in E^N \mid c = a + b,\ a \in A\},\quad A,\ b \in E^N \tag{1}$$

이동을 이용한 불림 연산은 합집합의 개념을 도입한 형태로써 식 (2)와 같이 정의되며 영상의 전 부분을 확장시켜주는 효과를 나타낸다. 그리고 영상을 축소시키는 효과를 나타내는 녹임 연산은 식 (3)과 같이 정의되며 교집합의 형태로 표현이 가능하다.

$$A \oplus_b B = \bigcup_{b \in B} (A)_b$$

$$= \{c \in E^N \mid c = a + b,\ a \in A \text{ and } b \in B\} \tag{2}$$

$$A \ominus_b B = \bigcup_{b' \in B} (A)_{-b}$$

$$= \{X + b \in A,\ \forall b = B\} \tag{3}$$

2.2 그레이 준위 영상의 수리 형태학 연산자

2-차원 유클리디언 공간에서 처리의 대상이 되는 f는 원영상, k는 형태소 영상이라 할 때, 영상의 고주파 성분을 제거시키며 영상의 모든 부분을 밝게 하는 효과를 나타내는 그레이 준위 불림 연산은, 원 영상의 화소 값과 형태소의 화소 값을 더한 후 형태소의 창 크기 내에서 최대값을 찾는 연산이며 식 (4)와 같이 정의된다. 단 $(i, j) \in f$이다.

$$(f \oplus_g k)(x, y) = \max_{(i, j)} \{f(x-i, y-j) + k(i, j)\} \tag{4}$$

녹임 연산은 영상의 전 부분을 어둡게 만들며 영상의 밝은 정보와 형태소의 크기보다 작은 크기의 영역을 감소시키는 효과를 나타내며 원 영상의 화소 값에서 형태소의 화소 값을 뺀 후 그중 최소값을 구하는 연산자로서 식 (5)와 같이 표현된다.

$$(f \ominus_g k)(x, y) = \min_{(i, j)} \{f(x+i, y+j) - k(i, j)\} \tag{5}$$

Ⅲ. 형태학적 연산자의 단위 하드웨어 설계

수리 형태학은 비선형적인 특징을 가지며 간단한 영상의 부호화, 특징점 추출, 의료 영상의 인식등 영상처리의 모든 분야에서 폭넓게 이용되고 있다. 그러나 형태학적 연산자는 원 영상과 형태소 영상과의 연산이 화소 단위로 이루어지며 또한 동일한 연산을 반복적으로 수행하므로 많은 처리시간을 필요로 한다.

기존의 그레이준위 연산 처리를 위한 형태학적 연산자 하드웨어는 그레이 준위 영상을 이진 영상의 형태로 변환시키고 형태소의 값을 고정시켜 연산을 수행한 방법[7]와 가산기를 이용하여 최대값 또는 최소값을 구하는 형태의 방법[8]이 존재한다. 이진 영상의 형태로 변환시킨 전자의 하드웨어는 형태소의 값을 고정시켰기 때문에 하드웨어의 감소 효과가 있는 반면, 다른 형태소의 값을 이용하여 연산을 수행하고자 할 때는 하드웨어의 구조상 연산이 불가능하다. 가산기를 이용한 하드웨어는 최대값 또는 최소값을 구할

때 연산 시간이 전자의 하드웨어보다 길게 된다. 따라서 본 논문에서는 실 시간 처리의 기본 요소가 될 연산시간을 감소시키기 위하여 그레이 준위값을 이진 영상으로 재구성하는 방법을 이용하며 형태소의 값을 고정시킬 필요가 없도록 가변 형태소의 값을 갖는 하드웨어를 설계한다.

3.1과 3.2에서는 이진영상을 위한 수리형태학의 하드웨어와 그레이준위 영상을 이진영상으로 변환시키는 방법에 대하여 설명을 하며, 3.3에서는 형태소의 값을 고정시킨 기존의 하드웨어에 대하여 설명을 하고 3.4에서는 형태소의 값에 영향받지 않는 수리형태학 하드웨어에 관하여 전개한다.

3.1 이진 영상의 수리 형태학 연산자 하드웨어

이진 영상을 위한 형태학적 연산자 하드웨어는 간단한 논리 게이트만을 이용하여 그림 1(a), (b)와 같이 구현이 가능하다[7]. A는 2-차원 원 이진 영상이고 B는 형태소의 크기가 (N×N)인 영상일때 불림 연산자와 녹임 연산자를 하드웨어로 표현하였다. 이진영상에 대한 불림 연산을 표현한 식 (6)으로 부터 AND 게이트 N^2개와 OR 게이트 1개를 사용하여 그림 1(a)와 같이 불림 하드웨어로 표현할 수 있다[7]. 단 $(x, y)\in A$이다.

$$(A \oplus_b B)_{(x,y)} = OR_{i,j}[A_{(x-i,y-j)} \cdot B_{(i,j)}] \tag{6}$$

반면에 녹임 연산자의 하드웨어는 식 (7)에서 보이고 있으며 OR 게이트 N^2, 인버터 N^2 그리고 AND 게이트 1개를 이용하여 그림 1(b)와 같이 나타낸다.

$$(A \ominus_b B)_{(x,y)} = AND_{i,j}[A_{(x+i,y+j)} + \overline{B_{(i,j)}}] \tag{7}$$

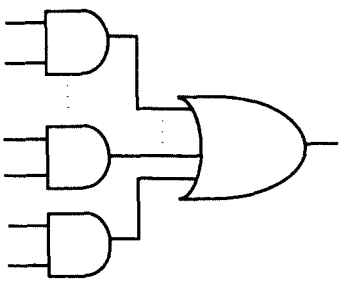

그림 1. (a) 불림 연산자 하드웨어
Fig. 1 (a) Dilation Operator

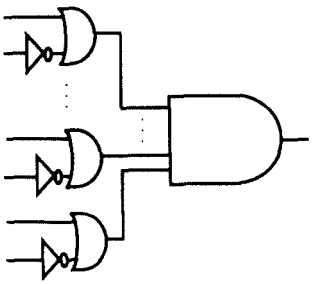

그림 1. (b) 녹임 연산자 하드웨어
Fig. 1 (b) Erosion Operator

3.2 그레이 준위 영상의 이진 영상으로의 변환

하드웨어의 시간 지연 문제점을 해결하기 위하여 그레이 준위값을 이진 영상으로 변환하여, 이진 영상의 하드웨어 구성방법을 이용하면 빠른 연산이 가능한 하드웨어를 설계할 수 있다. 그레이 영상의 값을 이진 영상의 값으로 변환시키는 방법은 다음의 기본 정의 식으로부터 얻을 수 있다[7].

1) 영상에서의 벡터 표현은 창 크기의 범위 즉, N×N안에 있는 화소들을 일렬로 나열한 형태로 식 (8)과 같이 표현된다.

$$\text{vector } \vec{f} : \overrightarrow{f_{1,N^2}} = (f_1, f_2 \ldots, f_{N^2}) \tag{8}$$

2) 그레이 준위 영상의 값을 이진 영상의 값으로 변환시키는 *Slice* S는 영상이 $(-\infty \sim +\infty)$에 존재할 경우, 그레이 준위 입력값 $\vec{fi}$가 해당되는 범위의 값과 같거나 큰 경우에는 '1'의 값을, 작은 경우는 '0'의 값을 가지며 식 (9)와 같이 표현 된다. 그림 2(a)와 (b)는 임의의 입력값에 대하여 $i=3$인 $S[\vec{f3}]$의 연산 예를 보여준다.

$$S[\vec{fi}]_{(x,y)} = \begin{cases} 1 & \text{if } y=i \text{ and } f(x) \geq y \\ 0 & elsewhere \end{cases} \tag{9}$$

x → 0 1 2 3
y ↓ 5 3 4 2

(a) 그레이 준위 입력값 : $\vec{f}$

x→ / y↓	0	1	2	3
5	0	0	0	0
4	0	0	0	0
3	1	1	1	0
2	0	0	0	0
1	0	0	0	0
:	:	:	:	:
-∞	0	0	0	0

(b) $S[\vec{f3}]$

그림 2. (a) 그레이 준위 입력값 $\vec{f}$ (b) $S[\vec{f3}]$
Fig. 2 Example of Slice

3) *Umbra* U는 입력값이 현재의 그레이 준위보다 큰 경우, $-\infty$에서 입력값 까지의 범위를 '1'로 하고 작은 경우에는 '0'으로 하는 연산자이며 식 (10)과 같이 표현된다.

$$U[f]=\{(x, y)\mid y\le f(x)\} \tag{10}$$

그리고 *Umbra* 연산은 *Slice* 연산을 이용하여 식 (11)과 같이 표현 가능하다. 그림 3은 그림 2(a)의 입력값에 대하여 *Umbra* 연산의 수행 예를 보여준다. M은 원 영상 f의 최대값이다.

$$U[\vec{f}]=S[\vec{f0}]\cup S[\vec{f1}]\cup\cdots S[\overrightarrow{f(M-1)}]$$
$$=\bigcup_{i=0}^{M-1} S[\vec{fi}] \tag{11}$$

→ / y↓	0	1	2	3
5	1	0	0	0
4	1	0	1	0
3	1	1	1	0
2	1	1	1	1
:	:	:	:	:
-∞	1	1	1	1

그림 3. Umbra 연산
Fig. 3 Umbra Operation

4) *Top* T는 이진 영상의 값을 그레이 준위값으로 변환시키는 연산으로 *Umbra* 연산을 수행하였을 경우의 '1'인 최대의 입력 레벨값을 말한다.

$$T[A]=\max\{y\mid(x, y)\in A\} \tag{12}$$

3.3 형태소의 값을 제한시킨 기존의 하드웨어

그레이 준위 영상의 값을 이진 영상의 값으로 재구성한 후, 변환된 이진 영상을 이용하여 하드웨어를 구현한 기존의 방법[7]은 형태소 영상과 처리하고자 하는 원 영상의 최대값 크기로 제한을 하여 하드웨어를 구성하였다. 이 방법은 하드웨어의 양을 감소시킬 수 있으나 일정한 값, 즉 구현된 하드웨어의 최대값 이외의 값을 갖는 영상에 응용하기는 불가능하다.

그레이 준위 원 영상을 f, 형태소 영상을 k라 하고 *Umbra* 연산과 *Top* 연산을 이용하여 불림 연산을 표현하면 식 (13)처럼 나타낸다.

$$f\oplus_g k=T[U(f)\oplus_b U(k)] \tag{13}$$

원 영상의 최대값을 P, 고정된 형태소의 최대값을 Q로 고정시키고 이진 영상의 하드웨어를 이용하기 위하여 식 (13)에 Slice 연산을 적용하여 식 (14)로 전개하였다[7].

$$f\oplus_g k=T[\bigcup_{j=0}^{P}\bigcup_{i=0}^{Q}(S[\vec{fj}]\oplus_b S[\vec{ki}])]$$
$$=\{\{[\overrightarrow{f(Q+1)_b}\oplus_b\overrightarrow{k0_b}]\cup[\overrightarrow{fQ_b}\oplus_b\overrightarrow{k1_b}]\cup\cdots\cup$$
$$[\overrightarrow{f1_b}\oplus_b\overrightarrow{kQ_b}]\}+\cdots$$
$$+\{[\overrightarrow{fP_b}\oplus_b\overrightarrow{k(Q-1)_b}]\cup[\overrightarrow{f(P-1)_b}\oplus_b\overrightarrow{kQ_b}]\}$$
$$+\{\overrightarrow{fP_b}\oplus_b\overrightarrow{kQ_b}\}\}+\overrightarrow{Q_c} \tag{14}$$

축입 연산자도 같은 방법으로 전개가 가능하며 식 (15)처럼 표현된다[7].

$$f\ominus_g k=T[(\bigcap_{j=0}^{P}S'[\vec{fj}])\ominus_b(\bigcup_{i=0}^{Q}S[\vec{ki}])]$$
$$=\{\{\overrightarrow{f1_b}\ominus_b\overrightarrow{kQ_b}\}+\cdots$$

$$+\{[\overrightarrow{fQ_b} \ominus_b \overrightarrow{kQ_b}] \cap [\overrightarrow{f(Q-1)_b} \ominus_b \overrightarrow{k(Q-1)_b}] \cap \cdots$$

$$[\overrightarrow{f1_b} \ominus_b \overrightarrow{k1_b}] + \cdots + \{[\overrightarrow{fP_b} \ominus_b \overrightarrow{kQ_b}] \cap$$

$$[\overrightarrow{f(P-1)_b} \ominus_b \overrightarrow{k(Q-1)_b}] \cap \cdots \cap$$

$$[\overrightarrow{f(P-Q)_b} \ominus_b \overrightarrow{k0_b}]\}\} - \overrightarrow{Q_c} \quad (15)$$

단, $\overrightarrow{fQ_b}$는 그레이 준위 Q 일때의 벡터값이며 $\overrightarrow{Q_c}$는 형태소 영상의 최대 상수벡터이다.

식 (14)와 (15)에서 전개된 것처럼 기존의 방법은 형태소의 최대값 $\overrightarrow{Q_c}$에 의존하는 하드웨어가 되며 불림 연산을 위해서는 가산기가 필요하며 녹임 연산을 위해서는 감산기가 필요로 하는 하드웨어임을 알 수 있다. 따라서 특수한 경우를 제외하고 임의의 값을 갖는 일반적인 수리 형태학의 하드웨어로 사용하기에는 불가능하다.

3.4 일반화된 형태학적 연산자의 하드웨어

이진 영상을 이용한 기존 하드웨어는 형태소의 최대값에 의존을 하지만 연산 시간이 빠르므로 일반화된 단위 하드웨어도 연산 처리시간을 단축하기 위하여 이진 영상을 이용한 방법을 사용한다. 그러나 일반화된 하드웨어는 원 영상과 형태소의 최대값에 영향을 받지 않도록 설계를 하며 불림 연산이나 녹임 연산시 가산기나 감산기를 사용하지 않는 하드웨어가 되어 연산 시간의 감소 효과를 얻을 수 있다.

일반화된 불림 연산은 기존의 방법[7]과는 달리 원 영상과 형태소의 값에 제한을 두지 않고 있으며 식 (13)에서 형태소의 최대값부터 전개를 한 반면 본 논문에서는 j=0, i=1부터 전개를 하고 stacking 성질[7] 적용하여 식 (16)과 같이 표현하였고 형태소의 최대값을 더해주는 연산 부분은 존재하지 않음을 알 수 있다. M과 N은 원 영상, f, 형태소 영상 k의 최대값이다.

$$f \oplus_g k = T[(\bigcup_{j=0}^{M-1} S[fj] \ominus_b \bigcup_{i=1}^{N} S[ki]$$

$$= T[\bigcup_{j=0}^{M-1} \bigcup_{i=1}^{N} (S[fj] \oplus_b S[ki])]$$

$$= \{\{\overrightarrow{f0_b} \oplus_b \overrightarrow{k1_b}\}$$

$$+\{[\overrightarrow{f1_b} \oplus_b \overrightarrow{k1_b}] \cup [\overrightarrow{f0_b} \oplus_b \overrightarrow{k2_b}]\}$$

$$+\{[\overrightarrow{f2_b} \oplus_b \overrightarrow{k1_b}] \cup [\overrightarrow{f1_b} \oplus_b \overrightarrow{k2_b}] \cup$$

$$[\overrightarrow{f0_b} \oplus_b \overrightarrow{k3_b}]\} + \cdots$$

$$+\{[\overrightarrow{f(M-1)_b} \oplus_b \overrightarrow{k1_b}] \cup [\overrightarrow{f(M-2)_b} \oplus_b \overrightarrow{k2_b}]$$

$$\cup \cdots \cup [\overrightarrow{f(M-N)_b} \ominus_b \overrightarrow{N_b}]\} \quad (16)$$

녹임 연산은 형태소의 최대값에 영향을 받지 않는 연산식이 존재하며 불림 연산식과는 다르게 전개를 한다. 즉 식(15)와 달리 원 영상의 최상위 레벨과 형태소의 최하위 레벨로 부터 전개를 하며 이를 표현하면 식(17)로 나타낼 수 있고 식 (15)처럼 상수값을 빼주는 연산 부분이 존재하지 않는다.

$$f \ominus_g k = T[(\bigcap_{j=1}^{M} S'[fj]) \ominus_b (\bigcup_{i=0}^{N} S[ki])]$$

$$= T[(S'[fM]) \ominus_b (S[k0])] \cap \{(S'[fM]) \ominus_b (S[k])\}$$

$$\cap \{(S'[f(M-1)]) \ominus_b (S[k0])\} \cap \cdots \cap$$

$$\{(S'[fM]) \ominus_b (S[kN])\} \cap \cdots \cap$$

$$\{(S'[f(M-N)]) \ominus_b (S[k0])\} \cap \cdots \cap$$

$$\{(S'[fN+1]) \ominus_b (S[kN])\} \cap \cdots \cap$$

$$\{(S'[f1]) \ominus_b (S[k0])\}\}$$

$$= \{\{\overrightarrow{fM_b} \ominus_b \overrightarrow{k0_b}\}$$

$$+\{[\overrightarrow{fM_b} \ominus_b \overrightarrow{k1_b}] \cap [\overrightarrow{f(M-1)_b} \ominus_b \overrightarrow{k0_b}]\} + \cdots$$

$$+\{[\overrightarrow{fM_b} \ominus_b \overrightarrow{kN_b}] \cap \cdots \cap [\overrightarrow{f(M-N)_b} \ominus_b \overrightarrow{k0_b}]\} + \cdots$$

$$+\{[\overrightarrow{f(N+1)_b} \ominus_b \overrightarrow{kN_b}] \cap \cdots \cap [\overrightarrow{f1_b} \ominus_b \overrightarrow{k0_b}]\}\} \quad (17)$$

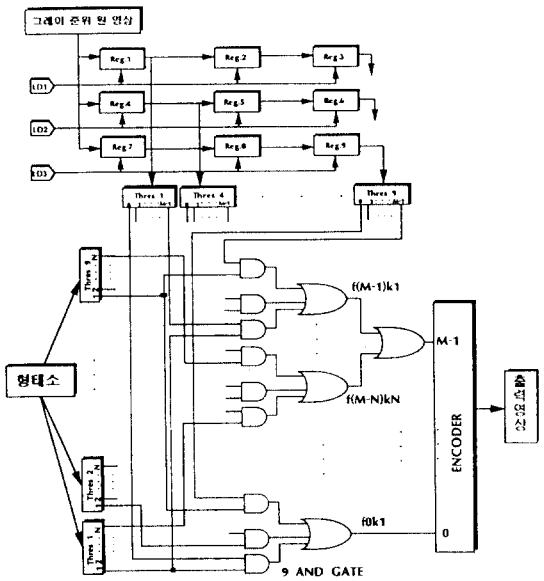

그림 4. 그레이 준위 영상의 불림 하드웨어
Fig. 4 Dilation Operator Hardware for Gray Scale

그림 4는 식 (16)을 이용하여 설계한 그레이 준위 영상처리가 가능하고 원 영상과 형태소의 최대값에 영향을 받지 않으며 형태소의 크기가 3×3인 일반화된 불림 연산자 하드웨어의 블럭도이다. 입력 부분을 행렬 형태로 구성하여 한번 읽어온 화소값은 오른쪽 레지스터 블럭으로 이동이 되며 다음 연산을 위하여 윈 영상에서 3개의 화소값만을 입력하므로 실제적인 연산 처리 시간을 단축할 수 있는 장점이 있다. Thres 블럭은 그레이 준위값을 이진 영상의 형태로 바꾸어 주는 블럭이며 예로써 '3'의 값은 '1' '1' '1' '0' '0' ... 으로 변환된다. 게이트들로 구성된 부분이 이진 영상의 값을 이용한 연산이 이루어지는 블럭이며 Encoder 블럭은 '1'과 '0'으로 이루어진 스트림을 '1'의 개수만큼 그레이 준위값으로 변환 시켜주는 블록이다.

그림 4와 참고문헌[7]의 그림 6에서 단위 게이트를 통과하는 시간이 같다고 가정할 때 이진화 방법을 이용한 두가지 방법 모두 시간지연이 같음을 알 수 있다. 그러나 본 논문에서 입력부분을 병렬로 설계함으로써 N번 클럭후 하나의 결과가 나오며 기존의 방법[7]은 N^2번후 결과가 출력되므로 처리시간을 $1/N$만큼 줄일 수 있다. 단, N은 형태소의 크기이다.

Ⅳ. 단위 하드웨어를 이용한 대 단위 병렬 하드웨어 구성

수리 형태학을 이용한 영상처리에서는 형태소의 크기(Size)와 값(Value)이 영상 처리 결과에 많은 영향을 주며, 그중 형태소의 크기는 중요한 역할을 하며 형태소의 크기를 변화시키면서 연산을 수행하는 다해상도(Multi-resolution) 영상처리 분야등[2]에 많이 응용된다. 그러나 구현된 단위 하드웨어의 형태소 크기와 다른 크기의 하드웨어를 이용하고자 할 때, 응용하는 분야마다 하드웨어를 구현한다는 것은 어려움이 따르며 구현된다 할지라도 하드웨어 낭비를 초래한다. 따라서 형태소의 크기가 변할 때 단위 하드웨어를 병렬로 구성하여 형태소의 크기에 능동적으로 대처할 수 있는 방법이 필요하다. 그리고 이러한 방법은 구현된 단위 하드웨어의 형태소의 크기가 응용하고자 하는 형태소의 크기보다 작은 경우와 큰 경우로 나누어 표현할 수 있다.

4.1과 4.2에서 구현된 하드웨어의 형태소보다 큰경우와 작은 경우를 나누어 설명을 하며 단위 하드웨어를 이용한 형태소의 크기에 영향을 받지 않는 하드웨어 설계 방법에 관하여 설명을 한다.

4.1 형태소가 큰 경우

사용하고자 하는 형태소의 크기가 $(2m+1\times 2m+1)$이고 구현된 단위 하드웨어의 크기가 $(N\times N)$이라 가정하자. 단, m은 $\neq 0$인 정수이다. 이때 이용하고자 하는 형태소의 크기가 구현된 형태소의 크기보다 클 경우 즉, $(2m+1\times 2m+1)\geq(N\times N)$인 경우 단위 하드웨어 여러 개를 동시에 병렬로 처리하여 $(2m+1\times 2m+1)$ 크기의 형태소 연산과 같은 결과값을 얻을 수 있다.

형태소의 크기가 구현된 하드웨어보다 큰 경우의 불림 연산은 단위 하드웨어를 병렬로 구성하여 식 (18)로 전개할 수 있다. 형태소가 겹치는 부분의 처리는 마지막 블럭에서 한번만 겹치는 방법으로 구성하였고 각각의 단위 하드웨어 블럭부분에서는 연산된 최대값이 출력되며 그중 최종의 최대값은 소프트웨어나 하드웨어로 처리하여 구하게 된다. 녹임 연산인 경우에도 마찬가지로 식 (19)처럼 단위 하드웨어를 병렬로 구성한 후 최소값을 구함으로써 형태소가 큰 경우에도 단위하드웨어를 사용하여 구현할 수 있다.

$$(f \oplus_g k)(x, y) = \underset{(a, b)}{MAX} \{f(x-a, y-b) + k(a, b)\}$$

$$= MAX\{(f \oplus_g k_1), (f \oplus_g k_2), \cdots, (f \oplus_g k_n)\} \quad (18)$$

$$(f \ominus_g k)(x, y) = \underset{(a, b)}{MIN} \{f(x+a, y+b) - k(a, b)\}$$

$$= MIN\{(f \ominus_g k_1), (f \ominus_g k_2), \cdots, (f \ominus_g k_n)\} \quad (19)$$

단, 식 (18)과 (19)에서 $k = k_1 \cup k_2 \cup \cdots \cup k_n$이다.

5×5 크기의 형태소를 이용하여 연산을 수행하고자 할 경우, 구현된 단위 하드웨어의 형태소의 크기가 3×3인 경우의 예를 그림 5에서 보여주고 있다.

4.2 형태소가 작은 경우

이용하고자 하는 형태소의 크기가 구현된 단위 하드웨어의 형태소 크기보다 작은 경우 즉, $(2m+1 \times 2m+1) < (N \times N)$일 경우에는 $N + (2m+1)$ 부분만을 연산 처리하는데 사용하면 되므로 연산 수행에 참가하지 않는 부분을 연산 결과에 영향을 주지 않는 값으로 초기화하여 설계할 수 있다.

불림 연산을 수행하기 위한 방법은 이진 영상의 형태로 구현되기 때문에 이용되지 않는 부분을 '0'으로 초기화하여 전설한다. 식 (20)은 불림 연산을 분해하여 나타낸 식이며 하드웨어로 구현시 '∪'은 그림 4에서 나타낸 것처럼 OR 게이트로 구현이 되므로 사용되지 않는 $\{\overrightarrow{f_{(2m+1)+1, N}} \oplus_b \overrightarrow{k_{(2m+1)+1, N}}\}$부분은 OR 게이트의 영향을 받지 않도록 주변의 값들을 $\vec{0}$으로 초기화함으로써 구현한다.

$$\overrightarrow{f_{1, N}} \oplus_b \overrightarrow{k_{1, N}} = \{\overrightarrow{f_{1, (2m+1)}} \oplus_b \overrightarrow{k_{1, (2m+1)}}\}$$

$$\cup \{\overrightarrow{f_{(2m+1)+1, N}} \oplus_b \overrightarrow{k_{(2m+1)+1, N}}\}$$

$$= \{\overrightarrow{f_{1, (2m+1)}} \oplus_b \overrightarrow{k_{1, (2m+1)}}\} \cup \vec{0}$$

$$= \{\overrightarrow{f_{1, (2m+1)}} \oplus_b \overrightarrow{k_{1, (2m+1)}}\} \quad (20)$$

반면에 녹임 연산인 경우에는 불림 연산과 같은 방법으로 식 (21)로 전개할 수 있으며 '∩'은 AND gate로 구현이 되며 $\{\overrightarrow{f_{(2m+1)+1, N}} \ominus_b \overrightarrow{k_{(2m+1)+1, N}}\}$ 부분은 AND 게이트에 영향을 주지 않기 위하여 주변의 값들을 크기가 '1'인 $\vec{1}$로 초기화한다.

$$\overrightarrow{f_{1, N}} \ominus_b \overrightarrow{k_{1, N}} = \{\overrightarrow{f_{1, (2m+1)}} \ominus_b \overrightarrow{k_{1, (2m+1)}}\}$$

$$\cap \{\overrightarrow{f_{(2m+1)+1, N}} \ominus_b \overrightarrow{k_{(2m+1)+1, N}}\}$$

$$= \{\overrightarrow{f_{1, (2m+1)}} \ominus_b \overrightarrow{k_{1, (2m+1)}}\} \cap \vec{1}$$

$$= \{\overrightarrow{f_{1, (2m+1)}} \ominus_b \overrightarrow{k_{1, (2m+1)}}\} \quad (21)$$

그림 6은 구현된 단위 하드웨어의 형태소 크기보다 작을 경우 불림 연산을 위하여 주변을 '0'으로 초기화를 한 그림이다.

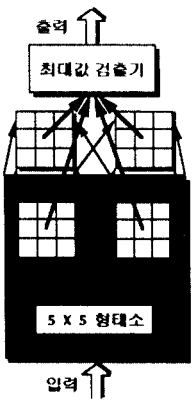

그림 5. 형태소가 큰 경우
Fig. 5 Large Structuring Element

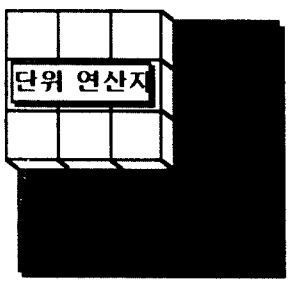

그림 6. 형태소가 작은 경우
Fig. 6 Small Structuring Element

V. 시뮬레이션 및 결과

본 논문에서는 하드웨어 시뮬레이션 언어인 VHDL를 이용하여 제안된 구조의 정확성을 증명하기 위하여 수리 형태학의 하드웨어를 모의 실험하였다.

실험은 Lenna 영상을 이용하였으며 3×3인 형태소 크기를 사용하여 불림 연산과 녹임 연산을 수행하였다. 그리고 형태소 영상의 값을 각각 다르게 하여 모의 실험을 하였다. 대단위 하드웨어는 5×5를 구현하기 위해서 3×3 단위 하드웨어 4개를 이용하여 구현을 하였다. 개선된 하드웨어를 사용하여 그림 7의 원영상으로부터 그림 8의 형태소의 값이 '7'일때 불림 연산의 결과를 얻을 수 있으며 그림 9는 형태소의 값이 '3'일때의 녹임 연산의 결과이다.

그레이 준위 영상의 수리 형태학의 단위 하드웨어는 크게 4부분으로 구성되어 진다. 각각의 블럭은 원영상의 입력 부분, 읽어들인 그레이 준위 값을 Umbra 형태로 변환시켜 이진 영상의 모습으로 전환시키는 블럭, 형태소와 변환된 이진 영상과의 연산 블럭, 마지막으로 출력되는 값을 다시 그레이 준위 영상 값으로 변환시켜주는 블럭이 있다. 입력 부분을 행렬 형태로 구성함으로써 전체 연산 시간을 크게 단축할 수 있다. 그리고 제안된 단위 하드웨어를 구성하는데 필요로 하는 하드웨어의 양은 형태소의 최대값을 제한시킨 하드웨어의 양과 같음을 식 (22)으로 부터 알 수 있으며 필요로 하는 게이트의 총 갯수는

$$S=\frac{M(M-1)}{2}\times(N^2+1)+(M-2) \qquad (22)$$

이다. 단, M은 그레이 준위 레벨, N은 형태소의 크기이다. 그러나 만약 형태소의 크기를 일정한 크기로 제한시킨다면 식 (22)에서 계산한 하드웨어의 양보다 적게 필요로 함을 알 수 있다. 물론 이때 설계된 하드웨어는 특정한 크기만을 필요로 할때만 유효하며 다양한 크기의 형태소를 필요로 하는 분야에서는 이용할 수 없다.

가산기를 이용한 하드웨어[8]는 입력 부분, 가산기 부분, 최대값을 찾는 부분등 크게 3부분으로 구성되어 있다. 입력 부분은 직렬 구조로 이루어져 있기 때문에 $n\times n$번 클락후 하나의 최종 연산 결과를 출력하며 가산기 부분과 감산기 부분의 설계는 각각 7개의 게이트를 통과한 후 결과값이 출력된다. 그러나 본 논문에서 제안된 하드웨어는 입력부분이 병렬로 구성되어 있으며 n번 클락후 출력이 생성이 되며 전체 9개의 게이트를 통과하면 최종 결과값이 출력되게 구성되었다. 따라서 기존의 하드웨어 구성보다 연산 시간이 빠름을 알 수 있다.

그림 7. Lenna 영상
Fig. 7 Lenna

그림 8. 불림 결과
Fig. 8 Result of Dilation

그림 9. 녹임 결과
Fig. 9 Result of Erosion

Ⅵ. 결 론

본 논문에서는 그레이 준위 영상 처리를 위한 형태학적 연산자의 하드웨어를 제안하였다. 제안된 단위 하드웨어는 연산 처리 시간을 빠르게 하기 위하여 이진 영상의 형태로 변환시켜 연산을 수행하는 방법을 채택하였으며 또한 임의의 형태소의 값을 사용할 수 있도록 설계를 하였다. 그리고 3×3과 같은 형태소의 크기를 이용하여 영상을 처리할때, 입력 부분을 행렬 형태로 구성하여 기존의 하드웨어보다 처리 속도를 약 60% 이상 단축할 수 있었다.

수리 형태학에서는 형태소의 크기와 값이 영상의 결과를 얻는데 중요한 역할을 하므로 한가지 형태소 크기가 아닌 다양한 크기의 형태소를 사용하여 연산을 수행하는 것이 필요하다. 따라서 본 논문에서는 제안된 단위 하드웨어를 병렬로 구성하여 다양한 크기의 형태소를 이용하는 분야에 능동적으로 사용할 수 있도록 구성하였다. 그리고 형태학적 연산자의 하드웨어는 형태소의 크기가 증가할 때 게이트의 수가 늘어나는 문제점이 발생할 수 있다. 이러한 게이트의 수가 증가하는 문제점은 가장 작은 크기의 형태소를 갖는 하드웨어를 병렬로 사용하여 해결 가능함을 보였다.

수리 형태학을 이용한 응용 분야가 증가함에 따라 실 시간 영상 처리를 위한 여러 가지 하드웨어의 개발이 진행되었으며 향후 다양한 연산 처리를 위하여 미디언 필터나 형태학적 연산자등을 포함하는 실 시간 연산 처리가 가능한 일반화된 하드웨어(Rank Order Filter)에 대한 연구가 필요하다.

참 고 문 헌

1. 이 임걸, 최 인아, "Gray scale 형태학적 변환법과 산업응용을 위한 특징정 검출", 대한전자공학회 하계 학술 발표 논문집, 제 9 권, 1호, 1993.
2. A. Toet, "A Morphological pyramid image decomposition," Pattern Recognition Letters, Vol.9, pp.255-261, 1989.
3. R. C. Gonzalez, R. E. Woods, Digital Image Processing, Addison Wesley, 1992.
4. J. Serra, Image Analysis and Mathematical Morphology, Academic Press, 1988.
5. A. K. Jain, Fundamentals of Digital Image Processing, Engleod Cliffs, NJ: Prentice Hall, 1990.
6. R. M. Haralick, S. R. Sternberg, X. Zhuang, "Image Analysis Using Mathematical Morphology," IEEE Trans. on PAMI, Vol.9, No.4, July, 1987.
7. F.Y.Shih, O.R.Mitchell, "Threshold Decomposition of Gray-Scale Morphology into Binary Morphology," IEEE Trans. on PAMI, Vol.11, No.1, pp.31-42, Jan. 1989.
8. 문 성룡 "그레이 스케일 형상학을 이용한 하이브리드 메디안 필터의 설계" 전북대학교 박사학위 논문, 1992.
9. D. L. Perry, VHDL, R. R. Donnelley and Sons Company.
10. Vantage Anaysis Systems, Inc., VantageSpreadsheet V3.1 User's Guide, 1992.
11. 한국 VHDL 사용자 모임, VHDL 이론, 1993.
12. S. Y. Kung, VLSI Array Processors, Prentice Hall, 1988.

박 동 선(Dong Sun Park) 정회원
한국 통신학회 논문지 제 21 권 제 18 호 참조
현재 : 전북대학교 컴퓨터·정보통신 공학부 조교수

신 진 욱(Jin Wook Shin) 정회원
1969년 10월 1일생
1993년 : 전북대학교 정보통신공학과(공학사)
1995년 : 전북대학교 정보통신공학과(공학 석사)
1997년 1월 : 삼보정보통신 기술연구소 연구원