영상처리의 윤곽선 검출을 위한 시스톨릭 배열

박 덕 원†

요 약

본 논문에서는 영상의 경계선 검출을 하는 시스톨릭어레이를 제안하였다. 영상을 실시간 처리하는 것은 국부적인 연산자의 많은 연산으로 인하여 많은 어려움이 따른다. 경계선 검출을 위한 국부적 연산자는 한 화소의 이웃해 있는 다른 화소를 이용하여 경계선을 검출하는 데에 이용되나 기존의 컴퓨터에서는 빈번한 입출력의 요구로 인하여 실시간 처리에서 요구하는 계산능력을 충분하게 제공하지 못한다. 그래서 이 논문에서는 처리 방법이 규칙적이고, 보통의 대역폭을 가지고 있으면서도 경계선 검출이나 라플라시안 처리에 적합한 시스톨릭어레이를 제안하였다.

Systolic Arrays for Edge Detection of Image Processing

Duck-Won Park†

ABSTRACT

This paper proposed a Systolic Arrays architecture for computing edge detection on images. It is very difficult to be processed images to real time because of operations of local operators. Local operators for computing edge detection are to be used in many image processing tasks, involve replacing each pixel in an image with a value computed within a local neighborhood of that pixel. Computing such operators at the video rate requires a computing power which is not provided by conventional computer. Through computationally expensive, it is highly regular. Thus, this paper presents a systolic arrays for tasks such as edge detection and laplacian, which are defined in terms of local operators.

1. 서 론

인간은 보는 것, 즉 화상이나 영상을 통해서 많은 정보를 비교적 단 시간에 지각할 수 있으며 시각, 청각, 촉각, 미각, 후각을 통해서 원하는 정보를 습득하고 있다. 그 중에서도 시각이 가장 중요한 감각기라고 할 수 있으며 인간은 화상을 통해서 원하는 모든 정보를 습득하고 행동한다고 할 수 있다. 이런 관계로 영상을 컴퓨터로 처리하려고 하는 시도는 오래 전부터 활발하게 진행되어 지금은 이러한 성과가 부분적으로 이루어져 문자인식, 원격탐사, 지문인식, 목표추적, 영상영화, 영상회의, 산업자동화 그리고 로봇 등의 많은 분야에서 인간의 생활을 편리하게 하는데 이용되고 있다. 그러나 현재까지는 영상처리과정의 몇 가지 어려움 때문에 아주 적은 부분에서만이 이용되고 있다. 영상처리는 그 어떤 것에 비해서 많은 정보량을 갖고 있으며, 국부적 연산자(Local Operators)를 처리해야 되는 문제점 때문에 더욱 더 실시간 처리에 어려움이 있다. 실시간 처리를 하려면 처리할 양이 적든지 또는 처리과정이 소프트웨어적이 아니라 하드웨어적인 방법으로 처리를 할 수 있을 때 가능하다고 하겠다. 그렇지만 영상처리는 이 분야에 대한 연구가 지속적으로 이루어

† 정 회 원 : 세명대학교 컴퓨터과학과 교수
논문접수 : 1998년 7월 6일, 심사완료 : 1999년 7월 28일

검출에도 불구하고 이 분야에 대한 특수한 하드웨어의 부족과 현재까지의 컴퓨터가 갖고 있는 처리속도의 문제점 때문에 실시간 처리의 문제는 극히 한정된 분야에서만 가능하다 하겠다. 최근 들어 가격이 저렴하면서도 계산속도가 빠른 소형 반도체 소자들을 생산할 수 있는 VLSI 기술이 급격하게 발전하고 있으며 이런 반도체 기술에 힘입어 영상처리에서도 실시간 처리를 위한 새로운 아키텍처 논문이 많이 발표되고 있으며 계속적으로 많은 연구가 진행되는 분야라 할 수 있다[1].

이 논문에서는 영상처리의 실시간 처리에 많은 제한이 되어 왔던 국부적 연산자의 처리를 하드웨어로 처리가 가능하도록 제안하였다. 국부적 연산자에 의한 계산의 요구는 영상처리의 전처리에 해당되는 영상의 변환의 일종인 윤곽선 검출과 평활화 조작에서 많이 이루어진다. 그래서 이 논문에서는 영상처리에서 필수적인 처리 분야라고 할 수 있는 윤곽선 검출이 많은 계산으로 인해 실시간 처리가 어려우므로 윤곽선 연산자를 이용하여 동시에 두가지의 윤곽선 검출 및 라플라시안(Laplacian)변환이 가능한 시스톨릭어레이(Systolic Arrays)를 제안했다. 시스톨릭어레이는 H.T.Kung에 의해서 제안된 아키텍처로서 데이터나 컨트롤의 흐름이 인접해 있는 Cell들간에 이루어지므로 모듈러하고, 규칙적으로 작업이 수행될 수 있다. 이런 작업은 반복적으로 Cell을 거치면서 이루어지므로 Cell은 간단하고 이해하기 쉽다. 그리고 시스톨릭어레이는 고도로 파이프라인 된 동기화 다중처리가 가능하며 VLSI칩으로 구현하기 쉽고 시스템의 동기가 유지되는 한 무한한 확장이 가능하다[2-6].

본 논문에서 제안한 이 시스톨릭어레이는 웨이트(Weight)값에 따라서 방법이 단순하고 결과가 좋은 Sobel 경계선 검출, Prewitt 그리고 라플라시안의 결과를 구할 수 있도록 설계하였다. 그러므로 소프트웨어적인 방법에 의해서는 실시간 처리가 어려웠던 것을 여기서는 보통의 메모리 대역폭을 갖고 있으면서도 실시간 처리가 가능하도록 하였다. 또한 대부분의 특수목적으로 설계된 하드웨어가 오직 하나의 결과만을 구할 수 있는데 비해서 유연성이 있도록 설계를 하였다.

본 논문의 구성은 2장에서는 윤곽선 검출에 대한 기본 이론으로서 주로 Sobel, Prewitt, 라플라시안 등에 대해서 기술했으며, 3장에서는 국부적 연산자의 처리가 콘벌루션(Convolution)의 형태이므로 주로 콘벌루션을 위한 시스톨릭어레이에 대해서 자세하게 기술했으며, 4장에서는 이 논문에서 제안한 윤곽선 검출을 위한 새로운 아키텍처에 대해서 각각의 Cell의 구조와 시스톨릭어레이로 적용한 것을 보여준다. 5장에서는 여기서 제안한 새로운 아키텍처를 평가하는 부분으로 메모리로부터 입력 영상이 불려지는 횟수 및 하나의 국부적인 연산을 위해서 필요로 하는 연산횟수를 기존의 소프트웨어적인 방법과 여기서 제안한 방법과 비교 평가 하여 보았다. 마지막으로 결론으로 이 논문에서 제안한 아키텍처의 특징과 앞으로의 과제에 대해서 기술하였다.

2. 윤곽선 검출

영상처리 시스템에서는 영상의 특징을 추출해내는 것이 영상을 처리하고, 분석하고, 인식하는데 매우 중요한 요소가 된다. 이러한 영상의 특징 중에서 윤곽선은 화소(Pixel)의 빛의 밝기나 색의 갑작스런 변화를 나타내는 특징이 있으며 물체의 테두리나 물체와 물체간의 경계를 나타낸다.

추출된 윤곽선은 산업용 로봇에서와 같은 시각장치 등에서 물체의 크기, 형태, 상대적인 위치를 인식하는데 기본적인 요소가 되며, 영상을 전송하고자 할 때 많은 정보량으로 인해 생기는 전송에서의 지연 및 병목현상을 윤곽선을 이용함으로서 해결할 수도 있으며 영상의 화소함수 $f(x,y)$를 미분하는 것에 따라 화소가 급격하게 변화하는 부분을 추출할 수 있다.

미분 영상 자체는 여러 가지 값을 갖는 영상이므로 윤곽선의 위치를 구하기 위해서는 이 미분 영상을 임계값으로 처리해서 0, 1의 2진 영상으로 변환시켜야 한다. 이 임계값을 어떻게 결정하는가는 아주 중요하며 이 값이 아주 작은 값이라면 윤곽선의 폭이 넓어지고 아주 큰 값이라면 윤곽선이 부분적으로 절단된 형태가 된다.

윤곽선 검출이 되는 1차 미분은 식 (1)과 같으며 이것을 이산적으로 근사시켜 X방향의 미분치와 Y방향의 미분치로 나누어서 표현한 것이 식 (2)와 식 (3)과 같다.

$$Gf(x,y)=\frac{df(x,y)}{dx}+\frac{df(x,y)}{dy} \tag{1}$$

$$G_x=f(x+1,y)-f(x,y) \tag{2}$$

$$G_y=f(x,y+1)-f(x,y) \tag{3}$$

윤곽선의 추출은 식 (4)와 같이 표현할 수 있다. 그러나 이 식은 계산 양이 많고 하드웨어로 구현하는데 어려움이 있어서 식 (5)와 같이 절대치로 근사시켜서 사용할 수 있다. 그러나 1차 미분은 벡터 양이므로 이것은 스칼라 양에 비해서 기억해야 하는 데이터 양이 증가하고 방향을 고려하지 않으면 안 된다. 또한 같은 기울기의 넓은 영역에서는 그 전 영역이 윤곽선으로 추출되는 경우가 있다.

$$G=\sqrt{G_x^2+G_y^2} \tag{4}$$

$$G=|G_x|+|G_y| \tag{5}$$

윤곽선 검출을 위한 2차 미분은 스칼라 양이며 식 (6)과 같다. 이것은 4근방 라플라스 연산자 형태이고 이산적으로 근사시킨 것이 식 (7)과 같다.

$$\nabla^2 f(x,y)=\frac{d^2f(x,y)}{dx^2}+\frac{d^2f(x,y)}{dy^2} \tag{6}$$

$$\begin{aligned}\nabla^2 = {} & f(x+1,y)+f(x-1,y)+f(x,y+1)\\ & +f(x,y-1)-4f(x,y)\end{aligned} \tag{7}$$

2차 미분은 1차 미분보다 잡음에 대해 민감하고 잡음성분을 강조하는 것이므로 이것을 적용할 때는 주의를 해야만 하며 보다 정확한 윤곽선 검출을 위해서는 평균화 처리를 한 후에 미분이나 라플라시안 처리를 해야 한다. 이것을 또한 Mask 형태로 표시하면 (그림 1)과 같다. (그림 1)의 a)는 위의 식에서 보여 주는 4근방 라플라스 연산자 ∇^2을 3×3 Mask로 표현한 것이고 b)는 8근방 라플라스 연산자 ∇^2을 3×3 Mask로 표현한 것이다. 이외에도 실제로 많이 이용되는 윤곽선 검출 연산자로서는 (그림 1)의 c), d), e)의 Robert 연산자, Sobel 연산자, Prewitt 연산자가 있다.

0	1	0
1	-4	1
0	1	0

a) 4근방 라플라스 연산자

1	1	1
1	-8	1
1	1	1

b) 8근방 라플라스 연산자

0	1
-1	0

1	0
0	-1

c) Roberts 연산자

-1	0	1
-1	0	1
-1	0	1

-1	-1	-1
0	0	0
1	1	1

d) Prewitt 연산자

-1	0	1
-2	0	2
-1	0	1

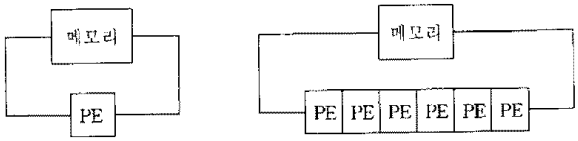
-1	-2	-1
0	0	0
1	2	1

e) Sobel 연산자

(그림 1) 각종의 윤곽선 연산자

3. 콘벌루션을 위한 Systolic Arrays

시스톨릭어레이란 각 Processing Element(PE)들이 바로 이웃한 PE와 연결되어 서로 데이터를 주고받으면서 계산을 실행하고 그 결과를 방출하는 PE들의 네트워크로서 기존의 컴퓨터로서 실시간 처리를 하려면 방대한 자료로 인해 처리하는 과정에서 입출력 병목 현상이 생겨 처리 속도가 떨어지는 단점을 보완하기 위하여 한번 메모리에서 가져온 데이터는 각각 PE들을 거치면서 최대한도로 사용된 후 다시 메모리로 보내므로 입출력 병목 현상을 해결하고 보통의 메모리 대역폭(Bandwidth)을 가지고 높은 계산율을 가져올 수 있으며 이 방법은 입출력 횟수 보다 더 많은 계산을 요구하는 문제에 적합하다[3,7,8]. (그림 2)는 이것을 고전적인 방법과 비교해 본 것이다.



a) 고전적인 프로세서 b) 시스톨릭어레이 프로세서

(그림 2) 고전적인 프로세서와 시스톨릭어레이 프로세서의 비교

3.1 콘벌루션(Convolution)

영상에서 사용되는 것이 1차원 콘벌루션 보다는 2차원 콘벌루션이 더 많이 이용되고 있으며 주로 입출력의 속도가 시스톨릭어레이의 Cell의 속도보다 빠른 경우에 주로 사용하며 입력 영상을 X라 하고 커널(Kernel)을 W라 하면 식 (8)과 같다.

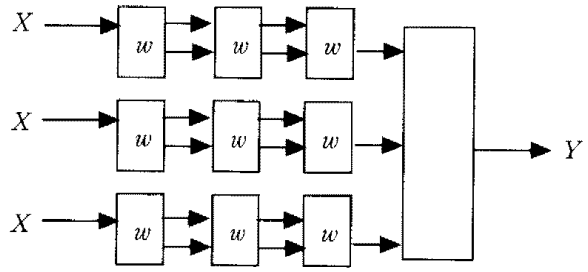
$$Y_{rs}=\sum_{i=0}^{k-1}\sum_{j=0}^{p-1} W_{i+1,j+1}\bullet X_{i+r,j+s} \tag{8}$$

$$(X=X_{ij}, i=0,1,2,3,\ldots,m, j=0,1,2,3,\ldots,n)$$

$$(W=W_{ij}, i=0,1,2,\ldots,k, j=0,1,2,3,\ldots,p)$$

여기서 입력 영상 X는 $m\times n$ 이고 커널 W의 크기는 $k\times p$일 때, $r=m-k+1$, $s=n-p+1$이다. 2차원 콘벌루션은 (그림 3)과 같이 1차원 콘벌루션 배열 k개를 동시에 사용함으로서 구현할 수 있으나 k값이 커지면 높은 입출력 대역폭을 요구하므로 실제 처리에 있어서는 1차원 콘벌루션으로 변환하여 사용하는 경우가 많다. 특히 메인 시스템과 시스톨릭어레이 사이의 주고

반는 대역폭이 병목현상인 경우에는 1차원이 2차원 보다 안전하게 글로벌 클락(Clock)에 의해서 동기화가 가능하다. 그래서 여기서는 1차원으로 변환한 두 가지를 소개하며 이들 중 영상처리에 가장 적합한 k개의 1차원 콘벌루션을 이용한 2차원 콘벌루션을 자세하게 소개한다[7-9].



(그림 3) 1차원 배열을 k개 누적하여 얻은 2차원 시스톨릭 배열

3.2 2차원 콘볼루션을 완전한 1차원으로 변환

일반적으로 n차원 콘벌루션을 계산하는 문제는 1차원 콘벌루션으로 변환하여 계산한다. 그러므로 2차원 콘벌루션도 1차원 콘벌루션으로 변환하여 계산을 하는 것이 편리하다. 2차원 콘벌루션을 1차원 콘벌루션으로 변환하는 것은 우선 2차원 영상과 2차원 커널을 1차원으로 변환해야 한다[9-11]. 우선 2차원 영상을 1차원으로 변환을 하면 식 (9)와 같이 표현될 수 있다.

$$X=[X_1*, X_2*, X_3*, \ldots\ldots\ldots\ldots, X_m*] \quad (9)$$
$$(X_i*=X_{i1}, X_{i2}, X_{i3}, \ldots\ldots, X_{in})$$
$$(i=1,2,3,\ldots\ldots, m)$$

또한 2차원 커널을 1차원으로 변환하는 것은 식 (10)과 같이 표현될 수 있는데 여기서는 입력 영상의 열과 커널의 열의 차이 만큼 0이 삽입된다.

$$W=[W_1*, (n-p)!0, W_2*, \ldots\ldots\ldots, (n-p)!0, W_k*] \quad (10)$$
$$(W_i*=W_{i1}, W_{i2}, W_{i3}, \ldots\ldots, W_{ip})$$
$$(i=1,2,3,4,\ldots\ldots, k)$$
$$(U!v=v\text{를 }U\text{번반복})$$

예로서 입력 영상이 5×4이고 커널의 크기가 3×2인 것을 1차원 형식의 입력 영상과 커널로 변환하면 식 (11)과 (12)처럼 바꿀 수 있다.

$$X=(X_{11}, X_{12}, X_{13}, X_{14}, X_{21}, \ldots\ldots, X_{53}, X_{54}) \quad (11)$$

$$W=(W_{11}, W_{12}, 0, 0, W_{21}, W_{22}, 0, 0, W_{31}, W_{32}) \quad (12)$$

3.3 2차원 콘벌루션을 k개의 1차원으로 변환

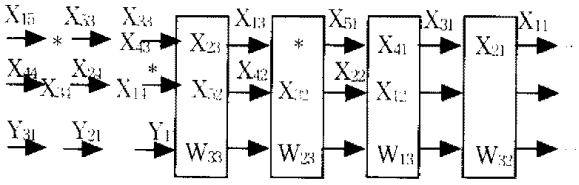
2차원 콘벌루션에서의 입출력 병목 현상을 해결하기 위하여 1차원 배열을 k개 사용한 것으로 1차원 배열을 k개 사용한 것이다. 예로서 입력 영상이 5×5, 커널이 3×3인 것을 식 (8)을 이용하여 결과 Y를 구하기 위해서는 다음과 같이 입력영상을 1차원 형식으로 변환해야 한다.

Y_{11}, Y_{12}, Y_{13}를 위한 입력 스트림→ $X_{11}, X_{21}, X_{31}, X_{21}, X_{22}, \ldots\ldots$

Y_{21}, Y_{22}, Y_{23}를 위한 입력 스트림→ $X_{21}, X_{31}, X_{41}, X_{22}, X_{32}, \ldots\ldots$

Y_{31}, Y_{32}, Y_{33}를 위한 입력 스트림→ $X_{31}, X_{41}, X_{51}, X_{32}, X_{42}, \ldots\ldots$

그리고 이것으로부터 결과 Y를 구하기 위한 시스톨릭어레이는 (그림 4)와 같다. 여기서 각 Cell에서 입력영상 X를 받아들이는 것은 동일한 스트림(Stream)에서 연속적으로 3개씩 교대로 받아들이면 된다.



(그림 4) 2차원 콘벌루션을 위한 1차원 시스톨릭어레이

4. 윤곽선 검출을 위한 Systolic Arrays

윤곽선 검출은 주어진 영상에서 영역들 간의 경계에 해당하는 윤곽선을 찾아내는 방법 중에서 국부연산자(Local Operator)를 사용하는 방법은 성능이 떨어지고 구한 윤곽선 값에 대해 임계값을 정해주어야 하는 문제점이 있으나 과정이 간단하고 실시간 처리가 가능하여 많이 사용되고 있다. 국부연산자의 대표적인 것으로는 Sobel, Prewitt, Roberts 그리고 라플라시안(Laplacian) 연산자 등이 있다.

이들을 이용한 윤곽선 검출은 X방향의 미분치를 나타내는 S_x와 Y방향의 미분치를 나타내는 S_y가 있으며 이들 S_x와 S_y값을 각각의 절대치를 구해 더해 주

면 된다. (그림 5)는 윤곽선 검출을 위한 연산자를 시스톨릭어레이에 적용시키기 위해서 일반적인 형태로 변형한 것이다. 여기서 α값이 2이면 Sobel 연산자이고 1이면 Prewitt 연산자이다.

a	b	c
d	e	f
g	h	i

a) 입력영상

-1	0	1
$-\alpha$	0	α
-1	0	1

b) X 방향의 연산자

-1	α	1
0	0	0
1	α	1

c) Y 방향의 연산자

(그림 5) 입력 영상과 윤곽선 검출을 위한 연산자

X방향의 연산자와 Y방향의 연산자를 이용하여 X방향의 미분치 G_x와 Y방향의 미분치 G_y를 구하면 식 (9)와 식 (10)과 같다. 그리고 4근방, 8근방 라플라시안 연산자는 식 (11)과 식 (12)와 같으며, 식 (13)은 식 (9)과 식 (10)의 미분치를 이용하여 (그림 5)의 입력 영상에서 화소 e에 대한 변화도를 구한 것이다.

$$G_x=(g+\alpha h+i)-(a+\alpha b+c) \tag{9}$$

$$G_y=(c+\alpha f+i)-(a+\alpha d+g) \tag{10}$$

$$L_{xy}=b+d+f+h-4e \tag{11}$$

$$L_{xy}=a+b+c+d+f+g+h+i-8e \tag{12}$$

$$G_e=\sqrt{G_x^2+G_y^2} \tag{13}$$

그러나 식 (13)은 계산이 많고 복잡하므로 식 (14)와 같이 절대값으로 계산을 할 수 있다. 이렇게 식을 간소화하면 하드웨어로 설계하는데 편리하다.

$$G_e=|G_x|+|G_y| \tag{14}$$

4.1 시스톨릭어레이의 Cell 구조

시스톨릭어레이에 적용을 하려면 먼저 적용 가능한 형태로의 수식 변환이 필요하다. 그래서 우선 윤곽선 검출 연산자의 일반적인 수식을 구하기 위하여 (그림 6)과 같이 윤곽선 검출 연산자를 3×3, 입력 영상을 5×5라하고 이것으로부터 각 방향에서의 변화도를 구하는 식을 유도한다.

W_{11}	W_{12}	W_{13}
W_{21}	W_{22}	W_{23}
W_{31}	W_{32}	W_{33}

a) 윤곽선 연산자

X_{11}	X_{12}	X_{13}	X_{14}	X_{15}
X_{21}	X_{22}	X_{23}	X_{24}	X_{25}
X_{31}	X_{32}	X_{33}	X_{34}	X_{35}
X_{41}	X_{42}	X_{43}	X_{44}	X_{45}
X_{51}	X_{52}	X_{53}	X_{54}	X_{55}

b) 입력 영상

(그림 6) 윤곽선 검출을 위한 윤곽선 연산자와 입력 영상

예로서 화소 X_{22}에 해당되는 변화도를 구하려면 X방향의 변화도와 Y방향의 변화도를 구하여야 되는데, 여기서 X방향의 변화도를 G_{x22}, Y방향의 변화도를 G_{y22}라 하면 식 (15)와 식 (16)처럼 구할 수 있다. 이 식을 이용하여 식 (17)과 같이 X_{22}의 최종 변화도 G_{22}를 구하게 된다.

$$G_{x22}=(W_{13}X_{13}+W_{23}X_{23}+W_{33}X_{33}) +(W_{11}X_{11}+W_{21}X_{21}+W_{31}X_{31}) \tag{15}$$

$$G_{y22}=(W_{13}X_{13}+W_{23}X_{23}+W_{33}X_{33}) +(W_{11}X_{11}+W_{21}X_{21}+W_{31}X_{31}) \tag{16}$$

$$G_{22}=|G_{x22}|+|G_{y22}| \tag{17}$$

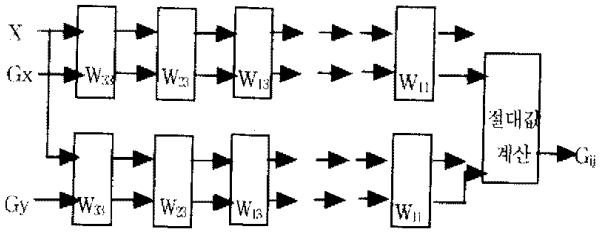
이렇게 구하는 과정을 시스톨릭어레이로 적용을 하려고 콘벌루션 형태로 변환하면 식 (18)과 같이 구할 수 있다.

$$G_{rs}=\sum_{i=0}^{k-1}\sum_{j=0}^{p-1} W_{i+1,j+1}\bullet X_{i+r-1,j+s+1} \tag{18}$$

식 (18)에서 입력 영상을 $X=X_{ij}, i=1,2,3,\ldots\ldots, m, j=1,2,3,\ldots\ldots,n$라 하고 Mask를 $W=W_{ij}, i=j=1,2,3,(r=2,3,4,\ldots\ldots,m-k+2, s=2,3,4,\ldots\ldots,n-p+2)$라 하면 2차원 콘벌루션과 거의 같은 수식이 되며 이것을 2개의 9개 Cell로 1차원 시스톨릭어레이로 적용을 시킬 수 있다. 이것을 (그림 7)에 나타내었다.

(그림 7)에서 각 Cell에서 레지스터 W_{ij}는 여러 가지 윤곽선 검출을 하기 위해서 <표 1>과 같이 여러 가지의 값을 갖는다. 이 레지스터는 4개의 숫자를 저장하는 것으로 첫 번째 값은 Sobel 연산자를 위한 것이고, 두 번째 값은 Prewitt 연산자를 구하기 위한 것이며,

네 번째 숫자는 4근방의 라플라스 변환 값을 위한 것이다. 그리고 마지막 값은 8근방 라플라스 변환을 위한 값이다.

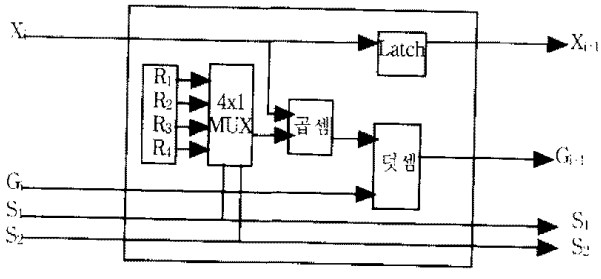


(그림 7) 윤곽선 검출을 위한 시스톨릭어레이의 개략도

〈표 1〉 각 Cell에서의 레지스터 값

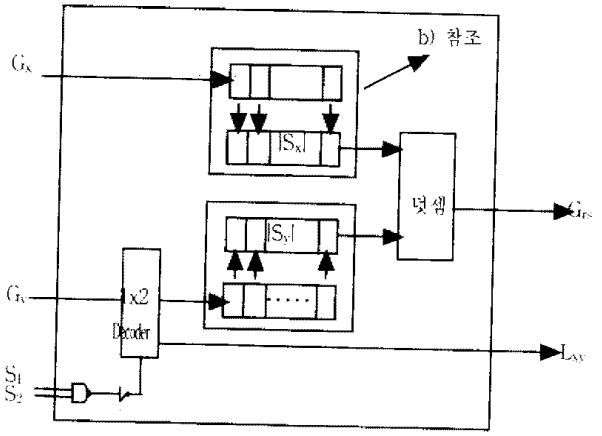
X,Y방향 / 레지스터	X 방향을 위한 각 Cell에서의 레지스터 값				Y 방향을 위한 각 Cell에서의 레지스터 값			
W_{11}	-1	-1	0	1	1	1	0	1
W_{12}	0	0	1	1	-2	1	1	1
W_{13}	1	1	0	1	-1	-1	0	1
W_{21}	2	-1	1	1	0	0	1	1
W_{22}	0	0	-4	8	0	0	-4	-8
W_{23}	2	1	1	1	0	0	1	1
W_{31}	1	1	0	1	1	1	0	1
W_{32}	0	0	1	1	2	1	1	1
W_{33}	1	1	0	1	1	1	0	1

Cell은 전체 19개로서 18개의 Cell은 X방향과 Y방향의 변화 값을 구하기 위한 것이고 마지막 하나의 Cell은 앞에서 구한 X, Y의 변화도 값을 입력으로 하여 절대값을 구하는 것이다. 그리고 이들 Cell의 구조는 일반적인 Cell(X, Y의 변화도 값을 구하기 위한 Cell)과 마지막 Cell로 구분하여 볼 수 있으며 Cell의 구조는 일반적인 Cell은 (그림 8)과 같이 입력영상 X에 윤곽선 검출 연산자의 값을 곱해 이들 값을 누적하고, 마지막 Cell은 절대값을 계산하는 Cell로서 (그림 9)와 같이 두 가지 출력을 갖는데 하나의 출력 L은 라플라스 변환한 값이고 다른 하나의 출력 G_{ij}는 윤곽선 검출을 위한 것으로 두 개의 입력의 절대값을 구해 이것을 더함으로서 구해진다.

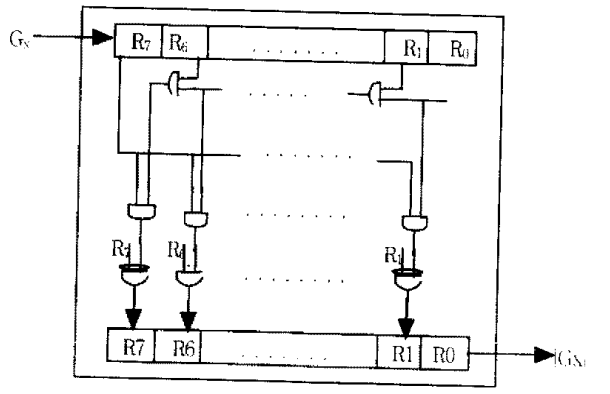



```
If (S1S2=00)
 then  /Sobel 연산자를 이용한 윤곽선 검출/
                Gi-1 = Gi + R1Xi
                Xi-1 = Xi
                S1 = S1,   S2 = S2
 else If (S1S2=01)
      then  /Prewitt 연산자를 이용한 윤곽선 검출/
                Gi-1 = Gi + R2Xi
                Xi-1 = Xi
                S1 = S1,   S2 = S2
      else If (S1S2=10)
                then /4근방 Laplacian 연산자를 이용한 미분치계산/
                     Gi-1 = Gi + R3Xi
                     Xi-1 = Xi
                     S1 = S1,   S2 = S2
                else /8근방 Laplacian 연산자를 이용한 미분치계산/
                     Gi-1 = Gi + R4Xi
                     Xi-1 = Xi
                     S1 = S1,   S2 = S2
                else
 else
```

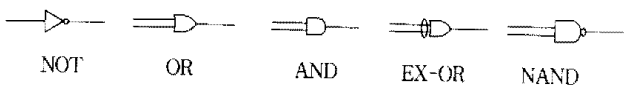
(그림 8) 일반적인 Cell의 구조와 연산 알고리즘



a) 마지막 Cell의 구조



b) a)의 Cell에서 강조한 부분의 구조



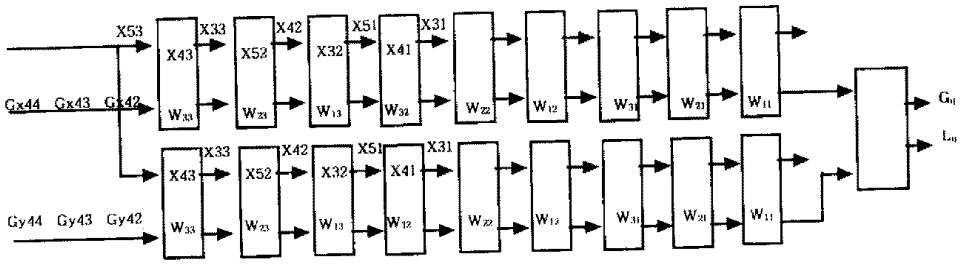
(그림 9) 마지막 Cell의 구조

4.2 시스톨릭어레이를 적용

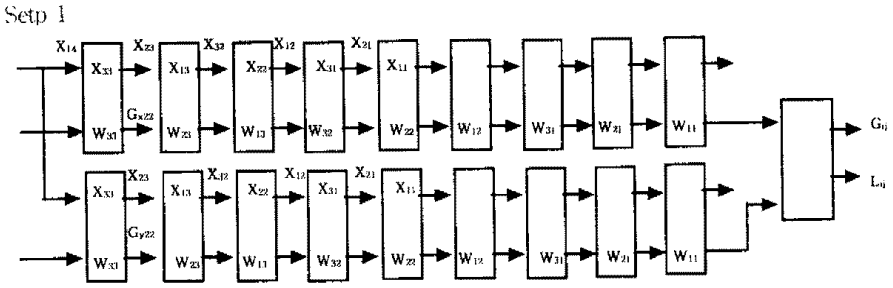
위의 이러한 구조에서 입력영상 X는 출력 스트림

G의 1/2의 속도로 진행하므로 각각의 Cell에서는 입력 스트림을 저장하기 위한 부가적인 레지스터를 필요로 하며 전체 결과 중 하나의 결과 G_{ij}를 구하기 위해서 Cell의 1/3만이 사용된다.

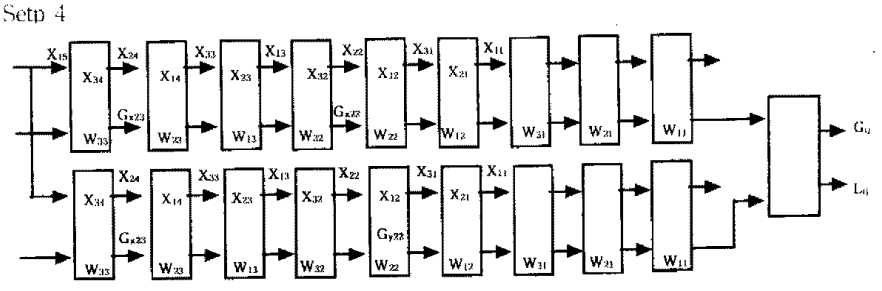
위의 이러한 구조에서 입력영상 X는 출력 스트림 G의 1/2의 속도로 진행하므로 각각의 Cell에서는 입력 스트림을 저장하기 위한 부가적인 레지스터를 필요로 하며 전체 결과 중 하나의 결과 G_{ij}를 구하기 위해서 Cell의 1/3만이 사용된다. G_{22}를 구하기 위한 시스톨릭어레이의 예가 (그림 10)과 같으며 실제로 이 값을 구하는 과정은 (그림 10)에 Snapshot으로 나타내었다. (그림 11)은 3 단계마다 하나의 결과를 구하고 각각의 Snapshot의 결과는 일반적인 윤곽선 검출을 위한 G_{ij}와 실제 4가지의 윤곽선 검출 중에서 Sobel 연산자를 이용한 윤곽선 검출 결과 S_{ij}를 보여 준다. 이때 S_1S_2=00이다.



(그림 10) 윤곽선 검출을 위한 시스톨릭어레이

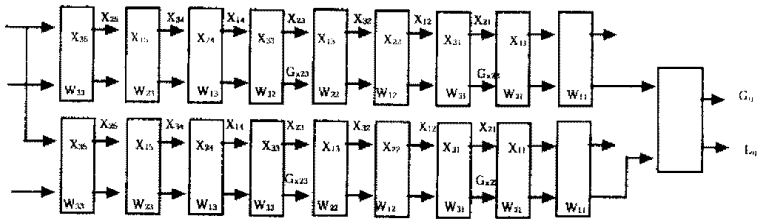


$$G_{x22} = W_{33}X_{33}$$
$$G_{y22} = W_{33}X_{33}$$
$$S_{x22} = X_{33}$$
$$S_{y22} = -X_{33}$$



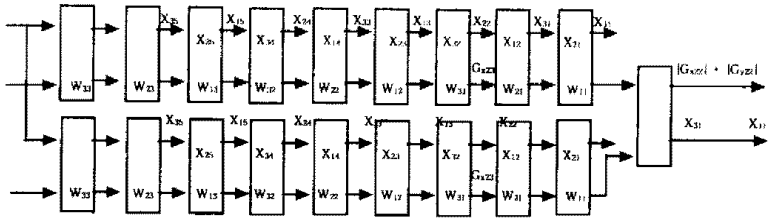
$$G_{x22} = W_{33}X_{33} + W_{23}X_{23} + W_{13}X_{13} + W_{32}X_{32}$$
$$G_{y22} = W_{33}X_{33} + W_{23}X_{23} + W_{13}X_{13} + W_{32}X_{32}$$
$$S_{x22} = X_{33} + 2X_{23} + X_{13}$$
$$S_{y22} = -X_{33} + X_{13} - 2X_{32}$$

Setp 7



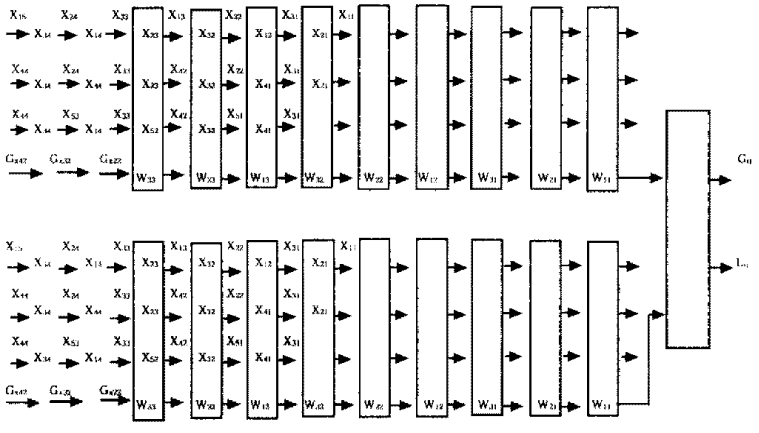
$$G_{x22} = W_{31}X_{33} + W_{23}X_{23} + W_{13}X_{13} + W_{32}X_{32} + W_{22}X_{22} + W_{12}X_{12} + W_{31}X_{31}$$
$$G_{y22} = W_{31}X_{33} + W_{21}X_{23} + W_{13}X_{13} + W_{32}X_{32} + W_{22}X_{22} + W_{12}X_{12} + W_{31}X_{31}$$
$$S_{x22} = X_{33} + 2X_{23} + X_{13} - X_{31}$$
$$S_{y22} = -X_{33} + X_{13} - 2X_{32} + 2X_{12} - X_{31}$$

Setp 10

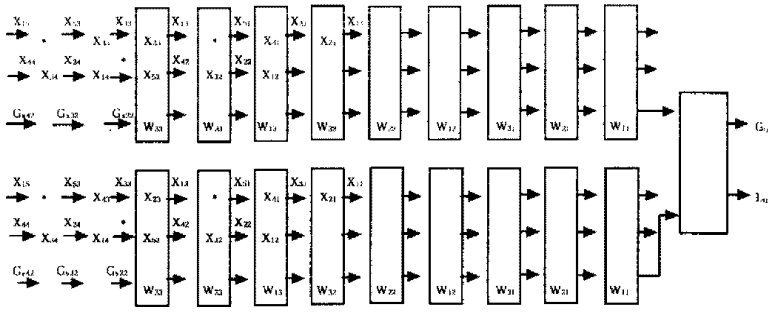


$$G_{22} = |G_{x22}| + |G_{y22}|$$
$$S_{22} = |S_{x22}| + |S_{y22}|$$

(그림 11) G22를 구하는 시스톨릭어레이의 Snapshot



(그림 12) 3개의 시스톨릭어레이를 하나로 합친 시스톨릭어레이



(그림 13) 윤곽선 검출을 위한 시스톨릭어레이

5. 평 가

콘벌루션의 계산은 동일한 입력을 여러 번 반복해서 사용함으로서 실제 CPU에서 처리되는 시간에 비해서 입출력에 걸리는 시간이 훨씬 더 크게 되어 전체 처리 시간을 떨어뜨린다. 만약에 이러한 처리를 소프트웨어적인 방법에 의해서 처리를 한다면 실시간 처리를 하는데 많은 어려움이 있다. 그래서 본 논문에서는 보통의 메모리 대역폭을 갖으면서도 높은 처리 효율을 갖는 시스톨릭어레이를 이용하여 설계를 하였다.

이 새로운 시스톨릭어레이의 평가의 두 가지로 나누어서 하였다. 첫 번째는 입출력에 많은 시간이 걸리므로 이것을 기존의 소프트웨어적인 방법과 이 논문에서 제안한 새로운 구조에 대해서 비교 평가한 것이다. 여기서 처리하려고 하는 입력 영상의 크기는 $n\times n$으로 하였고 이것을 처리하기 위해서 메모리로부터 CPU로 가져오는 횟수를 비교하였다. 두 번째는 소프트웨어적인 방법과 이 논문에서 제안한 새로운 구조에 대해서 하나의 결과 G_{ij}(한 화소의 경계선 검출 결과)를 구하는데 필요한 연산의 횟수를 비교 평가하였다.

5.1 메모리로부터 가져오는 횟수

(1) 기존의 소프트웨어적인 방법

기존의 소프트웨어에 의한 방법은 하나의 결과 G_{ij}를 구하기 위해서 중요부분의 프로그램이 다음과 같이 식 (21)처럼 구하게 된다. 여기서는 Sobel 윤곽선 연산자의 실제 값을 넣어서 계산한 것이다.

$$G_{x22}=(X_{33}+2X_{23}+X_{13})-(X_{31}+2X_{21}+X_{11}) \tag{19}$$

$$G_{y22}=(X_{33}+2X_{32}+X_{31})-(X_{13}+2X_{12}+X_{11}) \tag{20}$$

$$G_{22}=|G_{x22}|+|G_{y22}| \tag{21}$$

위의 프로그램에서 메모리로부터 가져오는 횟수는 X방향의 G_x와 Y방향의 G_y를 구하기 위해서 가져오는 횟수와 절대값을 구하기 위해서 가져오는 횟수를 더한 것과 같다. 여기서 $n\times n$ 입력영상으로부터 결과 G를 구하기 위해서 메모리를 엑세스하는 횟수를 $CNT1$이라 하면 식 (22)와 같이 계산된다. 이 식에서 2를 곱한 것은 X방향의 G_x와 Y방향의 G_y를 구하는 것이 같기 때문이다. 또한 절대값을 계산하기 위해서 메모리로부터 엑세스하는 횟수를 $CNT2$라 하면 식 (23)과 같이 계산된다. 그러므로 전체 메모리로부터 가져오는 횟수는 식 (24)와 같이 $CNT1$ 과 $CNT2$의 합으로 표현할 수 있다.

$$CNT1=[4+(8\times 1)+((n-1)\times 4\times 2)+((n-4)\times 4\times 3)+((n-4)(n-4)\times 6)]\times 2=12n^2-56n+72 \tag{22}$$

$$CNT2=(n-2)(n-2)\times 2=2n^2-8n+8 \tag{23}$$

$$CNT3=CNT1+CNT2=14n^2-64n+80 \tag{24}$$

입력장치의 발전으로 스캐너, 카메라 등의 선명도가 높아져서 입력영상의 크기는 갈수록 커지는 추세이며 실제로 n값은 아주 큰 값을 갖는다. 그래서 $n^2\gg n$이 되므로 식 (24)는 다음의 식으로 간소화시킬 수 있다.

$$CNT\fallingdotseq 14n^2 \tag{25}$$

(2) 이 논문에서 제안한 방법

이 논문에서 제안한 방법은 시스톨릭어레이의 특성에 따라서 입력 영상이나 윤곽선 연산자의 크기에 무관하게 한번씩만 메모리로부터 가져오면 되므로 전체 가져오는 횟수는 식 (26)와 같이 n^2이 된다.

$$CNT_s=n^2 \tag{26}$$

5.2 출력을 구하기 위해서 필요한 연산의 횟수

(1) 기존의 소프트웨어적인 방법

Sobel 윤곽선 검출에 대한 예로서 하나의 결과 G_{22}를 구하기 위한 연산의 횟수는 G_{x22}, G_{y22}를 구하는데 필요한 연산 횟수를 구해서 더하면 된다. G_{x22}를 구하기 위한 연산은 Sobel 윤곽선 연산자가 절대값이 0, 1 그리고 2로 되어있는데 곱셈은 2일 때만 하면 되므로 2번 필요하고 덧셈은 1, 2일 때만 하므로 5번 필요하다. 이것은 G_{y22}에 대해서도 같으므로 식 (27)과 같이 구할 수 있다. 위의 이 연산이외에도 G_{x22}와 G_{y22}의 절대값을 구하여 더하는 연산이 필요하다. 이것은 식 (28)과 같다. 마지막으로 전체 횟수는 식 (27)과 식 (28)을 더한 것과 같다.

$$OP1 = (2p+5q)\times 2 \tag{27}$$

$$OP2 = q+2r \tag{28}$$

$$OP3 = 4p+11q+2r \tag{29}$$

여기서 p는 곱셈에 걸리는 시간, q는 덧셈에 걸리는 시간 그리고 r은 절대값 계산에 걸리는 시간이다. 식 (29)에서의 $OP3$는 하나의 화소를 구하는데 걸리는 시간이므로 전체 화소를 구하는 시간은 $(n-2)(n-2)$개의 화소에 대해서 구한 것은 식 (30)과 같다.

$$OP4 = (4p+11q+2r)(n-2)(n-2) \tag{30}$$

위의 식 (30)에서 실제 $p \gg q$, $p \gg r$로서 곱셈하는 시간이 덧셈하는 시간이나 절대값을 계산하는 시간이 훨씬 더 걸리고 입력영상의 크기가 윤곽선 검출 연산자에 비해서 월등하게 크므로 $(n-2)$는 대략 n이라 하면 식 (31)와 같이 간소화시킬 수 있다.

$$OP \fallingdotseq 4pn^2 \tag{31}$$

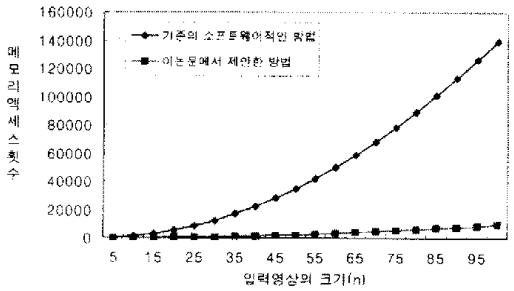
(2) 이 논문에서 제안한 방법

이 논문에서 제안한 방법은 매 사이클마다 하나의 결과를 구할 수 있으므로 실제 하나의 결과를 구하는데 걸리는 시간은 하나의 Cell에서 걸리는 시간이 된다. 즉 $p+q$가 된다. 그러므로 대략 n^2개의 결과를 구하는데 걸리는 시간은 식 (32)과 같이 약 pn^2이라 할 수 있다. 단 그 이외의 추가되는 시간은 고려하지 않았음.

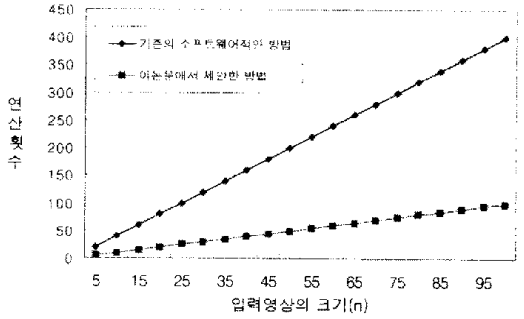
$$OP_c \fallingdotseq pn^2 \tag{32}$$

5.3 두 가지 방법의 비교 평가

첫 번째의 메모리에서 가져오는 횟수를 구해 본 것은 기존의 소프트웨어적인 방법이 식 (24)로서 이 논문에서 제안한 n^2과 비교해 보면 n값이 크다고 하면 대략 14배정도 데이터를 가져오는 횟수가 줄어드는 것을 알 수가 있으며 이것을 입력영상의 크기에 따라서 비교해 보면 (그림 14)와 같다. 또한 두 번째로 전체의 결과를 구하는데 필요한 연산의 횟수는 기존의 소프트웨어적인 방법이 $4pn^2$ 이며 이 논문에서 제안한 방법은 pn^2으로 대략 4배정도 차이가 나며 이것을 입력영상의 크기에 따라서 비교해 보면 (그림 15)와 같다. 결국은 데이터를 가져오는 횟수가 14배정도 줄어들고, 하나의 결과를 계산하는데 걸리는 시간이 4배정도 빠른 것을 알 수 있다. 이러한 수치는 별도의 추가되는 시간을 고려하면 훨씬 더 많은 차이가 날 것이다. 그리고 이러한 결과는 (그림 14)와 같다.



(그림 14) 메모리 엑세스 횟수의 비교



(그림 15) 연산횟수의 비교

6. 결 론

영상처리에서는 많은 정보량으로 인하여 실시간 처리에 어려움이 많아 응용이 어느 한정된 분야에서만이 원할하게 이루어지고 있는 실정이다. 그래서 본 논문에서는 영상처리를 위한 새로운 아키텍처인 시스톨릭 어레이를 이용한 방법을 제안했다. 영상처리과정에서 특히 국부적인 연산자의 계산을 요하는 부분인 영상처리의 전처리라 할 수 있는 평활화 과정, 선 등을 검출하기 위한 과정 또는 영상의 특징을 구하기 위한 과정으로 윤곽선 검출 등의 분야 등이 많은 계산으로 인하여 실시간에 어려움이 있다. 이것을 기존의 소프트웨어적인 방법으로는 한계가 있으므로 하드웨어로 제안을 한 것이다. 이러한 시도는 다른 논문에서 먼저 발표를 했으나 여기서는 대부분의 모든 하드웨어가 갖고 있는 공통점인 하나의 결과를 위해서 이용될 수밖에

없다는 문제점이 있는데 이 논문에서는 하나의 하드웨어로서 몇 가지 중요한 결과를 선택에 의해서 구할 수 있도록 설계를 하였다.

실제 영상처리를 하는 경우 정보량이 많고 이 많은 정보는 국부적인 연산을 필요로 하므로 여러 번 메모리로 불러어져야 하는데 이것은 메모리 대역폭으로 인하여 처리속도를 급격하게 떨어뜨리게 된다. 이런 단점을 보완하는 새로운 아키텍처의 개념이 많이 발표되었으나 그 중에서도 시스톨릭어레이가 가장 적합하다. 그래서 이 논문의 평가는 입력 영상이 메모리로 불리어지는 횟수를 비교 평가했으며, 하나의 국부적인 연산자를 이용하여 계산을 하는데 필요로 하는 연산횟수를 비교 평가하였다. 평가 결과 첫 번째로 메모리에서 가져오는 횟수를 구해 본 것은 입력 영상이 $n \times n$ 일 때 기존의 소프트웨어적인 방법이 n 값이 크다고 할 때 대략 $14n^2$으로 이 논문에서 제안한 n^2과 비교해 14배정도 데이터를 가져오는 횟수가 줄어드는 것을 알 수 있었으며, 두 번째로 하나의 결과를 구하는데 필요한 연산의 횟수는 기존의 소프트웨어적인 방법이 곱셈만을 비교했을 때 4배 차이가 난다. 결국은 데이터를 가져오는 횟수가 14배정도 줄어들고, 결과를 계산하는데 걸리는 시간이 4배정도 빠른 것을 알 수 있다. 이러한 수치는 별도의 추가되는 시간을 고려하면 훨씬 더 많은 차이가 날 것이다.

앞으로의 과제는 이 논문을 영상처리의 전 분야에 적용할 수 있도록 확장하는 것이다. 그렇게 하기 위해서는 시스톨릭어레이의 Cell에 대한 연구와 이 아키텍처로 매핑(Mapping)하는 문제를 더 연구 보완해야 할 것이다.

참 고 문 헌

[1] P.S.Sastry, N.Ranganathan, "A VLSI Architecture for Approximate Tree matching," IEEE Trans. on Computers, Vol.47, No.3, pp.346-352, March, 1998

[2] P.Kornerup, "A Systolic, Linear-Array Multiplier for a Class of Right-Shift Algorithms," IEEE Trans. on Computers, Vol.43, No.8, pp.892-898, August, 1994

[3] H.T.Kung, "Why Systolic Architecture," Computer Magazine 15(1), pp.37-46, January, 1982.

[4] Kai.Hwang and Faye A.Briggs, Computer Architecture and Parallel Processing, Mcgraw. Hill Series in Computer Organization and Architecture, pp.769-807, 1984.

[5] A.L. Fisher and H.T.Kung, "Synchronizing Large VLSI Processor Arrays," IEEE Trans. on Computers, Vol.c-34, No.8, August, 1985.

[6] H.T.Kung, L.M.Ruane and D.W.L.Yen "Two-level Pipelined Systolic Arrays for Multi-dimensional Convolution," Image and Vision Computing 1(1), pp.30-36, February, 1983.

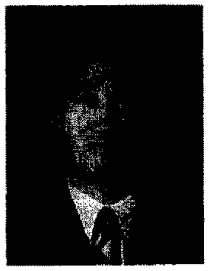
[7] H.T.Kung and R.L.Picard "One-dimensional Systolic Arrays for Multidimensional Convolution and Resampling," VLSI for Pattern Recognition and Image Processing, pp.9-24, Springverlag, 1984.

[8] Ronald Davis and Dave Thomas, "Integrated Circuits," NCR Cooperation Ft. Collins, March, 1985.

[9] H.T.Kung and S.W.Song, "A Systolic 2_D Convolution Chip," Multicomputers and Image Processing : Algorithms and Programs, pp.373-384, Academic Press, 1982.

[10] A.L. Fisher, H.T.Kung and L.M.Monier, "Architecture of the PSC : A Programmable Systolic Chip," Proceeding of the 10th Annual Symposium on Computer Architecture, June, 1983.

[11] G.E.Bridges, "Dual Systolic Architecture for VLSI Digital Signal Processing System," IEEE Trans. on Computers, Vol.c-35, No.10, pp.916-923, October, 1986.



박 덕 원

e-mail : pdw403@venus.semyung.ac.kr

1986년 숭실대학교 전자계산학과 졸업(학사)

1988년 숭실대학교 대학원 전자계산학과 졸업(공학석사)

1997년 충남대학교 대학원 계산통계학과 졸업(이학박사)

1988년~1991년 충남전문대학(조교수)

1991년~현재 세명대학교 컴퓨터과학과(부교수)

관심분야 : 컴퓨터아키텍처, 병렬처리, 영상처리