Journal of Knowledge Information Technology and Systems(JKITS), Vol. 10, No. 3, pp. 377~388, June 2015

Journal of Knowledge Information Technology and Systems

ISSN 1975-7700

http://www.kkits.or.kr

Image Preprocessing using Systolic Arrays

Hee-Seok Kang[1], **Jie-Young Lee**[2], **Durk-Won Park**[*2]

[1]*Hanra jeongmil Engineering*
[2]*School of Computer science, Semyung University*

ABSTRACT

This paper proposed a systolic array architecture for computing image preprocessing. It is very difficult to be processed images to real time because of operations of local operators. Local operators for computing image preprocessing are used in many image processing tasks, involve replacing each pixel in an image with a valve computed within a local neighborhood of that pixel. Computing such operators at the video rate requires a computing power which is not provided with software method by conventional computer. In this paper, we improved these problem by using systolic arrays. Once the data has been fetched from memory, it was used a lot of times in inside computer system. As a result, image data fetched from memory were used in each cell in CPU many times before it was return to main memory. There, we were able to obtain high throughput with the usual memory bandwidth. A architecture for processing of image preprocessing consisted of one dimensional systolic arrays. A cell has three mask value for image preprocessing. Each cell is executed in parallel at the same time. We had simulated its architecture, and not only achieve good results but also reduce processing time.

KEYWORDS: systolic arrays, cells, image preprocessing, local operator, memory bandwidth, mask value,

ARTICLE INFO: Received 13 May 2015, Revised 12 June 2015, Accepted 12 June 2015.

1. 서 론

최근에는 디지털 영상 장치의 발전으로 일상생활에 눈부신 변화를 가져왔다. 특히 초음파, 전자현미경 그리고 컴퓨터 영상 등의 발전은 일상생활뿐 만아니라 우주분야와 군사 분야에서도 혁신적

[*]Corresponding autho is with the School of Computer Science, Semyung University.
E-mail address: pdw403@semyung.ac.kr

인 변화를 가져오고 있다. 그러나 처리 속도가 빨라지는 것보다 처리량이 더 방대하게 증가해서 실시간에 우리가 원하는 결과를 얻기에는 일부분 부족한 면이 있다. 또한 여러 분야에서 영상의 고화질화는 이러한 문제점을 더 가중시키고 있다. 방대한 양의 고화질 영상을 처리하기 위해서는 소프트웨어적인 방법으로는 우리가 원하는 결과를 전부 얻을 수는 없다. 영상처리는 컴퓨터로 처리하는 그 어떤 업무보다도 많은 정보량을 갖고 있으며 이 정보는 화소로 표시된다. 결국 각 화소를 국부적으로 처리해야 되는 문제점 때문에 더욱 더 실시간 처리에 어려움이 따른다. 실시간 처리를 하려면 처리할 양이 적든지 또는 처리과정이 소프트웨어적이 아니라 하드웨어적인 방법으로 처리를 할 수 있을 때 가능하다. 이 분야에 대한 특수한 하드웨어의 부족과 컴퓨터가 갖고 있는 처리속도의 문제점 때문에 실시간 처리의 문제는 한정된 분야에서만 가능하다. 가격이 저렴하면서 계산속도가 빠른 소형 반도체 기술이 발전함에 따라 영상처리에서도 실시간 처리를 위한 새로운 아키텍처의 연구가 지속적으로 진행되고 있다[1-4].

이 논문에서는 영상처리의 실시간 처리에 많은 어려움이 있었던 국부적 연산자의 처리를 시스톨릭 어레이로 동시에 처리가 가능하도록 제안하였다. 국부적인 연산이 빈번하게 일어나는 전처리 중에 하나인 엠보싱, 블러링 그리고 샤프닝을 처리하는 시스톨릭 어레이를 제안한 것이다. 시스톨릭 어레이는 H.T.Kung에 의해서 제안된 아키텍쳐로 특수목적의 병렬처리기이다[5-9]. 간단한 연산능력을 가진 여러 개의 내부연산 스텝 프로세서로 구성되어 있으며, 이 프로세서들은 정규적이고 국부적인 통신링크로 연결된 네트워크이다. 그래서 이 시스톨릭 어레이는 처리 횟수 보다는 입출력 횟수가 많은 분야에서 폭 넓게 사용되고 있다[10-15].

2. 연구 배경

2.1 시스톨릭 어레이

시스톨릭 어레이란 각 내부 연산 스텝 프로세서들이 연속적으로 연결되어 서로 데이터를 주고받으면서 계산을 실행하고 그 결과를 출력하는 네트워크이다. 영상처리 등에서 기존의 컴퓨터로 실시간 처리를 하려면 방대한 자료로 인해 처리하는 과정에서 입출력 병목 현상이 생겨 처리속도가 떨어진다. 이런 단점을 보완하기 위해서 한번 메모리에서 가져온 데이터는 각 내부 연산 스텝 프로세서를 거치면서 최대한도로 사용된 후 다시 메모리로 보내므로 입출력 병목 현상을 해결하고 보통의 메모리 대역폭을 가지고도 높은 계산 속도를 갖는다. 또한 이 방법은 입출력 횟수보다 더 많은 계산을 요구하는 문제에 적합하다. 주로 많이 이용되는 내부 연산 스텝 프로세서의 구성은 <그림1>과 같다.

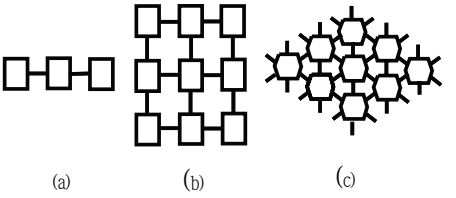

그림1. 시스톨릭어레이의 일반적인 구성
Figure 1. Various systolic array configurations

2.2 영상 전처리

화소 단위 처리는 영상 조합, 영상 변경과 수정을 위한 근본적으로 중요한 디지털 영상처리 도구이지만 화소 단위 처리는 장면 내용을 공간단위로 바꾸지 못한다. 이것은 화소 단위 처리가 하나의

입력 화소를 그 대응하는 출력 화소로 사상시키는 화소 단위의 행동을 하기 때문이다. 화소 단위 처리는 그 처리 특성상 이웃화소를 고려하지 못하지만 필터링처리는 중심화소 주변의 화소 그룹을 대상으로 동작한다. 인접한 화소는 처리될 영역내의 변화에 대한 정보를 제공하며 방대한 양의 처리가 반복되어 이루어진다. 기본적인 2차원 필터링 기법에는 샤프닝(Sharpening), 블러링(Blurring), 엠보싱(Embossing) 그리고 에지검출과 같은 방법들이 있으며, 영상처리 알고리즘에 있어서 많이 사용된다. 2차원 필터링의 처리는 회선처리(Convolution)에 의해 처리되며, 이는 커다란 원본 이미지 위를 마스크가 움직이며 주변 화소 값을 고려해 함께 연산을 하기 때문에 연산 량이 매우 많으며 메모리 접근을 많이 필요로 한다. 이 마스크가 움직이며 행렬의 승산을 반복하기 때문에 일반의 시스템에서는 많은 처리시간이 요구된다. 회선처리는 처리될 화소 주변의 화소로 어떠한 수행을 할 것인지를 계산하기 위해서 사용하는 방법이며, 이것은 신호처리나 분석에서 사용하는 방법이다. 마스크란 원본영상에 사상되어 처리 될 가중치이고, 각각의 처리 방법에 따라 다른 가중치 값을 가지고 있다.

본 논문에서는 엠보싱, 블러링, 샤프닝을 처리하는 시스톨릭 어레이를 제안하였으며, 각각의 마스크 값은 <그림2>와 같다.

엠보싱 처리를 위한 마스크의 특징은 가운데 가중치가 다른 계수를 상쇄시키도록 구성해서 경계선을 검출하게 된다. 이 경계선에서 양각한 효과를 얻을 수 있다. 마스크에는 음의 계수 값 -1이 있으므로 회선처리로 생성된 영상의 화소 값은 음수이다. 화소가 양의 값이 되도록 하려면 회선 처리한 뒤 화소에 일정한 상수 값을 더해 주면 된다. 블러링 처리를 위한 마스크는 모든 계수가 양수로 전체 합은 1이다. 그런데 디지털 영상에서 세세한 부분은 화소 값이 극단적인 값에 속한다. 이 극단적 값을 제거하는 대표적인 방법이 바로 평준화로, 평균값으로 대체하는 것이다. 블러링 마스크의 가중치는 평균을 구하는 데 사용되므로 모두 값이 같다.

-1	0	0
0	0	0
0	0	1

(a)엠보싱 마스크 (a) Mask of embossing

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

(b)블러링 마스크 (b)Mask of blurring

-1	-1	-1
-1	9	-1
-1	-1	-1

(c)샤프닝 (c)Mask of sharpening

그림2. 엠보싱, 블러링 그리고 샤프닝 마스크
Figure2. Mask of embossing, blurring and sharpening

샤프닝은 블러링과 반대되는 효과를 보이는 기법으로 영상 강화라고도 한다. 고주파에 해당하는 상세한 부분을 더욱 강조하여 대비 효과를 증가시킨다. 이 기법은 흐린 영상을 개선하여 선명한 영상을 생성하는 데 주로 사용된다. 샤프닝 처리에서 사용되는 마스크의 한 가운데는 양의 값, 바깥 경계에는 음의 값이 있으며, 그 합은1이 된다. 가운데에 있는 큰 양수 값과 주변의 작은 음수 값이 경계선을 더욱 두드러지게 해서 영상의 고주파 성분을 강조한다. 위와 같은 마스크들은 회선처리 기법에 의해 처리되며 <그림3>에 5x5 크기의 입력영상과 3x3의 마스크가 회선처리 되는 것을 보여준다.

I_1	I_2	I_3	I_4	I_5
I_6	I_7	I_8	I_9	I_{10}
I_{11}	I_{12}	I_{13}	I_{14}	I_{15}
I_{16}	I_{17}	I_{18}	I_{19}	I_{20}
I_{21}	I_{22}	I_{23}	I_{24}	I_{25}

(a) 입력영상 (a) Input image

M_1	M_2	M_3
M_4	M_5	M_6
M_7	M_8	M_9

(b) 마스크 (b) Mask

그림3. 회선처리의 예
Figure 3. Example of convolution

이 <그림3>에서 제일 먼저 회선처리로 구해지는 I7을 구하면 다음과 같다.

$$I_7 = I_1 \times M_1 + I_2 \times M_2 + I_3 \times M_3 + I_6 \times M_4 + I_7 \times M_5 + I_8 \times M_6 + I_{11} \times M_7 + I_{12} \times M_8 + I_{13} \times M_9 \quad (1)$$

입력영상(a)의 I7의 값은 <그림3> (b)이 마스크 값이 입력영상 (a)에 사상되어 구해진다. 이렇게 하여 I7의 값을 구한 후 마스크는 다시 오른쪽으로 옮겨 I8을 중심으로 마스크가 사상되어 처리된 후 한 줄에 대한 처리가 끝나면 다음 줄로 이동하여 다시 한 화소씩 처리를 한다. 회선 처리 시 고려되어야 할 것들이 있는데 우선 마스크의 크기는 홀수x홀수 크기를 만족해야 하며, 정방행렬이여야 한다. 이때 마스크의 중앙값이 회선처리 될 영상의 화소에 사상되어 처리가 된다. 이러한 연산이 입력영상에 대해 국부적으로 빈번하게 연산되기 때문에 기존의 소프트웨어적인 방법에서는 연산속도가 떨어지고 메모리 낭비가 심하다. 회선처리는 식(2)로 정의할 수 있다.

$$\text{output\_pixel}[x,y] = \sum_{m=(x-k)}^{x+k} \sum_{n=(y-k)}^{y+k} (I[m,m] * M[m,n]) \quad (2)$$

여기서 output_pixel[x,y]는 회선처리로 출력한 화소, I[m,n]는 입력영상의 화소, M[m,n]는 입력영상에 대응되는 마스크 값을 말한다. 새로운 화소의 좌표는 x, y이고, 이것을 생성하는 데 관여하는 주변 화소의 수는 k값이 결정한다. 예를 들어 k=1 이면 관여하는 화소의 수는 3x3 배열이므로 총 아홉 개다. 그리고 배열의 중심부가 새로운 화소의 위치 x, y가 된다. 본 논문에서는 경계 부분을 고려하지 않고 처리 하였지만 보다 정확한 처리를 위해서는 경계 부분을 고려해야 한다.

3. 영상 전처리를 위한 시스톨릭 어레이

필터링 처리는 하나의 화소를 처리하기 위해서 현재 화소를 포함하여 총 9개의 화소가 필요하다. 필터링 처리의 특성상 한 번의 연산이 끝난 후 다시 옆으로 한 칸 옮겨진 뒤, 같은 연산이 반복되기 때문에 한번 메모리로부터 인출되었던 화소가 다음 처리에서 다시 인출되어 사용되어야 하므로 소프트웨어적인 방법으로는 속도가 많이 떨어진다. 그래서 이런 처리에 효율적인 하드웨어인 시스톨릭 어레이를 이용하여 영상의 일부 전 처리라 할 수 있는 엠보싱, 블러링 그리고 샤프닝을 처리 한 것이다.

3.1 제안한 시스톨릭 어레이 셀 구조

세 가지 종류의 전처리를 위한 시스톨릭 어레이의 셀 내부구조는 <그림4>와 같다. 각각의 전처리 값을 누적해 나가는 Y값의 초기 값은 0이며, 이 값이 처음 덧셈 연산자에 들어가고 입력 영상 값 Xij와 마스크 값 W가 곱해진 후, Y값과 누적된 값을 방출 하는 흐름을 보여주고 있다.

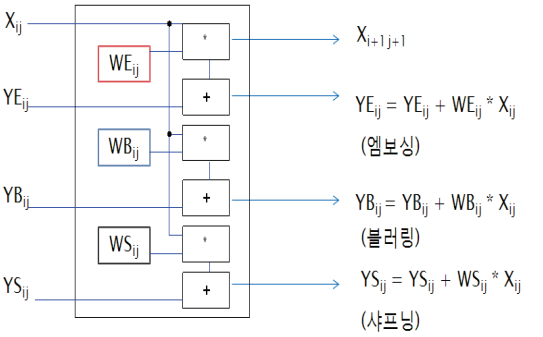

그림4. 기본 셀 구조
Figure 4. Implementation of the basic cell

이렇게 설계한 내부 구조를 이용하여 1차원 배

열 3개를 누적하여 2차원 시스톨릭 어레이로 설계한 것이 <그림5>이다. <그림5>의 동작은 연속적으로 8개의 셀을 통과하며 9번의 곱셈과 8번의 덧셈 연산을 한 후 첫 번째 화소 점에 대한 각각의 필터링 처리의 최초 결과 값이 방출된다. 9번째 사이클에서 최초 결과 값이 방출된 후, 매 사이클에서 결과 값이 나오게 된다.

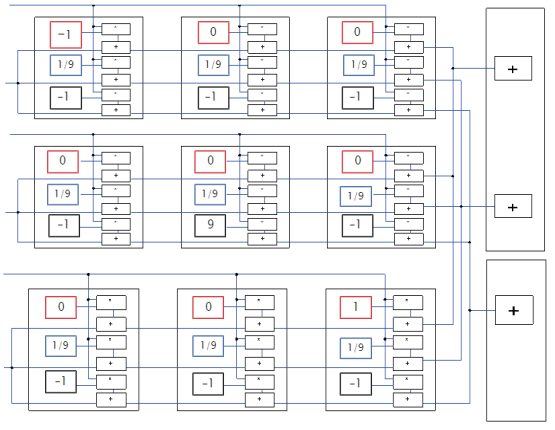

그림5. 1차원 배열 3개를 이용한 2차원 시스톨릭 어레이
Figure 5. 2-D systolic array obtained by stacking three 1-D arrays

<그림5>에서 첫 번째 행은 엠보싱 마스크 값을 나타내었고, 두 번째 행의 값은 블러링 마스크 값이며 마지막 행은 샤프닝 마스크 값을 의미한다. 전체 셀은 9개로 구성되어 있고 각 셀은 엠보싱 마스크 값, 블러링 마스크 값 그리고 샤프닝 마스크 값 하나 씩을 가지고 있다. 입력 영상은 1차원 형태로 연속적으로 받아들이고 각 3개의 배열에서 나온 결과 값을 더하여 누적하는 형태이다[16-18].

제안하는 시스톨릭 어레이는 1차원 배열의 형태이다. 필터 마스크의 값은 각 셀의 내부에 저장되어 있고 입력 영상은 1차원 형태로 받아들인다. 입력 순서는 1차원이되, 역순으로 받아들이게 된다. <그림6>의 입력 영상을 예로 들었을 때, 처음 X22을 처리하게 되는데 이때 입력 영상의 순서는 X33, X23, X13, X32, X22,...,X11 와 같이 역순으로 입력이 된다. X22의 처리가 끝난 후 X23을 처리하기 위해서는 다시 X34, X24, X14, X33,...,X12가 순서대로 입력되게 된다. 셀은 총 9개이며 각 셀에는 엠보싱, 블러링, 샤프닝 마스크의 값을 가지고 있다. 입력 영상의 순서가 역순이듯 각각의 셀 내부에 저장되어 있는 마스크들의 값도 역순으로 배치되어있다. 하나의 셀에는 3가지 필터 마스크 값이 들어있고 총9개의 셀에 27개의 마스크 값이 저장되어 있다. Xij는 입력 영상의 좌표 값을 의미하고 Yij는 3가지 필터링 처리의 누적 합을 구하는 변수이다. 9개의 셀을 모두 거친 후에는 하나의 화소처리에 대한 결과를 방출하는 형태이다.

X_{11}	X_{12}	X_{13}	X_{14}	X_{15}	X_{16}	X_{17}	X_{18}
X_{21}	X_{22}	X_{23}	X_{24}	X_{25}	X_{26}	X_{27}	X_{28}
X_{31}	X_{32}	X_{33}	X_{34}	X_{35}	X_{36}	X_{37}	X_{38}
X_{41}	X_{42}	X_{43}	X_{44}	X_{45}	X_{46}	X_{47}	X_{48}
X_{51}	X_{52}	X_{53}	X_{54}	X_{55}	X_{56}	X_{57}	X_{58}
X_{61}	X_{62}	X_{63}	X_{64}	X_{65}	X_{66}	X_{67}	X_{68}
X_{71}	X_{72}	X_{73}	X_{74}	X_{75}	X_{76}	X_{77}	X_{78}
X_{81}	X_{82}	X_{83}	X_{84}	X_{85}	X_{86}	X_{87}	X_{88}

그림6. 입력 영상
Figure 6. Input image

3.2 시스톨릭 어레이를 이용한 영상 전처리

총 9개로 구성된 시스톨릭 어레이는 처음에는 모든 동시에 실행되지 않고 입력이 첫 번째 셀로 들어오면 처리한 후 다음 셀로 보내어 지고 이 셀에서 처리한 결과가 다시 다음 셀로 보내주기 때문에 최초의 결과 값이 나오는 9번째 사이클까지 처리가 되어야 모든 셀이 동작을 하게 된다. 이후 모든 셀들이 활성화되며, 매 사이클마다 결과 값이 출력된다[13].

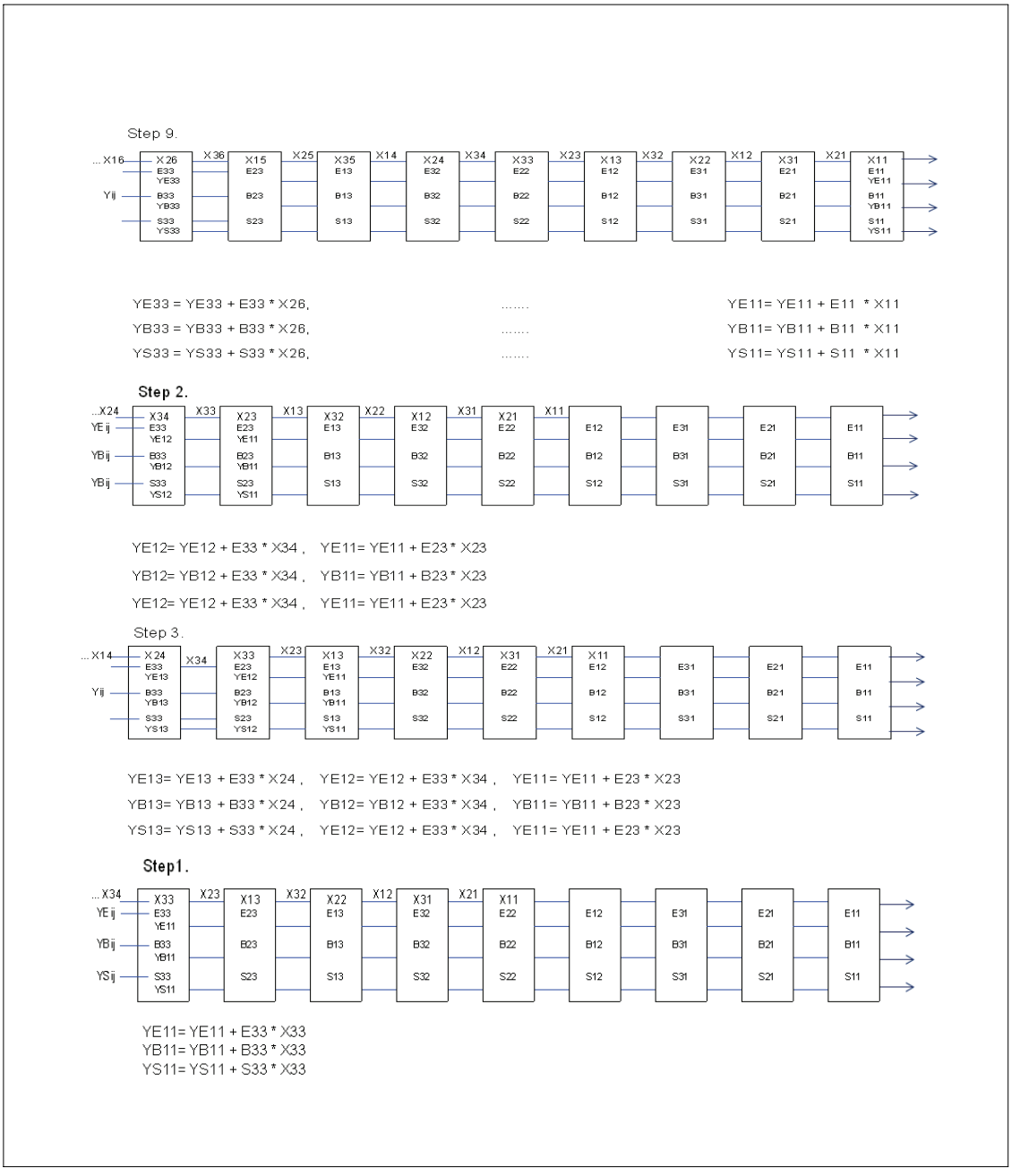

그림7. Y11을 구하는 시스톨릭 어레이의 연속 과정
Figure7. 9 consecutive steps of the computation of Y11

<그림7>은 매 사이클 마다 시스톨릭 어레이에서 연산되는 과정을 보여주는 스냅 샷을 설명할 것이다. 스텝1은 최초의 사이클에서 첫 번째 셀이 동작하는 것을 보여준다. 입력 영상 X33을 받아 각 마스크 값 E33, B33, S33, 과 곱하여 Y에 누적하여 다음 셀로 보내준다. 두 번째 사이클에서는 첫 번째 셀은 다시 새로운 영상의 입력을 받고, 스텝2는 첫 번째 셀로부터 받은 누적 값과 입력 영상을 연산하여 세 번째 셀로 방출한다. 이렇게 매 사이클이 증가하는데 따라서 그 결과 값 다음 셀로 입력되어 연산된 후 결과 값을 또 다른 셀의 입력으로 들어간다. 그렇게 해서 9번 째 사이클이 되면 스텝9와 같이 Y11의 결과 값이 9개의 셀에서 모두 연산된 후 최초로 첫 번째로 회선처리 된 결과를 얻게 된다. 그리고 매 사이클 마다 엠보싱 회선 처리한 결과, 블러링 회선 처리한 결과 그리고 샤프닝 회선 처리한 결과를 동시에 얻게 된다.

4. 시뮬레이션 및 평가

영상 전처리와 같이 회선처리를 하는 기법은 동일한 입력을 여러 번 반복사용하기 때문에 메모리의 입력에 비해 처리되어 출력되는 속도가 떨어져 CPU의 효율을 떨어뜨리는 요인이 된다. 이러한 처리를 소프트웨어적인 방법으로 처리할 경우 실시간 처리에 있어서 많은 어려움이 따른다. 이에 이 논문에서는 입출력에서의 병목현상을 없애고 높은 처리율을 보여주는 시스톨릭 어레이를 제안하여 첫 번째로는 소프트웨어 구현에 의해서 정상적으로 처리되는 것을 확인하였으며, 두 번째로는 기존의 소프트웨어적인 방법으로 처리한 경우와 시스톨릭 어레이를 적용하였을 때 입력 영상을 메모리로부터 CPU로 가져오는 횟수를 비교하였다. 여기서 회선처리를 하기 위해서 입력영상을 메모리로부터 CPU로 가져오는 횟수의 비교는 기존의 방법이 소프트웨어로 처리하는 것이고 여기서는 시스톨릭 어레이로 제안한 하드웨어이기 때문에 현실적으로 단순 비교는 어렵다. 그래서 회선처리를 할 때에 CPU에서 처리되는 시간은 빠르나 메모리로부터 가져오는 시간이 많이 걸리므로 이 횟수를 비교한 것이다.

4.1 시뮬레이션

소프트웨어로 시뮬레이션 했을 때 사용한 언어는 Visual C++ 6.0을 사용해 프로그램 하였고 512x512 크기의 로우 포맷 파일의 그레이 영상 LENA 이미지를 적용하였다. <그림8>은 구현한 소프트웨어에 대한 플로차트이다. 입력영상의 넓이와 높이만큼 반복 루프를 수행하며 그 안에서 마스크와 입력 영상의 화소가 곱셈과 덧셈 연산이 이루어지는 과정을 보여주고 있다. 연산이 끝나면 결과 값을 메모리에 저장하고 다시 마스크를 옮긴 후 연산되는 과정을 보여주고 있다.

<그림9>는 각각을 3가지 방식으로 전처리를 한 결과이다. (a)는 엠보싱 처리되어 양각한 효과를 보이고, (b)는 블러링 처리를 한 것으로 평균 값을 이용하여 전체 이미지가 부드럽게 처리되었다. (c)는 샤프링 처리를 한 것으로 더욱 강조된 느낌의 처리 결과를 보여주고 있다.

4.2 평가

기존의 소프트웨어적인 처리방법에서 경계 부분들의 메모리 접근 횟수는 <그림10>과 같다. 이 영상의 크기를 $n \times n$의 크기라 했을 때 빈 공간은 9번 모두 접근되는 것을 의미하며, 9번 모두 접근되는 부분의 메모리 접근 횟수는 $(n-4)^2 \times 9$ 번이고 그 외 나머지 경계 부분들의 총 접근 횟수는 252번이다.

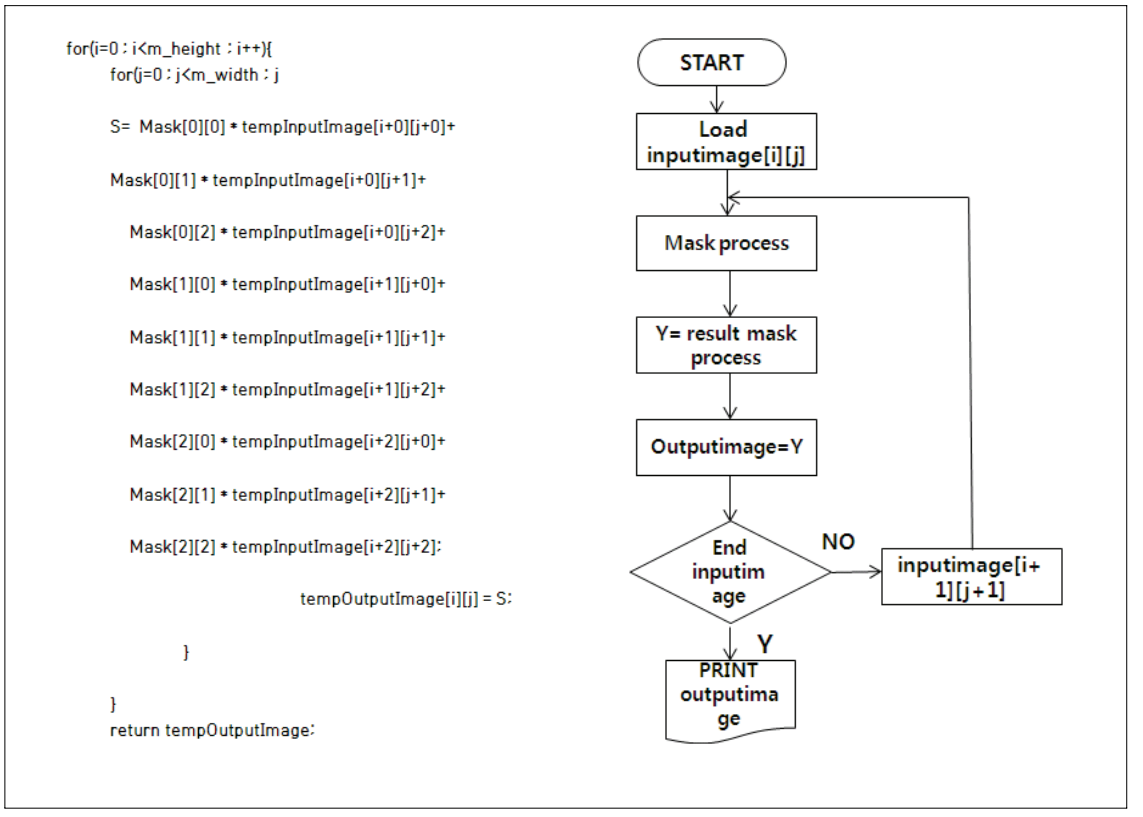

그림8. 구현한 소프트웨어의 플로차트
Figure 8. Flowchart of implemented software

(a) 엠보싱 (a)Embossing (b) 블러링 (b)Blurring (c) 샤프닝 (c)Sharping

그림9. 전처리한 이미지
Figure 9. Preprocessed image

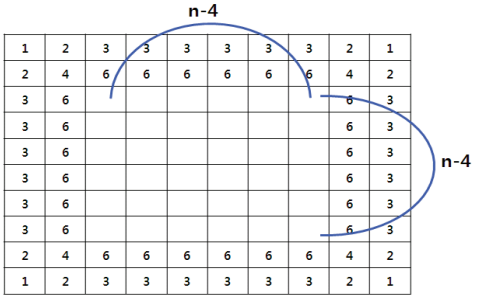

그림10. 화소들의 메모리 접근 횟수
Figure 10. Memory access number of image pixels

이식을 풀면 엠보싱, 블러링, 샤프닝 처리를 위해서 각각 $9n^2-72n+396$의 입력 횟수가 필요하다는 것을 알 수 있다. 그리고 제안한 시스톨릭 어레이를 적용하여 처리 할 경우 시스톨릭 어레이의 특성에 따라 입력 영상이나 마스크의 크기에 무관하게 메모리로부터 한번 가져오면 처리가 끝날 때까지 최대한으로 사용되므로 입력영상이 $n\times n$의 크기 일 때, 단지 n^2번의 입력만이 필요하다. 기존의 소프트웨어적인 처리 방법이 $9n^2-72n+396$번의 입력 횟수를 보이는 반면 제안한 시스톨릭 어레이의 경우는 n^2번의 입력 횟수를 보이며 동시에 3가지 전처리를 하기 때문에 <표1>과 같이 약 27배 정도 입력 횟수가 적어진 것을 알 수 있다.

표1. 개선된 입력 횟수
Table 1. Improved input number

		화소의 입력횟수
소프트웨어에 의한 방법	엠보싱	$9n^2-72n+396$
	블러링	$9n^2-72n+396$
	샤프닝	$9n^2-72n+396$
	합계	$27n^2-216n+1188$
본 논문에서 제안한 방법	엠보싱	n^2
	블러링	
	샤프닝	
	합계	n^2
개선된 입력 횟수	입력 횟수가 약 $\frac{1}{27}$로 줄어들음	

5. 결 론

본 논문에서는 영상처리의 필터링처리를 위한 새로운 아키텍처인 시스톨릭 어레이를 제안하였다. 기존의 소프트웨어적인 방법으로 영상을 실시간으로 처리하는 것은 많은 계산으로 인하여 실시간 처리에 한계가 있으므로 영상처리에 아주 효율적인 시스톨릭 어레이를 이용한 방법을 제안한 것이다. 실제 영상처리를 하는 경우 정보량이 많고 이 많은 정보는 국부적인 연산이 필요로 하므로 여러 번 메모리로 불리어져야 하는데 이것은 메모리 대역폭으로 인하여 처리속도를 급격하게 떨어뜨리게 된다. 이런 단점을 보완하는 아키텍처가 많이 발표되었으나 그 중에서도 시스톨릭 어레이가 가장 적합하다.

이 논문의 평가는 입력 영상이 메모리로 불리어지는 횟수를 비교 평가 하였으며, 제안된 시스톨릭 어레이를 사용한 경우에는 일반적인 소프트웨어로 처리한 횟수 $9n^2-72n+396$번 에 비해 시스톨릭 어레이를 적용한 경우 n^2번만의 입력이 필요했다. 이 횟수를 비교하면 대략 9배의 정도의 메모리 입력 횟수가 줄어든 것을 알 수 있고 동시에 3가지 처리를 병행하므로 기존의 소프트웨어적인 처리에 비해 대략 27배 정도의 적은 메모리 입력 횟수를 보였다. 그러나 이것은 단순하게 영상 전처리를 위해 입력 영상이 회선처리를 위해 메모리로부터 CPU로 읽혀오는 횟수만을 비교한 것이나, 실제는 기존의 소프트웨어적인 처리를 여기서는 하드웨어로 제안한 것이므로 실제 처리할 경우에 속도가 기존의 방법보다는 훨씬 더 빠를 것이다. 그리고 동일한 3x3크기의 마스크를 처리할 때 초기 저장 값만 다르게 주면 다른 전처리에 대해서도 유동적으로 처리가 가능하다. 또한 이 논문에서는 3x3크기 마스크 처리를 위하여 9개의 셀을 사용하였지만, 5x5마스크의 경우에는 25개의 셀, 7x7마스크의 경우에는 49개의 셀을 동일한 방법으로 추가하여

처리가 가능하다. 마지막으로 영상처리의 전 분야에 확장 적용이 가능하도록 더 연구・보완해야 한다.

References

[1] Durk-Won Park, *Systolic arrays for edge detection of image processing*, Korea Information Processing Society, Vol. 6, No. 8, pp. 2222-2232. 1999.

[2] Jae-Jin Lee, and Gi-Young Song, *Design of a bit-level super-systolic array*, The Institute of Electronics and Information Engineers, Vol. 42, Sd No. 12, 2005.

[3] Kenji Murakami, Makoto Aboshi, Koji Kinoshita, and Masaharu Isshiki, *Fast line detection by hough trasform using inter-image operation*, IEEJ Transactions on Electronics, Information and Systems, Vol. 133, No. 9, pp. 1539-1548, 2013.

[4] Lional Gueguen, Santiago Velasco-Forero, and Pierre Soille, *Local mutual information for dissimilarity-based image segmentation*, Journal of Mathematical Imaging and Vision, Vol. 48, No. 3, 2014.

[5] H.T.Kung, *Why systolic architecture*, Computer Magazine 15(1), pp. 37-46, 1982.

[6] 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, 1983.

[7] P.Kornerupm, *A systolic, linear-array multiplier for a class of right-shift algorithms,* IEEE Trans, on Computers, Vol. 43, No. 8, pp. 892-898, 1994.

[8] Nobuo Kochi, Kazuo Kitamura, Takeshi Sasaki, and Shunichi Kaneko, *3D modeling by integrating point cloud and 2D images through 3D edge-matching and its application to architecture modeling*, IEEJ Transactions on Electronics, Information and Systems, Vol. 133, No. 6, pp. 1160-1172, 2013.

[9] Yonghun Park, Junsang Seo, and Cheolhong Kim, *Implementation of a multi-core prototype system for image processing algorithms*, The Journal of Korea Institute of Next Generation Computing, Vol. 10, No. 4, 2014.

[10] Jin-Pyo Hong, and Kee-Young Yoo, *Computing space matrices for designing planar systolic arrays*, The Korea Institute of Information Scientists and Engineers, Vol. 21, No. 7, 1994.

[11] Yun-Ho Kim, *A method of determining the space matrices for systolic arrays*, The Korea Institute of Information Scientists and Engineers, Vol. 26, No. 2, 1999.

[12] Jae-Cheol Ha, and Sang-Jae Moon, *Design of montgomery modular multiplier based on systolic array,* Korea Institute of Information and Communication Engineers, Vol. 9, No. 1, pp. 135-146, 1999.

[13] Keon-Jik Lee, Young-Jun Heo and Kee-Young Yoo, *Fast modular exponentiation on a systolic array*, Korea Institute of Information and Communication Engineers, Vol. 8, No. 1, pp. 39-52, 1998.

[14] Yun Yang, and Shinji Kimura, *Optimal planar jumping systolic array design for matrix multiplication : The 20th workshop on circuits and systems in Karuizawa*, pp. 23-24, April 2007.

[15] Jin-Ho Lee, and Hyun-sung Kim, *Systolic architecture for digit level modular multiplication/squaring over GF(2m)*, The Korea Institute of Information Security and Cryptology, Vol. 18, No. 1, pp. 41-47, 2008.

[16] Y0ng-Sung Kim, *A study on improving the performance of one dimensional systolic array processor for matrix. vector operation using sub-matrix*, The Journal of Information Technology, Vol. 10, No. 3, pp. 34-45, 2007.

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

[18] H. T. Kung, and R. L. Picard, *One-dimensional systolic arrays for multimensional convolution and resampling,* VLSI for Pattern Recognition and Image Processing, pp. 9-24, 1984.

시스톨릭 어레이를 이용한 영상 전처리

강희석[1], 이지영[2], 박덕원[2]
[1]*한라정밀 엔지니어링*
[2]*세명대학교 컴퓨터학부*

요 약

이 논문에서는 영상의 전처리를 하는 시스톨릭 어레이를 제안하였다. 영상을 실시간으로 처리하는 것은 국부적인 연산들이 빈번하게 일어나기 때문에 처리하는데 어려움이 따른다. 여상의 전처리를 위한 국부적 연산자는 한 화소의 이웃에 있는 다른 화소를 이용하여 처리를 하는 것으로 기존의 컴퓨터에서는 빈번한 입출력의 요구로 인하여 실시간 처리에서 요구하는 계산 능력을 충분하게 제공하지 못한다. 이 논문에서는 이러한 단점을 보완하기 위해서 한번 메모리에서 인출해온 데이터는 각 셀에서 최대한으로 사용한다. 그래서 보통의 메모리 대역폭을 가지고도 빠른 처리가 이루어지도록 하였다. 제안한 하드웨어는 영상 전처리를 위해 1차원 시스톨릭 어레이로 구성하였으며, 한 개의 셀에는 3가지 영상 전처리를 위한 마스크 값이 들어있으며 각 셀을 거치면서 결과 값을 누적해 가도록 설계하였다. 이렇게 설계된 것을 시뮬레이션 한 결과 정상적으로 처리가 되는 것을 확인 하였으며 영상처리를 할 때 가장 처리속도를 떨어트리는 빈번한 입력의 횟수를 현저하게 줄여서 처리속도를 빠르게 하였다.

Hee Seok Kang received the B.S. and M.S. degree in the Department of Computer Science from the Semyung University in 2011 and 2013. He has been working in the Hanra Pricision Engineering

E-mail address: zii918@naver.com

Jie Young Lee received the M.S. and Ph.D. degree in the Department of Electronics from the Sungkyunkwan University in 1983 and 1988. He worked as a professor in Department of Electronics R.O.K Naval Academy from 1984 to 1992. He has been a processor in School of Computer Science, Semyung University since 1992. His current research interests include OS and parallel processing..

E-mail address: ljy409@venus.semyung.ac.kr

Durk Won Park received the B.S. and M.S. degree in the Department of Computer Science from the Soongsil university in 1982 and 1988. He completed his Ph.D. in the Department of Computer Science from the Chungnam University in 1997. He worked as a Professor in Department of Computer Science, Daeduk College from 1988 to 1991. He has been a processor in School of Computer Science, Semyung University since 1991. His current research interests include parallel processing and image processing.

E-mail address: pdw403@semyung.ac.kr