論文93-30B-5-4

# 파이프 라인 방법을 이용한 이차원 FIR 디지탈 필터의 실시간 구현

# (The Real-Time Implementation of Two-Dimensional FIR Digital Filter using Pipe-Line Method)

尹 馨 泰*, 李 根 泳**

(Hyung Tae Yoon and Keun Young Lee)

要 約

본 논문은 실시간 화상처리를 위한 2차원 FIR 디지탈 필터의 하드웨어 구현을 기술하였다. 일반적으로 신호처리에서 가장 시간지연이 되는 연산은 곱셈처리이며 이 연산을 피하기위하여 Pelid와 Liu는 분산산술방법을 제안하였다. 본 논문에서 제안한 구현방법은 Pelid의 ROM Lookup 방법을 2차원 FIR 필터로 확장하는 방법과 파이프라인방법을 이용하여 메모리억세스과정과 연산과정, 두 과정을 병렬처리하는 방법을 제안하였다. 이 경우 제안방법은 기존방법보다 거의 두배에 가까운 처리속도를 가질수 있었으며 실시간처리가 가능하였다.

**Abstract**

This paper describes the hardware implementation of 2-D FIR digital filter for a real-time image processing. Generally, the most time-consuming operation in signal processing is the multiplication operation. To avoid it in digital filter, Pelid and Liu proposed the distributed arithmetic method for the one-dimensional case. The implementation method proposed in this paper is to extend Pelid's method to two-dimensional FIR filter using simple ROM lookup table and to use the technique of pipelining two main operations of memory access and arithmetic. As a result, the speed of our proposed hardware implementation is two times faster than that of conventional methods and can be close to the real time speed.

## Ⅰ. 서론

최근 컴퓨터와 소자들이 급속도로 발전됨으로써 화상과 같은 다차원 신호의 사용이 증가하여 2차원 디지탈 신호처리에 대한 새로운 인식과 연구가 진행되었다. 특히 지진데이타, 중력, 자기장 데이타, 생의학 화상처리, 그리고 선형 감쇠된 화상의 복구와 개선같은 많은 응용분야에서 이용되고 있으며 이러한 샘플된 2차원 데이타를 처리하기위한 2차원 필터에 관한 관심이 증가하였다. 특히 화상처리인 경우 비선형 시스템은 화상을 왜곡시키는 특성을 가지므로 선형위상과 안정성을 갖는 2차원 FIR 필터의 구현은

* 正會員, 仁德專門大學 事務自動化科
(Dept. of O.A. Eng., Induk Ins. of Design)
** 正會員, 成均館大學校 電子工學科
(Dept. of Elec Eng.,Sungkyunkwan Univ.)
接受日字: 1992年 9月 8日

중요한 의미를 갖고 있다. Pelid와 Liu [1,2] 는 누산기구조를 갖는 분산산술방법(DAM : Distributed Arithmetic Method)을 이용하여 1차원 FIR 필터의 구현을 제안하였다. 이 방법은 2차원인 경우에도 적용가능하며 실행속도면에서 효과적인 방법이 될 수 있다. 그러나 누산기구조는 많은 연산시간을 필요로 하므로 참고문헌 [3] 에서는 누산기를 제거한 구조를 이용함으로써 더 높은 데이타처리율을 갖는 방법을 제안하였지만 실시간처리하기에는 다소 문제가 있다. 따라서 본 논문에서는 1차원 분산산술방법을 2차원으로 확장시키는 방법과 파이프라인방법을 이용하여 병렬처리함으로써 실시간 처리를 할 수 있는 2차원 필터 구현방법을 제안하였다.

## Ⅱ. 분산 산술 방법

일반적으로 필터링이란 입력과 필터계수간의 컨볼류션(Convolution)으로 처리되며 2차원인 경우의 컨볼류션은 입력화상과 2차원 필터계수들간의 곱의 합으로 표현할 수 있다. 컨볼류션의 곱셈과정은 연산시간의 지연요소로써 실시간 처리를 어렵게 한다. Peled 와 Liu 는 곱셈처리에서 연산가능한 유한 결과값을 미리 계산하여 메모리에 저장한 후 반복되는 덧셈과 자리이동 동작(shift operation)으로 곱셈을 처리하는 분산산술방법을 제안하였으며 이 방법은 2차원인 경우도 적용가능하다.

2차원 FIR 디지탈 필터는 비순환(non-recursive) 구조를 가지므로 설계와 구현방법이 용이하며 N x N 차수의 2차원 전달함수는 식(1)과 같으며 [4]

$$H(Z_1,Z_2)=\sum_{i=0}^{N}\sum_{j=0}^{N}a_{i,j}\cdot Z_1^{-i}Z_2^{-j} \qquad (1)$$

선형 차분방정식으로 표현하면 식(2)와 같이 쓸 수 있다.

$$y_{m,n}=\sum_{i=0}^{N}\sum_{j=0}^{N}a_{i,j}\cdot X_{m-i,n-j} \qquad (2)$$

그림 1은 필터차수 N = 2 인 필터링 과정을 나타낸 것으로써 필터 계수 $a_{i,j}$가 입력 화상과 겹쳐져 있으며 2차원 컨볼류션은 입력 계수와 겹쳐진 샘플간의 곱들의 합으로 계산된다. 따라서 식(2)를 직접구현방법(direct realization method)으로 구현하면 그림 2와 같으며 하나의 출력값 $y_{m,n}$ 을 얻기 위해서는 9번의 곱셈, 8 번의 덧셈이 필요하다. 여기서 $Z_1$과 $Z_2$는 행과 열의 지연소(delay element)를 의미한다.

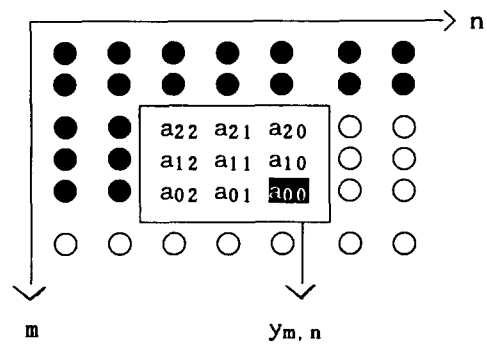



그림 1. 2차원 컨볼류션 (2x2 차)
Fig. 1. The 2-D convolution ( 2 x 2 order).

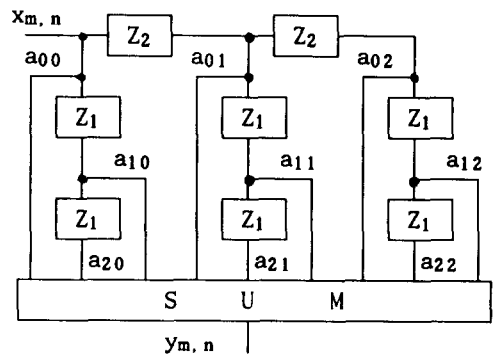



그림 2. 직접구현방법
Fig. 2. The Direct Realization Method.

입력과 출력신호는 +1 과 -1 사이의 값이며 부호비트를 갖는 B 비트 길이의 2 의 보수형으로 표현할 경우 입력신호 $x_{m,n}$은 식(3)과 같이 쓸 수 있다. [3]

$$X_{m-i,n-j}=\sum_{s=1}^{B-1}X^s_{m-i,n-j}\cdot 2^{-s}-X^0_{m-i,n-j} \qquad (3)$$

식(3)을 식(2)에 대입하면 식(4)와 같은 2의 보수형 출력 $y_{m,n}$ 을 얻을 수 있으며

$$y_{m,n}=\sum_{i=0}^{N}\sum_{j=0}^{N}a_{i,j}\cdot\left(\sum_{s=1}^{B-1}X^s_{m-i,n-j}\cdot 2^{-s}-X^0_{m-i,n-j}\right) \qquad (4)$$

식(4)의 누적합의 순서를 재정렬하여 다시 쓰면 식(5)과 같이 쓸 수 있다.

$$y_{m,n}=\sum_{s=1}^{B-1}\left[\sum_{i=0}^{N}\sum_{j=0}^{N}(a_{i,j}\cdot X^0_{m-i,n-j})\right]2^{-s}-\left[\sum_{i=0}^{N}\sum_{j=0}^{N}(a_{i,j}\cdot X^s_{m-i,n-j})\right] \qquad (5)$$

식(5)에서 대괄호를 친 항들은 모든 입력값들의 s

번째 비트들이 가질 수 있는 경우의 수에 대하여 연산가능한 필터계수 조합값을 의미한다. 이 항들을 $F^s(\cdot)$와 $F^0(\cdot)$와 같은 조합된 함수로 표현하면 식(6)과 같이 최종적인 2의 보수형 선형 차분방정식을 얻을 수 있다.

$$y_{m,n} = \sum_{s=1}^{B-1} F^s(\cdot)\cdot 2^{-s} - F^0(\cdot) \tag{6}$$

$$\begin{cases} F^s(X^s_{m,n}, X^s_{m,n-1}, \cdots, X^s_{m-N,n-N}) \\ \quad = a_{00}\cdot X^s_{m,n} + a_{01}\cdot X^s_{m,n-1} + \cdots + a_{NN}\cdot X^s_{m-N,n-N} \quad (s = 1, \cdots, B-1) \\ F^0(X^0_{m,n}, X^0_{m,n-1}, \cdots, X^0_{m-N,n-N}) \\ \quad = a_{00}\cdot X^0_{m,n} + a_{01}\cdot X^0_{m,n-1} + \cdots + a_{NN}\cdot X^0_{m-N,n-N} \end{cases}$$

따라서 식(6)의 $F^s(\cdot)$와 $F^0(\cdot)$의 값을 미리 알고 있다면 전체 연산과정은 B-1 번의 덧셈과 자리이동, 그리고 한번의 뺄셈으로 구현할 수 있으며 본 논문에서는 편의상 분산산술방법 1이라고 하겠다. 식(6)을 위한 블록도는 그림 3 과 같으며 $Z_1$과 $Z_2$로 표현되는 지연부, $F^s(\cdot)$와 $F^0(\cdot)$값을 저장하는 메모리부, 덧셈과 자리이동을 할 수 있는 누산기(accumulator)를 포함한 연산부, 그리고 제어부로 구성된다.

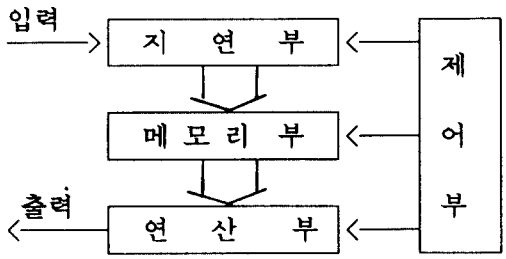



그림 3. 분산산술방법의 블록도
Fig. 3. The block diagram of DAM.

Pelid-Liu 가 제안한 분산산술방법 1은 하나의 출력값을 얻기위해서 B-1 번의 덧셈, 결과값에 대한 누적계산, 그리고 한번의 뺄셈이 필요하므로 전체 B 개의 클럭이 있어야 한다. 그러나 식(6)을 식(7)과 같이 미리 자리이동시킨 계수조합값 $F'^s(\cdot)$를 이용하고 뺄셈을 덧셈처리하기 위하여 미리 2의 보수를 취한 값을 이용하면 누산기대신 다단형 가산기를 이용하여 처리할 수 있으며 본 논문에서는 편의상 분산산술방법 2 라고 하겠다.

$$y_{m,n} = \sum_{s=0}^{B-1} F'^s(\cdot) \tag{7}$$

$$\begin{cases} F'^s(X^s_{m,n}, X^s_{m,n-1}, X^s_{m,n-2}, \cdots, X^s_{m-2,n-2}) = F^s(\cdot)\cdot 2^{-s} \\ F'^0(X^0_{m,n}, X^0_{m,n-1}, X^0_{m,n-2}, \cdots, X^0_{m-2,n-2}) = F^0(\cdot) \end{cases}$$

## Ⅲ. 분산 산술 구현

화상이란 행과 열을 갖는 2차원 데이타로써 한 화상을 처리하기 위해서는 화상의 전체의 내용을 저장하여야 한다. 그러나 그림 2 와 같이 필터차수가 제한되는 경우에는 그림 1의 $Z_1$과 $Z_2$로 표현한 지연소의 화상정보만 필요하며 입력화상의 크기와 필터차수에 따라 결정된다. 256x256 크기의 이진 화상(binary image)인 경우 입력화상의 일부분을 저장할 지연부는 그림 4와 같이 필터 차수와 같은 길이의 행들만 저장할 수 있으면 된다. 그림 4는 NxN 차의 고차 필터인 경우의 지연부로써 N개의 256x1 비트 자리이동레지스터와 (N+1)xN 개의 1 비트 D 플립플롭으로 구성된다. 지연부의 출력선은 메모리의 입력으로 연결되어 계수조합값이 저장된 메모리의 주소를 지정하게 된다. 그러나 화소의 밝기가 B 비트인 화상인 경우에 그림 4는 입력값의 s 번째 비트만을 저장하는 소지연부로 동작되므로 B 비트 단위의 연산처리를 하기위해서는 그림 5와 같이 각 s 번째 비트만을 저장할 수 있는 B 개의 소지연부로 구성해야한다.

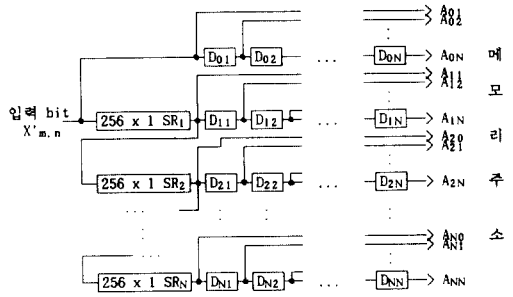



그림 4. 소지연부 블록도 (N x N 차)
Fig. 4. The block diagram of the small delay part.

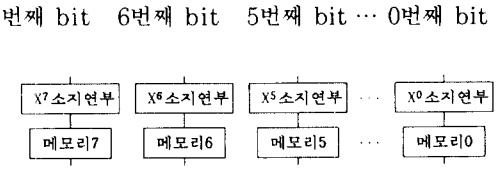



그림 5. 지연부의 블록도 ( B = 8 )
Fig. 5. The block diagram of the delay part.

컨볼류션에 필요한 곱셈을 덧셈만으로 연산하기위해서는 계수조합값을 미리 알고 있어야 한다. 식(6)의 $F^s(\cdot)$ 값은 입력의 s 번째 비트들이 발생될 수 있는 경우의 수에 따라 계산될 수 있는 필터 계수들의 조합값이며 식(7)의 $F'^s(\cdot)$값은 $F^s(\cdot)$값을 S 비트만큼 오른쪽으로 자리이동한 값이다. 이 값들은 미리 계산하여 메모리에 저장하여야 한다. 분산산술방법 1인 경우에는 각 비트에 대한 계수조합값이 모두 동일하므로 하나의 메모리와 멀티플렉싱(multiplexing)기법을 이용한 구조를 가지며 부호비트에 대한 계수조합값 $F^0(\cdot)$ 값은 한번의 뺄셈에 해당하므로 뺄셈을 할 수 있는 구조를 가져야하며, 그러나 분산산술방법 2인 경우에는 모든 계수조합값이 다르므로 그림 5와 같이 B개의 메모리를 가져야 하며 $F^0(\cdot)$값을 $-F^0(\cdot)$값으로 저장함으로써 덧셈처리할 수 있다. 모든 데이타나 연산처리를 B 비트의 정밀도와 NxN 차수의 2차원 필터 계수값을 이용하는 경우 각 비트당 연산 가능한 결과의 총 개수는 필터 계수의 수에 종속하며, (N+1)(N+1)개의 계수값이 존재하므로 분산산술방법 1과 2에서 필요한 메모리는 $2^{(N+1)(N+1)}$바이트와 $Bx2^{(N+1)(N+1)}$ 바이트가 필요하다. 메모리의 값은 (N+1)x(N+1) 개의 입력에 대한 필터계수의 조합값을 가지며, 입력의 S 번째 비트를 저장하는 소지연부의 출력을 메모리의 입력 주소로 사용하면 S 번 자리 이동한 곱셈 결과값을 얻을 수 있다. 따라서 각 메모리의 출력값을 더함으로써 최종적인 출력값을 구할 수 있다. 표.1은 2 x 2 차인 경우 입력값에 따라 발생되는 $F^0(\cdot)$값의 메모리 주소와 필터계수 조합값을 나타낸 것이다.

표 1. 메모리값과 주소 (2 x 2 차)
Table 1. Memory address and data (2 x 2 order).

| 주 소 값 | 메 모 리 값 |
|---|---|
| 000000000 | 0.0 |
| 000000001 | $a_0$ |
| 000000010 | $a_1$ |
| 000000011 | $a_0 + a_1$ |
| 000000100 | $a_2$ |
| 000000101 | $a_0 + a_2$ |
| ... | ... |
| 111111111 | $a_0 + a_1 + a_2 + a_3 + a_4 + a_5 + a_6 + a_7 + a_8$ |

분산 산술방법 1의 연산부는 그림 6과 같은 누산구조를 가지며 누산기의 출력은 자리이동을 위하여 한 비트 오른쪽으로 이동하여 재입력되며, 자리이동시 부호를 유지하기위하여 D7과 D6은 같이 입력된다.

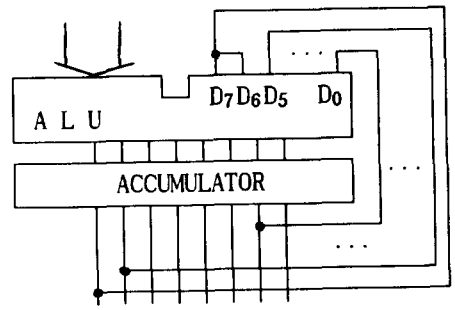



그림 6. 분산산술방법 1의 연산부
Fig. 6. The ALU part of DAM 1.

분산산술방법 2인 경우에는 이미 메모리에 자리이동한 계수조합값과 뺄셈인 경우의 2 의 보수 계수조합값으로 저장되므로 그림 7과 같이 B 비트 가산기로 구성된 단순한 다단형 덧셈장치로 구성할 수 있다. 따라서 하나의 출력은 하나의 클럭 주기동안 발생되며 이 클럭 주기는 지연부의 자리이동시간, 메모리 억세스 시간, 다단 덧셈장치 통과시간을 합한 시간이 되며 각 소자들의 처리시간에 따라 실시간 처리가 가능한다.

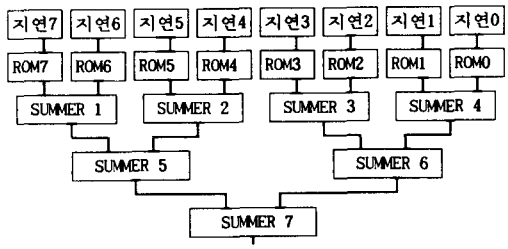



그림 7. 분산산술방법 2의 블록도
Fig. 7. The diagram of DAM 2.

## Ⅳ. 파이프 라인 구현

파이프라인방법이란 반복되는 종속 과정들을 각기 독립적인 과정들로 분리하여 각 독립과정이 각기 다른 시간대에서 동작된다는 점을 이용하여 여러 데이타가 시간차를 두고 동시에 병렬처리함으로써 더 높은 데이타처리율을 얻는 방법을 말하며 이러한 방법은 신호처리에서 많이 이용되고 있으며 필터연산에서도 적용할 수 있다.

분산산술방법 2를 이용하는 경우 연산처리시간은 하나의 클럭주기동안 이루어지며 크게 지연부의 자리이동시간과 메모리 억세스시간, 그리고 가산기들을

통과하는 시간으로 나눌 수 있다. 이 연산처리과정은 표 2 와 같이 메모리를 억세스를 하는 동안에 연산부는 대기상태로 있고, 반면에 메모리에서 출력되는 조합값들이 연산부를 통과하여 연산되는 동안에 메모리 억세스는 대기상태이므로 이 시간대를 파이프라인방법을 이용하여 병렬처리하면 표 2의 파이프라인방법인 경우처럼 입력 1 을 연산하는 동안 메모리부는 입력 2를 처리함으로써 하나의 클럭동안 두개의 출력값을 얻을 수 있어 처리속도를 배로 증가시킬 수가 있다.

그러나 데이타 충돌을 피하기위해서는 현재의 출력값이 연산부를 완전히 통과한 후에 다음 입력값이 지연부로 입력되어 메모리를 지정하여야 한다. 따라서 다음 출력값이 계산될때까지 현재의 출력값이 변하지 않고 보존되야하므로 메모리부와 연산부 사이, 그리고 연산부 다음에 래치를 사용하여 각 장치의 출력값들이 손실되지않도록 저장해야한다. 파이프 라인방법의 제어신호는 그림 8과 같으며 전체 블록도는 그림 9에 나와 있다.

표 2. 분산산술방법과 파이프라인방법의 처리과정
Table 2. The Process of DAM and Pipeline method.

| | | | | | |
|---|---|---|---|---|---|
| 분산산술 2 | 지연부 | 동작 | 대기 | 동작 | … |
| | 메모리부 | 동작 | 대기 | 동작 | … |
| | 연산부 | 대기 | 동작 | 대기 | … |
| 파이프라인 | 지연부 | 동작 | 동작 | 동작 | … |
| | 메모리부 | 입력 1 | 입력 2 | 입력 3 | … |
| | 연산부 | . | 입력 1 | 입력 2 | … |

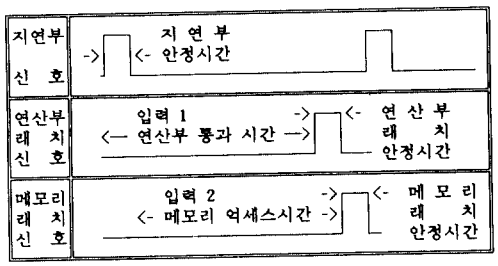



그림 8. 파이프 라인 방법의 제어 신호
Fig. 8. The control signal of Pipeline method.

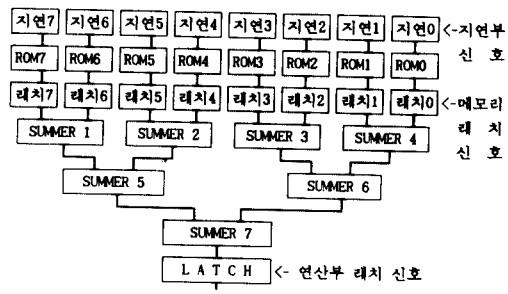



그림 9. 파이프라인방법의 블록도
Fig. 9. The block diagram of Pipeline method.

## Ⅳ. 실험 및 결과

256x256의 해상도를 갖으며 초당 30 매의 화상을 처리하기 위해서는 초당 $1.966 \times 10^6$ 화소를 처리해야 하며 한 화소당 508.6 ns의 시간이 필요하다. 또한 512x512 크기의 화상인 경우에는 초당 $7.864 \times 10^6$ 화소를 처리해야 하므로 한 화소당 127.1 ns의 시간이 필요하다. 본 논문에서는 256x256 크기의 화상을 처리할 수 있는 2x2 차수의 2차원 FIR 디지탈 필터를 설계하여 로직 시뮬레이션 프로그램(Logic Simulation Program)을 이용하여 하드웨어를 시뮬레이션하였으며 이 경우 입, 출력 데이타의 길이와 연산처리 정밀도는 8 비트 길이로 하였다. 그림 7과 그림 9의 구성인 경우 사용한 소자의 종류와 소자 처리시간을 표 3 에 나타내었으며 표 4에 각 구현방법에 따른 최대 출력시간과 데이타 처리속도를 나타내었다.

표 3. 소자의 특성
Table 3. The IC spec.

| 소 자 종 류 | 소자이름 | 처리시간 |
|---|---|---|
| SHIFT REGISTER | TDC1006J | 30 ns |
| D FLIP FLOP | 74S174 | 8 ns |
| MEMORY (PROM) | AM27S191 | 50 ns |
| LATCH | 74S373 | 10 ns |
| ADDER | 74S283 | 15 ns |
| BUFFER | 74S244 | 6 ns |

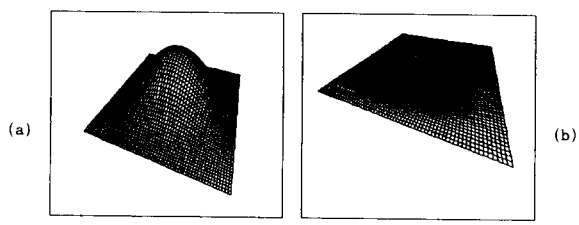
최대 출력시간은 하나의 출력값이 나올때까지의 시간으로써 지연부 통과시간, 메모리억세스시간, 그리고 연산부를 통과하는 시간의 합이 된다. 분산산술방법 1인 경우 최대 출력시간은 30+8x(50+6+15+10)

=678 ns 이며 1.5 MHz의 데이타 처리 속도를 갖으므로 실시간 처리가 어려움을 알 수 있다. 분산산술 방법 2인 경우 최대 출력시간은 (30+50+15x3)=125 ns 가 소요되며 데이타 처리속도는 8 MHz 로써 실시간 처리를 할 수 있다.

표 4. 최대 출력시간과 데이타 처리율
Table 4. The max output time and data process time.

| | 최대출력시간 | 데이타처리율 | 실시간 |
|---|---|---|---|
| 256x256 화상<br>512x512 화상 | 508 ns<br>127 ns | 1.9 MHz<br>7.8 MHz | |
| 분산산술1<br>분산산술2<br>파이프라인 | 678 ns<br>125 ns<br>90 ns | 1.5 MHz<br>8.0 MHz<br>11.1 MHz | no<br>yes<br>yes |

또한 파이프라인 방법을 사용하는 경우 최대 출력시간은 연산부 통과시간과 메모리부 통과시간 중 더 긴쪽이 선택된다.




(a) (b)

(c) (d)

그림 10. 진폭 응답과 필터링화상 (2x2 차)
(a) 저역 통과 필터 (b) 고역 통과 필터
(c) 저역 통과 화상 (d) 고역 통과 화상

Fig 10. The Amplitude response and the filtering image
(a) low-pass filter, (b) high-pass filter,
(c) low-pass image, (d) high-pass image.

연산부 통과시간은 덧셈장치 통과시간과 래치 안정시간을 합한 것으로 (15x3+10)=55 ns이 되며 메모리부 통과시간은 지연부안정시간, 메모리 억세스시간, 그리고 래치 안정시간을 합한 것으로써 (30+50+10)=90 ns 이므로 11.1 MHz 의 데이타 처리 속도를 갖으며 분산산술방법 1과 2보다 더 높은 데이타 처리율을 갖음을 알 수 있다. 가장 긴 처리 시간을 갖는 소자는 메모리 소자와 자리이동 레지스터로써 보다 빠른 소자의 선택에 따라 더 빠른 데이타 처리율을 갖을 수 있다. 그림 10은 2x2 차수를 갖으며 차단주파수가 $\omega_p$=0.4 , $\omega_s$=0.6 인 2차원 저역통과필터와 고역통과필터의 주파수응답과 시뮬레이션한 필터링 화상을 보여주고 있다.

## V. 결 론

본 논문에서는 기존의 1차원 디지탈 필터의 분산산술 구현방법을 이용한 2차원 FIR 디지탈 필터의 하드웨어 구현 방법과 파이프 라인 방법을 이용하여 병렬 처리함으로써 더 높은 데이타 처리율을 갖는 구현 방법을 제안하였다. 고차필터 구성인 경우는 소지연부만 확장하면 되며 메모리의 데이타만 바꾸어줌으로써 다양한 필터를 구성할 수 있는 융통성을 가지고 있다. 또한 실시간처리가 가능하며 단순한 구성을 가지므로 VLSI 구현이 용이하고 화상처리 응용에 적합하다고 사료된다.

## 參 考 文 獻

[1] A.Pelid and B.Liu, "A new hardware realization of digital filters," *IEEE Trans.* ASSP, vol ASSP-22, pp 456-462, Dec. 1974.
[2] F.J.Taylor, "A distributed arithmetic MFIR filter," *IEEE Trans.* ASSP, vol ASSP-32, no.1, pp 186, Feb. 1984.
[3] H.Jaggernauth and A.N.Venetsanopoulos, "Real-time image processing by distributed arithmetic implementation of two-dimensional digital filters," IEEE Trans. ASSP, vol. ASSP-33, no 6. pp 1546-1555, Dec. 1985.
[4] D.E.Dugeon and R.M.Mercereau, Multidimensional Digital Signal Processing, Englewood Cliffs, N.J : Prentice Hall, 1984.

[5] J.Lim, Two-Dimemsional Signal and image processing, Prentice Hall, Edition, 1980.
[6] Texas Instruments,Inc.,The TTL Data Book for Design Engineers,Dallas, TX, 1981.
[7] Advanced Micro Devices,Inc.,MOS/LSI Data Book,Sunnyvale,CA,1980.

## 著 者 紹 介

**尹 馨 泰(正會員)**
1962年 4月 15日生. 1984年 성균관대학교 전자공학과(공학사). 1986年 성균관대학교 대학원(공학석사). 1986年 성균관대학교 대학원(공학박사). 1991年 ~ 현재 인덕전문대학 사무자동화과 조교수. 주관심분야는 이차원 신호처리, 화상처리 등임.

**李 根 泳(正會員)**
1947年 12月 30日生. 1973年 전남대학교 전자공학과(공학사). 1975年 한양대학교 대학원 전자공학과(공학석사). 1978年 한양대학교 대학원 전자공학과(공학박사). 1979年 3月 ~ 1980年 2月 Denmark 공과대학(연구). 1987年 9月 ~ 1988年 8月 영국 Loughborough 대학(연구). 1977年 3月 ~ 1981年 8月 광운공대 조교수. 1981年 9月 ~ 현재 성균관대학교 전자공학과 교수. 주관심분야는 영상처리, 신경회로망, 패턴인식 등임.