論文 88-25-7-16

VLSI 병렬 연산을 위한 여현 변환 알고리듬

(Discrete Cosine Transform Algorithms for the VLSI Parallel Implementation)

趙 南 翊,** 李 商 郁*

(Nam Ik Cho and Sang Uk Lee)

要 約

본 논문에서는 이산 여현 변환 DCT(discrete cosine transform)의 병렬 연산을 위한 VLSI 구현 방법을 두가지 제안하였다. 먼저 DFT(discrete fourier transform)를 직접 systolic 구조로 구현한 구조를 약간 수정함으로써 DCT를 구현 가능하게 하였다. 다른 한가지 방법으로 소인수 DFT 알고리듬을 이용한 소인수 DCT 알고리듬을 제안하고 소인수 DFT 알고리듬의 systolic 구조로 이 알고리듬의 병렬 연산이 가능함을 보였다. 이 방법은 변환할 데이타의 길이가 서로소인 홀수들의 곱으로 분해될 때 가능하나 일반적인 FDCT(fast DCT) 알고리듬들을 병렬 처리 구조 하드웨어로 구성하였을 때보다 곱셈기가 적게 든다는 장점이 있다.

Abstract

In this paper, we propose two different VLSI architectures for the parallel computation of DCT (discrete cosine transform) algorithm. First, it is shown that the DCT algorithm can be implemented on the existing systolic architecture for the DFT (discrete fourier transform) by introducing some modifications. Secondly, a new prime factor DCT algorithm based on the prime factor DFT algorithm is proposed. And it is shown that the proposed algorithm can be implemented in parallel on the systolic architecture for the prime factor DFT. However, proposed algorithm is only applicable to the data length which can be decomposed into relatively prime and odd numbers. It is also found that the proposed systolic architecture requires less multipliers than the structures implementing FDCT (fast DCT) algorithms directly.

I. 서 론

이산 여현 변환 DCT(discrete cosine transform)은 영상 또는 음성 데이타의 감축을 위한 변환 코딩 등에 가장 많이 사용되는 변환으로서 특히 변환하려는 데이타의 상관 관계가 1에 가까우면 그 성능이 매우 우수한 것으로 알려져 있다.[1,2] 그런데 대부분의 변환 알고리듬과 마찬가지로 DCT는 계산량이 많아서 실시간 계산이 어렵다. 이와 같은 문제점을 해결하기 위하여 FDCT(fast discrete cosine transform),[3] Lee 알고리듬[4] 등 여러가지의 고속 DCT

*正會員, **準會員, 서울大學校 制御計測工學科
(Dept. of Control and Instrumentation Eng., Seoul Nat'l Univ.)

接受日字 : 1987年 12月 29日

알고리듬들이 제안되었다. 그러나 영상 데이타와 같은 정보량이 매우 많은 데이타들은 이러한 고속 알고리듬으로도 실시간 처리가 매우 어렵다. 따라서 이와같은 고속 알고리듬을 일반적인 Von-Neumann 형태의 컴퓨터를 이용하여 순차적인 계산을 하는 방법과는 근본적으로 다른 병렬 연산(parallel processing)을 이용하여 DCT를 실시간으로 계산하는 방법에 대한 연구가 필요하다. 병렬 연산은 알고리듬 내에서 서로 독립적으로 계산될 수 있는 부분은 서로 다른 프로세서에서 계산될 수 있도록 하는 것이다. 최근에는 급속히 발달하고 있는 VLSI(very large scale integration) 기술을 이용하여 낮은 비용, 작은 부피와 빠른 속도로 병렬 연산을 수행하는 systolic 구조가 제안되었다.[5,6] 따라서 본 논문의 목적은 DCT를 systolic 구조에 의한 병렬 연산으로 수행하는 알고리듬 및 구조를 제안하는데 있다.

Systolic 구조는 병렬 처리할 알고리듬을 독립적으로 계산되는 간단한 부분들로 분해하고 각 부분의 계산을 수행하는 프로세서들을 구성하여 이들을 규칙적으로 가능하면 하나의 VLSI 칩에 집적시키는 것이다. 이 구조의 특성을 네가지로 요약하면 모듈화(modularity), 규칙성(regularity), 국부 연결성(local connectivity), 그리고 pipelining 이다. 즉 알고리듬을 서로 독립적인 부분들로 나누어 각 부분들이 수행되는 프로세서들을 하나의 칩에 배열 할 때 이들이 가능하면 같은 구조를 갖도록 하고, 또한 각 프로세서들이 규칙적인 연산과 데이타 교환을 하도록 하여 설계시의 복잡성과 비용을 낮출 수 있다. 그리고 VLSI에서는 각 요소 간의 연결에 많은 칩 면적이 필요하므로 VLSI내에 배열된 각 프로세서간의 연결, 즉 데이타 교환은 인접한 프로세서들 간에만 이루어지도록 하고 pipelining이 잘 되도록 하여 칩의 집적도를 높일 뿐만 아니라 계산 속도를 빠르게 할 수 있다. 여러 종류의 신호처리 알고리듬들이 systolic 구조로 구현되고 있으나,[7,8] FDCT, Lee 알고리듬등의 고속 DCT 알고리듬들은 systolic 구현이 매우 어렵다. 그 이유는 이들의 신호 흐름도를 보면 systolic 구조의 조건인 규칙성이나 국부 연결성이 부족하기 때문이다. 그리고 이 알고리듬들은 2^n 포인트의 변환에 대해서만 가능하므로 다른 포인트의 변환에 대한 하드웨어 구현은 비효율적이라는 단점이 있다. 그런데 N 포인트 DCT는 2N 포인트 DFT로 계산이 가능하다.[1] 따라서 N 포인트 DCT를 2N 포인트 FFT(fast fourier transform)를 이용하여 계산할 수 있으나 FFT도 FDCT와 같은 이유로 systolic 구현이 어렵다. 반면에 DFT 자체는 벡터와 벡터의 곱을 계산하는 systolic 구조[7]를 변형하여 수행이 가능하다. 그 한가지 예로서 Bayoumi[9]는 N 포인트 DFT 계산에 N 개 정도의 PE(processing element)를 사용하는 systolic 구조를 제안하였다. 따라서 이러한 DFT의 systolic 구조를 이용하여 N 포인트 DCT를 계산하는 경우에는 약 2N개의 PE가 필요하게 되며 복잡성이 두배로 증가하게 된다. 본 논문의 목적은 N 포인트 DCT를 병렬연산으로 처리할 때 2N 포인트 DFT의 systolic 구조를 이용하는 것 보다 복잡성이 낮은 systolic 구조를 구현하는 것이다. 이를 위하여 FDCT나 FFT 보다는 순차적 계산으로는 계산 속도가 낮더라도 systolic 구현에 적합한 알고리듬과 구조를 제안한다. 그 한가지 방법으로서 Bayoumi[9]의 systolic 구조를 약간 개선하여 N 포인트 DCT를 계산하는데 N 개의 PE 만이 필요한 구조를 제안한다. 다른 한가지 방법으로서 systolic 구현이 가능한 소인수 DFT 알고리듬[10,11]을 이용하여 DCT를 다차원의 작은 DFT로 분해한 후 이를 DFT의 systolic 구조로 계산하는 방법을 제안한다. 이미 DCT를 작은 포인트들의 DCT로 분해하는 Yang[12] 등의 소인수 DCT 알고리듬이 있으나 이 알고리듬에서의 각각의 작은 포인트에 대한 systolic 구현 방법이 없으며 더우기 FDCT 에서와 같은 butterfly stage가 있으므로 systolic 구조로 구현하기는 어렵다.

Ⅱ. DCT의 직접적인 systolic 구현

본 장에서는 고속 알고리듬을 사용하지 않고 DFT를 직접 systolic 구조로 구현하는 방법 중 가장 프로세서의 수가 적은 Bayoumi의 구조[9]를 개선하고 이의 N 포인트 DFT 구조를 이용하여 N 포인트 DCT를 계산하는 방법을 제안한다. 변환할 데이타를 X(n), DFT 출력을 Y(k) 라 할 때 DFT는 식(1)과 같이 정의된다.

$$Y(k) = \sum_{n=0}^{N-1} X(n)\, W_N^{nk} \qquad (1)$$

$$W_N = e^{-j2\pi/N},\ k=0,1,\cdots,N-1$$

그림 1이 5 포인트의 DFT를 계산하는 Bayoumi의 systolic 구조이다. 일반적으로 N 포인트 DFT를 이 구조를 이용하여 계산하려면 그림1(b)와 같은 PE(processing element)를 N−1개 연결하고 데이타를 X(N−1) 부터 X(0) 의 역순서로 입력시켜야 한다. 따라서 구조 자체는 간단하나 데이타를 역순으로 입력시켜야 한다는 단점이 있다. 이 구조에 데이타를 X(0) 부터 순서대로 입력시킬 때의 출력을 A(k) 라 하면 식(2)와 같다.

$$A(k) = \sum_{n=0}^{N-1} X(N-n-1)\, W_N^{nk} \qquad (2)$$

그러므로 DFT의 결과 Y(k) 와 A(k) 간에는 다음과 같은 관계식이 성립한다.

$$A(k) = W_N^{-k}\, Y(k)^* \qquad (3)$$

그런데 그림 1 의 systolic 구조에서는 DFT의 결과와 함께 W_N^k가 출력으로 나오므로 그림 2 에서 보는 바와 같이 그림1(b) 의 PE를 하나 더 연결하여 $W_N^k A(k)$ 가 출력으로 나오도록 하고 이 결과의 허수부의 부호만을 반대로 하면 DFT의 올바른 결과 Y(k) 가 얻어진다. 즉 그림1(b) 의 PE를 그림1(a) 에 한 개만 더 연결함으로써 데이타를 순서대로 입력시킬 수 있고, 따라서 데이타 조작이 간편하게 되는 그림2의 구조를 얻을 수 있다.

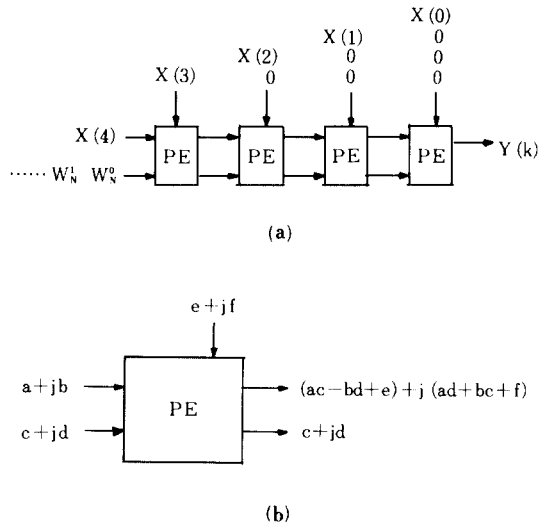


그림 1. Bayoumi의 DFT를 위한 systolic구조
(a) 전체적인 구조
(b) 각 PE의 기능

Fig. 1. Systolic architecture for DFT by Bayoumi.
(a) The architecture.
(b) The function of each PE.

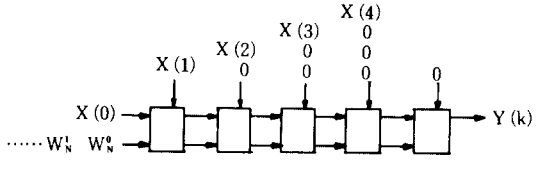


그림 2. 개선된 Bayoumi의 systolic 구조
Fig. 2. Improved systolic architecture for DET.

이와 같이 얻어진 개선된 Bayoumi의 DFT 구조를 이용하여 식(4)의 DCT를 계산하기 위하여 식(5)와 같은 T(k)를 정의하면 식(6)에 의하여 DCT를 얻을 수 있다.

$$Y(k) = c(k) \sum_{n=0}^{N-1} X(n) \cos[(2n+1)k\pi/2N],$$

$$c(k) = \begin{cases} \frac{1}{\sqrt{2}}, & k=0 \\ 1, & k\neq 0 \end{cases} \qquad (4)$$

$$T(k) = \sum_{n=0}^{N-1} X(n)\, U_N^{nk} \qquad (5)$$

$$U_N = e^{j\pi/N},\ k=0,1,\cdots,N-1$$

$$Y(k) = c(k)\ \mathrm{Re}\{e^{jk\pi/2N}\, T(k)\} \qquad (6)$$

그림 2 의 구조에서 $W_N^0, \cdots$대신에 $U_N^0, \cdots$을 입력시켜 얻을 수 있는 출력 B(k) 는 다음과 같다.

$$B(k) = \left[\sum_{n=0}^{N-1} X(N-n-1)\, U_N^{nk} U_N^{k}\right]^*$$
$$= U_N^{-Nk} T(k) \qquad (7)$$

따라서 k가 짝수일 때, 즉 짝수번째 출력 B(k) 는 T(k) 와 같고 홀수번째의 출력들은 T(k) 와 부호만 다르다. 그러므로 식(6)에서 알 수 있는 바와 같이 그림 2 의 구조에 $e^{jk\pi/2N}$ 를 곱하는 프로세서를 연결하고 출력의 부호를 한번씩 바꿈으로써 DCT 변환을 얻을 수 있다. 이 구조는 한 스텝에 필요한 데이타의 입출력 양이 매우 작고 데이타 조작이 매우 간편하다는 장점이 있고 각 PE를 DSP(digital signal processor) 등의 프로세서를 이용하여 구성할 수도 있다. 이와 같은 방법으로 DST(discrete sine transform)[2] 를 얻는 것도 가능하다. 그러나 이에 대한 자세한 토의는 지면상 생략한다.

Ⅲ. 소인수 DCT 알고리듬과 systolic 구조

소인수 DFT 알고리듬은 솟수(prime number)포인트 DFT가 변환할 입력 데이타 X(n)과 W_N^n 과의 circular convolution으로 계산이 가능하다는 Rader의 정리[13] 와 Argawal과 Cooley[14] 의 고속 convolution 알고리듬에 의하여 이 계산을 고속으로 계산할 수 있다는 사실에 근거를 두고 있다. 소인수 DFT나 Winograd DFT[15] 는 변환하려는 데이타의 크기 N이 서로소인 $N_1, N_2, \cdots, N_r$의 곱으로 나타날 때 N 포인트 DFT를 $N_1 \times N_2 \times \cdots N_r$의 r차원 DFT로 분해하여 각 N 포인트의 DFT는 앞에서 설명한 고속 convolution 알고리듬을 이용한 고속 DFT 알고리듬

으로 계산하는 것이다. DFT의 소인수 알고리듬이 가능한 이유는 DFT의 kernel인 W_N가 $W_N^N=1$, $W_N^{a+b}=W_N^a W_N^b$의 성질을 갖고 있기 때문인데 식(4)의 DCT의 식에서 여현 함수는 이와 같은 성질을 갖지 않는다. 따라서 식(5)의 T(k)를 소인수 알고리듬으로 계산하고 식(6)을 이용하여 DCT를 계산한다. 그런데 식(5)의 U_N은 $U_N^{a+b}=U_N^a U_N^b$를 만족하지만 U_N^N이 −1 이므로 N을 주기로 하는 값이 아니라는 문제점이 있다. 따라서 소인수 DCT는 이런 문제점을 해결하여야 하므로 소인수 DFT 알고리듬보다는 약간 복잡한 과정을 거쳐 T(k)의 다차원 변환이 얻어질 수 있는데 이에대한 설명은 다음과 같다.

N이 서로 소인 N_1과 N_2의 곱이라고 할 때 소인수 DFT에서와 같은 방법으로 0부터 N−1 사이의 값인 n과 k를 식(8)로부터 얻을 수 있다. 여기에서 s_1과 s_2는 식 (9)를 만족하는 값이다.

$$n=N_2n_1+N_1n_2 \mod N \tag{8a}$$

$$k=N_2s_1k_1+N_1s_2k_2 \mod N \tag{8b}$$

$$n_1,\ k_1=0,1,\cdots,N_1-1$$

$$n_2,\ k_2=0,1,\cdots,N_2-1$$

$$N_2s_1=1 \mod N \tag{9a}$$

$$N_1s_2=1 \mod N \tag{9b}$$

식 (8)로부터 n과 k는 0에서 N−1사이의 값이지만 우선 편의상 "mod N"는 고려하지 않고, 즉 n과 k가 식 (8)과 같으며 N으로 나눈 나머지의 값이 아니라 그 값 자체라고 가정한다. 이 때 nk는 다음과 같다.

$$nk=N_2N_2n_1k_1s_1+N_1N_2k_1n_2s_1+N_2N_1n_1k_2s_2+N_1N_1k_2n_2s_2 \tag{10}$$

이것을 U_N^{nk}의 nk에 대입하여 보면 식 (11)이 된다.

$$U_N^{nk}=U_{N_1}^{N_2n_1k_1s_1}\cdot e^{j\pi k_1n_2s_1}\cdot e^{j\pi n_1k_2s_2}\cdot U_{N_2}^{N_1k_2n_2s_2} \tag{11}$$

이 식에서 둘째, 세째 항은 1이 되는 경우도 있고 −1이 되는 경우도 있다. 식 (11)에서 첫째와 네째 항만이 남아야 식 (5)의 T(k)의 변환이 2차원 변환이 되므로 식 (11)의 둘째, 세째 항의 곱이 항상 1이 되어야 한다. 따라서 $k_1n_2s_1$과 $n_1k_2s_2$가 항상 짝수가 되도록 하여야 하는데 n_1, n_2, k_1과 k_2는 변수이므로 s_1과 s_2를 짝수로 만드는 수 밖에 없다. 여기서 s_1과 s_2는 식 (9)의 해인데 식 (9)로부터 s_1과 s_2가 모두 짝수이려면 N_1과 N_2가 모두 홀수가 되어야 함을 알 수 있다. 따라서 식(5)의 T(k) 변환중에 N이 서로소인 홀수들의 곱으로 분해될 때에는 이를 다차원의 변환으로 분해할 수가 있다.

또 하나의 문제점은 원래의 U_N^{nk}와 식 (10)에 의한 U_N^{nk}가 서로 부호가 반대가 되는 경우가 있다는 것이다. 식 (10)에서 s_1과 s_2는 모두 짝수이므로 이 경우에 nk는 항상 짝수인데 원래의 nk는 n과 k가 홀수인 경우에는 홀수이므로 이 경우에 U_N^{nk}의 부호가 반대가 되는 것이다. 따라서 식 (10)의 nk를 식 (5)의 nk 대신에 대입한 다차원 변환에서 T(k)의 짝수번째 출력은 올바르게 얻어지지만 홀수번째 출력의 계산에는 홀수번째 입력 데이타에 원래의 U_N^{nk}와는 부호가 반대인 $-U_N^{nk}$가 곱하여지므로 홀수번째의 출력으로는 잘못된 결과가 얻어지게 된다. 따라서 홀수번째의 출력을 계산할 때에는 홀수번째 입력들의 부호를 반대로 하여 입력시켜야 올바른 결과를 얻을 수 있음을 쉽게 알 수 있다. 물론, 이 경우에는 짝수번째 출력에 잘못된 결과가 나타난다. 그러나 이러한 문제점은 원래의 입력 데이타에 대하여 이러한 변환을 수행하여 짝수번째 출력을 얻고 또 한번은 홀수번째 입력 데이타들의 부호를 반대로 하여 홀수번째 출력을 얻음으로써 해결할 수 있다. 이와 같은 두번의 수행은 서로 독립적이므로 병렬 연산으로 처리하면 수행 시간은 증가하지 않는다.

소인수 DCT 변환의 예로 N=3×5, 즉, 15 포인트의 경우를 보면 n, k는 다음과 같다.

$$n=5n_1+3n_2 \mod 15 \tag{12a}$$

$$k=5s_1k_1+3s_2k_2 \mod 15 \tag{12b}$$

$$5s_1=1 \mod 3 \tag{13a}$$

$$3s_2=1 \mod 5 \tag{13b}$$

여기서 식 (13)의 해는 식 (14)와 같은 값들이다.

$$s_1=2,\ 5,\ 8,\ 11,\ 14,\ \cdots \tag{14a}$$

$$s_2=2,\ 7,\ 12,\ 17,\ 22,\ \cdots \tag{14b}$$

따라서 s_1과 s_2를 모두 짝수로 할 수 있다. 식 (14)에서 s_1과 s_2를 모두 2로 선택하면 식 (12b)는 다음과 같다.

$$k=10k_1+6k_2 \mod 15 \tag{15}$$

따라서 nk는 식 (16)과 같고 이를 U_N^{nk}의 nk 대신에 대입하면 식 (17)이 된다.

$$nk=50n_1k_1+30n_1k_2+30n_2k_1+18n_2k_2 \tag{16}$$

$$U_{15}^{nk}=U_3^{10n_1k_1}U_5^{6n_2k_2} \tag{17}$$

따라서 식 (18)의 15 포인트 T(k) 변환은 식 (19)와 같은 3×5의 2차원 변환이 된다.

$$T(k)=\sum_{n=0}^{14} X(n)\ e^{j\pi nk/15} \tag{18}$$

$$T(k_1,k_2)=\sum_{n_2=0}^{4}\left[\sum_{n_1=0}^{2} X(n_1,n_2)\ e^{-j2\pi n_1k_1/3}\right] e^{j6\pi n_2k_2/5} \tag{19}$$

앞의 예로부터 N이 서로 소인 홀수들의 곱일 때 그것이 다차원의 DFT 또는 DFT와 비슷한 변환이 됨을 알았다. 그리고 앞에서 설명한 바와 같이 이와 같은 2차원 변환을 독립적으로 한번은 원래의 데이타에 대하여 수행하여 T(k)의 짝수번째 출력을 얻고, 또 한번은 홀수번째 입력의 부호를 반대로 하고 수행하여 홀수번째 출력을 얻어야 하므로 15 포인트 T(k) 변환의 수행 예는 그림 3과 같이 나타낼 수 있다.

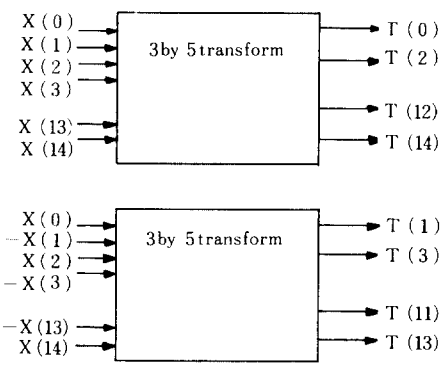


그림 3. 식(19)의 2차원 변환에 의한 식(18)의 15포인트 변환의 예

Fig. 3. The example of 15-opint transformation of eqation (18) by the two-dimensional transform of (19).

식 (19)의 3×5의 2차원 변환에서 3 포인트와 5 포인트의 구체적인 변환예를 살펴보면 3포인트 변환의 경우에는 kernel이 $e^{-j2\pi/3}$이므로 3 포인트 DFT를 사용하면 됨을 알 수 있다. 그런데 5 포인트 변환의 경우를 보면 kernel이 $e^{j6\pi/5}$이므로 이는 $e^{-j4\pi/5}$와 같고 따라서 $e^{-j2\pi/5}$를 W_5라 할 때 이는 W_5^2와 같다. 따라서 5포인트 변환은 식 (20)과 같은 행렬과 벡터의 곱으로 나타낼 수 있는데 원래의 DFT 식은 식 (21)과 같다. 식 (20)과 (21)을 서로 비교하면 원래의 5포인트의 DFT 알고리듬에 X(1) 대신 X(3)을, X(2)에 X(1), X(3)에 X(4), X(4)에 X(2)를 대입함으로써 식 (19)의 5포인트 계산이 가능함을 알 수 있다. 이상에서 알 수 있는 바와 같이 식 (19)의 2차원 변환은 분해된 작은 포인트의 DFT를 앞에서 설명한 systolic 구조 또는 owens[11]등의 병렬연산 구조를 이용하여 수행할 수 있다. Owens[11] 등이 제안한 prime factor DFT의 병렬연산 구조는 엄밀한 systolic 구조의 정의에 의하면 systolic 구조라고 할 수 없다. 그 이유는 이 구조가 systolic 구조의 조건 중에 각 element가 똑같아야 된다는 조건을 만족하지 않기 때문이다. 그러나 실제로는 이 구조에서의 각 element들은 거의 비슷한 기능을 수행하며 VLSI 설계시에도 각각의 차이점이 매우 작다. 따라서 각 element가 똑같지는 않아도 modularity는 매우 높으므로 이를 systolic 구조라 볼 수 있다.

$$\begin{bmatrix}Y(0)\\Y(1)\\Y(2)\\Y(3)\\Y(4)\end{bmatrix}=\begin{bmatrix}1&1&1&1&1\\1&W_5^2&W_5^4&W_5^1&W_5^3\\1&W_5^4&W_5^3&W_5^2&W_5^1\\1&W_5^1&W_5^2&W_5^3&W_5^4\\1&W_5^3&W_5^1&W_5^4&W_5^2\end{bmatrix}\begin{bmatrix}X(0)\\X(1)\\X(2)\\X(3)\\X(4)\end{bmatrix} \tag{20}$$

$$\begin{bmatrix}Y(0)\\Y(1)\\Y(2)\\Y(3)\\Y(4)\end{bmatrix}=\begin{bmatrix}1&1&1&1&1\\1&W_5^1&W_5^2&W_5^3&W_5^4\\1&W_5^2&W_5^4&W_5^1&W_5^3\\1&W_5^3&W_5^1&W_5^4&W_5^2\\1&W_5^4&W_5^3&W_5^2&W_5^1\end{bmatrix}\begin{bmatrix}X(0)\\X(1)\\X(2)\\X(3)\\X(4)\end{bmatrix} \tag{21}$$

식(18)의 T(k)를 식(19)의 2차원 변환을 이용하여 계산하는 소인수 DCT 알고리듬을 정리하면 다음과 같다.

(1) 식 (12)와 같은 방법으로 n과 k를 분해하되 식 (13)의 해인 s_1, s_2, …가 짝수가 되도록 한다.

(2) 식 (16)과 같이 nk를 분해하고 n_1, n_2 포인트의 kernel을 식 (17)의 예에서 본 바와 같이 구한다.

(3) 식 (20), (21)의 예에서와 같은 방법으로 다차원 변환에서의 각 포인트의 kernel을 DFT의 kernel과 비교하여 이것이 다를 경우에는 DFT로 계산이 가능하도록 입력의 순서를 바꾼다.

(4) 입력과 출력을 식 (12)를 이용하여 소인수 DFT 알고리듬에서와 같은 방법으로 다차원 array에 배열한다.

(5) $n_1\times n_2\times\cdots$의 다차원 변환을 수행한다.

(6) 짝수번째 출력을 얻어낸다.

(7) 홀수번째의 입력의 부호를 반대로 하여 (4)-(5)를 수행하고 여기서 홀수번째 출력들을 얻어 낸다.

단 여기서 짝수번째 출력과 홀수번째 출력은 서로 독립적으로 계산할 수 있다.

이 알고리듬의 systolic 구조는 앞에서 설명한 바

와 같이 소인수 DFT의 systolic 구조를 이용하여 가능하므로 식 (19)의 출력은 15포인트 DFT의 출력과 마찬가지로 식 (15)에 의하여 그림 4와 같은 순서로 나타난다. DCT의 결과는 식(6)에서와 같이 T(k)에 $e^{jk\pi/2N}$를 곱하여 얻을 수 있으므로 그림 5에서 보는 바와 같이 PE 들을 배열하고 $e^{jk\pi/30}$ 들을 그림 4의 순서대로 입력시켜 그림 6에서와 같이 각 PE가 입력들의 곱의 실수부를 출력시키도록 하여 DCT의 결과를 얻을 수 있다. 그림 6의 PE가 이 경우와는 달리 입력들의 곱의 허수부를 출력시키도록 하면 DST를 얻는 것도 가능함을 쉽게 알 수 있다.

15-point DFT processor	0			9	3	12	6	0
			4	13	7	1	10	
		14	8	2	11	5		

그림 4. 15포인트 DFT 구조에서의 출력의 순서
Fig. 4. The order of outputs in 15 point DFT structure.

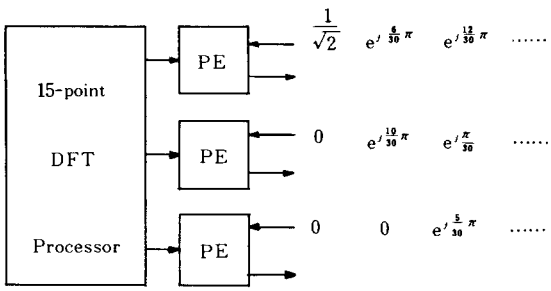


그림 5. 15포인트 DCT를 위한 systolic 구조
Fig. 5. The systolic architecture for 15-point DCT.

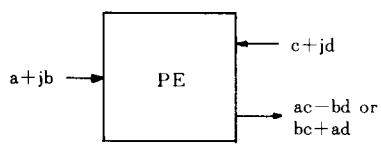


그림 6. 그림 5의 각 PE의 기능
Fig. 6. The function of each PE in Fig. 5.

Ⅳ. 각 구조의 비교

본 장에서는 3장에서 설명한 소인수 DCT의 병렬연산 구조, 2장에서의 DCT의 직접적인 systolic 구조와 FDCT의 병렬 연산 방법간의 비교를 한다. VLSI에서 각 element 간의 배선이 많은 면적을 차지하는데 이것은 직접적인 비교가 어려우므로 생략하고 곱셈기가 덧셈기 보다는 면적을 더 많이 차지하므로 각 알고리듬에 필요한 곱셈기의 수와 외부 프로세서에서와 연결의 용이성 등으로 비교를 한다.

솟수 포인트 DFT의 병렬연산에 필요한 곱셈기의 수는 표 1과 같다. 이 표에서 n 포인트 DFT 계산에 필요한 곱셈기의 수를 f(n) 이라 하자. 예를 들어 f(5)는 4를 말한다. N 포인트 DFT가 $n_1 \times n_2$의 2차원 변환으로 분해되었고 n_1이 n_2보다 크다고 하면 n_1 포인트의 변환을 먼저 하는 것이 유리한데 그 이유는 먼저 수행되는 DFT는 실수 입력에 대한 변환이고 그 후에 수행되는 것은 복소수 입력에 대한 변환이기 때문이다. 따라서 n_1 포인트 DFT 수행에 필요한 곱셈기의 수는 $f(n_1)$ 이고 n_2 포인트 변환에 필요한 곱셈기의 수는 $2f(n_2)$가 된다. 그리고 Ⅲ장에서 밝혔듯이 이러한 수행을 독립적으로 두번 수행하여야 하므로 식 (5)의 T(k) 를 얻는데 총 $2f(n_1)+4f(n_2)$ 개의 곱셈기가 필요하다. 그리고 그림 6으로부터 알 수 있듯이 T(k)를 이용하여 DCT 출력을 얻으려면 $4n_2$ 개의 곱셈기가 더 필요하므로 결국 DCT의 최종 출력을 얻는데 필요한 곱셈기의 수는 $2f(n_1)+4f(n_2)+4n_2$ 가 된다. Ⅱ장에서 설명한 DCT의 직접적인 systolic 구조에는 각 PE 마다 4개씩의 곱셈기가 필요하므로 결국 4N 개의 곱셈기가 필요하다.

각 방법의 병렬 연산 구조에 필요한 곱셈기의 수를 몇 개의 포인트에 대하여 비교한 것이 표 2 이다.

여기서 "DCT"는 Ⅱ장에서 설명한 DCT의 직접적인 systolic 구현을 말하며 "소인수 DCT"는 Ⅲ장의 소인수 DCT를, "FDCT"는 Chen-Smith[3] 의 FDCT 알고리듬을 의미한다. 그리고 괄호 안의 수는 이 알고리듬보다 곱셈의 수가 작은 Lee 알고리듬[4]

표 1. 소인수 DFT에 필요한 곱셈수
Table 1. The number of multpliers required for the prime point DFT.

포 인 트	곱 셈 수
3	1
5	4
7	8
9	8

표 2. 각 알고리듬의 병렬처리에 필요한 곱셈기 수

Table 2. The number of multipliers required for the paralled implementation. of DCT algorithm.

	DCT	소인수 DCT	FDCT (Lee)
7	28	30	
8	32		16 (12)
9	36	34	
15	60	24	
16	64		44 (32)
21	84	32	
32	128		116 (80)
35	140	52	
45	180	52	
63	252	76	
64	256		292 (192)

을 의미한다. 여기에서 FDCT 나 Lee 알고리듬의 경우는 이의 신호흐름도 전체를 하드웨어로 구성한다는 가정하에 이 알고리듬에 필요한 곱셈수를 나타낸 것이고 소인수 DCT 의 경우에는 앞에서 설명한 식으로 계산한 것이며 DCT 의 경우는 하나의 PE에 필요한 곱셈기를 4 개로 한 것이다.

포인트 수가 10 이하일 때는 소인수 DCT 의 구조가 FDCT 의 병렬연산 구조나 Lee 알고리듬의 경우보다 보다 불리하지만 포인트 수가 커질수록 곱셈기가 크게 줄어든다는 것을 알 수 있다. 또한 직접적인 DCT 의 구조는 곱셈기의 수가 포인트 수에 비례하여 늘어나므로 처음에는 FDCT 보다 곱셈기가 많으나 포인트가 증가할수록 그 수는 줄어든다. 그러나 소인수 DCT 의 구조보다는 많게 됨을 알 수 있다.

각 병렬처리 구조에서 수행 속도에 영향을 미치는 throughput 과 외부 프로세서와의 데이타 교환량을 나타내는 I/O bandwidth 를 비교하여 보면 소인수 DCT 의 경우가 직접적인 DCT 보다 이 값들이 모두 크므로 계산 속도는 소인수 DCT 가 더 빠르지만 데이타 조작은 직접적인 DCT 가 더 간편함을 알 수 있다. FDCT 는 전체를 systolic 구조로 구현한 예가 없으므로 본 논문에서 제안한 방법들과의 throughput 이나 I/O bandwidth 에 대한 비교는 하지 않았다.

V. 결 론

본 논문에서는 VLSI 병렬 연산을 위한 DCT 알고리듬과 그 systolic 구조에 대한 연구를 하였다. 한가지 방법으로서 N 포인트 DFT 의 systolic 구조를 변형하여 N 포인트 DCT 를 계산하는 방법을 연구하였다. 이 구조는 외부와의 데이타 교환이 매우 간단하다는 것이 장점이다. 다른 한가지 방법으로 소인수 DCT 알고리듬을 개발하여 이를 systolic 구조로 구현하는 것이다. 소인수 DCT 알고리듬은 N 이 서로소인 홀수들의 곱으로 분해될 때 DCT 가 여러개의 작은 포인트의 DFT 로 분해되어 이들 각각을 DFT 의 systolic 구조를 이용하여 수행함으로써 병렬연산이 가능하다.

본 논문에서 제안한 소인수 DCT, 직접적인 DCT 의 병렬 처리 구조와 Chen-Smith 의 FDCT 알고리듬을 직접 병렬 처리 구조로 한 경우를 비교하여 보았는데 곱셈기의 숫자를 비교하면 작은 포인트의 경우에는 소인수 DCT 구조나 직접적인 DCT 의 병렬연산 구조에 필요한 곱셈기가 FDCT 의 구조에 필요한 곱셈기보다 많지만 포인트가 증가할수록 소인수 DCT 에 필요한 곱셈기의 수는 상대적으로 적어지므로 소인수 DCT 가 가장 유리하다는 것을 알 수 있었다. 이들의 병렬처리 구조를 VLSI 로 구현할 때 여기에서 소개한 소인수 DCT 나 직접적인 DCT 의 병렬 연산 구조는 systolic 구현이 가능하므로 systolic 구현이 거의 불가능한 FDCT 의 병렬 처리 구조보다 VLSI 면적, 제작의 용이성 등에서 유리하다.

본 논문에서 연구한 내용은 1 차원 데이타에 대한 DCT 에 관한 것이었다. 따라서 본 논문에서 제안한 구조를 이용하여 음성등의 1 차원 데이타는 간편하게 DCT 변환을 할 수 있다. 그러나 2 차원 데이타인 영상 데이타를 1 차원 DCT 알고리듬을 이용하여 DCT 변환을 수행하는 경우에는 데이타 조작이 상당히 불편하게 된다. 이러한 문제점을 해결하기 위하여 최근에는 2 차원 DCT 자체에 대한 고속 알고리듬이 연구되고 있다.[16,17] 그리고 지금까지는 이에 대한 활발한 연구는 없지만 앞으로 2 차원 DCT 에 대한 systolic 구조도 반드시 연구되어야 할 것이다.

參 考 文 獻

[1] N. Ahmed, T. Natarajan, K.R. Rao, "Discrete cosine transform," *IEEE Trans, on Commun.*, vol. COM-23, pp. 90-93, Jan. 1974.

[2] R.J. Clarke, *Transform Coding of Images*, Academic Press, 1985.

[3] W.H. Chen, C.H. Smith, S.C. Fralick, "A fast computational algorithm for discrete

cosine transform," *IEEE Trans. on Communications,* vol. COM-25, pp. 1004-1009, Nov. 1977.

[4] B.G. Lee, "A new algorithm to compute the discrete cosine transform," *IEEE Trans. on ASSP.,* vol. ASSP-32, pp. 1243-1245, Dec. 1984.

[5] H.T. Kung, "Why systolic architecture?" *IEEE Computer,* pp. 37-46, Jan. 1982.

[6] S.Y. Kung, "VLSI array processors," pp. 4-22, July, 1985.

[7] S.Y. Kung, et. al., Ed., *VLSI and Modern Signal Processing* Prentice Hall, 1985.

[8] P.R. Cappello, et. al., Ed., *VLSI Signal Processing, IEEE Press,* 1984.

[9] M.A. Bayoumi, G.A. Jullien, W.C. Miller, "A VLSI array for computing the DFT based on RNS," ICASSP 86, Tokyo, pp. 2147-2150.

[10] D.P. Kolba, T.W. Parks, "A prime factor algorithm using high speed convolution," *IEEE Trans. on ASSP,* vol. ASSP-25, pp. 281-294, Aug. 1977.

[11] R.M. Owens, J. Ja'ja', "A VLSI chip for the Winograd/Prime factor algorithm to compute the discrete Fourier transform," *IEEE Trans. on ASSP,* vol. ASSP-34, pp. 979-989, Aug. 1986.

[12] P.N. Yang, M.J. Narashimha, B.G. Lee, "A prime factor decompsotion algorithm for the computation of the discrete cosine transform," International Conf. on Computers, Systems and Signal Processing, Dec. 10-14, 1984.

[13] C.M. Rader, "Discrete Fourier transforms when the number of the data samples is prime," *Proc. IEEE,* vol. 56, pp. 1107-1108, June, 1968.

[14] R.C. Argawal, J.W. Cooley, "New algorithms for digital convolution," *IEEE Trans. on ASSP,* vol. ASSP-25, pp. 392-410, Oct. 1977.

[15] S. Winograd, "On computing the discrete Fourier transform," MATHEMATICS OF Computation, vol. 32, pp. 175-199, Jan., 1978.

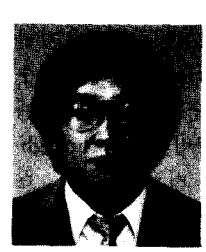
[16] F.A. Kamangar, K.R. Rao, "Fast algorithms for the 2-D discrete cosine transform," *IEEE Trans. on Computers,* vol. C-32, pp. 899-906, Sep. 1982.

[17] M.A. Haque, "A two-dimensional fast cosine transform," *IEEE Trans. on ASSP,* vol. ASSP-33, pp. 1532-1549, Dec. 1985.*

著 者 紹 介

趙 南 翊(準會員)

1964年 1月 19日生. 1986年 2月 서울대학교 제어계측공학과 졸업 (B. S) 현재 서울대학교 제어계측공학과 석사과정 재학중. 주관심분야는 디지탈 신호처리등임.



李 商 郁(正會員)

1949年 8月 11日生. 1973年 2月 서울대학교 전기공학과 졸업. 1980年 1月 University of Southern California 전기공학과 졸업(Ph.D) 현재 서울대학교 제어계측 공학과 부교수 주관심분야는 디지탈 신호처리등임.