2D DCT/IDCT의 행, 열 주소생성기를 위한 파이프라인 구조 설계

노진수+, 박종태++, 문규성+, 성해경+++, 0이강현+
+조선대학교 전자정보통신공학부 MSI Lab.
++ 춘해대학
+++한양여자대학
http://multimedia.chosun.ac.kr
khrhee@chosun.ac.kr

Design on Pipeline Architecture for the Low and Column Address Generator of 2D DCT/IDCT

Jin Soo NOH+, Jong Tae PARK++, Gyu Sung MOON+,Hae Kyung SEONG, and 0Kang Hyeon Rhee+
+School of Electronics, Info-Commumication Eng. Chosun University
++Choonhae College
+++Hanyang Women's College

ABSTRACT

This paper presents the pipeline architecture for the low and column address generator of 2D DCT/IDCT(Discrete Cosine Transform/Inverse Discrete Cosine Transform). For the real time process of image data, it is required that high speed operation and small size hardware.

In the proposed architecture, the area of hardware is reduced by using the DA(distributed arithmetic) method and applying the concepts of pipeline on the parallel architecture.

As a results, the designed pipeline of the low and column address generator for 2D DCT/IDCT architecture is implemented with an efficiency and high speed compared as the non-pipeline architecture. And the operation speed is improved about 50% up. The design for the proposed pipeline architecture of DCT/IDCT is coded using VHDL.

1. 서 론

영상압축을 위한 효율적인 기법 중에서도 1974년에 발표된 DCT(discrete cosine transform)[1,2]는 영상압축 효과가 다른 영상변환 방법보다 우수하여 국제 표준화 그룹의 영상압축 기본 알고리즘으로 채택되어 신호처리 분야에 널리 응용되고 있으며, 이의 역변환으로서의 IDCT (Inverse Discrete Cosine Transform)도 영상복원에 많이 사용되는 기법으로 JPEG[3]이나 MPEG[4][5] 등에서 널리 사용되고 있다.

본 논문에서는 고속의 DCT/IDCT 소자를 설계하기 위하여, 곱셈기 대신 ROM 테이블을 사용하는 chen 알고리즘에 의한 분산연산 (DA : distributed arithmetic)[6][7] 방법과 전치 메모리의 억세스를 위하여 행렬분리(row column decomposition)[8] 계산방법을 이용한 파이프라인 구조의 2D DCT/IDCT의 행, 열 주소생성기를 설계한다.

제안된 설계를 위한 행,열 주소생성기의 동작은 행,열 주소생성의 분리계산을 위하여 행 단위로 1D DCT/IDCT를 행한 다음, 이 결과를 열 단위로 1D DCT/IDCT를 행하여 2D DCT/IDCT를 구성한다. 1D와 1D 사이에 Row start와 Column start 블럭을 파이프라이닝 구조로 설계하여 DCT/IDCT의 동작 속도를 향상시킨다.

2. DCT/IDCT의 이론적 배경

DCT 알고리즘은 영상 압축을 위한 효과적인 코딩기술로서 영상 데이터를 트랜스폼-영역으로 변환한 것이고, IDCT는 처리된 트랜스폼-영역의 데이터를 영상 데이터로 변환시킨 것이다.

2차원 DCT/IDCT의 일반식은 식(1,2)과 같다 [9,10].

$$F(u,v)=\frac{2}{N}C(u)C(v)\sum_{u=0}^{N-1}\sum_{v=0}^{N-1}f(x,y)\cos\frac{(2x+1)u\pi}{2N}\cos\frac{(2y+1)v\pi}{2N} \quad \cdots\cdots\cdots (1)$$

$$f(x,y)=\frac{2}{N}\sum_{u=0}^{N-1}\sum_{v=0}^{N-1}C(u)C(v)F(u,v)\cos\frac{(2x+1)u\pi}{2N}\cos\frac{(2y+1)v\pi}{2N} \quad \cdots\cdots\cdots (2)$$

, where u, v, x, y = 0, 1, · · · , N-1

$\sqrt{2}$, u,v = 0

C(u), C(v) = 1 , otherwise 0

여기서 u,v는 변환 영역의 좌표이고, x,y는 화소 영역의 공간 좌표이다

2D DCT를 구현하는 방법은 크게 두가지로 구분할 수 있다. 첫째, 2D DCT의 분해 특성을 이용하는 행열분해 기법으로서, 2개의 1D DCT와 행렬 전치를 통한 RCA(row column algorithm)방법이 있다. 이 경우는 2D 영상에 대해 행(열)방향으로 1D DCT/ IDCT를 수행한 후, 그 결과를 전치 메모리에 의해 위치를 바꾼 다음 다시 열(행)방향으로 1D DCT/IDCT를 수행함으로써 2D DCT/IDCT를 유도할 수 있다. 이 방법은 1D DCT/IDCT 구조로부터 2D DCT/IDCT를 연산할 수 있으므로 구조가 간단해지는 장점이 있고, 분산연산과 결합하여 사용함으로써 고속화가 가능하다. 둘째는 분해 특성을 이용하지 않고 2D DCT의 계산식으로 부터 직접 고속 알고리즘을 유도하여 2D DCT를 구현하는 방법의 NRCA(no row column algorithm) 방법이 있다.

분산연산 방식은 DCT와 같이 입력 데이터 값과 고정된 계수와의 내적 연산을 수행하는데 매우 효율적인 방법으로 곱셈기를 사용하지 않고 레지스터와 누적기만으로 곱셈연산을 수행하므로 구조가 간단하고, 규칙적이며 하드웨어의 크기를 감소시키는 장점을 가지고 있다. 최근에는 분산 연산 방식이 DCT를 하드웨어로 구현하는데 더 효율적임이 인정되고 있으며 분산연산에 근거한 DCT 구조에 관한 많은 연구결과가 발표되어있다.

1차원 DCT/IDCT의 일반식은 식(3,4)와 같다.

$$F(k) = 1/4\ C(K)\sum_{m=0}^{N-1}f(m)\cos[(2m+1)K\pi / 16] \quad \cdots\cdots\cdots (3)$$

$$f(m) = 1/4\ C(K)\sum_{m=0}^{N-1}F(k)\cos[(2m+1)K\pi / 16] \quad \cdots\cdots\cdots (4)$$

$$c(K)=\begin{pmatrix} 1/\sqrt{2},\ K=0 \\ 1,\ C=1,2,\ldots,7 \end{pmatrix}$$

위에서 표현한 1차원 DCT를 일반적인 행렬식으로 나타내면 식 (5)과 같다.

$$\begin{bmatrix} Xe \\ Xo \end{bmatrix} = \begin{bmatrix} C_{N/2} & C_{N/2} \\ S_{N/2} & -S_{N/2} \end{bmatrix} \begin{bmatrix} x_f \\ x_r \end{bmatrix}$$

$$\begin{bmatrix} x_f \\ x_r \end{bmatrix} = \begin{bmatrix} C^t_{N/2} & S_{N/2} \\ C^t_{N/2} & -S_{N/2} \end{bmatrix} \begin{bmatrix} Xe \\ Xo \end{bmatrix} \quad \cdots\cdots\cdots (5)$$

X_e : 짝수항을 갖는 N/2 포인트 트랜스폼 계수 백터

X_o : 홀수항을 갖는 N/2 포인트 트랜스폼 계수 백터

x_f : x의 앞부분 반절을 갖는 N/2-포인트 DCT 행렬

x_r : x의 뒷부분 반절(역방향으로)을 갖는 N/2-포인트 DCT 행렬

$S_{N/2}$: N/2 × N/2 대칭행렬

$C_{N/2}$: N/2-포인트 DCT행렬

그리고 2차원 DCT/ IDCT는 열(또는 행)방향으로 1차원 DCT/IDCT를 한 뒤 행렬을 치환하고 다시 이를 행 (또는 열)방향으로 1차원 DCT/IDCT를 행하여 얻을 수 있다. Chen의 알고리듬은 cosine의 주기 특성을 이용하여 불필요한 중복 계산을 줄인 것으로 식 (6,7)로 표현된다.

$$\begin{bmatrix} X0 \\ X2 \\ X4 \\ X6 \end{bmatrix} = \begin{bmatrix} A & A & A & A \\ B & C & -C & -B \\ A & -A & -A & A \\ C & -B & B & -C \end{bmatrix} \begin{bmatrix} x0+x7 \\ x1+x6 \\ x2+x5 \\ x3+x4 \end{bmatrix}$$

$$\begin{bmatrix} X1 \\ X3 \\ X5 \\ X7 \end{bmatrix} = \begin{bmatrix} D & E & F & G \\ E & -G & -D & -F \\ F & -D & G & E \\ G & -F & E & -D \end{bmatrix} \begin{bmatrix} x0-x7 \\ x1-x6 \\ x2-x5 \\ x3-x4 \end{bmatrix} \quad \cdots\cdots\cdots (6)$$

$$\begin{bmatrix} x0 \\ x1 \\ x2 \\ x3 \end{bmatrix} = \begin{bmatrix} A & B & A & C \\ A & C & -A & -B \\ A & -C & -A & B \\ A & -B & A & -C \end{bmatrix} \begin{bmatrix} X0 \\ X2 \\ X4 \\ X6 \end{bmatrix} + \begin{bmatrix} D & E & F & G \\ E & -G & -D & -F \\ F & -D & G & E \\ G & -F & E & -D \end{bmatrix} \begin{bmatrix} X1 \\ X3 \\ X5 \\ X7 \end{bmatrix}$$

$$\begin{bmatrix} x7 \\ x6 \\ x5 \\ x4 \end{bmatrix} = \begin{bmatrix} A & B & A & C \\ A & C & -A & -B \\ A & -C & -A & B \\ A & -B & A & -C \end{bmatrix} \begin{bmatrix} X0 \\ X2 \\ X4 \\ X6 \end{bmatrix} - \begin{bmatrix} D & E & F & G \\ E & -G & -D & -F \\ F & -D & G & E \\ G & -F & E & -D \end{bmatrix} \begin{bmatrix} X1 \\ X3 \\ X5 \\ X7 \end{bmatrix} \quad \cdots\cdots\cdots (7)$$

이 식을 계산한 후 행렬을 바꾼 후에 다시 같은 계산을 하면 2차원 DCT/IDCT의 결과가 나온다. 식 (6,7)에서 곱셈의 수가 줄어든 것을 볼 수 있고 같은 행렬을 한 번은 더하고 한 번은 뺀 것을 알 수 있다. 따라서 곱셈의 수가 줄어든다.

3. DCT/IDCT의 구현

2차원 DCT/IDCT는 1차원 DCT/IDCT를 수행하고 이 결과를 전치 RAM(transpose RAM)에 저장한다. 그리고 행과 열을 바꾸어서 엑세스하여 다시 1차원 DCT/IDCT의 연산을 수행하면 그 결과가 2차원 DCT/IDCT의 연산 결과가 된다. 2D DCT/IDCT 구조는 그림 1과 같다.

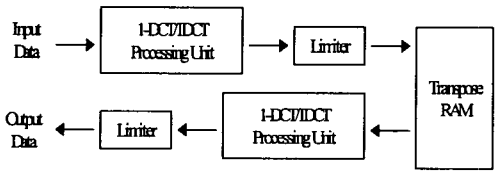


그림 1. 2D DCT/IDCT의 구조

그림 1에서 1D DCT/IDCT는 승산기를 사용하는 대신 ROM 테이블에 저장된 데이터를 이용해서 분산연산 방식을 구현한다. 식(8,9)의 각 행 A_kX_k는 내적이므로 분산연산을 적용할 수 있다.[6]

$$y = \sum_{k=1}^{K} A_k X_k \quad \cdots\cdots\cdots \quad (8)$$

식 (8)에서 X_k를 2의 보수로 표현하면 식 (9)과 같다.

$$X_k = -b_{k0} + \sum_{n=1}^{N-1} B_{kn} 2^{-n} \quad \cdots\cdots \quad (9)$$

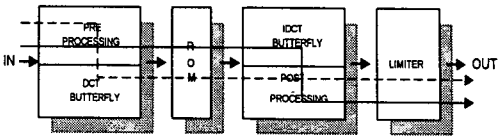
$$y = \sum_{k=1}^{K} A_k[-bk0 + \sum_{n=1}^{N-1} 2^{-n}]$$

$$y = \sum_{n=1}^{N-1}[\sum_{k=1}^{K} A_k b_{kn}]2^{-n} + \sum_{k=1}^{K} A_k(-b_{k0})$$

$$\cdots\cdots\cdots \quad (10)$$

식 (10)에서, b_{kn}은 0 또는 1의 값이므로 우변의 첫 항은 2^k개의 조합이 가능하다. 그리고 우변 둘째 항의 b_{k0}는 부호 비트(MSB)이므로 모두 2×2^k개의 값을 기억하고 있으면 y를 계산할 수 있다. k비트는 2×2^K개의 값을 기억하고 있는 ROM 테이블의 주소가 되고, N-1번의 쉬프트와 순차적인 합으로 계산이 완료된다. 그리고 여기서 부호 비트를 위하여 ROM 테이블의 크기가 2배가 되었는데, MSB가 주소가 될 때 가산 대신 감산을 하면 ROM의 크기는 2^k으로 줄어든다. 1D DCT/IDCT의 흐름도는 그림 2와 같다.

그림 2에서 입력 데이터는 전처리기의 레지스터를 통하여 병렬로 인가되고, 이 데이터는 DCT 연산 동안에 입력 열의 정렬을 위해 버터플라이 단을 거치게 되고 IDCT 연산 동안에는 통과하게 된다. 이 단계를 거친 데이터는 ROM 테이블을 거친 후, DCT는 버터플라이 단을 통과 한 이후, 후처리기를 통과하게 되고 IDCT는 버터플라이 단을 거친 이후 후처리기를 통과하게 된다. 그리고 나서 다시 제한기(limiter)를 통과하여 16비트를 출력한다.



DCT 흐름도 ---
IDCT 흐름도 ——

그림 2. 1D DCT/IDCT 블록도

4. 파이프라이닝 기법

파이프라인은 명령어 실행이 분업화되어 하나의 프로세스를 서로 다른 기능을 가진 여러 개의 서브 세그멘트로 나누어 각각의 단계에 있는 세그멘트가 동시에 서로 다른 데이터를 취급하도록 하는 기법이다. 파이프라인이 아닌 경우 하나의 명령이 수행되려면 각 단계를 거쳐 명령 수행이 끝날 동안 모든 과정이 기다리게 된다. 파이프라인 기법이 사용되면 각각의 단계는 자신의 일만 처리하는 명령어 실행이 계속된다.

1차원 DCT/IDCT는 분산 연산을 통해서 구현되는데, 분산 연산은 곱셈기를 사용하는 대신 ROM에 저장된 데이터를 이용해서 구현하는 방식이다. 분산 연산은 내적을 계산하는데 유용한 방법이다. k비트가 ROM의 주소가 되어 ROM의 내용이 출력되어 덧셈기로 인가되어 전 상태의 값과 더하여진다. 그리고 한 비트가 쉬프트 되어 다음의 ROM 출력값과 더하여 진다. 이 과정을 반복하다가 MSB가 ROM의 주소가 되면 덧셈기가 뺄셈을 수행하여 최종 결과를 출력한다.

본 논문에서도 2D를 행 방향으로 처리하는 동안 열 방향의 처리가 동시에 이루어 질 수 있도록 2D 구조를 각각 1D와 1D로 나누어 하나의 1D에서는 행 방향으로 처리를 하는 동안에 다른 1D에서는 열 방향을 처리할 수 있도록 행과 열을 시작할 수 있는 블록을 구성하여 속도의 향상을 가질 수 있도록 한다. 1D가 만들어진 이후 ROW을 start할 수 있는 블럭과 Column을 start 할 수 있는 블록을 만들어 1D에서 Row가 행해진 상태가 '1'이면 Row을 start할 수 있는 신호를 제공하고, '0'이면 Column을 start할 수 있는 신호를 생성하는 파이프라인 구조는 Row 상태가 "001000"일 때 다른 1D에서 Column을 start가 되고, Go 신호가 '1'이고

event 상태일 때 Row start 한다. 파이프라인을 적용한 DCT/IDCT의 블록도는 그림3과 같고, DCT/IDCT의 데이터 흐름도는 그림 4와 같다.

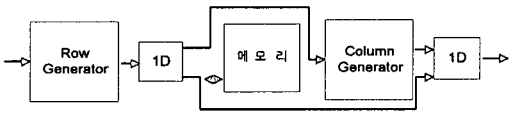


그림 3. 파이프라인 구조의 DCT/IDCT의 블럭도

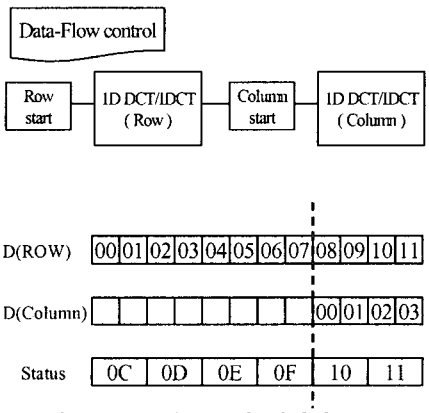


그림 4. DCT/IDCT의 데이터 흐름도

2-D DCT/IDCT 전체 블록과 합성된 결과는 그림 5, 6과 같다. 그리고 기존의 1D와 파이프라인을 사용한 2D의 특성비교는 표 1과 같다.

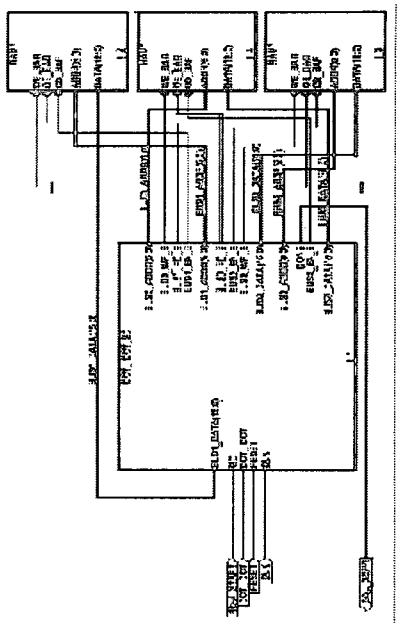
표 1. 기존 1D와 파이프라인을 사용한 2D의 특성비교

	8X8 블록의 행해진 시간 (ns)
Pipeline 을 사용하지 않은 1D	10495(행) + 10495(열)
Pipeline 을 사용한 2D	11936 (행, 열)

5. 결 론

본 논문에서는 고속의 DCT/IDCT를 위하여 고속의 가산을 수행하는 분산연산 기법과 영상정보처리의 속도 향상을 위하여 2D DCT/IDCT의 행, 열 주소생성기를 2D를 1D로 분해한 파이프라인 개념으로 처리하였다. 그리고 연산은 누산기 대신에 새로운 가산기를 설계하여 분산연산 방법을 적용하여 전체 시스템의 연산 속도 성능을 약 50% 정도 향상시킬 수 있었다.

그림 5. 2-D DCT/IDCT 최종 블럭



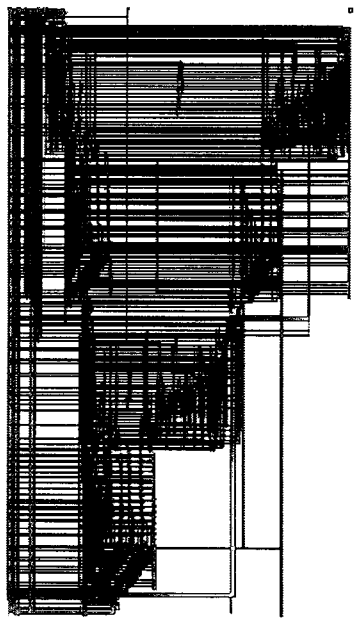


그림 6. 2-D DCT/IDCT 합성 결과

참 고 문 헌

[1]. 박종태, "고속 DCT/IDCT 소자 설계" 조선대학 교 석사학위논문, 1998.2

[2]. N.Ahmed, T. Natarajan, and K.R.Rao, "Discrete Cosine Transform," IEEE trans. comput., vol. C-23, PP. 90-93, Jan 1974.

[3]. J.H.Lee, D.W.Kang, "Frequency Dependent

(Runlength, Level) Coding of DCT Coefficients," 전자 공학회 논문지 94-31B-11-9 pp.1673-1682.

[4]. N.T Cho, S.U. Lee, "DCT Algorithms for VLSI Parallel Implementation," IEEE Trans. Acoust. Speech, Signal Processing, vol. ASSP38, pp.121-127, no.1,Jan. 1990.

[5]. B.G.Lee. "A New algorithm To Compute the Discrete Cosine Transfer", IEEE Trans. ASSP, Vol. ASSP-32,no.6, pp.1243-1245, 1984.(ASSP: Acoustics, Speech and Signal Processing)

[6]. S.A.White, "Applications of Distributed Arithmetic to Digital Signal Processing: a Tutorial Review", IEEE ASSP Magazine, Vol.6, No.3,pp.4-19, July 1989

[8]. M. Sun, T. Chen, and A. M. Gottlieb, "VLSI implementation of a 16x16 discrete cosine transform," IEEE Trans. Circuits and Systems, Vol. 36, No. 4, pp.610-617, Apr. 1989.

[7]. J.S.Choi, "The Structure of 2-D DCT/IDCT for Real-Time Video Applications", 한국 통신학회 논문지, '95-11, Vol. 20, No.11, pp.3219-3233.

[9]. 민경욱, 서기범, 김기현, 정정화, "효율적인 파이프라인 구조를 갖는 2-D IDCT 하드웨어설계, DSP 설계 및 응용 워크숍 논문집, 95

[10]. Y.F. Jang, et. al., "A 100-MHz 2-D DCT Core Processor," IEEE Trans, on Consummer Electronics. vol. 40, no. 3. 1994. 8