論文94-31B-3-11

고속 DCT 알고리듬을 이용한 DCT 및 IDCT 구조

(An Architecture for the DCT and IDCT using a Fast DCT Algorithm)

李 勝 旭*, 任 綱 彬*, 鄭 華 子**, 鄭 己 鉉***, 金 容 得***

(Seung Wook Lee, Kang Bin Yim, Hwa Ja chung,
Gi Hyun Jung and Yong Deak Kim)

要 約

본 논문은 고속 DCT 알고리듬[11]을 이용한 이산 코사인 변환(DCT : Discret Cosine Transform) 및 IDCT(Inverse DCT) 처리를 위한 VLSI 구조 구현에 관한 것이다. 본 논문에서 제시한 VLSI 구조에 적용된 고속 DCT 알고리듬은 기존의 고속 DCT 알고리듬과는 달리 DCT 및 IDCT 의 연산 과정에서 처리 속도에 직접적으로 영향을 미치는 곱셈 연산을 쉬프트 및 덧셈 연산으로 대체함으로써 곱셈 연산으로 인한 문제를 근본적으로 해결한 비가역적 DCT 알고리듬이다. 제안한 VLSI 구조는 모듈성, 규칙성 및 멀티프로세싱 능력을 가지고 있다. 본 논문에서 제시한 VLSI 구조의 성능을 시뮬레이션 소프트웨어인 "Compass"를 이용하여 설계 및 검증하였다. 이러한 시뮬레이션의 결과는 압축 및 복원 영상의 화질, 처리 속도, VLSI 구조면에서 우수성을 입증하였다.

Abstract

This paper proposes an implementation of DCT (Discrete Cosine Transform) and IDCT (Inverse DCT) using a fast DCT algorithm with shift and addition operations instead of multiplications [11] Based on the proposed algorithm, a new VLSI architecture for the DCT and the IDCT is proposed. It shows modularity, regularity and capability for multiprocessing. Its performance is also simulated by a simulation software, "Compass". The results of the simulation provide the quality of decompression images, the increase in processing speed, representing the superiority of the proposed architecture.

Ⅰ. 서 론

G4 FAX, HDTV(High Definition TV), 영상 전화, 레이저 인쇄기, 전자식 정지 영상 카메라, 동화상 카메라, VCR 등 영상 신호를 다루는 기기들은 다량의 정보를 처리하여야 하며 큰 전송 대역폭이 요구된다. 또한 최근 주목을 받고 있는 ISDN(종합정보통신망:Integrated Service Digital Network)으로 영상 신호를 전송하기 위해서는 높은 데이터의 압축이 요구된다. 영상 데이터 압축을 위해서는 일반적으로 예측 부호화 기법, 변환 부호화 기법이 주로 사용되고 있으며, 이들의 변환이나 조합으로 성능 향상을

* 準會員, *** 正會員, 亞洲大學校 電子工學科
(Dept.of Elec. Eng., Ajou Univ.)
** 正會員, 서울産業大學 電子計算學科
(Dept.of Compter Science, S.N.P.U)
接受日字 : 1993年 7月 1日

시도하고 있다.

영상 데이터 압축 기법의 하나인 DCT(이산 코사인 변환: Discrete Cosine Transfom)는 N×N 영상 신호들을 적은 수의 영상 신호에 정보를 집중시키고 나머지 신호들은 '0' 또는 작은 값을 갖도록 처리한다. 이러한 특성은 통신 선로의 대역폭 감소 문제 및 정보 저장에 필요한 기억용량의 감축 문제를 동시에 해결하기 때문에 디지탈 영상 신호처리 사용시 발생하는 경제성 문제를 해결하는 핵심 기술의 하나로 연구되고 있으며, JPEG (Joint Photographic Expert Group)[9], MPEG (Motion Picture Expert Group)[17] 등에서 국제 표준 규격으로 권고한 영상 압축 알고리듬들 중 DCT를 기반으로 하는 알고리듬을 제시하였다.(그림 1, 2 참조) 그러나 이 영상 압축 기법은 많은 계산량으로 인해 실시간 영상 데이터 처리에 어려움이 있다.

많은 고속 DCT 알고리듬들[1-6]은 계산량을 줄이기 위해 변환 행렬을 분해하여 상당수의 계수를 1 또는 0으로 대치시켰다. 따라서 이들 알고리듬들을 적용하는 경우 곱셈 횟수는 감소하고 코사인 항의 종류도 줄어든다. 또한, 이의 하드웨어 구현을 위한 다양한 구조들[6]-[8]이 제안되어 왔다. 이와 같은 고속 DCT 알고리듬들은 대부분 1차원 DCT를 위한 것인데 일반적으로 2차원 신호인 영상 신호에 대한 2차원 DCT의 계산에는 N×N 입력신호에 대하여 행(열) 방향으로 1차원 DCT를 수행한 후에 다시 열(행) 방향으로 1차원 DCT를 수행하는 행렬 분해 기법(Row-column method)[7]이 많이 사용된다. 즉, N×N 의 2차원 DCT 변환을 위하여 2N개의 1차원 DCT를 순차적으로 수행하는 것이다. 한편, 이 방법 외에도 2차원 DCT를 빠르게 계산하기 위하여 2차원 입력 데이타를 직접 다루는 2차원 고속 DCT 알고리듬들과 이의 하드웨어 구현을 위한 구조들도 발표되었다.[7]

기존의 DCT 처리의 VLSI 구현에 관한 연구는 입력된 영상 신호 처리에 있어서 ROM과 RAM등 메모리 소자들을 이용하여 변환 행렬 값을 Look-up table로 만들고 이를 이용하여 고속 DCT 알고리즘에 적용한 구조[10], 시스톨릭 어레이 구조를 응용한 구조[11]-[13] 그리고, 분산 연산을 이용한 구조[8] 등이 발표되었다.

한편, HDTV 수상기, VCR 등의 압축된 영상의 복원만을 필요로 하는 기기는 영상 복원을 위한 전용 회로를 요구한다. 만약, 이 기기들에 입력된 영상이 영상 데이타 압축 기법의 하나인 DCT로 압축된 영상이라면 원화상으로 복원하기위해서는 IDCT를 거쳐야 한다. 특히, MPEG등에서 권고한 동영상의 영상 압축 알고리듬의 움직임 보상과 추정을 위한 하드웨어 구현을 위해서는 고속의 IDCT 처리가 요구된다.

이와 같이 본 논문은 영상 압축 및 복원을 위해 요구되는 DCT 및 IDCT를 VLSI로 구현하기 위하여 논문 [14] 에서 제시한 고속 DCT 알고리듬을 이용한 DCT 및 IDCT 구조 구현에 관한 것으로 Row-column 행렬 분해 기법을 이용하여 2차원 DCT 및 IDCT 처리하였다. 제시한 하드웨어 구조는 "Compass"을 이용하여 HDL로 설계하고, 성능을 평가하였다.

본 논문의 순서는 다음과 같다. 먼저, Ⅱ장에서는

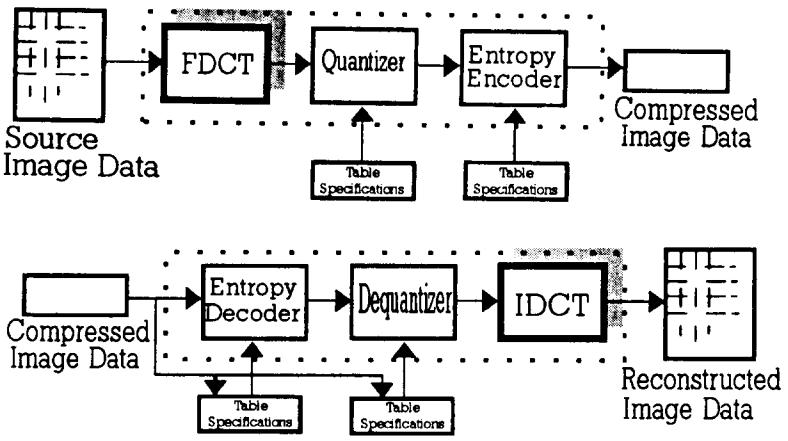


그림 1. JPEG baseline 알고리듬 블럭도[9].
Fig. 1. Block diagram of JPEG baseline algorithm.

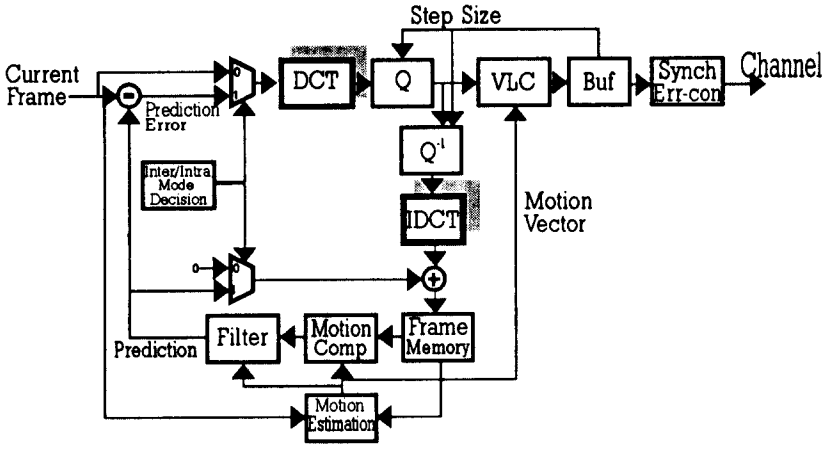


그림 2. MPEG의 움직움 추정 및 보상 알고리듬 블럭도 [17]
Fig. 2. Block diagram of Motion estimation and compensation algorithm in MPEG.

논문 [14] 에서 제안된 알고리듬을 적용한 DCT 및 IDCT 하드웨어 구조에 관하여 설명하고 Ⅲ장에서는 Ⅱ장에서 설명한 하드웨어 구조에 대한 시뮬레이션 결과에 관하여 설명하며, 마지막으로 Ⅳ장에서는 결론을 내린다.

Ⅱ. DCT 및 IDCT 하드웨어 구조

DCT 및 IDCT 하드웨어 구현을 위하여 고속 DCT 알고리듬 [14] 을 이용하였다. 먼저, 2차원 N차 DCT 및 IDCT를 수식으로 표현하면 식(1)과 식(2)와 같다.

$$T_{uv} = \frac{2}{N} C_u C_v \sum_{x=0}^{N-1}\sum_{y=0}^{N-1} F_{xy} \cos\frac{(2x+1)u}{2N}\pi \cos\frac{(2y+1)v}{2N}\pi \quad (1)$$

단, $C_u, C_v = \frac{1}{\sqrt{2}}$; $u,v = 0$

$C_u, C_v = 1$; $u,v \neq 0$

$$G_{uv} = \frac{2}{N} \sum_{u=0}^{N-1}\sum_{v=0}^{N-1} C_u C_v T_{uv} \cos\frac{(2x+1)u}{2N}\pi \cos\frac{(2y+1)v}{2N}\pi \quad (2)$$

위의 DCT 정의식을 [14] 에서 제시한 고속 DCT 알고리듬으로 표현하면 식(3)과 식(4)와 같다. 편의상 $\sqrt{2}/N$ 및 $2C_uC_v/N$의 상수 부분 (scale factor)을 생략하였다(설계시에 보상하였다).

$$T_{uv} = \sum_{x=0}^{N-1}\sum_{y=0}^{N-1} F_{xy} \cos\frac{(2x+1)u}{2N}\pi \cos\frac{(2y+1)v}{2N}\pi \quad (3)$$

$$T_{uv} = \sum_{i=0}^{N-1}\left(\sum_{j=0}^{M-1} b_j \times RS\left(\sum_{x=0}^{N-1}\left(\sum_{i=0}^{M-1} a_i \times RS(F_{xy})_i\right)\right)_j\right) \quad (4)$$

단, a_i는 해당 코사인 계수의 2진수 표현에 i번째 비트 코드이며, b_j도 해당 코사인 계수의 2진수 표현에 j번째 비트 코드이다. $RS(F_{xy})_i$는 $F_{xy} \times \cos(i\pi/2N)$의 근사 함수이다.

[14] 에서 제시한 고속 DCT 알고리듬은 기존의 고속 DCT 알고리듬과는 달리 DCT 및 IDCT 의 연산과정에서 처리 속도에 직접적으로 영향을 미치는 곱셈 연산을 쉬프트 및 덧셈 연산으로 대체함으로써 곱셈 연산으로 인한 문제를 근본적으로 해결하였다.

[14] 에서 제시된 고속 DCT 알고리듬을 간단히 설명하면 다음과 같다. 입력된 영상 데이타를 하나의 레지스터에 저장하고, 이 데이타를 1비트씩 오른쪽으로 수차례에 걸쳐 쉬프트하여 각각의 레지스터에 저장한다. 이 입력된 영상 데이타와 곱셈 연산을 하는 코사인 계수를 2진 코드로 표시할 경우 이 코드의 '1' 인 비트에 대응되는 레지스터 내에 저장된 값만을 선택하여 더해줌으로서 영상 데이타와 코사인 계수와의 곱셈 연산이 쉬프트와 덧셈 연산으로 대치되는 것이다.

식(4)를 이용하여 DCT 연산을 하였을 경우 약 0.4%의 오차를 갖는다.(표1 참조) 또한, 식(4)를 이용한 소프트웨어 시뮬레이션 결과(그림3, 그림4 참조)를 참조로 하고 하드웨어 구현의 용이성을 고려하여, 1비트씩 오른쪽으로 쉬프트된 영상 데이타를 저장할 쉬프트 레지스터의 수(이하 M이라 한다)를 8로, 오른쪽 쉬프트에 의해 절사되는 비트로 발생하는 계산상의 오차를 줄이기 위하여 입력된 압축 영상 값을 왼쪽으로 레벨 쉬프트 해 주는 비트 수(이하 SL 이라 한다)를 4로 하였다.

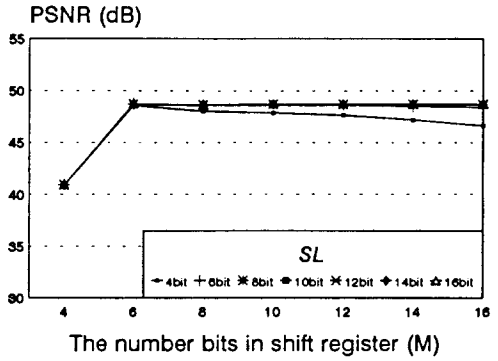


그림 3. M 및 SL에 따른 PSNR [14]

Fig. 3. PSNR for different values of M and SL.

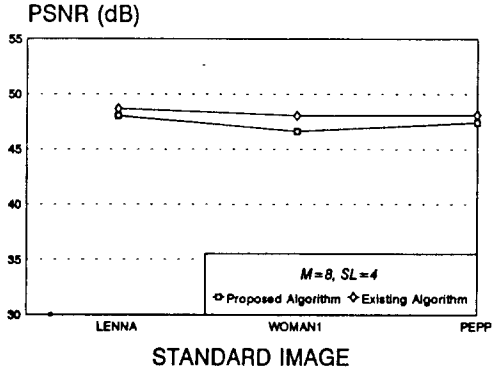


그림 4. M=8, SL=4 일 때 각 표준영상들의 평균 PSNR [18]

Fig. 4. Average PSNR for standard images with M=8, SL=4.

표 1. N=8, M=8 일 때 코사인 값 및 2진 코드 [18]

Table 1. Cosine values and their binary codes with N=8, M=8.

n		cos(nπ/16)	M 값 8
1	2진수		0.1111110
	10진값	0.9807852804032	0.984375
2	2진수		0.1110110
	10진값	0.9238795325113	0.921875
3	2진수		0.1101010
	10진값	0.8314696123025	0.828125
4	2진수		0.1011011
	10진값	0.7071067811865	0.7109375
5	2진수		0.1000111
	10진값	0.5555702330196	0.5546875
6	2진수		0.0110001
	10진값	0.3826834323651	0.3828125
7	2진수		0.0011001
	10진값	0.1950903220161	0.1953125

이 알고리듬을 적용한 전체적인 8×8 2차원 DCT 하드웨어 구조가 본 논문에서 제시되었으며, 이 하드웨어 구조는 Row-column 접근 방식 [7] 으로 구현되었다. 이의 구성도는 그림 5와 같다.

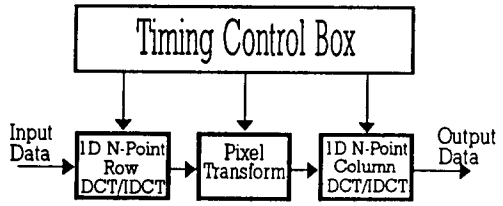


그림 5. 2차원 DCT row-column 접근 방식의 블럭도 [7]

Fig. 5. Block diagram for the row-column approach for 2D DCT.

N×N 2차원 입력 영상 신호 행렬을 [X] , 변환 행렬을 [C] 라 할 때 2차원 DCT를 일반적으로 행렬을 사용하여 표현하면 식(5)와 같다.

$$Y = C \cdot X \cdot C^T \tag{5}$$

단, X : N×N 데이터 행렬

C^T : C의 전치 행렬

여기서, 행렬 C는 다음과 같이 나타낼 수 있는 N×N 행렬이다.

$$C = [\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} \cos(\frac{(2x+1)u}{2N})\pi \cos(\frac{(2y+1)v}{2N})\pi]$$

식(5)의 계산을 위한 일반적인 방식의 행렬 기법은 먼저 각 행(열)을 따라 1차원 DCT를 수행하고 행렬 전치 후에 다시 이 결과를 열 방향으로 1차원 DCT를 수행하는 것이다. 즉, 행방향의 1차원 DCT는 식(6)과 같이 정의하고 이를 행렬로 표현하면 식(7)과 같이 정의된다. 식(7)에서 보는 바와 같이 변환 계수 C_0를 C_4로 대치하였다. 이는 하드웨어 설계시 레지스터를 줄이고 DCT 처리시 처리속도를 고려하여 C_u나 C_v의 값이 $1/\sqrt{2}$이 될 경우에 이 값을 DCT 변환 행렬에 같이 포함시키기 위하여 $\cos(4\pi/16)$로 대치하였다.

$$G = X \cdot C^T \tag{6}$$

$$[G]=[X]\begin{bmatrix} C_4 & C_1 & C_2 & C_3 & C_4 & C_5 & C_6 & C_7 \\ C_4 & C_3 & C_6 & C_7 & C_4 & C_1 & C_2 & C_5 \\ C_4 & C_5 & C_6 & C_1 & C_4 & C_7 & C_2 & C_3 \\ C_4 & C_7 & C_2 & C_5 & C_4 & C_3 & C_6 & C_1 \\ C_4 & C_7 & C_2 & C_5 & C_4 & C_3 & C_6 & C_1 \\ C_4 & C_5 & C_6 & C_1 & C_4 & C_7 & C_2 & C_3 \\ C_4 & C_3 & C_6 & C_7 & C_4 & C_1 & C_2 & C_5 \\ C_4 & C_1 & C_2 & C_3 & C_4 & C_5 & C_6 & C_7 \end{bmatrix} \quad (7)$$

단, X : 입력된 8 × 8 영상 신호

G : 1차원 DCT 처리 후 출력된 8 × 8 영상 신호

C_n : 직교행렬의 코사인 계수($C_n = \cos(n\pi/16)$)

그러면, 2차원 DCT 행렬 [Y] 는 행렬 [G] 를 다시 열 방향으로 1차원 DCT를 행하여 얻어지는데 이 과정을 식으로 나타내면 식(8)과 같다.

$$Y = C \cdot G \quad (8)$$

또한, N×N 2차원 IDCT를 일반적으로 행렬을 사용하여 표현하면 식(9)와 같다. 즉, 위에서 소개된 식(5)을 역으로 행렬 연산하면 된다.

$$X = C^T \cdot Y \cdot C \quad (9)$$

식(9)의 행렬 연산은 각 행(열)을 따라 1차원 IDCT를 수행하고 행렬 전치 후에 다시 이 결과를 열 방향으로 1차원 IDCT를 수행하는 일반적인 행렬 방식을 이용할 수 있으며 이를 수식으로 나타내면 식(10)과 식(11)로 나타낼 수 있다.

$$G = Y \cdot C \quad (10)$$

$$X = C^T \cdot G \quad (11)$$

이와 같은 Row-column 접근 방식을 적용하여 2차원 DCT 및 IDCT 하드웨어를 구성하였으며 세부적인 하드웨어 구성은 다음과 같다.

1. DCT 하드웨어 구조

1) 한 화소 처리를 위한 하드웨어 구조

8×8 2차원 DCT 처리를 위한 구조에서 한 화소 처리를 위한 하드웨어 구조는 그림 6과 같은 모듈로 구성된다. 그림 6의 하드웨어 구조는 식(7)에서 보는 바와 같이 변환 행렬 계수의 부호를 고려하지 않을 경우, 입력된 영상 신호 X와 $\cos(i\pi/16)$ (i = 1, 2, 3,…,7)의 7가지의 곱셈 연산을 처리하는 구조이다. 이는 입력된 한 화소 값에 대한 8개의 변환 계수와의 곱셈이 동시에 이루어지므로 행렬 곱셈 속도를 개선시킬 수 있다.

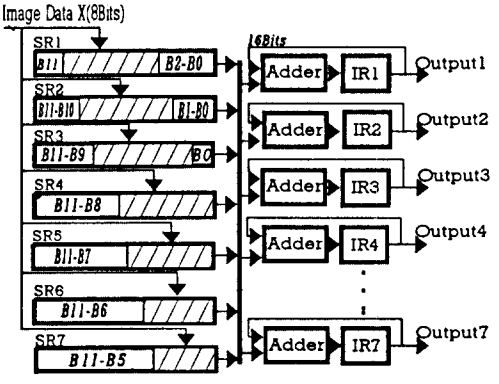


그림 6. 한 화소 처리를 위한 하드웨어 구조

Fig. 6. Hardware architecture for processing a pixel.

이의 하드웨어 동작을 설명하면 다음과 같다. 식(3)에서 $\cos(0\pi/16)$를 표1에서와 같이 8비트 2진 코드로 표시하면 '1.0000000B'이다. 2^0의 자리의 '1'에 대응되는 값은 입력된 영상 데이타를 계산상의 오차를 줄이기 위하여 왼쪽으로 쉬프트해주는 수 SL값 만큼 왼쪽으로만 쉬프트한 후 레지스터에 저장된 값이다. 그러나, 변환 행렬 계수 $\cos(0\pi/16)$가 $\cos(4\pi/16)$로 대치됨으로서 논문 [14] 에서 제시된 알고리듬에서 SL값 만큼 왼쪽으로만 쉬프트한 값을 저장하는 레지스터는 필요하지 않다. 즉, 쉬프트 레지스터의 수 M을 8로 정하여 하드웨어를 구성하여도 7개의 레지스터만을 필요로 한다.

그림 6에서 입력된 영상 신호 X가 7가지 변환 계수와의 곱셈 연산이 이루어진다. 입력된 영상 신호 X와 변환 계수와의 곱셈 연산을 통한 출력 값을 고려하여 그림 6을 수식으로 표현하면 아래와 같다.

$$Output\ N = X \times \cos(N\pi/16) \quad (12)$$

단, X : 입력된 영상 신호

N : 1, 2, 3,…, 7

SL값 만큼 입력된 영상 값을 왼쪽으로 쉬프트한 다음 SRn(n=1,2,3,…,7) 레지스터의 n에 해당하는 것 만큼 오른쪽으로 1비트씩 쉬프트하여 각각의 레지스터에 저장되며, 이때 IRn(n=1,2,3,…,7) 레지스터

들은 '0' 값으로 초기화 된다. 먼저, SR1이 인에이블(enable) 되어 저장된 데이터를 내보내면, 입력된 영상 신호 X와 변환 행렬 계수들과의 곱셈 연산에 있어서 이 값을 필요로 하는 곳에서만 즉, 변환 계수들을 2진 코드로 표현할 경우 2^1의 자리가 '1'인 IR_1, IR_2, IR_3, IR_4, IR_5 레지스터 값과 덧셈 연산이 이루어지고 이 연산 결과값은 다시 IR_1, IR_2, IR_3, IR_4, IR_5 레지스터에 저장된다.(표1참조) 다음에 SR2가 인에이블 되어 저장한 값을 내보내면, 이 값을 필요로 하는 곳 즉, IR_1, IR_2, IR_3, IR_6 레지스터 값과 덧셈 연산이 이루어지고 이 연산 결과값은 다시 IR_1, IR_2, IR_3, IR_6 레지스터에 저장되어 진다.(표1참조) 이와 같은 방식으로 한 화소에 대하여 7가지의 변환 행렬 계수와의 곱셈이 동시에 이루어지게 된다.

또한, 행방향의 1차원 DCT 결과 행렬 [G] 에 대한 열방향의 1차원 DCT에서도 한 화소에 대한 하드웨어 구조는 위에서 소개한 한가지 모듈로써 표현이 가능하다. 이는 본 논문에서 소개된 하드웨어 구조가 모듈성을 가지고 있으며 또한, 여러가지 변환 계수와의 곱셈 연산을 동시에 수행하는 특징을 가지고 있음을 보여주는 것이다.

2) 8×1 1차원 DCT 처리를 위한 하드웨어 구조

8×1 1차원 DCT 처리를 위한 하드웨어 구조에 관하여 간단히 설명하기 위하여 식(8)를 식(13)과 같이 나타내어 입력 화소 행렬의 한 열에 대한 하드웨어 구조를 설명하고자 한다. 이의 하드웨어 구조는 그림 7과 같이 구성되어지며, 이 구조의 동작은 다음과 같다.

먼저, PP0에서는 한 화소에 대하여 전치 변환 행렬의 열방향으로 각 DCT 변환 계수에 대하여 곱셈 연산이 이루어진다. 즉, 식(13)에서 입력 화소 값 X_0는 변환 행렬 계수 C_1, C_2, C_3, C_4, C_5, C_6, C_7 에 대한 7가지의 곱셈 연산이 이루어지며, 이의 하드웨어 구조는 그림 6의 모듈과 같다. 나머지 입력 화소에 대한 변환 계수와의 곱셈 연산은 앞서 설명한 그림6의 하드웨어 구조를 이용하여 연산이 가능하다. 이와 같이, 각각의 PPN(N = 0, 1, 2, ···, 7)에서 변환 행렬 계수에 대한 64개 연산 결과값이 얻어질 수 있다. 이 값들 중 변환 행렬 계수의 부호에 맞게 A&S 연산기(그림 7.참조)에서 8개씩 덧셈 및 뺄셈 연산이 이루어진다.

이 연산 결과 값은 계산상의 오차를 줄이기 위하여 왼쪽으로 SL값만큼 쉬프트된 후에 계산되어진 값이므로 SL만큼 오른쪽으로 쉬프트 되어진 값을 취한다. 이 결과값들이 8×1 1차원 DCT 처리된 연산 결과 행렬 [G] 의 행렬 요소이다.

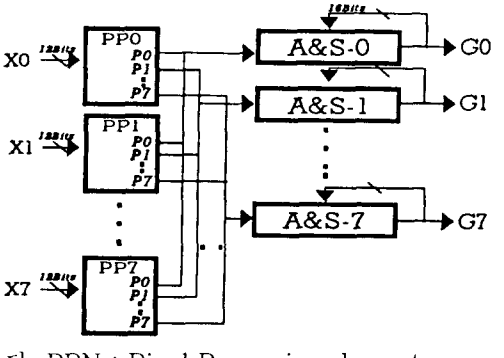
$$\begin{bmatrix} G_0 \\ G_1 \\ G_2 \\ G_3 \\ G_4 \\ G_5 \\ G_6 \\ G_7 \end{bmatrix} = \begin{bmatrix} C_4 & C_4 & C_4 & C_4 & C_4 & C_4 & C_4 & C_4 \\ C_1 & C_3 & C_5 & C_7 & C_7 & C_5 & C_3 & C_1 \\ C_2 & C_6 & C_6 & C_2 & C_2 & C_6 & C_6 & C_2 \\ C_3 & C_7 & C_1 & C_5 & C_5 & C_1 & C_7 & C_3 \\ C_4 & C_4 & C_4 & C_4 & C_4 & C_4 & C_4 & C_4 \\ C_5 & C_1 & C_7 & C_3 & C_3 & C_7 & C_1 & C_5 \\ C_6 & C_2 & C_2 & C_6 & C_6 & C_2 & C_2 & C_6 \\ C_7 & C_5 & C_3 & C_1 & C_1 & C_3 & C_5 & C_7 \end{bmatrix} \begin{bmatrix} X_0 \\ X_1 \\ X_2 \\ X_3 \\ X_4 \\ X_5 \\ X_6 \\ X_7 \end{bmatrix} \quad (13)$$

단, X_n : 입력된 영상 신호
G_n : DCT 처리 후 출력된 영상 신호
C_n : 변환행렬의 코사인 계수
$(C_n = \cos(n\pi/16))$

3) 8×8 2차원 DCT 하드웨어 구조

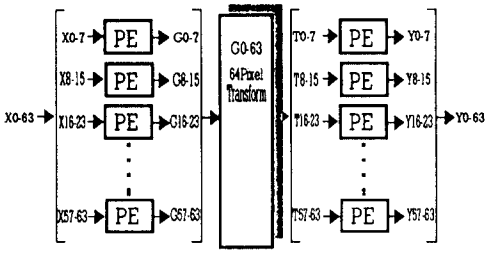
8×8 2차원 DCT 하드웨어 구조는 그림 8과 같이 구성되어지며, 이의 동작은 다음과 같다. 먼저, 입력 화소 값 $X_{0\text{-}64}$이 입력 되어지고 이 값들은 각각의 PE (Processing element)에서 행방향으로 1차원 DCT 처리되어진다. 여기서 PE의 구조는 그림 7과 같은 구조를 갖는다. 행방향으로 1차원 DCT 처리되어진 연산 결과 값 $G_{0\text{-}64}$는 다음 열방향의 1차원 DCT 처리를 위하여 각각 8개씩 행렬의 열방향으로 모아져 다시 다음 단계의 PE로 입력되어지고 이 곳에서 열방향으로 1차원 DCT 처리되어 얻어진 결과값 $Y_{0\text{-}64}$는 8×8 2차원 DCT 처리되어진 결과값과 동일하다.

또한, 이 하드웨어 구조는 각각의 PE의 구조를 8×8 2차원 IDCT 처리를 위한 하드웨어 구조로 변형하면 8×8 2차원 IDCT 처리가 가능하다. 본 논문에서 제시된 8×8 2차원 IDCT 하드웨어 구조도 이를 이용하여 구성하였다.(차후의 8×8 2차원 IDCT 하드웨어 구조는 생략하겠다.)



단, PPN : Pixel Processing element
A&S-N : Adder & Subtractor
N : 0, 1, 2, 3 ···, 7

그림 7. 8×1 1차원 DCT 처리를 위한 하드웨어 구조
Fig. 7. Hardware architecture for 8×1 1D DCT.



단, PE : Processing elemnet

그림 8. Row-column 접근 방식을 이용한 8×8 2차원 DCT 하드웨어 구성도

Fig. 8. Hardware configuration for 8×8 2D DCT using Row-column approach.

2. IDCT 하드웨어 구조

1) 한 화소 처리를 위한 하드웨어 구조

8×8 2차원 IDCT 처리를 위한 구조에서 한 화소 처리를 위한 하드웨어 구조는 그림 9, 10, 11과 같이 세가지 모듈로 구성되어진다. 그림 9의 하드웨어 구조는 식(10)에서 보는 바와 같이 입력된 영상 신호 X_{N0}(N=0,1,2,3,…,7:행번호)과 변환 행렬에서 첫번째 행의 계수들인 $\cos(4\pi\ 16)$와 곱셈 연산을 처리하는 구조이다. 그림 10의 구조는 입력된 영상 신호 X_{N1}(N=0,1,2,3,…,7:행번호)과 변환 계수 행렬에서 부호를 고려하지 않을 경우에 두번째 행의 계수들인 $\cos(1\pi\ /16)$, $\cos(3\pi\ /16)$, $\cos(5\pi\ /16)$, $\cos(7\pi\ /16)$와의 곱셈 연산을 처리하는 구조이며, 그림 11의 구조는 입력된 영상신호 X_{N2}(N=0,1,2,3,…,7:행번호)와 변환 계수 행렬에서 세번째 행의 계수들인 $\cos(2\pi\ /16)$, $\cos(6\pi\ /16)$와의 곱셈 연산을 처리하는 구조이다. 변환 계수 행렬의 나머지 행들의 연산은 동일한 방법으로 위에서 소개된 모듈들을 이용하여 처리할 수 있다. 그림 9, 10, 11에서 소개된 모듈들의 하드웨어 동작은 앞서 설명한 그림 6의 하드웨어와 동일하게 동작한다.

2) 8×1 1차원 IDCT 처리를 위한 하드웨어 구조

8×1 1차원 IDCT 처리를 위한 하드웨어 구조에 관하여 간단히 설명하기 위하여 식(11)를 식(14)와 같이 나타내어 입력 화소 행렬의 한 열에 대한 하드웨어 구조를 설명하고자 한다. 이의 하드웨어 구조는 그림 7과 같이 구성되며, 이 구조의 동작은 다음과 같다.

먼저, PP0에서는 한 화소에 대하여 전치 변환 행렬의 열방향으로 각 IDCT 변환 계수에 대하여 곱셈 연산이 이루어진다. 즉, 식(14)에서 입력 화소 값 X0는 변환 행렬 계수 C4에 대한 곱셈 연산이 이루어지며, 이의 하드웨어 구조는 그림 9의 모듈과 같다. 입력 화소 값 X1에 대한 PP1은 그림 10의 하드웨어 모듈과 같으며, 입력 화소 값 X2에 대한 PP2는 그림 11의 하드웨어 모듈과 또한 같다. 이와 같이, 각각의 PPN(N = 0,1,2, ,7)에서 변환 행렬 계수에 대한 64개 연산 결과값이 얻어질 수 있다. 이 값들 중 변환 행렬 계수의 부호에 맞게 A&S 연산기(그림 7.참조)에서 8개씩 덧셈 및 뺄셈 연산이 이루어진다.

이 연산 결과 값은 그림 7의 8×1 1차원 DCT 하드웨어에서와 같이 SL값 만큼 오른쪽으로 쉬프트 되어진 값을 취한다. 이 결과값들이 8×1 1차원 IDCT 처리된 연산 결과 행렬 [Y] 의 행렬 요소이다.

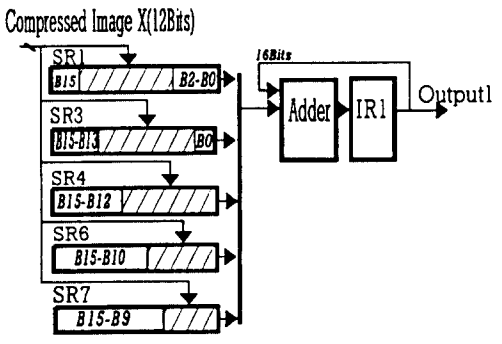


그림 9. 입력 화소 XN0(N=0,1,2,3, ,7 :행번호)를 위한 하드웨어 구조

Fig. 9. Hardware architecture for pixel XN0 (N=0,1,2,3, ,7: row number).

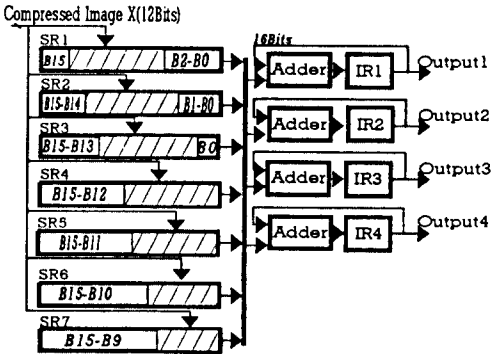


그림 10 입력 화소 X_{N0}(N=0,1,2,3,… ,7 :행번호)를 위한 하드웨어 구조

Fig. 10 Hardware architecture for pixel X_{N0} (N=0,1,2,3,…,7: row number).

$$\begin{bmatrix} Y_0 \\ Y_1 \\ Y_2 \\ Y_3 \\ Y_4 \\ Y_5 \\ Y_6 \\ Y_7 \end{bmatrix} = \begin{bmatrix} C_4 & C_1 & C_2 & C_3 & C_4 & C_5 & C_6 & C_7 \\ C_4 & C_3 & C_6 & C_7 & C_4 & C_1 & C_2 & C_5 \\ C_4 & C_5 & C_6 & C_1 & C_4 & C_7 & C_2 & C_3 \\ C_4 & C_7 & C_2 & C_5 & C_4 & C_3 & C_6 & C_1 \\ C_4 & C_7 & C_2 & C_5 & C_4 & C_3 & C_6 & C_1 \\ C_4 & C_5 & C_6 & C_1 & C_4 & C_7 & C_2 & C_3 \\ C_4 & C_3 & C_6 & C_7 & C_4 & C_1 & C_2 & C_5 \\ C_4 & C_1 & C_2 & C_3 & C_4 & C_5 & C_6 & C_7 \end{bmatrix} \begin{bmatrix} X_0 \\ X_1 \\ X_2 \\ X_3 \\ X_4 \\ X_5 \\ X_6 \\ X_7 \end{bmatrix} \quad (14)$$

단, X_n : 입력된 영상 신호

Y_n : IDCT 처리 후 출력된 영상 신호

C_n : 변환행렬의 코사인 계수($C_n = \cos(n\pi/16)$)

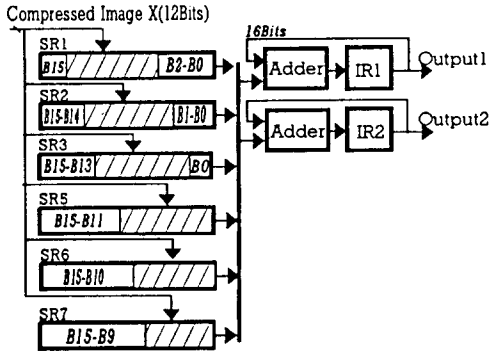


그림 11. 입력 화소 X_{N2}(N=0,1,2,3,…,7 : 행번호)을 위한 하드웨어 구조

Fig. 11. Hardware architecture for pixel X_{N2} (N=0,1,2,3,…,7: row number).

Ⅲ. 시뮬레이션 결과 및 고찰

1. DCT 하드웨어 시뮬레이션

고속 DCT 알고리듬을 DCT 정의식에 적용한 8×8 2차원 DCT를 "Compass" Simulator를 이용하여 설계 및 시뮬레이션 하였으며, 그 구조는 그림 6, 7, 8에서 보는 바와 같다.

한 화소처리를 위한 하드웨어 구조는 그림 6과 같이 변환 행렬의 열방향에 따라 구성하였으며, 입력되는 화소 값을 8비트로 가정하여 설계하였다. 이는 입력된 영상의 한 화소 값을 Gray scale로 0에서 255의 값을 갖기 때문이다.

입력 화소값을 저장하는 SRn(n = 1, 2, ··, 7 : 그림 6.참조)은 입력된 영상 신호 8비트에 계산상의 오차를 줄이기 위하여 왼쪽으로 쉬프트해 주는 비트수 SL값을 더한 12비트로 구성하였다. 또한, 덧셈기(그림 6.참조)는 계산상에서 발생하는 캐리를 생각하여 16비트로 구성하여 사용하였다. 하드웨어 시뮬레이션 결과는 입력된 한 화소에 대해 8가지의 변환 계수에 대한 곱셈을 처리하고 연산결과 값을 출력하는데 기준 클럭을 T(이하 클럭을 T라 한다)라고 했을 경우 7.5T가 소요되었다. 실시간 처리 속도를 얻기 위하여 GATE의 전파지연 시간(T_{PHL}:TransPort Hign to Low, T_{PLH}:TransPort Low to Hign)을 참조[15, 16] 하여 시뮬레이션을 하였으며, T = 40ns일 경우 303.6ns 가 소요되었다. 이 시뮬레이션에 관한 타이밍 도는 그림 12와 같다.

그림 12에서 데이타 버스의 SRn(n = 1, 2, 3,…, 7)은 입력된 영상 신호가 쉬프트되어져 각각의 레지스터에 저장된 값들이 인에이블되어 나타나는 값이다. 이 값들은 각각의 덧셈기에 전달된다. 또한, OutputN(N = 1, 2, 3,…,7)에 나타나는 'E'는 변환 계수의 2진 코드의 '1'인 비트에 대응하여 IRn 레지스터들이 활성화되는 것을 보여준다. 그리고, 'ZZZ'은 IRn 레지스터가 'High impedance' 상태를 의미하며, 'DO'는 IRn 레지스터에 저장된 최종 연산 결과 값이 다음 단계로 전달되는 것을 나타낸다.

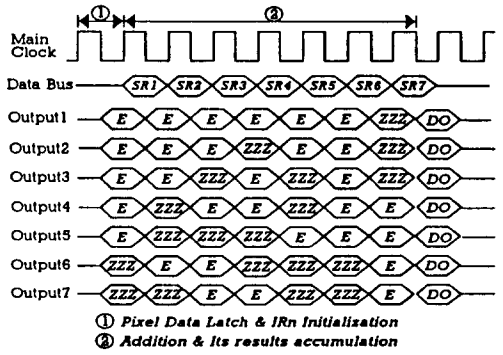


그림 12. 한 화소 처리를 위한 하드웨어의 타이밍 도

Fig. 12. Timing diagram of hardware for a pixel process.

식(7)과 같은 행방향의 1차원 DCT를 수행하기위하여 그림 7과 같은 하드웨어를 구성하고 1차원 DCT 처리 결과 행렬 [G] 를 얻기위한 시뮬레이션을 하였다. 8×8 입력 화소 행렬 [X] 의 요소들중 X_{00} 화소 값은 변환 행렬의 첫번째 row 행렬 요소에 대하여 PP0(그림 7.참조)에서 DCT 변환 계수와의 곱셈이 수행되어지고, X_{01} 화소 값은 변환 행렬의 두번째 row 행렬 요소에 대하여 PP1(그림 7.참조)에

서 DCT 변환 계수와의 곱셈이 수행되도록 하드웨어를 구성하였다. 나머지 같은 행의 입력 화소들에 대해서도 같은 방식으로 PPN(N = 2,3, ··,7)을 구성하였다. 각각의 PPN 모듈에서의 연산 결과 값 64개 중에서 8개씩 행방향의 행렬 연산에 맞추어 덧셈 및 뺄셈 연산기(그림 7.참조)에서 연산이 이루어진다. 덧셈과 뺄셈은 각각의 변환 계수의 부호에 의하여 결정되어진다. 입력 화소 행렬 [X] 의 다른 행들도 앞서 제시된 하드웨어 구조를 이용하므로, 또 다른 형태의 하드웨어 구성의 필요성을 갖지 않는다. 이는 본 연구에서 제시된 하드웨어 구성은 모듈성과 규칙성이 있음을 보여주는 것이다.

8개의 입력된 영상 신호를 행방향으로 DCT 처리하는데는 15.5T가 소요되었다. 또한, 64개의 입력 화소 값을 행방향으로 DCT 처리하는데 소요된 클럭은 입력 화소 행렬의 각 행 방향으로 8개씩 동시에 처리되므로 8개의 입력 화소를 DCT 처리하는데 소요된 클럭과 같다. 즉, 64개의 입력된 화소 값을 행방향으로 1차원 DCT 처리하는데 15.5T 만을 필요로 하였다. 위의 예처럼 T = 40ns 일 경우 행방향 DCT 처리 소요 시간은 624.6ns 이다. 이 시뮬레이션에 관한 타이밍 도는 그림 13과 같다.

그림 13에서 데이타 버스의 PPN(N = 0, 1, 2, ···,7)은 한 화소에 대한 변환 계수와의 곱셈 연산이 이루어진 값들(그림 6.참조)이 클럭에 동기되어 덧셈 및 뺄셈기로 전송되는 것을 보여주는 것이다. 이때 반 클럭앞에 각각의 덧셈 및 뺄셈기의 기존에 저장된 연산 결과 값이 귀환하여 다시 덧셈 및 뺄셈기에 입력된다. 이 값들중 처음으로 귀환되는 H'0000 값은 덧셈 및 뺄셈기의 초기화 된 값을 나타내는 것이다. 또한, 'DO'는 그림 12에서의 'DO'와 동일하게 다음 단계로 전달되는 최종 연산 결과 값을 가르킨다.

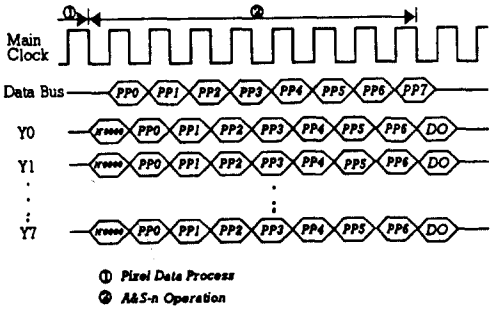


그림 13. 8×1 1차원 DCT 하드웨어의 타이밍 도
Fig. 13. Timing diagram of hardware for 8 ×1 1D DCT.

식(8)의 열방향 1차원 DCT를 수행하기 위한 하드웨어 구성은 앞서 구성된 행방향의 1차원 DCT 하드웨어 구성을 이용하였다. 행렬 [G] 의 행렬 요소 값들은 각각 열방향으로 8개씩 나누어져 각각 하드웨어 모듈(그림 8.참조)에 입력된다. 그러나 행방향의 1차원 DCT 처리된 [G] 값은 SL값 만큼 오른쪽으로 쉬프트되어 열방향의 1차원 DCT 처리단계로 전달되어진다. 즉, 이 값들은 12비트로 구성되고, SRn 레지스터들은 이 12비트에 SL을 더한 16비트로 각각 구성된다.

8×8 2차원 DCT 정의식에서 상수부분 중 1 /√2 부분은 변환 계수로 전환되어 계산되어졌으나, 나머지 상수부분인 1 /4 은 열방향의 1차원 DCT 하드웨어구성에서 덧셈 및 뺄셈 연산기(그림 6 참조)의 입력부분으로 전달되는 16비트중 상위 14비트만이 처리되도록 설계하여 2비트를 오른쪽으로 쉬프트시켜줌으로서 이 상수부분을 계산하였다. 이와같이 연산처리해 줌으로서 2차원 DCT 처리 결과 행렬 [Y] 의 DC값 Y_{00}에서의 overflow를 방지하는 효과를 얻을 수 있었다. 또한, 최종 2차원 DCT 처리가 끝난 연산 결과 값에서 계산상의 쉬프트로 인한 오차를 줄이기 위해 왼쪽으로 쉬프트해주는 비트 수 SL값을 고려하여 상위 12비트만을 8×8 2차원 DCT 연산 결과 값으로 취하였다.

열방향의 1차원 DCT 처리 속도는 전단계인 행방향의 1차원 DCT 처리를 거친 결과값이 다음 단계로 전달되는 시간을 포함하여 15.5T를 얻을 수 있었다. 이 값은 행방향의 1차원 DCT 처리 속도와 같았다. 최종 2차원 DCT 처리가 끝난 결과 행렬 [Y] 을 얻을 수 있었다.

2. IDCT 하드웨어 시뮬레이션

고속 DCT 알고리듬을 식(2)의 IDCT 정의식에 적용한 8×8 2차원 IDCT를 설계 및 시뮬레이션 하였으며, 그 구조는 그림 7, 8, 9, 10, 11에서 보는 바와 같다. 이의 시뮬레이션 결과는 DCT 하드웨어 시뮬레이션 결과와 동일하다.

한 화소처리를 위한 하드웨어 구조는 그림 8, 9, 10과 같이 변환 행렬의 열방향에 따라 세가지 모듈로 구성하였으며, 입력되는 화소 값을 12비트로 가정하여 설계하였다. 이는 압축된 영상 블럭의 DC 값은 상당히 큰 값이며, 최대 DC 값은 약 2885이므로 이 값을 고려하였기 때문이다.

입력 화소값을 저장하는 SRn(n = 1,2, ··,7: 그림 9, 10, 11.참조)은 입력된 압축 화소 12비트에 계산상의 오차를 줄이기 위하여 왼쪽으로 쉬프트해주는 비트 수 SL값을 더한 16비트로 구성하였다. 또

한, 덧셈기(그림 9, 10, 11.참조)는 계산상에서 발생하는 자리올림을 생각하여 16비트로 구성하여 사용하였다. 하드웨어 시뮬레이션 결과는 입력된 한 화소에 대해 8가지의 변환 계수에 대한 곱셈을 처리하고 연산결과 값을 출력하는데 DCT 경우와 같이 7.5T가 소요되었다.

식(14)의 행방향의 1차원 IDCT를 수행하기 위하여 그림 7과 같은 하드웨어를 구성하고 1차원 IDCT 처리 결과 행렬 [G] 를 얻기위한 시뮬레이션을 하였다. 이 시뮬레이션의 결과에 의하여 행방향의 1차원 DCT 처리속도와 동일함을 알 수 있었다.

식(11)의 열방향 1차원 IDCT를 수행하기 위한 하드웨어 구성은 앞서 구성된 행방향의 1차원 IDCT 하드웨어 구성을 이용하였다. 행렬 [G] 의 행렬 요소 값들은 각각 열방향으로 8개씩 나누어져 각각 하드웨어 모듈(그림 8.참조)에 입력된다. 그러나 행방향의 1차원 IDCT 처리된 [G] 값은 SL값 만큼 오른쪽으로 쉬프트되어 열방향의 1차원 IDCT 처리단계로 전달된다. 즉, 이 값들은 12비트로 구성되고, SRn 레지스터들은 이 12비트에 SL을 더한 16비트로 각각 구성된다.

8×8 2차원 IDCT 정의식에서 상수부분 $1/\sqrt{2}$, 1/4은 8×8 2차원 DCT 처리에서와 동일하게 $1/\sqrt{2}$은 변환 계수로 전환하여 변환 행렬에 포함시켰으며, 1/4은 최종 연산 결과 값들 중에서 SL값과 나머지 상수부분인 1/4 값을 고려하여 하위 6비트를 오른쪽으로 쉬프트하여 버리고, 나머지 10비트 중에서 하위 8비트만을 최종 연산 결과로 얻었다. 마지막으로 최종 연산 결과를 최고, 최저치를 넘지않게 보정되도록 보정 회로를 첨가하였다. 8비트로 구성된 데이터인 경우, 식(15)와 같이 복원된 화소 값을 보정해 주었다.

열방향의 1차원 IDCT 처리 속도는 전단계인 행방향의 1차원 IDCT 처리를 거친 결과값이 다음 단계로 전달되는 시간을 포함하여 15.5T를 얻을 수 있었다. 이 값은 행방향의 1차원 IDCT 처리 속도와 같았다. 최종 2차원 IDCT 처리가 끝난 결과 행렬 [Y] 을 얻을 수 있었다.

$$\begin{cases} if\ Y<0, & then\ Y=0 \\ if\ Y>255, & then\ Y=255 \end{cases}$$

Ⅳ. 결 론

본 연구에서는 영상처리 분야에서 많이 이용되고 있는 DCT 및 IDCT 처리를 위한 고속 전용 VLSI 구조를 제시하였으며, 많은 처리시간을 요하는 곱셈 연산을 덧셈 연산 및 쉬프트 연산으로 처리하기 위한 비가역적 근사법을 이용하였다. 하드웨어 구조의 우수성은 Logic simulator를 통하여 31T 만에 8×8 2차원 영상 블럭을 DCT 처리할 수 있었음을 입증하였다. 또한, 이 처리 시간은 8×8 2차원 IDCT 처리 시간과 동일함을 알 수 있었다.

본 연구에서 제시된 하드웨어 구조는 scalable한 구조를 갖기 때문에 단순한 변형으로 입력되는 영상 블럭의 크기를 조정할 수 있다. 또한, 규칙성과 모듈성을 가지고 있기때문에 VLSI 설계에 매우 유익한 구조이다.

※ 본 연구는 아주대학교 정착연구비 지원에 의해 이루어진 것임.

參 考 文 獻

[1] R.Rao, P.Yip, Discrete Cosine Transform Algorithms, Advantages, Application., New York, Academic press,1990.

[2] M.Vetterli, H.Nussbaumer, "Simple FFT and DCT algorithms with reduced number of operations," Signal Processing, vol.6, pp.267-278, Aug. 1984.

[3] N.Ahmed, T.Natarajan, K.Rao, "Discrete cosine transform," IEEE Trans., Comput., vol.C-23, pp.90-93, Jan. 1974.

[4] B.G.Lee, "A new algorithm for the discrete cosine transform," *IEEE Trans. Acoust.*, Speech, and Signal Process., vol.ASSP-32, no.6, pp.1243-1245, Dec. 1984.

[5] M.A.Haque, "A two-dimensional fast cosine transform," IEEE Trans Acoust., Speech, and Signal Process., vol. ASSP-33, no.6, pp.1532-1538, Dec. 1985.

[6] H.S.Hou, "A fast recursive algorithm for computing the cosine transform," *IEEE Trans. Acoust.*, Speech, and Signal Process., vol. ASSP-35, no.10, pp.1455-1461, Oct.1987.

[7] H.S.Lim, D.S.kim, N.I.Cho, and S. U.Lee, "Systolic Arrays for 2-D DCTand Other Orthogonal Transforms,"

KITE., vol.27, Num.7, July 1990.

[8] W.H.Chen, C.H. Smith and S.C. Fralick, "A fast computational algorithm for discrete cosine transform," *IEEE Trans Commun.*, vol.COM-25, pp. 1004-1009, Nov. 1977.

[9] JPEG Technical Specification, PostScript, May 4, 1991.

[10] A.Artieri, S.Kritter, F.Jutand, and N. Demassieux, "A one chip VLSI for real time two-dimentional discrete cosine transform," 1988 Intl. Symp. on Circuits and Systems, pp.701-704, Helsinki, Finland, June 1988.

[11] M.Afghahi, S.Matsumura, J.Pencz, B. Sikstrom, U.Sjostrom, and L. Vandendorpe, "Block operation in digital signal processing with application to TV coding," Speech, and Signal Process., vol.13, pp.385-387

[12] B.Sikstroem, L.Wanhammer, M. Afghahi, and J.Pencz, "A high speed 2-D discrete cosine transform chip," Integration the VLSI Journal, vol.5, pp.159-169, June 1987.

[13] I.Defilippis, U.Sjoestroem, M.Ansorge, F.Paellandini, "A 2 dimensional 16 point discrete cosine transform chip for real time video applications," Douzieme colloque sur le traitement du signal et des images, pp.813-816, Juan-Les-Pins, France, June 12-16, 1989.

[14] H.J.Chung, S.J.Kim, G.H.Jung, Y.D. Kim, "A DCT algorithm using shift and addition," KICS, vol.18, no.6, pp. 773-778, June, 1993.

[15] Data book," HSGIK/10K channelless gate array," Hyundai Electronics industries Co., Ltd., Korea, 1991.

[16] Data book," SEC KG50000 CELL LIBRARY," Samsung Electronics industries Co., Ltd., Korea, 1992.

[17] MPEG Video Simulation Model Three (SM3), Simulation Model Editorial Group, July, 1990.

[18] Seung-Wook Lee, Hwa-Ja Chung, Gi-Hyun Jung, Youg-Deak Kim, "A study on implimenting DCT without using multiplier and memory units", Procedings of KITE Fall Conference '92, vol.15, num.2, Nov.21, 1992

著者紹介

李 勝 旭(準會員)

1968年 7月 26日生. 1991年 2月. 아주대학교 전자공학과(공학사). 1992年 3月 ~ 현재 아주대학교 전자공학과 석사과정중. 주관심 분야는 컴퓨터 구조, ASIC 설계 및 ASSP 설계 등임.

任 綱 彬(準會員)

1969年 4月 27日生. 1992年 2月. 아주대학교 전자공학과(공학사). 1992年 3月 ~ 현재 아주대학교 전자공학과 석사과정중. 주관심 분야는 영상처리 및 멀티미디어 시스템 설계 등임.

•

鄭 華 子(正會員) 第 29卷 B編 第 6號 參照
현재 서울산업대학 전산계산학과 교수

•

鄭 己 鉉(正會員) 第 29卷 B編 第 7號 參照
현재 아주대학교 전자공학과 교수

•

金 容 得(正會員) 第 30卷 B編 第 5號 參照
현재 아주대학교 전자공학과 교수