論文 95-11-11-24

실시간 비디오 응용을 위한 2차원 DCT/IDCT의 구조

正會員 崔 長 植*

The Structure of 2-D DCT/IDCT for Real-Time Video Applications

Jang Sik Choi* Regular Members

要 約

본 논문에서는 서로 다른 8×8 블럭과 2개의 4×8 블럭을 처리하면서, 구조적인 정형성을 가지고, ITU-T의 H.261 IDCT 에러 규정치를 만족하는 정방향/역방향 2-D DCT 구조를 제시하였다. 이 구조는 국제 표준 알고리듬인 JPEG, H.261, MPEG 표준과 DVCR(Digital Video Cassette Recorder) 규격을 모두 만족한다. 제안한 2-D DCT는 행렬 분해 방식의 구조로 2개의 1-D DCT와 행렬치환기로 구성된다. 1-D DCT는 분산산술처리 방식과 고속 알고리즘의 절충형 구조이며, 행렬치환기는 레지스터와 멀티플렉서의 조합을 통해 행렬간의 상호 치환을 수행한다.

ABSTRACT

In this paper, we proposed the architecture of 2-D DCT/IDCT that is optionally possible to process a 8x8 block and two 4x8 blocks and satisfies the accuracy specification of ITU-T with a structural regularity. This architecture satisfies JPEG, H.261, MPEG and DVCR(Digital Video Cassette Recorder). The proposed 2-D DCT of the row-column separation method requires two 1-D DCT and a transposition network. The 1-D DCT is a compromise between distributed arithmetic and fast algorithm. The transposition network that consists of registers and multiplexers processes the row-column transposition.

* 한국전자통신연구소
論文番號 : 95304-0904
接受日字 : 1995年 9月 4日

Ⅰ. 서 론

이산여현변환 DCT(Discrete Cosine Transform)는 화상 정보를 주파수 정보로 변환시키거나 반대로 주파수 정보를 화상 정보로 변환시켜 주는 방법[3] 중의 하나로 신호처리 응용 분야에 많이 사용된다. DCT는 화상 전체에 대해 변환을 할 경우 많은 처리 시간이 소요되므로 원 이미지를 8x8 크기의 블럭으로 나누어 각 블럭에 대해 DCT 변환을 한다. 이때 8×8 블럭 내에 있는 화소값들은 급격히 변하지 않으므로 변환을 수행하고 나면 대부분의 공간 주파수 성분들은 영이거나 또는 거의 영에 가까울 정도로 매우 작아 부호화할 필요가 없게 된다. 그래서 JPEG[2], H.261[9], MPEG[13,14] 표준의 DCT는 8 x 8의 화소 블럭 크기만을 처리하도록 규정하고 있다. 그러나 DVCR(Digital Video Cassette Recorder) 규격에서는 8×8 블럭과 함께 2개의 4×8 블럭도 처리할 것을 요구[11,21]하고 있다.

DCT는 많은 계산량을 요구하는 것으로 2-D DCT의 직접 계산시 N×N 블럭의 경우 $O(N^4)$의 곱셈을 요구해, 8×8 이미지의 경우 4096번의 곱셈을 수행해야 한다. 이는 고속으로 동작할 수 있는 DCT 구조를 필요로 하는데 기본적으로 많은 비용과 실리콘 면적을 차지하는 곱셈기의 수를 최소 한도로 줄이는 것이 중요하다. 그리고 산술 복잡도가 적고, VLSI의 정형성(regularity)을 크게 하고, 좋은 에러 성능(error-performance)을 가지며, 이동 휴대 화상 전화와 같은 응용 분야를 위해서 전력 소비가 적어야 한다.[10]

지금까지 제안된 DCT 구조들은 크게 직접(direct)으로 계산하는 NRCA(No Row Column Algorithm) 방법과 DCT의 분리 가능성(separability)를 이용하여 2개의 1-D DCT와 치환을 통한 RCA(Row Column Algorithm) 방법으로 구분할 수 있다. 그리고 DCT의 분리 가능성을 이용한 2-D DCT의 구조에서 1-D DCT는 매트릭스와 벡터 입력 값을 가지고 계산하는 직접(direct)방법[8,18,22]과 매트릭스 계산에 사용되는 곱셈기의 수를 최소한으로 줄이기 위한 고속(fast) 알고리즘[7,17,24]으로 구분된다. 직접 방법에는 분산산술처리(distributed arithmetic)를 이용한 방법[8], 시스톨릭 어레이를 이용한 방법[15,16]들이 있고 고속 알고리즘의 대표적인 것으로 Lee[7]와 Hou[17]의 알고리즘이 있다. 고속 알고리즘들은 분산산술처리 방식보다 곱셈기의 수를 줄일 수 있으나 고정 길이의 레지스터 사용으로 인한 라운드오프(roundoff) 에러가 발생하고 버터 플라이(butterfly) 연산으로 인한 불규칙적인 구조, 복잡한 라우팅, 많은 실리콘 면적과 긴 설계 시간을 요구하는 단점이 있다. 직접(Direct)방법의 하나인 분산산술처리는 실리콘 면적을 적게 하기 위해 곱셈기를 사용하지 않고 덧셈기, 레지스터와 메모리로만 구성된 RAC(ROM and Accumulator in Cascade)을 통해 곱셈을 수행하나, RAC의 수가 많아지면서 하드웨어가 증가하는 단점이 있다.

위의 단점들을 보완하여 본 논문에서 제안하는 DCT 변환의 구조는 분산산술처리 방법과 고속 알고리즘의 혼합 구조로 간단하면서 비용이 저렴하고 실리콘 면적을 적게 차지하는 특성을 가진다. 또한 8x8의 화소 블럭은 물론, 2개의 4x8 블럭의 처리도 지원하고 있다.

본 논문의 2장에서는 서로 다른 입력 블럭을 처리하는 2-D DCT/IDCT의 알고리즘에 대해, 3장에서는 DCT의 전체적 구조에 대해, 4장에서는 DCT의 주요 기능 블럭의 구조 및 동작에 대해 설명한다. 5장에서는 ITU-T H.261의 IDCT 에러 규정치와 본 구조의 에러 규정치를 비교, 설명한다. 6장에서는 DCT의 설계 검증 과정에 대해 설명한다. 마지막으로 7장에서 결론을 맺는다.

Ⅱ. 2-D DCT/IDCT

1. 8×8 DCT/IDCT 알고리즘

주어진 8×8 픽셀 블럭의 2-D DCT는 정방향인 경우 식(1), 역방향인 경우 식(2)로 정의된다.

위 식 (1), (2)에서 의 코사인 함수는 각각 식 (3)으로 정의된다.

$$F(u,v)=\frac{1}{4}C(u)C(v)\sum_{x=0}^{7}\sum_{y=0}^{7}f(x,y)C_1(x,u)C_1(y,v) \quad (1)$$

$$f(x,y)=\frac{1}{4}\sum_{u=0}^{7}\sum_{v=0}^{7}C(u)C(v)F(u,v)C_1(x,u)C_1(y,v)$$

$$C(i)=\begin{cases}\frac{1}{\sqrt{2}}, & if \quad i=0\\ 1, & if \quad i\neq 0\end{cases} \quad (2)$$

$$C_f(x,u)=\cos\frac{(2x+1)\pi u}{16},$$

$$C_f(y,v)=\cos\frac{(2y+1)\pi v}{16} \quad (3)$$

u, v는 2-D DCT 변환 후의 계수(coefficient) $F(u, v)$의 위치를, x, y는 픽셀 블럭 내의 픽셀 $f(x, y)$의 위치를 나타내며, 이때 u, v, x, y는 0에서 7의 범위를 가진다. 식 (1)은 $f(0,y),\ldots,f(7,y)$의 8×1 벡터와 8×8 매트릭스인 $C_f(x,u)$와의 곱셈 형태로 각 열에 대해 8×1 DCT가 수행되어 8×8 매트릭스가 생성된다. 이 매트릭스에서 열 순서의 8×1 입력 벡터는 다시 $f(x,0),\ldots,f(x,7)$의 값이 되어 $C_f(y,v)$의 8×8 매트릭스와의 곱셈 형태로 각 행에 대해 두 번째의 8×1 DCT가 수행된 값이 산출되어, 최종 2-D DCT 변환된 8×8 매트릭스가 만들어진다. 이 과정은 입력 8×8 픽셀 블럭에 대해 방향으로 1-D DCT를 계산하고, 열 방향으로 1-D DCT를 수행하여 8×8 계수 블럭을 생성하는 것과 같으므로, 식 (1)은 행과 열이 분리된 형태의 식 (4)로 표현이 가능하다.

$$Z=(X^TC)^TC = C^TXC \quad (4)$$

식 (4)에서 Z는 2-D DCT가 수행된 계수 블럭, X는 픽셀 블럭, C는 8×8 코사인 매트릭스, CT는 C가 행렬 치환된 매트릭스를 의미한다.

식 (2)의 역방향 2-D DCT는 DCT 계수 블럭을 $C_f(y,v)$의 8×8 매트릭스와 열 순서로 곱한 다음, 그 값에 행 순서로 $C_f(x,u)$의 8×8 매트릭스와 곱하므로 원래의 픽셀 블럭을 복원한다. 따라서 역방향도 행렬이 분리된 식의 형태로 정의할 수 있다.

2. 2개의 4×8 DCT/IDCT 알고리즘

2개의 4×8블럭을 처리하기 위한 정방향, 역방향 DCT의 정의식은 식 (5), (6)과 같다.

$$F(u,v)=\frac{1}{4}C(u)C(v)\sum_{x=0}^{3}\sum_{y=0}^{7}[f(x,2y)+f(x,2y+1)]C_f(x,u)C_f(y,v) \quad (5)$$

$$F(u,v+4)=\frac{1}{4}C(u)C(v)\sum_{x=0}^{3}\sum_{y=0}^{7}[f(x,2y)-f(x,2y+1)]C_f(x,u)C_f(y,v)$$

$$f(x,2y)=\frac{1}{4}\sum_{u=0}^{3}\sum_{v=0}^{7}C(u)C(v)[F(u,v)+F(u,v+4)]C_f(x,u)C_f(y,v)$$

$$f(x,2y+1)=\frac{1}{4}\sum_{u=0}^{3}\sum_{v=0}^{7}C(u)C(v)[F(u,v)-F(u,v+4)]C_f(x,u)C_f(y,v)$$

$$C(i)=\begin{cases}\frac{1}{2\sqrt{2}}, & if \quad i=0\\ \frac{1}{2}, & if \quad i\neq 0\end{cases} \quad (6)$$

위의 식에서 는 각각 아래와 같이 정의된다.

$$C_f(x,u)=\cos\frac{(2x+1)\pi u}{16},$$

$$C_f(y,v)=\cos\frac{(2y+1)\pi v}{8} \quad (7)$$

식(5)의 정방향 처리 시 $[f(x,2y)\pm f(x,2y+1)]$는 $f(x,2y)$와 $f(x,2y+1)$로 분리된다. 분리된 $[f(0,2y),\ldots,f(7,2y)]$와 $[f(0,2y+1),\ldots,f(7,2y+1)]$의 8×1 입력 벡터는 의 8×8 매트릭스와의 곱셈을 통해 8×1 DCT의 결과치를 출력한다. 이 8×1 DCT는 각 열에 대해 수행되어 8×8 매트릭스를 생성한다. 이 매트릭스의 각 행에 대해 두 번째 DCT에서 $[f(0,2y),\ldots,f(0,2y+1)]$의 가감산이 이루어진다. 이 가감산의 결과치는 각각 2개의 4×1 입력 벡터가 되어 4×4 매트릭스인 $C_f(y,v)$와 곱해져 2개의 4×1 DCT 결과치가 생성된다. 그래서 식 (5), (6)의 정의식은 8×8 블럭의 행렬이 분리된 식 (4)의 형태로 표현이 가능하다.

Ⅲ. DCT 구조

8×8 블럭과 2개의 4×8 블럭을 처리하는 2-D DCT의 식 (4)는 그림 1의 구조로 2개의 1-D DCT와 행렬 치환기를 통해 정방향과 역방향 DCT 변환이 수행된다. 8×8 블럭을 처리하는 2-D DCT의 정방향시 $f(x, y)$

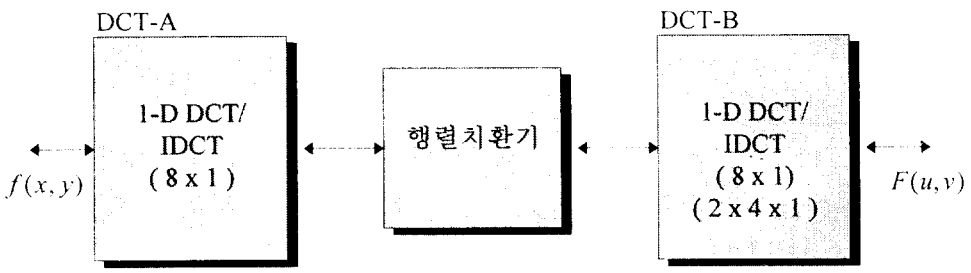


그림 1. 2-D DCT의 분해
Fig. 1. The decomposition of 2-D DCT

는 행 순서로 DCT-A에서 8×1 DCT가 수행된다. 이 결과는 행렬 치환되고 8×1 DCT인 DCT-B를 통해 F(u, v)의 결과치가 생성된다. 역방향의 동작은 정방향의 역순으로 $F(u, v)$가 입력되어 DCT-B, 행렬치환기, DCT-A를 거쳐 $f(x, y)$가 복원된다. 그리고 2개의 4×8 블럭의 정방향DCT 변환에서는 $f(x, y)$가 입력되어 DCT-A에서 8×1 DCT가 계산되고, 그 결과는 행렬 치환된 후 수직 방향의 이웃하는 화소끼리 가감산이 이루어진다. 이 값은 상, 하위로 구분되어 DCT-B에서 4×1 DCT가 각각 수행된 $F(u, v)$가 생성된다. 역방향의 경우는 계수 블럭의 8×1 입력 벡터에 대해 2개의 4×1 IDCT가 수행되고 행렬치환 후 8×1 IDCT를 거쳐 픽셀 블럭이 복원된다.

행렬 분해 구조의 2-D DCT에서 정방향 1-D DCT는 8×1 입력 벡터와 8×8코사인 매트릭스의 곱셈 형태로 표현되고, 이는 분해(decomposition) 과정을 통해 식 (8), (9)의 두 식으로 나타낼 수 있다.

$$\begin{vmatrix} y_0 \\ y_2 \\ y_4 \\ y_6 \end{vmatrix} = \begin{vmatrix} cos4\theta & cos4\theta & cos4\theta & cos4\theta \\ cos2\theta & cos6\theta & -cos6\theta & -cos2\theta \\ cos4\theta & -cos4\theta & -cos4\theta & cos4\theta \\ cos6\theta & -cos2\theta & cos2\theta & -cos6\theta \end{vmatrix} \begin{vmatrix} x_0 + x_7 \\ x_1 + x_6 \\ x_2 + x_5 \\ x_3 + x_4 \end{vmatrix} \quad (8)$$

$$\begin{vmatrix} y_1 \\ y_3 \\ y_5 \\ y_7 \end{vmatrix} = \begin{vmatrix} cos\theta & cos3\theta & cos5\theta & cos7\theta \\ cos3\theta & -cos7\theta & -cos\theta & -cos5\theta \\ cos5\theta & -cos\theta & cos7\theta & cos3\theta \\ cos7\theta & -cos5\theta & cos3\theta & -cos\theta \end{vmatrix} \begin{vmatrix} x_0 - x_7 \\ x_1 - x_6 \\ x_2 - x_5 \\ x_3 - x_4 \end{vmatrix} \quad (9)$$

$y_0, \ldots, y_7$는 1-D DCT후의 계수 값을, $x_0, \ldots, x_7$는 픽셀 입력 8×1벡터를 나타내며, θ는 $\frac{\pi}{16}$이다. 식 (8)은 재분해 과정을 통해 식 (10)으로 변환이 가능하다.

$$\begin{vmatrix} y_0 \\ y_4 \\ y_2 \\ y_6 \end{vmatrix} = \begin{vmatrix} cos4\theta & cos4\theta & 0 & 0 \\ cos4\theta & -cos4\theta & 0 & 0 \\ 0 & 0 & cos6\theta & cos2\theta \\ 0 & 0 & -cos2\theta & cos6\theta \end{vmatrix} \begin{vmatrix} x_0 + x_7 + x_3 + x_4 \\ x_1 + x_6 + x_2 + x_5 \\ x_1 + x_6 - x_2 - x_5 \\ x_0 + x_7 - x_3 - x_4 \end{vmatrix} \quad (10)$$

그리하여 정방향 8×1 DCT는 아래의 식 (11)~(13)으로 표현될 수 있어, 원래의 (8×8)×(8×1) 곱셈이 2개의 (2×2)×(2×1), 1개의 (4×4)×(4×1) 곱셈과 14번의 덧셈으로 바뀐다.

$$\begin{vmatrix} y_0 \\ y_4 \end{vmatrix} = \begin{vmatrix} cos4\theta & cos4\theta \\ cos4\theta & -cos4\theta \end{vmatrix} \begin{vmatrix} x_0 + x_7 + x_3 + x_4 \\ x_1 + x_6 + x_2 + x_5 \end{vmatrix} \quad (11)$$

$$\begin{vmatrix} y_6 \\ y_2 \end{vmatrix} = \begin{vmatrix} -cos2\theta & cos6\theta \\ cos6\theta & cos2\theta \end{vmatrix} \begin{vmatrix} x_1 + x_6 - x_2 - x_5 \\ x_0 + x_7 - x_3 - x_4 \end{vmatrix} \quad (12)$$

$$\begin{vmatrix} y_1 \\ y_3 \\ y_5 \\ y_7 \end{vmatrix} = \begin{vmatrix} cos\theta & cos3\theta & cos5\theta & cos7\theta \\ cos3\theta & -cos7\theta & -cos\theta & -cos5\theta \\ cos5\theta & -cos\theta & cos7\theta & cos3\theta \\ cos7\theta & -cos5\theta & cos3\theta & -cos\theta \end{vmatrix} \begin{vmatrix} x_0 - x_7 \\ x_1 - x_6 \\ x_2 - x_5 \\ x_3 - x_4 \end{vmatrix} \quad (13)$$

그리고 식 (11)~(13)의 분해는 더 이루어질 수 있으나, VLSI의 정형성과 산술 복잡도의 상호 보완 측면에서 더 이상의 분해는 하지 않았다.

역방향 1-D DCT는 정방향 DCT에서와 같이 (8×8)

×(8×1)의 매트릭스가 첫번 분할 과정을 거쳐 식 (14),(15)로 표현된다.

$$\begin{matrix}1\\2\end{matrix}\begin{vmatrix} x_0+x_7 \\ x_1+x_6 \\ x_2+x_5 \\ x_3+x_4 \end{vmatrix} = \begin{vmatrix} cos4\theta & cos4\theta & cos4\theta & cos4\theta \\ cos2\theta & cos6\theta & -cos6\theta & -cos2\theta \\ cos4\theta & -cos4\theta & -cos4\theta & cos4\theta \\ cos6\theta & -cos2\theta & cos2\theta & -cos6\theta \end{vmatrix} \begin{vmatrix} y_0 \\ y_2 \\ y_4 \\ y_6 \end{vmatrix} \quad (14)$$

$$\begin{matrix}1\\2\end{matrix}\begin{vmatrix} x_0-x_7 \\ x_1-x_6 \\ x_2-x_5 \\ x_3-x_4 \end{vmatrix} = \begin{vmatrix} cos\theta & cos3\theta & cos5\theta & cos7\theta \\ cos3\theta & -cos7\theta & -cos\theta & -cos5\theta \\ cos5\theta & -cos\theta & cos7\theta & cos3\theta \\ cos7\theta & -cos5\theta & cos3\theta & -cos\theta \end{vmatrix} \begin{vmatrix} y_1 \\ y_3 \\ y_5 \\ y_7 \end{vmatrix} \quad (15)$$

위의 식 (14), (15)는 재분해 과정을 거쳐 식 (16)~(18)로 표현된다.[1]

$$\begin{matrix}1\\4\end{matrix}\begin{vmatrix} x_0+x_7+x_3+x_4 \\ x_1+x_6+x_2+x_5 \end{vmatrix} = \begin{vmatrix} cos4\theta & cos4\theta \\ cos4\theta & -cos4\theta \end{vmatrix} \begin{vmatrix} y_0 \\ y_4 \end{vmatrix} \quad (16)$$

$$\begin{matrix}1\\4\end{matrix}\begin{vmatrix} x_1+x_6-x_2-x_5 \\ x_0+x_7-x_3-x_4 \end{vmatrix} = \begin{vmatrix} -cos2\theta & cos6\theta \\ cos6\theta & cos2\theta \end{vmatrix} \begin{vmatrix} y_6 \\ y_2 \end{vmatrix} \quad (17)$$

$$\begin{matrix}1\\2\end{matrix}\begin{vmatrix} x_0-x_7 \\ x_1-x_6 \\ x_2-x_5 \\ x_3-x_4 \end{vmatrix} = \begin{vmatrix} cos\theta & cos3\theta & cos5\theta & cos7\theta \\ cos3\theta & -cos7\theta & -cos\theta & -cos5\theta \\ cos5\theta & -cos\theta & cos7\theta & cos3\theta \\ cos7\theta & -cos5\theta & cos3\theta & -cos\theta \end{vmatrix} \begin{vmatrix} y_1 \\ y_3 \\ y_5 \\ y_7 \end{vmatrix} \quad (18)$$

식(16)~(18)의 역방향 1-D DCT의 수행 시 곱셈의 수는 정방향과 같은 수로 줄어든다. 특히 정방향과 역방향시 코사인 매트릭스의 값이 같아 정방향과 역방향에 따른 각각의 ROM을 사용하지 않아도 된다. 그리고 역방향 DCT가 정방향 DCT의 역으로 되어 있어 멀티플렉서 제어를 통해 데이터 흐름을 변경하면 단일 구조로 정방향/역방향의 DCT 변환이 가능하다.

2개의 4×8을 처리하는 정방향 1-D DCT도 분해 과정을 통한 매트릭스식으로 표현하면 식 (19)~(21)과 같다.

$$\begin{vmatrix} y_0 \\ y_4 \end{vmatrix} = \begin{vmatrix} cos4\theta & cos4\theta \\ cos4\theta & -cos4\theta \end{vmatrix} \begin{vmatrix} x_0+x_1+x_6+x_7 \\ x_2+x_3+x_4+x_5 \end{vmatrix} \quad (19)$$

$$\begin{vmatrix} y_6 \\ y_2 \end{vmatrix} = \begin{vmatrix} -cos2\theta & cos6\theta \\ cos6\theta & cos2\theta \end{vmatrix} \begin{vmatrix} x_2+x_3-x_4-x_5 \\ x_0+x_1-x_6-x_7 \end{vmatrix} \quad (20)$$

$$\begin{vmatrix} y_1 \\ y_3 \\ y_5 \\ y_7 \end{vmatrix} = \begin{vmatrix} cos4\theta & cos4\theta & cos4\theta & cos4\theta \\ cos2\theta & cos6\theta & -cos6\theta & -cos2\theta \\ cos4\theta & -cos4\theta & -cos4\theta & cos4\theta \\ cos6\theta & -cos2\theta & cos2\theta & -cos6\theta \end{vmatrix} \begin{vmatrix} x_0-x_1 \\ x_2-x_3 \\ x_4-x_5 \\ x_6-x_7 \end{vmatrix} \quad (21)$$

식 (19)~(21)를 8×8 블럭을 처리하는 8×1 DCT의 식 (11)~(13)과 비교하여 보면 식 (21)과 식 (13)의 코사인 매트릭스 값이 다르다. 이는 2개의 서로 다른 블럭을 지원하기 위해서는 별도의 코사인 매트릭스가 제공되어져야 함을 의미한다. 또한 코사인 매트릭스와 곱셈이 되는 입력 벡터의 계산 형태가 달라, 블럭의 크기에 따라 입력 벡터의 셔플을 제어하는 회로(셔플제어기)

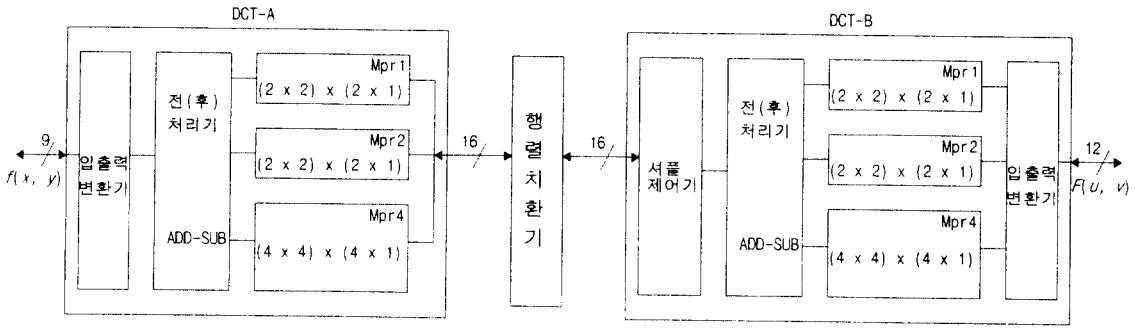


그림 2. 2-D DCT의 구조
Fig. 2. The architecture of 2-D DCT

가 필요하다.

그러면 앞서 논한 매트릭스 식들을 2개의 8×1 DCT와 행렬치환기로 구성된 2-D DCT로 표현하면 그림 2와 같다.

DCT-A에서는 8×1 DCT/IDCT가, DCT-B에서는 8x1 DCT/IDCT 또는 2개의 4×1 DCT/IDCT가 수행된다. DCT-A와 DCT-B에 사용되는 기능 단위(unit)들은 입출력 변환기, 전(후)처리기, Mpr1, Mpr2, Mpr4, 셔플제어기로 구분된다. DCT-A는 세부적으로 그림 3의 구조를 가진다. DCT-B는 그림 3의 구조에 셔플제어기가 추가되고 입출력 변환기가 맨 마지막 단계에 놓인다. 그리고 Mpr1, Mpr4의 구조가 DCT-A와 다르다. DCT-B의 Mpr1에서는 1/2가 수행되지 않으며, Mpr4에서는 2개의 서로 다른 블럭을 처리하기 위해 기존의 8×1 DCT용 ROM에다 4×1 DCT를 위한 ROM이 추가된다.

Ⅳ. 주요 기능 단위들의 구조 및 동작

이 장은 2-D DCT 변환을 위한 주요 기능 단위들의 구조 및 동작 과정에 관한 것으로, 매트릭스 곱셈을 위한 입력 벡터의 계산 과정, 병렬-시리얼 변환기, Mpr1, Mpr2, Mpr4의 매트릭스 곱셈을 하기 위해 사용된 RAC 그리고 행렬 치환을 담당하는 행렬치환기에 대해 설명하고 있다.

1. 매트릭스 곱셈을 위한 입력 벡터의 계산(전(후)처리기 : ADD-SUB)

매트릭스 곱셈을 하기 전 정방향(역방향)시는 전(후)처리에 해당하는 입력 벡터의 계산을 위한 가감산이 그림 3의 전(후)처리기(ADD-SUB)에서 이루어진다. 식 (11)~(13)에서 코사인 매트릭스와 곱해지는 입력 벡터의 계산을 위해서 먼저 $x_0,\ldots,x_7$의 입력에 대해 식 (22)가 수행되고, 그 결과인 A(1)~A(4)를 가지고 식 (23)이 수행되어 B(1)~B(4)의 결과치가 생성된다.

$$A(1)=x_0+x_7, A(2)=x_1+x_6, A(3)=z_2+x_5, A(5)=x_3+x_4,$$
$$A(5)=x_3-x_4, A(6)=x_2-x_5, A(7)=x_1=x_6, A(8)=x_0-x_7 \quad (22)$$

2개의 4×8 블럭을 처리하기 위해서는 식 (22)가 식

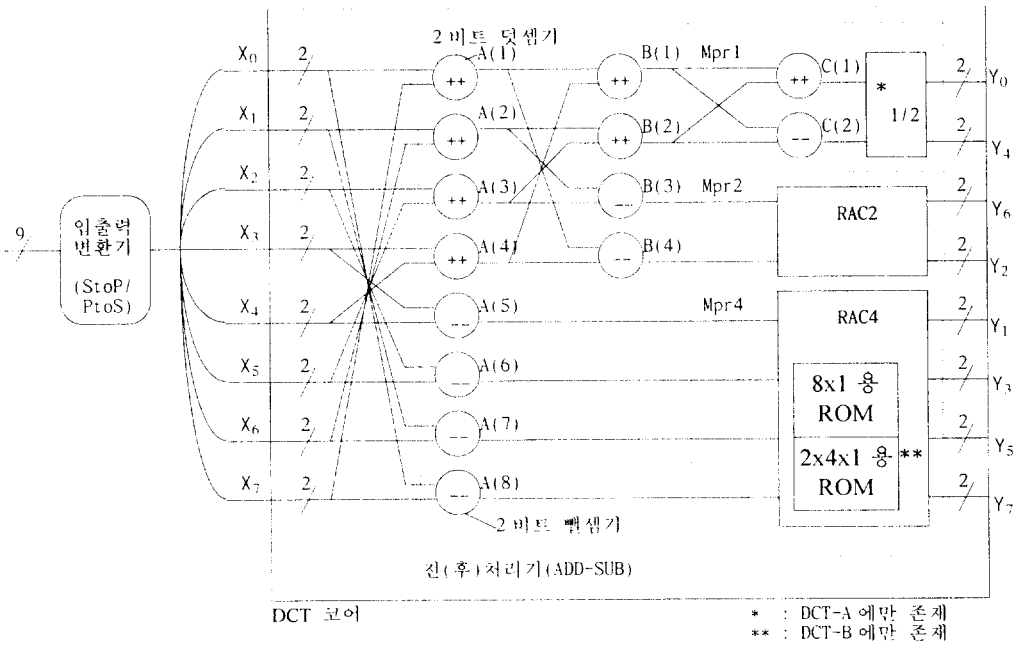


그림 3. 8×1 DCT 구조도
Fig. 3. The architecture of 8×1 DCT

(24)로 바뀌며, B(1)~B(4)의 계산은 8×8 블럭의 경우와 같이 식 (23)을 통해 이루어진다.

$$B(1)=A(1)+A(2), A(2)=A(3)+a(4), B(3) =A(4), B(4)=A(1)-A(2) \quad (23)$$

$$A(1)=x_0+x_1, A(2)=x_2+x_3, A(3)=x_4+x_5 A(4)=x_6+x_7, \quad (24)$$
$$A(5)=x_0-x_1, A(6)=x_2-x_3, A(7)=x_4-x_5, A(8)=x_6-x_7$$

그러므로 2개의 서로 다른 블럭을 처리하기 위해서는 식 (22)와 식 (24)를 모두 지원해야 한다. 따라서 A(1)~A(8)를 계산하는 2비트 덧셈기와 뺄셈기에 입력되는 $x_0, \ldots, x_7$의 셔플 제어를 위해 별도의 셔플제어기가 DCT-B에 사용된다.

역방향 DCT 변환 시는 정방향의 역순으로 동작하는데, 먼저 Mpr1, Mpr2의 결과치를 입력으로 B(1)~B(4)가 계산되고, 이 결과치에 대해 A(1)~A(8)의 계산이 이루어진다.

2. 입출력 변환기

입출력 변환기는 DCT 코어에서 계산된 결과를 외부로 출력하거나, DCT 변환을 위해 DCT 코어로의 데이터 입력을 담당하는 것으로 그림 4의 구조로 되어 있다.

정방향 DCT 변환을 위한 9비트의 픽셀 데이터는 DCT 코어로 클럭에 동기 되어 순차적으로 입력된다. 그리고 이 값은 클럭에 동기 되어 다음 레지스터로 시프트된다. 8 사이클 후 8개의 레지스터(SR1~SR8)에 입력된 픽셀 데이터는 병렬-시리얼 변환기로의 전달 신호에 의해 8개의 채널을 통해 병렬-시리얼 변환기로 전달된다. 병렬-시리얼 변환기는 시프트레지스터군으로부터 넘겨 받은 데이터를 DCT 코어로 클럭에 따라 LSB부터 2비트씩 순차적으로 출력한다.

역방향 DCT 변환시 병렬-시리얼 변환기는 클럭에 동기화되어 순차적으로 데이터를 2비트씩 받아들여 8사이클 후 시프트레지스터군으로 전달한다. 전달된 픽셀 데이터는 클럭에 동기화되어 시프트되면서 순차적으로 출력된다.

3. 매트릭스 곱셈(Mpr1, Mpr2, Mpr3)

1-D DCT의 변환 시 매트릭스 곱셈은 2개의 (2×2)×(2×1)과 1개의 (4×4)×(4×1)번 이루어진다. 이중 (2×2)×(2×1)의 곱셈을 수행하는 그림 3의 맨 상위 Mpr1은 입력 벡터에 을 곱하는 것으로 2-D DCT의 전체적인 면에서 보면 을 하는 것과 같으므로 곱셈 대신 2개의 1-D DCT에서 덧셈을 하고 첫번 째 1-D DCT에서 을 해주면 같은 결과를 얻게 된다. 그래서 DCT-A에서 Mpr1의 곱셈은 식 (25)를 통한 덧셈과, 그 결과인 C(1), C(2)에 대해 을 한다. DCT-B의 Mpr1에서는 식 (25)만 수행된다.

$$C(1) = B(1)+B(2), \; C(2) = B(1) - B(2) \quad (25)$$

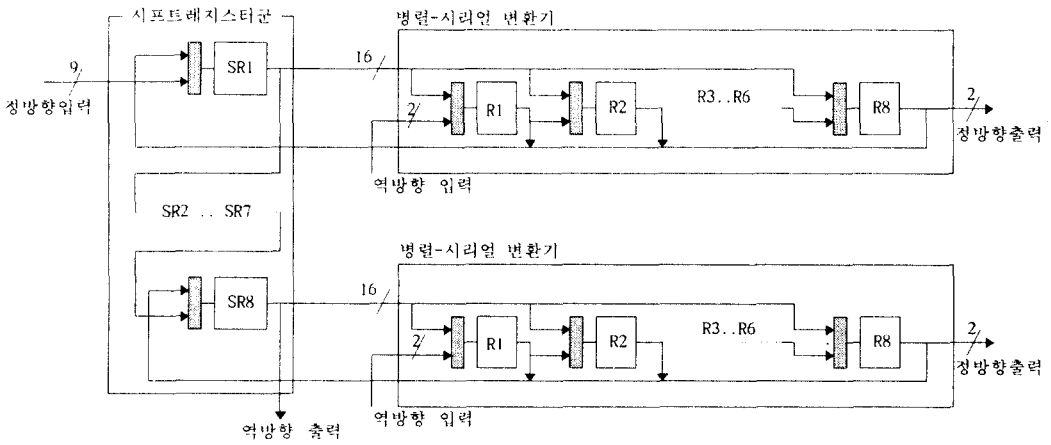


그림 4. 입출력 변환기 구조도
Fig. 4. The structure of input/output formatter

Mpr2, Mpr4의 매트릭스 곱셈은 ROM, 덧셈기와 레지스터로 연결되어 있는 RAC(ROM and Accumulator in Cascade)을 사용한다.[5,8] RAC에 대해 살펴보면, X가 변수들 $Y_0, \ldots, Y_3$에 의해 식 (26)에 따라 구해진다고 가정하자.

$$x=C_0Y_0+C_1Y_1+C_2Y_2+C_3Y_3 \tag{26}$$

또한 변수 X와 $Y_0, \ldots, Y_3$는 식 (27)과 같이 n비트의 2의 보수(2s complement) 값으로 표현되어 있다고 가정하자.

$$Y_0 = -2^{n-1}y_{0,n-1} + \sum_{i=0}^{n-2} 2^i y_{0,i} \tag{27}$$

그러면 X는 식 (28)과 같이 표현할 수 있다.

$$X = \sum_{k=0}^{3} C_k\left(-2^{n-1}y_{k,n-1} + \sum_{i=0}^{n-2} 2^i y_{k,i}\right)$$

$$ROM_{content} = \sum_{k=0}^{3} y_{k,i}C_k \tag{28}$$

위의 식 (28)에서 yk,i 들의 가능한 모든 경우에 대하여 미리 의 값을 구하여 $ROM_{content}$에 저장하여 놓고 $Y_0, \ldots, Y_3$의 비트 시리얼 입력에 대해 중간 값을 계속하여 더하면 최종적으로 원하는 X의 값을 얻을 수 있다. 위의 과정을 그림으로 나타내면 그림 5와 같다.

그림 3에서 RAC2와 RAC4는 입력 비트의 차이와 ROM의 내용에 따라 구분되며 그림 6의 구조로 되어 있다. RAC2는 입력 비트가 2, RAC4는 입력 비트가 4이다. 그리고 이 입력 비트 수에 해당하는 만큼의 RAC이 RAC2와 RAC4에 존재한다.

RAC은 그림 7(a)의 구조로 동작 과정을 살펴보면, 먼저 입력 비트에 따른 ROM 어드레싱(addressing)이 이루어져 해당 주소의 ROM 값(16비트 워드)이 상,하위 ROM에서 각각 출력되고, 이 출력 값은 18비트 덧셈기에 의해 가산된다. 이때 하위 ROM 값은 상위 ROM 값에 비해 한 자리 상위 비트에 대한 결과 치로서 한 비트 시프트레프트되어 17비트로 18비트 덧셈기에 입력된다.

18비트 덧셈기의 결과치는 19비트 덧셈기에 의해 AC에 저장하고 있는 값과 더해지며, 8사이클 후 최종 계산된 16비트 결과치는 병렬-시리얼 변환기로 전달된다. 병렬-시리얼 변환기는 16비트 결과치를 LSB부터 2비트씩 순차적으로 출력한다.

RAC 구조는 DCT-B의 RAC4내에 사용된 RAC을 제외하고 모두 같은 구조를 가진다. DCT-B의 RAC4에서 사용하는 RAC은 기존의 8+1 DCT용 ROM과 4+1 DCT용 ROM을 함께 가지고 있고, 블럭 크기에 따라 해당 ROM 값이 선택되어 매트릭스 곱셈이 수행되는데, 그 구조는 그림 7(b)와 같다.

분산산술처리에 의거한 DCT 구조는 매우 효율적인 것으로 나타나 있고, 이 때 RAC의 수와 RAC에 쓰이는 ROM의 크기는 분산산술처리의 DCT 구현에 필요한 회로 소자수 및 복잡도에 크게 영향을 미친다. 그런

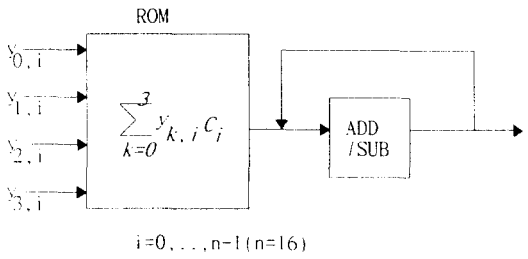
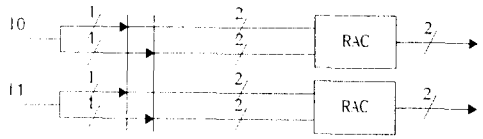


그림 5. RAC의 일반적 구조도
Fig. 5. The architecture of RAC

(a) RAC2



(b) RAC4

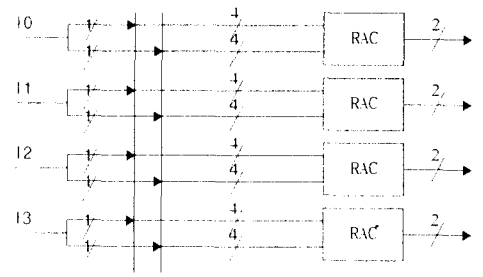
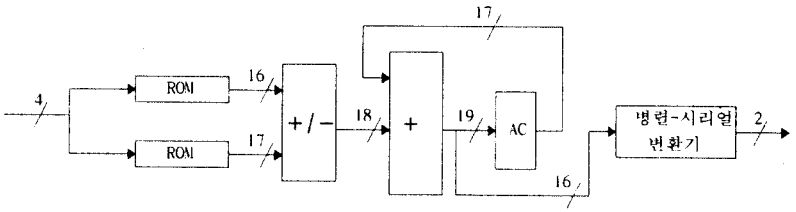


그림 6. RAC2/RAC4 구조도
Fig. 6. The structure of RAC2/RAC4

(a) DCT-A의 RAC4 구조



(b) DCT-B의 RAC4 구조

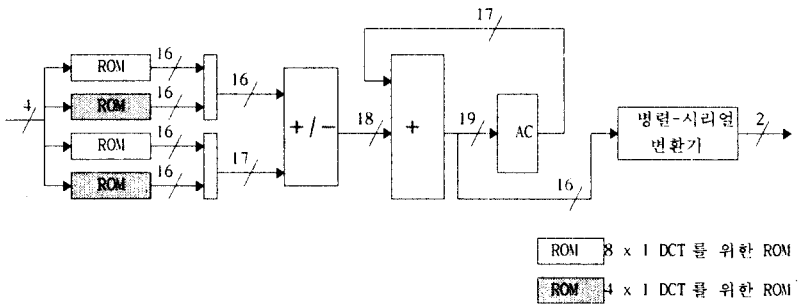


그림 7. RAC 구조도
Fig. 7. The structure of RAC

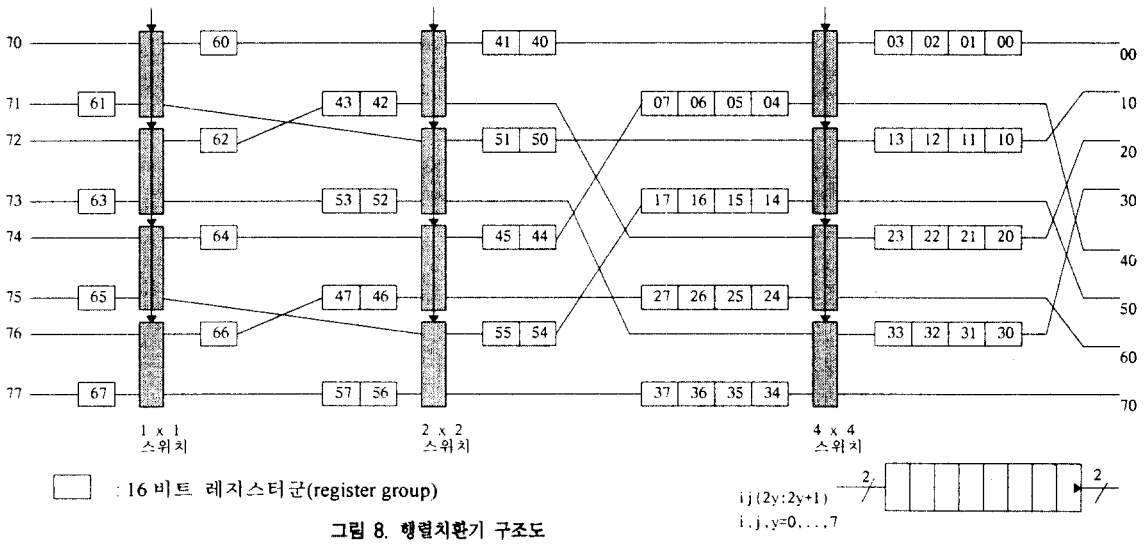


그림 8. 행렬치환기 구조도
Fig. 8. The architecture of a transposition network

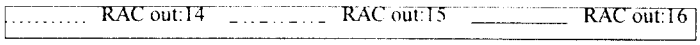
데 기존의 1-D DCT는 정방향과 역방향에 따라 RAC이 달라 하드웨어가 증가하는 문제가 있었다. 그리고 RAC내의 ROM 크기는 입력 벡터의 비트 수에 의해 결정되는 것으로 입력이 4-비트이면 ROM의 크기는 24가 된다. 그러나 본 DCT의 RAC에서는 덧셈기를 하나 추가하고 두개의 ROM을 사용하여 ROM의 크기를 24

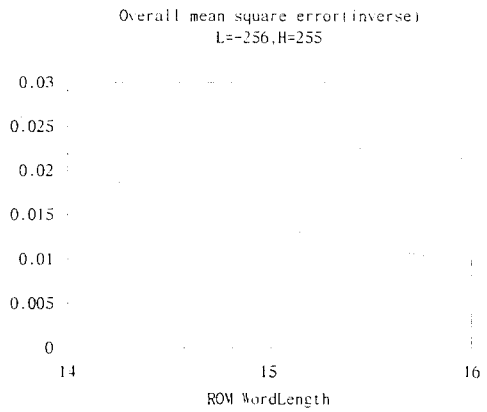
에서 2x22으로 줄였고, 정방향과 역방향시 같은 RAC을 사용하여 하드웨어의 효율성을 높였다.

4. 행렬 치환기

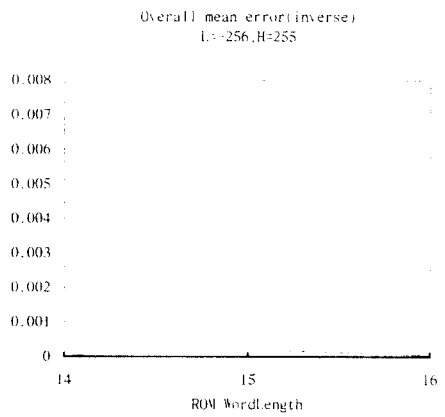
행렬치환기는 2-D DCT에서 행 순서를 열 순서로 또는 열 순서를 행 순서로 데이터를 치환한다. 행렬치환기의 구조는 수평, 수직으로 시프트를 하는 양방향 레지스터 어레이를 사용하는 방법[19]과 메모리를 사용하는 방법[6,8,12,23]등으로 구분할 수 있다.

양방향 레지스터 어레이를 통해 치환을 하는 방법은 짧은 임계 패스와 하드웨어 설계가 용이한 장점을 가지나, 많은 상호 연결이 필요하여 전체적으로 많은 실리콘 면적을 차지하는 단점이 있다. 메모리를 통한 행렬 상호 변환은 읽기와 쓰기가 동시에 수행되는 치환(transposition) RAM을 사용하거나, 핑퐁(ping-pong)방식으로 할 수 있다. 전자의 경우 RAM에 읽기와 쓰기가 한 클럭 주기안에 이루어져야 한다. 이는 RAM의 성능에 따른 지연 시간의 증가로 고속 동작시 제약이 된다. 그리고 후자의 경우 2배의 RAM과 2비트 시리얼 변환 회로가 사용되므로 하드웨어 부담이 증가한다.

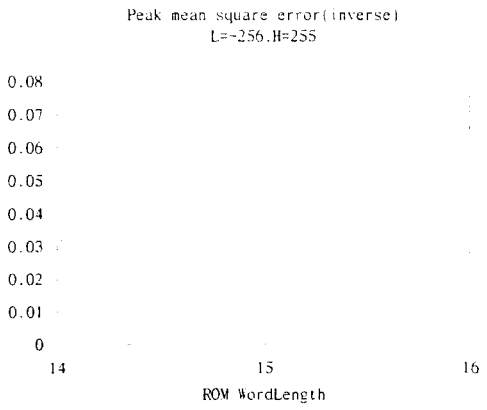




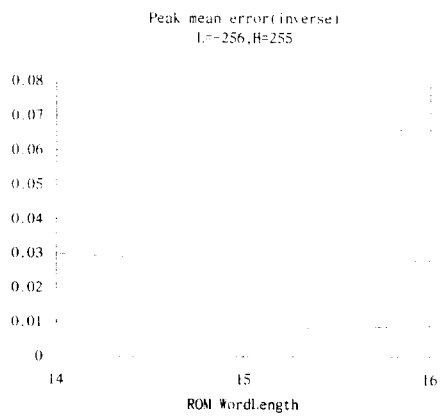
(a) overall 평균 평방 오차(역방향)
(a) Overall mean square error(inverse)



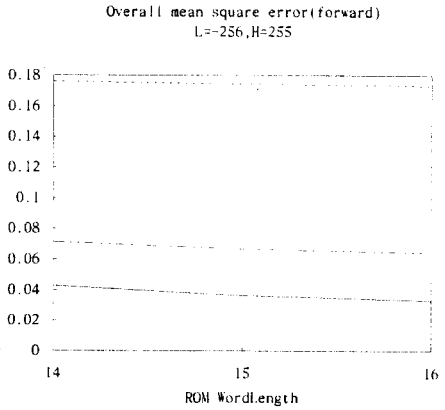
(b) overall 평균 오차(역방향)
(b) Overall mean error(inverse)



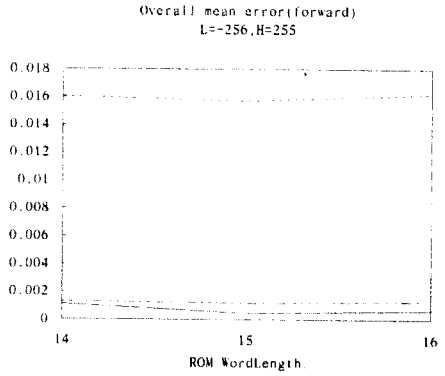
(c) 피이크 평균 평방 오차(역방향)
(c) Peak mean square error(inverse)



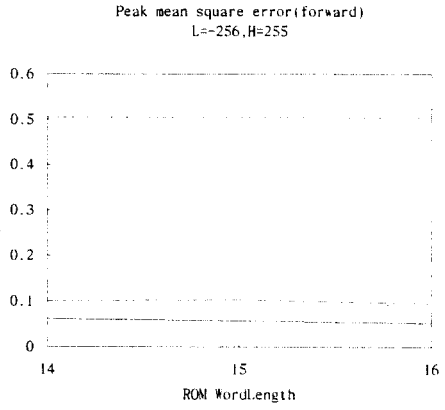
(d) 피이크 평균 평방 오차(역방향)
(d) Peak mean square error(inverse)



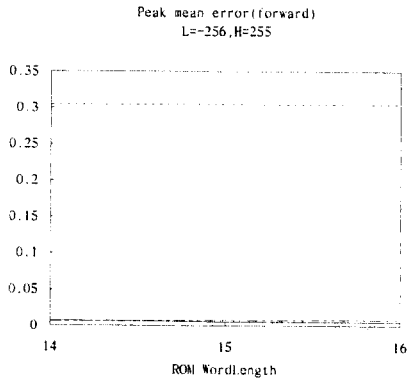
(e) overall 평균 평방 오차(정방향)
(e) Overall mean square error(forward)



(f) overall 평균 오차(정방향)
(f) Overall mean error(forward)



(g) 피이크 평균 평방 오차(정방향)
(g) Peak mean square error(forward)



(h) 피이크 평균 오차(정방향)
(h)Peak mean error(forward)

그림 9. 본 구조의 에러치
Fig. 9. The error performance of the proposed DCT

본 논문의 행렬치환기는 RAM을 사용한 구조보다 하드웨어와 지연 시간에 대한 요구 사항이 완화된 것으로 그림 8과 같이 멀티플렉서와 레지스터군의 조합인 1×1 스위치, 2×2 스위치, 4×4 스위치의 기본 단위들로 구성된다. 행렬치환기의 입출력은 8개의 채널을 통해 2비트씩 동시에 이루어지며[4], 입력된 2비트 데이터는 멀티플렉서 제어 신호에 의해 교차(cross)되거나 곧바로 통과된 후 클럭에 따라 다음 단계로 이동한다. 멀티플렉서 제어는 8클럭의 배수로 1×1 스위치에서는 8, 2×2 스위치에서는 16, 4×4 스위치에서는 32 클럭을 주기로 이루어진다.

V. DCT의 정확도

ITU-T의 H.261은 역방향 DCT의 정확도에 대한 규정[9]을 두고 있는데, 모든 DCT들은 이 규정을 만족시

켜야 한다. H.261의 IDCT 정확도 계산 과정은 여러 개의 -L(최소)과 H(최대) 사이의 값을 갖는 10,000 블럭의 8×8 랜덤(random) 화소 블럭을 발생시킨다. 화소의 범위에서 L은 최소값을, H는 최대값을 의미한다. 랜덤하게 발생된 화소 블럭 입력에 대해 64비트 부동 소수점 계산이 수행되어 적절한 DCT 계수가 구해진다. 그리고 이 DCT 계수를 가지고 64비트 부동 소수점 계산으로 역방향 DCT를 수행한 값과 제안한 DCT 회로가 수행한 역방향 DCT의 값을 비교한다. 물론 이때 제안된 역방향 DCT의 에러 오차가 주어진 규정치 값보다 작아야 한다.

예전에는 DCT의 설계에서 곱셈기의 숫자를 비용 측면에서 가장 중요한 요소로 보아 B.G.Lee[7]의 알고리즘을 포함한 고속 알고리즘이 직접(direct) 구현 방법보다 저렴하였다. 그러나 H.261의 에러 규정치를 만족하는 측면에서는 곱셈기의 크기, 채널 크기, 치환(transposition) RAM의 크기들이 커지므로 비용이 증가한다. 따라서 H.261 규정이 DCT 설계에 미치는 영향은 매우 크다.

본 논문에서는 H.261의 IDCT 에러 규정치를 만족하는지를 검증하기 위해 내부 정확도를 16비트로 하고 ROM의 워드 길이와 RAC 결과치의 비트 수를 달리하여 시뮬레이션을 수행하였다. 시뮬레이션은 3가지 경우의 (L=-256, H=255), (L=-300, H=300), (L=-5, H=5)에 대해 랜덤하게 발생시킨 10000 개의 8x8 블럭에 대해 이루어졌고, L=-256, H=255의 경우에 대해서만 정방향과 역방향으로 구분하여 H.261의 각 비교 항목에 관한 시뮬레이션 결과를 그림 9(a~h)에 나타내었다.

시뮬레이션 결과는 ROM의 워드 길이와 RAC의 결과치가 16비트 이하인 경우 IDCT의 에러 규정치를 모두 만족하지 못함을 나타내고 있다. 그래서 ROM의 워드 길이와 RAC의 결과치를 16비트로 H.261의 에러 규정치를 만족하는 본 구조의 에러 치를 요약하면 표 1과 같다.

Ⅵ. 설계 검증

DCT는 표준이 권고하는 사양을 모두 만족해야 한다. 하드웨어를 통한 직접적인 구조의 선정 및 구현에는 많은 시행착오가 생기며 이로 인해 전체적인 칩 설계의 기간이 길어진다.

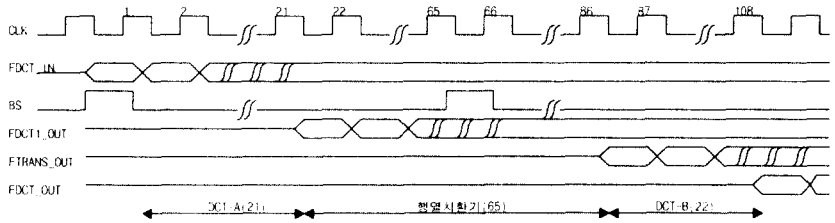
그래서 하드웨어의 구현에 앞서 표준을 만족하는 DCT 구조를 위해 소프트웨어 시뮬레이터를 설계하였다. 이 시뮬레이터는 C 언어로 구현이 되었고 추후 회로 설계 검증용으로 활용되었다. 또한 하드웨어를 고려하여 설계되어 있어 로직 설계 시 직접 매핑이 가능하였다. 소프트웨어 시뮬레이션을 거친 후의 회로 설계는 삼성의 1μ STD-50 표준셀 라이브러리를 사용하면서 추후 제작사 및 공정의 변경 등을 고려하여 제작사에 종속된 셀의 사용은 가급적 피하고 프리미트브(primitive)한 셀만을 사용하여 이루어졌다.

회로 설계의 검증은 우선 시뮬레이터의 결과와 일치하는 지의 여부를 확인, 회로의 수정 및 보완을 행하였다.[20] 회로 설계의 검증이 완료된 2-D DCT의 정방향 및 역방향 타이밍도는 그림 10과 같다. 정방향 변환된 DCT 계수는 픽셀 입력후 2개의 1-D DCT, 행렬치환기를 거치면서 108클럭 이후 부터 출력되고, 역방향 변

표 1. H.261 IDCT 규정치
Table 1. H.261 IDCT error performance

CCITT H.261		본 구조의 에러 치		
비교 항목	IDCT 에러 치	L,H=256,255	L,H=300,300	L,H=5,5
pixel peak error	1	1	1	1
Peak mean square error	0.06	0.013	0.0103	0.0116
overall mean square error	0.02	0.000938	0.000863	0.00931
peak mean eror	0.015	0.009	0.0068	0.00931
overall mean error	0.0015	0.00002	-0.00006	-0.00018

(a) 정방향 DCT 타이밍 다이어그램



(b) 역방향 DCT 타이밍 다이어그램

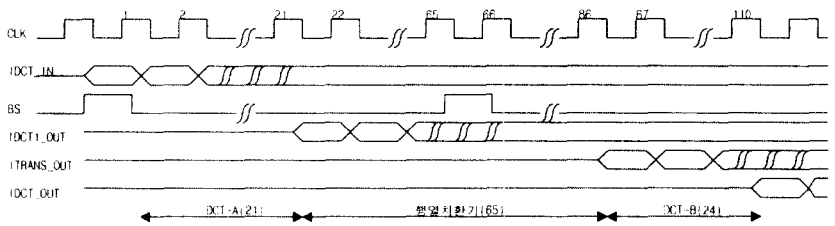


그림 10. 정방향/역방향 타이밍도
Fig. 10. Timing Diagram of 2-D DCT/IDCT

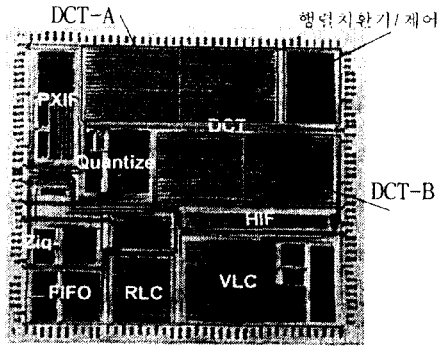


그림 11. JPEG 칩 내에서의 DCT 회로도
Fig. 11. The DCT circuit diagram in the JPEG Chip

환된 픽셀 데이타는 정방향의 역순으로 110 클럭 이후에 출력된다. 행렬치환기가 2개의 1-D DCT 보다 많은 처리 시간을 요구하는 것은 본 논문이 채택한 행렬치환기의 구조에 기인하는 것으로, 64개의 1-D DCT 결과를 행렬치환기내의 시프트 레지스터에 임시 저장하고 시프트와 멀티플렉서 제어를 통해 행렬치환을 수행하기 때문이다.

회로 설계의 검증이 완료된 본 논문의 DCT 구조에서 8x8 DCT 회로는 JPEG칩인 그림 11의JESP(Jpeg Engine for Scan layer Processing)에 사용이 되었다. JESP 칩은 한국전자통신연구소 미디어 연구실에서 개발한 JPEG 압축 복원 칩이다. 그림 11의 DCT-A, DCT-B는 1-D DCT이고, 행렬치환기/제어는 행렬치환기와 DCT의 칩제어를 담당하는 회로이다. JESP 칩내에서 8×8 DCT는 1μ CMOS 표준셀의 테크놀러지로 게이트 수는 30,000 게이트 수준이고 동작 클럭은 40MHz이다.

Ⅶ. 결 론

국제 표준 알고리듬인 JPEG, H.261, MPEG 표준에서는 8×8 화소 블럭에 대한 DCT를 근간으로 하나, DVCR 규격에서는 8×8 DCT 외에도 2개의 4×8 DCT 기능을 추가로 요구한다. 그래서 본 논문에서는 8×8 DCT와 2개의 4×8 DCT로 사용 가능한 VLSI 구조를 제시하였는데, 이는 구조적인 정형성을 확보하고, 실리콘 면적을 적게 하며, 고속으로 동작하는 측면에 중

점을 두었다. 특성을 살펴보면 구조는 분산산술처리 방식과 고속 알고리즘의 절충형을 기반으로, 블럭의 크기는 2가지로 8×8 화소 블럭과 2개의 4×8 화소 블럭이다. ITU-T H.261 에러 규정치를 만족하는 2-D DCT에서 픽셀 데이터는 9비트, DCT 계수는 12비트이다. 본 논문의 DCT 구조에서 8×8 DCT만을 처리하는 회로는 한국전자통신연구소 미디어 연구실에서 개발한 JPEG 압축 및 복원 칩[20]의 DCT로 사용이 되었다.

참고문헌

1. 김기철, 민병기, 최장식, "정방향/역방향 DCT를 위한 효율적인 VLSI 구조", 대한전자 공학회 추계종합학술대회 논문집, 1993.
2. JPEG-KOREA, "DIS 10918 Digital Compression and Coding of Continuos-Tone Still Images", 1992.
3. N.Ahmed,T.Natarajan, and K.R.Rao, "Discrete Cosine Transform", *IEEE Trans. Computers*, vol. COM-23, no. 1, pp.90-93, 1974.
4. JC. Carlac, P. Penard, JL. Sicre, "TCAD:a 27 MHz 8x8 Discrete Cosine Transform Chip", *ICASSP*, pp.2429-2432, 1989.
5. A.Peled and B.Liu, "A New Hardware Realization of Digital Filters", *IEEE Trans. Acoustics, Speech,and Signal Processing*, vol. ASSP-22, no. 6, pp.456-462, 1974.
6. S. Uramoto, Y. Inoue, "A. Takabatake, J. Takeda, Y. Yamashita, H. Terana,A 100-MHz 2-D Discrete Cosine Transform Core Processor", *IEEE J. Solid-State Circuits*, vol. 27, no. 4, pp.492-499, 1992.
7. B.G.Lee, "A New Algorithm To Compute The Discrete Cosine Transform", *IEEE Trans. Acoustics, Speech, and Signal Processing*, vol. ASSP-32, no. 6, pp.1243-1245, 1984.
8. M.T. Sun, T.C. Chen, and A.M. Gottlieb, "VLSI Implementation of a 16×16 Discrete Cosine Transform", *IEEE Trans. Circuits and Systems*, vol. 36, no. 4, pp.610-617, Apr., 1989.
9. ITU-R Rec.H.261(Formerly CCITT Rec.H.261), "Video Codec for Audiovisual Services at px64 kbit/s", Geneva, August, 1990.
10. E. Scopa, "A. Leone,R. Guerrieri, G. Baccarani, A 2D-DCT Lower-Power Architecture for H. 261 Coders", *ICASSP*, vol. 4, pp.3271-3274, 1995.
11. 박구만, "가정용 DVCR의 최신 기술 및 표준화 동향", 전자공학회지 제22권 제1호, pp.76-86, 1995.
12. 박현상, 이우용, 나종범, "HDTV용 8x8 DCT/IDCT Processor의 구현", DSP 설계 및 응용워크숍 논문집, pp.125-128, 1995.
13. ISO/IEC 11172-2(MPEG-1), "Information Technology - Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to about 1.5Mbits/s- Part2 Video", 1993.
14. ISO/IECT 13812-2(MPEG-2), "Committee Draft/ITU-T Draft Rec H.262,Information Technology - Generic Coding of Moving Pictures and Associated Audio - part2 Video", Nov. 5, 1993.
15. Chen-Mie Wu,Andy Chiou, "A SIMD-Systolic Architecture and VLSI Chip for the Two-Dimensional DCT and IDCT", *IEEE Trans.Consumer Electronics*, vol. 39, No. 4, pp.859-869, Aug., 1993.
16. Chin-Liang Wang, Chang-Yu Chen, "High-Throughput VLSI Architectures for the 1-D and 2-D Discrete Cosine Transforms", *IEEE Trans. Circuits and Systems forVideo Technology*, vol. 5, no. 1, pp.135-146, Feb., 1995.
17. H.S.Hou, "A Fast Recursive Algorithm for Computing The Discrete Cosine Transform", *IEEE Trans. Acoust, Speech, Signal Processing*, vol. ASSP-35, pp.1455-1461, Oct., 1987.
18. ZJ.Mou,F.Jutand, "A High-Speed Low-Cost DCT Architecture for HDTV Applications", *ICASSP*, vol. 2, pp.1153-1156, 1991.
19. U.Sjostrom,I.Defilippis, and M.Ansorg,

"Discrete Cosine Transform Chip for Real-time Video Applications", *ISCAS*, pp.1620-1623, 1990.

20. 민병기, 최장식, "JPEG 압축복원", DSP 설계 및 응용워크숍 논문집, pp.113-118, 1995.
21. J.Boyce, F. Lane, "Fast Scan Technology for Digital Video Tape Recorders", *IEEE Trans. Consumer Electronics*, vol. 39 , no. 3 ,pp.186-191, Aug., 1993
22. M.Madisetti, "A.N.Willson,A 100Mhz 2-D 8x8 DCT/IDCT processor for HDTV application", *IEEE Trans. CAS for Video Tech.*, vol. 5, pp.158-165, April, 1995.
23. 김기철, "이산여현변환을 위한 병렬 전치 메모리", 한국통신학회 논문지, 제20권 제6호, pp.1678-1689, 1995년, 6월.
24. N.I.Cho, S.U.Lee, "Fast Algorithm and implementation of 2-D DCT", *IEEE Trans. CAS*, vol. 38, pp.297-305, Mar., 1991.

崔 長 植(Jang Sik Choi) 정회원

1965년 8월 9일
1991년 2월 : 전북대학교 전자계산기공학과(학사)
1991년 1월~현재 : 한국전자통신연구소 연구원