論文2002-39SD-12-8

가변길이 다중비트 코딩을 이용한 DCT/IDCT의 설계

(Variable Radix-Two Multibit Coding and Its VLSI Implementation of DCT/IDCT)

金 大 原 *, 崔 峻 林 *

(Dae Won Kim and Jun Rim Choi)

요 약

본 논문은 가변길이 다중비트 코딩 알고리듬을 제안하고 DCT/IDCT(이산여현변환/역이산여현변환)설계에의 적용 과정을 제시한다. 가변길이 다중 비트 코딩은 일반적인 Booth's알고리듬과 같이 중첩에 의한 다중비트 코딩을 가변적인 방법을 사용하여 그 중 2의 멱승이 되는 부분 즉 2k의 SD(Signed Digit)을 생성하는 방법이다. 이렇게 발생된 SD는 곱셈에 있어서 2k의 부분적(Partial Product)을 생성하게 되고 이로 인해 필요한 하드웨어는 단순한 덧셈기와 쉬프트 연산에 필요한 플립플롭만 필요하게 되므로 설계과정에 있어서 칩의 면적과 속도 면에서 효율적인 방법이다. 본 논문에서는 이 알고리듬의 정의 및 증명과정과 실제 알고리듬 적용을 위한 DCT/IDCT의 설계방법을 논의하고 제작한 IDCT의 결과에 대해 논의한다. 설계된 IDCT칩은 병렬 고속 처리를 위한 8개의 PE(Processing Element)와 하나의 전치 메모리를 사용한 방법으로 54MHz에서 400Mpixels/sec의 동작속도를 가지며 HDTV 및 MPEG 디코더에 적용하여 동작을 검증하였다.

Abstract

In this paper, variable radix-two multibit coding algorithm is presented and applied in the implementation of discrete cosine transform(DCT) and inverse discrete cosine transform(IDCT). Variable radix-two multibit coding means the 2k SD (signed digit) representation of overlapped multibit scanning with variable shift method. SD represented by 2k generates partial products, which can be easily implemented with shifters and adders. This algorithm is most powerful for the hardware implementation of DCT/IDCT with constant coefficient matrix multiplication. This paper introduces the suggested algorithm, it's proof and the implementation of DCT/IDCT. The implemented IDCT chip with 8 PEs(Processing Elements) and one transpose memory runs at a rate of 400 Mpixels/sec at 54MHz frequency for high speed parallel signal processing, and it's verified in HDTV and MPEG decoder.

Keyword : Booth's algorithm, DCT, IDCT, overlapped scanning, signed digit(SD) arithmetic, distributed arithmetic(DA), high speed multiplication.

Ⅰ. 서 론

Booth's 알고리듬은 곱셈 연산 시 승수를 2비트씩 중첩시켜 코딩하는 방법을 이용하여, 생성된 부분적은 단순히 시프트와 덧셈만으로 구현되는 아주 널리 알려진 방법이다.[1] 이런 성과 이후 MacSorley[2]는 Booth가 제안한 알고리듬을 3비트씩 코딩하는 방법으로 변형하였고 Sam과 Gupta[3]에 의해서 위의 코딩방법을 다중비트에 적용하는 일반적인 방법으로 제시하고 이를 증명하였다. 이런 알고리듬들은 공통적으로 비트 형태로 표현된 수를 코딩할 때 하위 비트를 추가하여 이를 이전 비트와 중첩시키면서 코딩하는 방법으로 정해진 길

* 正會員, 慶北大學校 電子電氣工學部
(School of Electrical Engineering and Computer Science Kyungpook National University)

接受日字:2002年3月18日, 수정완료일:2002年11月27日

이 만큼만 시프트를 하면서 코딩하게 된다.[4] 그러나 이런 방법들은 부스 알고리듬처럼 2비트 코딩의 경우 효율적이지만 비트가 커짐에 따라 발생되는 SD(signed digit)의 값이 부가적인 덧셈연산을 필요로 하는 단점이 있다. 본 논문에서는 이 단점을 없애기 위해 정해진 길이만큼 시프트 하여 SD를 생성하는 방법대신 그 값이 2k가 되도록 SD를 생성하는 가변길이 다중비트 코딩방법을 제시하고 이에 따른 설계 방안에 대해 논하고자 한다. 이 방법은 비트 단위로 표현된 데이터의 코딩방법이 가변적이므로 고정된 승수를 사용하는 곱셈연산, 혹은 행렬연산에 적용되어질 수 있고 이런 연산들은 디지털 필터 혹은 각종 변환에 있어서 필수적인 연산들이다. 결과적으로 코딩된 값을 이용한 곱셈은 단순한 데이터 패스의 배선만으로 그 부분적을 구할 수가 있으므로 속도 및 면적 소모에도 큰 효율성을 발휘한다. 본 논문에서는 고정변수를 사용하는 DCT/IDCT의 설계를 통해 제시된 알고리듬을 검증하였다. DCT/IDCT 설계에 있어서 기본적인 곱셈을 수행할 기본모듈 PE(Processing Element)는 0.6μm triple metal CMOS 공정에서 Verilog HDL을 사용하여 시뮬레이션 한 결과 DCT의 경우 8.2K, IDCT의 경우 9.2K의 게이트 수로 설계가 가능하였다. 그리고 최종적으로 설계된 HDTV용 IDCT core의 경우 IEEE STD 1180-1990[5]을 만족하고 54MHz에서 400Mpixels/sec로 동작하는 빠른 속도를 보였다. 2장에서는 가변길이 다중비트 코딩에 대한 정의 및 증명을 기술하고 3장에서는 제안된 알고리듬을 이용한 하드웨어 적용 방법으로 DCT/IDCT core 설계를 전개하며, 4장에서는 IDCT core의 구현 결과를 설명한다.

Ⅱ. 2k SD 및 가변길이 다중비트 코딩

1. 일반적인 다중비트 코딩과 가변길이 다중비트 코딩

두 수의 곱셈에 있어서 승수 X가 곱셈기의 입력이고 곱해지는 피승수 Y는 고정된 계수일 경우 Y는 식 (1)과 같은 n비트의 2의 보수표현으로 나타낼 수 있다.

$$Y = -y_{n-1}\cdot 2^{n-1} + \sum_{q=0}^{n-2} y_q \cdot 2^q \qquad (1)$$

만약 Y를 일반적인 2^k (k≥1, k는 임의의 정수형) 형태의 SD 표현으로 나타낸다면 Y는 식 (2)와 같이 다시 나타낼 수 있다.

$$Y = \sum_{i=0}^{\frac{n}{k}-1} D_i \cdot 2^{ki}$$

$$where,\quad D_i = y_{ki-1} + \sum_{j=1}^{k-1} y_{ki+j-1}\cdot 2^{j-1} - y_{k(i+1)-1}\cdot 2^{k-1} \qquad (2)$$

식 (2)의 표현은 일반적인 경우 다중비트 코딩의 예이다. 정의에 의하여 y_{-1}=0이고 y_0의 오른쪽에 즉 최하위 비트로 첨가된다. <표 1>은 k 값이 바뀜에 따라 SD인 Di의 값에 대한 결과를 요약한 표이다.

표 1. 코딩되는 비트수에 따른 SD의 값
Table 1. SD of each coding schemes.

코딩된 비트수 (k+1)	D_i(SD의 값)
2 (k=1)	{-1, 0, 1}
3 (k=2)	{-2, -1, 0, 1, 2}
4 (k=3)	{-4, -3, ..., 0, ..., 3, 4}
5 (k=4)	{-8, -7, ..., 0, ..., 7, 8}
n (k=n-1)	$\{-2^{k-1}, -2^{k-1}+1, ..., 0, ..., 2^{k-1}-1, 2^{k-1}\}$

위의 <표 1>에서 보면 k값이 3이상이 될 때 SD의 값이 2^k가 아닌 경우에 있어서 부분적을 구해보면 D_i=3일때 $3X=2^1X+2^0X$ 혹은 D_i=7일 때 $7X=2^3X-2^0X$로 부가적인 덧셈 혹은 뺄셈의 연산이 필요하다. 그러나 만약 SD의 값이 항상 2k가 된다면 단순한 시프트만으로 부분적을 구할 수 있다. 가변길이 다중비트 코딩이란 2^k가 되는 SD의 값을 구하기 위해 고정된 k값을 두는 것이 아니라 k값을 가변 하면서 코딩하는 방법이다. 2^k SD의 일반적인 표현을 위하여 식 (2)의 D_i를 다른 방법으로 나타내도록 한다.

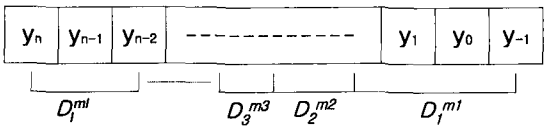


그림 1. n 비트를 가지는 Y의 2의 보수 표현
Fig. 1. 2's Complement expression of Y with n bits.

<그림 1>은 Y 값을 n 비트의 2의 보수 표현으로 나타낸 것이다. y_n은 MSB, y_0은 LSB, y_{-1}은 코딩을 위해 추가된 비트로 이전 설명과 같이 그 값은 0이다. 그리

고, D_l^{ml}에서 ml비트 개수로 코딩된 l번째의 SD값이고 그 값은 2^k로 표현되는 값이다. 또, l값은 Y의 최종 코딩된 SD의 개수를 의미한다. 여기서 일반적인 2^k SD를 표현하기 위하여 <그림 1>로부터 D_l^{ml}을 식 (2)와 같이 유도하여 보면, 식 (3)과 같다.

$$D_1^{m1} = y_{-1} + y_0 \cdot 2^0 + y_1 \cdot 2^1 + \Lambda\Lambda\Lambda - y_{m1-2} \cdot 2^{m1-2}$$
$$= y_{-1} + \sum_{i=0}^{m1-3} y_i 2^i - y_{m1-2} \cdot 2^{m1-2} \quad (3)$$

마찬가지로 D_2^{m2}부터 D_l^{ml}까지 나타내면 식 (4), (5), (6)과 같다.

$$D_2^{m2} = y_{m1-2} + \sum_{i=0}^{m2-3} y_{i+(m1-1)} 2^i - y_{m1+m2-3} \cdot 2^{m2-2} \quad (4)$$

$$D_3^{m3} = y_{m1+m2-3} + \sum_{i=0}^{m3-3} y_{i+(m1+m2-2)} 2^i - y_{m1+m2+m3-4} \cdot 2^{m3-2} \quad (5)$$

. . . . . .

$$D_l^{ml} = y_{m1+m2+\Lambda+m(l-1)-l}$$
$$+ \sum_{i=0}^{ml-3} y_{i+(m1+m2+\Lambda+m(l-1)-(l-1))} 2^i - y_{m1+m2+m3+\Lambda+m(l-1)-(l-2)} \cdot 2^{ml-2} \quad (6)$$

여기서 식 (6)은 2^k SD의 일반적인 식이나 이 식은 너무 복잡하므로 y의 index를 일반적인 방법으로 나타내어 위의 표현을 간단하게 표현할 수 있다. 그러므로 여기서 새로운 index인 Li를 정의한다.

$$Li = \sum_{j=1}^{i-1} m_j - i \quad (where \quad if \quad i=1, \quad then \sum_{j=1}^{i-1} m_j = 0) \quad (7)$$

결과적으로 식 (7)을 이용하여 나타낸 2^k SD의 일반적인 식의 표현은 식 (8)과 같다.

$$D_i^{mi} = y_{Li} + \sum_{j=0}^{mi-3} y_{j+(Li+1)} \cdot 2^j - y_{L(i+1)} \cdot 2^{mi-2}$$
$$(where, \quad mi \geq 3) \quad (8)$$

그러므로, 새로 정의된 D_i^{mi}로 (2)의 식처럼 Y를 표현하면 식 (9)와 같이 된다.

$$Y = D_1^{m1} \cdot 2^0 + D_2^{m2} \cdot 2^{m1-1} + D_3^{m3} \cdot 2^{m1+m2-2}$$
$$+ \cdots\cdots + D_l^{ml} \cdot 2^{m1+m2+m3+\cdots+m(l-1)-(l-1)} \quad (9)$$

식 (9)에서 우변의 제일 마지막 항을 다시 한번 살펴보면 이전 정의한 Li의 표현으로 다음과 같이 나타낼 수 있다.

$$m1+m2+m3+\Lambda\Lambda\Lambda+m(l-1)-(l-1) = \sum_{i=1}^{l-1} mi - l + 1 = Li + 1 \quad (10)$$

그러므로 최종적으로 Y를 2^k SD의 일반적인 형태로 나타내면 식 (11)과 같다.

$$Y = \sum_{i=1}^{l} D_i^{mi} \cdot 2^{Li+1}$$
$$where, \quad D_i^{mi} = y_{Li} + \sum_{j=0}^{mi-3} y_{j+(Li+1)} \cdot 2^j - y_{L(i+1)} \cdot 2^{mi-2}$$
$$Li = \sum_{j=1}^{i-1} m_j - i \quad (11)$$

2. 가변길이 다중비트 코딩의 예

2^k SD의 일반식이 실제 코딩에서 어떻게 사용되는지에 대하여 알아보도록 하겠다.

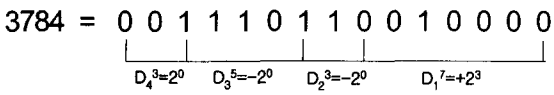


그림 2. 3784의 2의 보수표현 및 2^k SD
Fig. 2. 2's Complement of 3784 and SD with 2^k.

<그림 2>는 예를 들기 위하여 고정된 계수 3784의 2의 보수 표현이다. 만약 전체 2^k SD 개수 즉 l=4 이고 m1=7, m2=3, m3=5, m4=3 이면 식 (11)을 이용하여 다음과 같이 Li를 구할 수 있다.

$$L1 = -1$$
$$L2 = m1 - 2 = 5$$
$$L3 = m1 + m2 - 3 = 7$$
$$L4 = m1 + m2 + m3 - 4 = 11$$
$$L5 = m1 + m2 + m3 + m4 - 5 = 13$$

그러므로 위에서 구한 Li 값으로부터 $D_i$mi 값을 구해보면 다음과 같이 구할 수 있고 구한 결과는 3784과 일치함을 볼 수 있다.

$$D_1^7 = 0 + 0 \cdot 2^0 + 0 \cdot 2^1 + 0 \cdot 2^2 + 1 \cdot 2^3 + 0 \cdot 2^4 - 0 \cdot 2^5 = 8$$
$$D_2^3 = 0 + 1 \cdot 2^0 - 1 \cdot 2^1 = -1$$
$$D_3^5 = 1 + 0 \cdot 2^0 + 1 \cdot 2^1 + 1 \cdot 2^2 - 1 \cdot 2^4 = -1$$
$$D_4^3 = 1 + 0 \cdot 2^0 - 0 \cdot 2^4 = 1$$
$$Y = 8 \cdot 2^0 - 1 \cdot 2^6 - 1 \cdot 2^8 + 1 \cdot 2^{12} = 3784$$

그러나, 실제 코딩 과정에 있어서 l 및 ml의 값을 미리 구하는 것이 가변 길이 다중비트 코딩에 있어서 가

장 우선적인 일이다. 이런 작업을 하드웨어적으로 구현하기란 쉬운 일은 아니다 그러나 다음과 같은 방법으로 2^k SD를 찾는 것은 그다지 어려운 일은 아니다. 2^k SD를 찾는 방법을 다음 흐름도와 같이 요약할 수 있다.

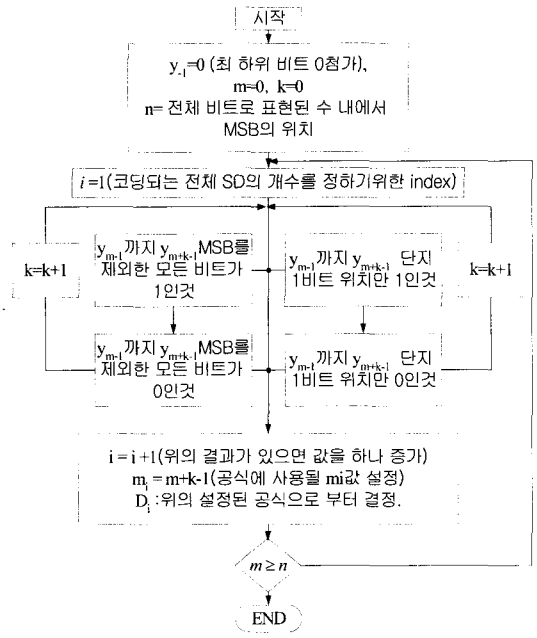


그림 3. 2^k SD를 찾기 위한 순서도
Fig. 3. The Flow chart for 2^k SD.

3. 가변길이 다중비트 코딩의 증명

가변길이 다중비트 코딩의 증명은 Sam 과 Gupta에 의한 방법을 이용하였다.[3] 증명과정은 여기서 구한 2k SD에 대한 결과를 Y에 대입하고 그 결과가 2의 보수 표현과 동일하면 된다. 위의 식 (11)로부터 Dimi를 대입하면 Y는 식 (12)와 같이 나타나고 이를 다시 정리하면 식 (13)과 같다.

$$Y=\sum_{i=1}^{l}(y_{Li}+\sum_{j=0}^{mi-3}y_{j+Li+1}\cdot 2^{j}-y_{L(i+1)}\cdot 2^{mi-2})\cdot 2^{Li+1} \tag{12}$$

$$\begin{aligned}Y=\;&(y_{-1}+y_0\cdot 2^0+y_1\cdot 2^1+\Lambda-y_{m1-2}\cdot 2^{m1-2})\cdot 2^0\\&+(y_{m1-2}+y_{m1-1}\cdot 2^0+y_{m1}\cdot 2^1+\\&\Lambda+y_{m1+m2-4}\cdot 2^{m2-3}-y_{m1+m2-3}\cdot 2^{m2-2})\cdot 2^{m1-1}\\&+(y_{m1+m2-3}+y_{m1+m2-2}\cdot 2^0+y_{m1+m2-1}\cdot 2^1+\\&\Lambda+y_{m1+m2+m3-5}\cdot 2^{m3-3}-y_{m1+m2+m3-4}\cdot 2^{m3-2})\cdot 2^{m1+m2-2}\\&+\Lambda\Lambda\Lambda\\&+(y_{m1+m2+\Lambda+m(l-1)-l}+y_{m1+m2+\Lambda+m(l-1)-(l-1)}\cdot 2^0+\\&+y_{m1+m2+\Lambda+m(l-1)-(l-2)}\cdot 2^1+\Lambda\Lambda\\&-y_{m1+m2+\Lambda+m(l-1)+ml-(l+1)}\cdot 2^{ml-2})\cdot 2^{m1+m2+\Lambda+m(l-1)-(l-1)}\end{aligned} \tag{13}$$

$2^{i+1}-2^i=2^i$ 을 이용하여 식 (13)의 괄호를 제거하면, 식 (14)와 같다.

$$\begin{aligned}Y=\;&y_{-1}+y_0\cdot 2^0+y_1\cdot 2^1+\Lambda+y_{m1-2}\cdot 2^{m1-2}+\\&\Lambda+y_{m1+m2-3}\cdot 2^{m2-2}+\Lambda\Lambda\Lambda\\&+y_{m1+m2+\Lambda+m(l-1)+ml-(l+2)}\cdot 2^{m1+m2+m3+\Lambda+m(l-1)+ml-l-2}\\&-y_{m1+m2+\Lambda+m(l-1)+ml-(l+1)}\cdot 2^{m1+m2+m3+\Lambda+m(l-1)+ml-l-1}\end{aligned} \tag{14}$$

위의 y에 대한 index를 살펴보면 식 (15)와 같다.

$$m1+m2+m3+\Lambda\;\Lambda\;\Lambda+m(l-1)+ml-l=n \tag{15}$$

그리고 $y_{-1}=0$이고 식 (15)를 적용하여 다시 정리하면 식 (16)과 같고 이는 2의 보수 표현인 식 (1)과 동일함을 알 수 있다.

$$\begin{aligned}Y=\;&-y_{m1+m2+\Lambda+m(l-1)+ml-(l+1)}\cdot 2^{m1+m2+m3+\Lambda+m(l-1)+ml-l-1}\\&+y_0\cdot 2^0+y_1\cdot 2^1+\Lambda+y_{m1-2}\cdot 2^{m1-2}+\\&\Lambda+y_{m1+m2-3}\cdot 2^{m2-2}+\Lambda\Lambda\\&+y_{m1+m2+\Lambda+m(l-1)+ml-(l+2)}\cdot 2^{m1+m2+m3+\Lambda+m(l-1)+ml-l-2}\\=\;&-y_{n-1}\cdot 2^{n-1}+y_0\cdot 2^0+y_1\cdot 2^1+\Lambda+y_{n-2}\cdot 2^{n-2}\\=\;&-y_{n-1}\cdot 2^{n-1}+\sum_{i=0}^{n-2}y_i\cdot 2^i\end{aligned} \tag{16}$$

4. 부호 비트의 확장 방법.

일단 코딩된 값으로 구한 부분적의 경우 부호 비트에 대한 고려는 하지 않은 상태이다. 그러므로 두 수의 곱셈을 최종적으로 얻기 위하여 위에서 구한 부분적의 합에 부호 비트의 확장으로 인한 값을 구해 더해 주는 과정이 필요하다. 이 값 역시 이전에 정의한 L_i 및 m_i를 이용하여 나타낼 수 있다. 만약 승수 X가 p비트를 가지고 피승수 Y가 q비트를 가진다면 부호 비트의 확장 표현은 식 (17)과 같다. 식에서 S_{mi}는 코딩되는 m_i 비트들의 최상위비트 즉 부호 비트를 의미한다.

$$\begin{aligned}S=\;&\sum_{i=1}^{l}S_{mi}\cdot\sum_{j=p+ki+Li+}^{p+q-l}2^j\\=\;&\sum_{i=1}^{l}S_{mi}\cdot 2^{p+ki+Li+1}-\sum_{i=1}^{l}2^{p+ki+Li+1}+2^{p+q-1}\end{aligned} \tag{17}$$

예를 들어 가변길이 다중비트 코딩이 m1, m2, m3, m4비트로 이루어지고 2^k SD 값이 $D_1^{m1}=2^{k1}$,

$D_2^{m2}=2^{k2}$, $D_3^{m3}=2^{k3}$, $D_4^{m4}=2^{k4}$로 이루어져 있다면 식 (17)에 의한 부호 비트 확장 결과 값은 식 (18)과 같이 구할 수 있다.

$$S = S_{m1}\cdot\sum_{i=p+k1}^{p+q-1}2^i + S_{m2}\cdot\sum_{\substack{i=p+k2\\+m1-1}}^{p+q-1}2^i + S_{m3}\cdot\sum_{\substack{i=p+k3\\+m1+m2-2}}^{p+q-1}2^i + S_{m4}\cdot\sum_{\substack{i=p+k4\\+m1+m2+m3-3}}^{p+q-1}2^i \quad (18)$$

Ⅲ. 가변길이 다중비트 코딩을 이용한 DCT/IDCT 설계

1. 설계 적용을 위한 DCT/IDCT 알고리듬

DCT 알고리듬[6]은 선형이고 분리 가능한 변환에 기본을 둔 N×N의 변환 계수 X를 얻기 위해 역시 N×N의 변환 행렬 A를 이용하여 행렬 연산을 수행하는 것으로 알려져 있다. 그리고, 이의 역변환인 IDCT 역시 가역성을 지니고 있으므로 N×N의 원래 값을 같은 특성을 이용하여 구할 수 있다. 그것은 다음 식 (19)와 같이 두 번의 곱셈과 한번의 전치 과정으로 나타낼 수 있다.

$$\boldsymbol{X} = \boldsymbol{AxA}^T \rightleftharpoons \boldsymbol{x} = \boldsymbol{A}^T\boldsymbol{XA} \quad (19)$$

하드웨어 적용을 위하여 식 (19)를 이용한 Chen의 알고리듬을 이용하도록 한다.[7] 이는 8×1 DCT/IDCT의 계산 방법으로 잘 알려진 것으로 식 (20)과 같다.

$$\begin{bmatrix}X_0\\X_2\\X_4\\X_6\end{bmatrix}=\begin{bmatrix}A & A & A & A\\B & C & -C & -B\\A & -A & -A & A\\C & -B & B & C\end{bmatrix}\cdot\begin{bmatrix}x^0+x^7\\x^1+x^6\\x^2+x^5\\x^3+x^4\end{bmatrix}$$

$$\begin{bmatrix}X_1\\X_3\\X_5\\X_7\end{bmatrix}=\begin{bmatrix}D & E & F & G\\E & -G & -D & -F\\F & -D & G & E\\G & -F & E & -D\end{bmatrix}\cdot\begin{bmatrix}x^0-x^7\\x^1-x^6\\x^2-x^5\\x^3-x^4\end{bmatrix}$$

$$\begin{bmatrix}x_0\\x_1\\x_2\\x_3\end{bmatrix}\begin{matrix}x_7\\x_6\\x_5\\x_4\end{matrix}=\begin{bmatrix}A & B & A & C\\A & C & -A & -B\\A & -C & -A & B\\A & -B & A & -C\end{bmatrix}\cdot\begin{bmatrix}X_0\\X_2\\X_4\\X_6\end{bmatrix}\pm\begin{bmatrix}D & E & F & G\\E & -G & -D & -F\\F & -D & G & E\\G & -F & E & -D\end{bmatrix}\cdot\begin{bmatrix}X_1\\X_3\\X_5\\X_7\end{bmatrix}$$

$$where,\quad A=\cos\frac{\pi}{4},\quad B=\cos\frac{\pi}{8},\quad C=\sin\frac{\pi}{8},$$
$$D=\cos\frac{\pi}{\cdot},\quad E=\cos\frac{3\pi}{\cdot},\quad F=\sin\frac{3\pi}{\cdot},\quad G=\sin\frac{\pi}{\cdot} \quad (20)$$

행렬식 (20)에서 첫 번째 두 번째 행렬식은 각각 DCT 우수, 기수 행렬의 계산이고, 세 번째 행렬식은 IDCT 계산을 위한 기수 및 우수 행렬식이다. 여기서 DCT를 위한 기수 행렬 그리고 IDCT 연산의 기수 행렬이 동일하고 DCT의 우수 행렬과 IDCT의 우수 행렬 역시 단순히 전치과정만 거치면 되므로 하드웨어 디자인에 있어서 매우 반복적인 구조를 가진다. 그러므로 PE의 설계는 이런 반복적인 구조를 단순한 하드웨어 설계로 유도할 수 있고 입력이 선택되면 단순한 데이터 패스의 변화로 입력 X와 가변길이 다중비트 코딩을 한 계수들과의 곱셈을 실행할 수 있다.

2. 가변길이 다중비트 코딩을 실행하는 PE

일반적으로 2차원 DCT/IDCT를 설계하는 방법 중 분산 알고리듬[8]을 이용한 방법을 많이 사용한다. 이 방법은 ROM을 이용한 방법으로 입력이 들어오면 미리 계산된 입력을 이용하여 부분적을 생성하고 이를 축적

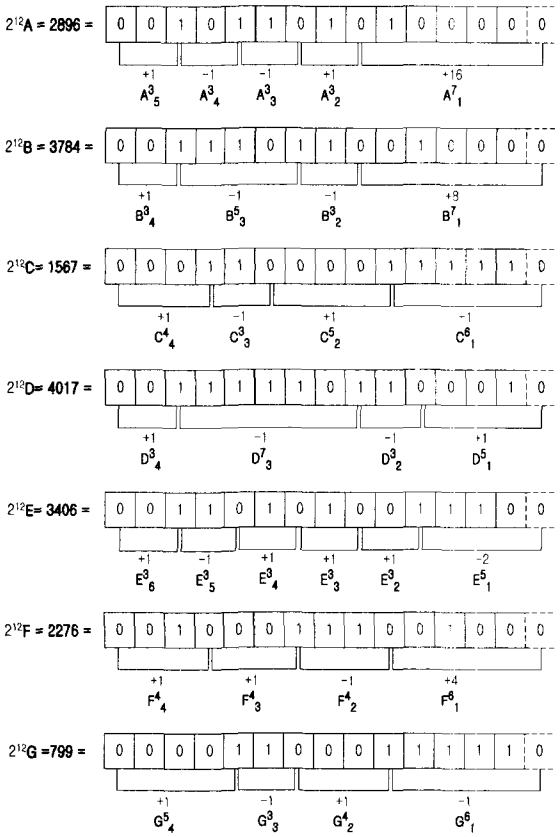


그림 4. 각 계수의 가변길이 다중비트 코딩 결과
Fig. 4. Variable Radix-2 mutibit Coding result of all coefficient for DCT/IDCT.

기(accumulator)를 이용하여 더하는 방법이다. IDCT 행렬식 연산을 예로 들어보면 3비트를 코딩하는 변형된 Booth 알고리듬의 경우 입력 하나를 곱하는데 7개의 부분적이 발생되고 하나의 확장된 부호 값을 더해주는 최종적으로 8번의 덧셈이 필요하다. 그러므로 한 열을 계산하기 위해서는 총 32번의 부분적을 더하는 덧셈이 필요하게 된다 만약 우리가 8개의 픽셀을 동시에 처리하기 위해서는 총 256개의 부분적을 더해야 한다. 그러나 분산 알고리듬을 사용하면 일반적인 방법으로 한 열을 계산하기 위해서는 16번, 총 128개의 부분적을 계산해야 하며 여기에 24개의 ROM이 필요하게 된다. 분산알고리듬을 이용한 방법이 비록 부분적의 개수는 적으나 부가적인 ROM이 필요하고 또 축적기의 사용으로 한번 연산에 많은 시간이 소모되므로 고속을 필요로 하는 영상신호에 있어서는 처리 속도를 증가시키기 어렵다. 가변길이 다중 비트 코딩은 각 행렬의 상수 계수에 적용되면 되므로 <그림 3>과 같이 코딩 할 수 있다.[9]

위의 DCT/IDCT 계수에 212을 곱한 이유는 내부 연산 시 12 비트의 크기로 계산되어 지기 때문이다. 코딩 결과를 바탕으로 하여 각 행렬에 대한 부분적의 개수를 구해 보면 식(21), (22)와 같다.

$$\begin{bmatrix}18\,pps\\18\,pps\\18\,pps\\18\,pps\end{bmatrix}=\begin{bmatrix}5As & 4Bs & 5As & 4Cs\\5As & 4Cs & 5As & 4Bs\\5As & 4Cs & 5As & 4Bs\\5As & 4Bs & 5As & 4Cs\end{bmatrix}\cdot\begin{bmatrix}X_0\\X_2\\X_4\\X_6\end{bmatrix}$$

$$\begin{bmatrix}18\,pps\\18\,pps\\18\,pps\\18\,pps\end{bmatrix}=\begin{bmatrix}4Ds & 6Es & 4Fs & 4Gs\\6Es & 4Gs & 4Ds & 4Fs\\4Fs & 4Ds & 4Gs & 6Es\\4Gs & 4Fs & 6Es & 4Ds\end{bmatrix}\cdot\begin{bmatrix}X_1\\X_3\\X_5\\X_7\end{bmatrix}\qquad(21)$$

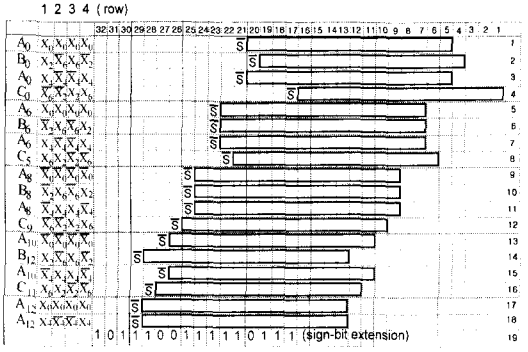
$$\begin{bmatrix}18\,pps\\18\,pps\\18\,pps\\18\,pps\end{bmatrix}=\begin{bmatrix}5As & 4Bs & 5As & 4Cs\\5As & 4Cs & 5As & 4Bs\\5As & 4Cs & 5As & 4Bs\\5As & 4Bs & 5As & 4Cs\end{bmatrix}\cdot\begin{bmatrix}X_0\\X_2\\X_4\\X_6\end{bmatrix}$$

$$\begin{bmatrix}18\,pps\\18\,pps\\18\,pps\\18\,pps\end{bmatrix}=\begin{bmatrix}4Ds & 6Es & 4Fs & 4Gs\\6Es & 4Gs & 4Ds & 4Fs\\4Fs & 4Ds & 4Gs & 6Es\\4Gs & 4Fs & 6Es & 4Ds\end{bmatrix}\cdot\begin{bmatrix}X_1\\X_3\\X_5\\X_7\end{bmatrix}\qquad(22)$$

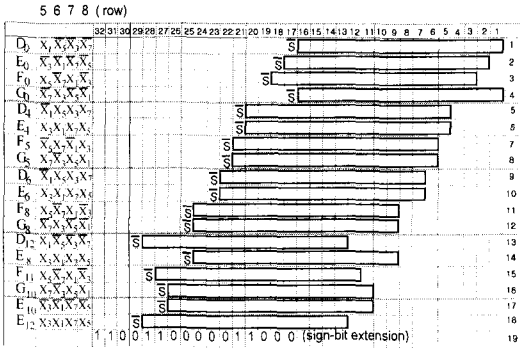
식 (21)은 DCT의 경우 부분적의 수이고 식 (22)의 경우는 IDCT의 경우이다. 위의 결과에서 볼 수 있듯이 IDCT의 경우 한번의 행의 계산에 18개의 부분적이 생성되고 전체적으로 144개의 미리 정해진 덧셈만 계산하면 된다. <표 2>는 각 곱셈방식에 따른 부분적의 개수를 나타낸다.

표 2. 각 곱셈별 부분적의 수 비교
Table 2. The comparison of partial product numbers.

일반적인 곱셈	분산 알고리듬	가변길이 다중비트 알고리듬
256	128+24ROMs	144



(a) IDCT even matrix



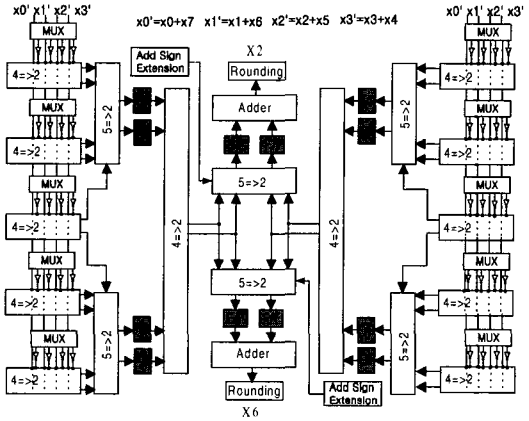
(b) DCT/IDCT odd matrix

그림 5. 각 계수에 대한 부분적
Fig. 5. The partial products' arrangement of each coefficient.

각 계수를 위의 제시한 알고리듬을 이용하여 코딩한 다음 각각 더하여 질 부분적을 구해야 한다. 이것은 입력이 들어오면 얼마만큼 시프트 되어야 하는지를 알기 위함이다. 아래 <그림 5>는 기수, 및 우수 행렬에 대한 부분적을 구한 예이다. <그림 5>에서 각 자릿수는 내부계산을 위하여 16비트로 확장된 비트로 입력이 들어

오면 시프트 되어야할 크기를 의미한다. 이렇게 계산된 부분적은 최종적으로 식 (17)을 이용하여 생성된 부호 비트의 확장 결과와 더해지게 된다. 각 비트에 대해 더해져야 될 비트 위치를 미리 알기 때문에 입력이 들어오면 각 입력 비트에 해당되는 값은 비트 단위의 덧셈기의 입력으로 미리 결정되어지고 덧셈기는 CSA (Carry Save Adder)를 이용하여 구현되어 질 수 있다. CSA는 multi-operand adder로 보통 4⇒2 압축기 혹은 5⇒2 압축기로 사용될 수 있다.

(DCT X2, X6 matrix)



(IDCT even and odd matrix)

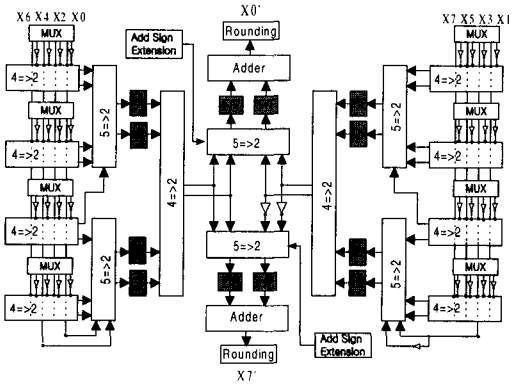


그림 6. DCT/IDCT를 위한 PE(Processing Element)
Fig. 6. The Processing Element for DCT/IDCT.

부분적의 개수가 19개이므로 고속처리를 위해 이를 동시에 더하기 위하여 다음 <그림 6>과 같은 설계가 필요하다.[10~12] <그림 6>의 첫 번째 그림은 DCT X2와 X6의 계수를 위한 PE이다. 식 (21)에서 DCT 우수 매트릭스를 보면 20개의 부분적을 더해야 하므로 5개의 4⇒2 압축기가 필요하고 이의 출력이 10개이므로 이를 다시 5⇒2 압축기를 사용하여 계산 할 수 있다. 그리고 입력에 멀티플렉서는 만약 이런 PE의 숫자를 줄이려면 멀티플렉서에서 계수에 따른 더해 주어야 하는 입력 비트의 자리를 결정하는 역할을 한다. 속도 향상을 위해 완전히 병렬로 처리 할 경우 이 MUX는 필요하지 않게 된다. 두 번째 그림은 IDCT 기수와 우수 매트릭스 모두에 적용 가능한 PE이다. 설계 방법은 위의 설명과 동일하고 18개의 부분적이므로 총 5개의 4⇒2 압축기와 3개의 5⇒2가 필요하게 된다.[13,14]

표 3. DCT/IDCT PE의 게이트 수 비교 (0.6 μm CMOS technology)
Table 3. The comparison of gate counts for PE (0.6μm CMOS technology)

Resource	DCT	IDCT
4:2 압축기	4800	3040
5:2 압축기	2520	4032
F/F	1008	1344
전체 게이트수	8640	9184

<표 3>은 <그림 6>의 PE 설계 시 사용되는 4⇒2 혹은 5⇒2압축기의 게이트 수 및 PE 전체 게이트 수를 나타낸 것이다. 표에서도 알 수 있듯이 전체적으로 약 9K 개의 게이트로 하나의 PE를 설계할 수 있음을 알 수 있다.

3. 2차원 DCT/IDCT를 위한 전치 메모리.

2차원의 DCT/IDCT를 위하여 1차원 DCT/IDCT 이후에 전치 시키는 방법을 취한다. 이때 행과 열을 바꾸는 전치 과정은 일반적으로 메모리를 사용하는 방법을 취한다. 본 논문에서는 고속의 동작을 위하여 8개의 픽셀 데이터를 위하여 동시에 처리하는 방법을 가지는 새로운 전치 메모리 구조를 제시한다.

<그림 7>은 전치 메모리의 동작 구조를 설명한 그림이다.

일단 입력이 열 방향으로 입력되면 매 클록마다 입력은 시프트 되고, 8클럭이 지나 9클럭이 되면 출력이 발생되는데 출력은 행 방향으로 빠져나간다 그리고 다시 입력은 행 방향으로 채워지고 이런 과정이 매 8 클럭마다 되풀이된다. 이렇게 동작하는 전치 메모리는 처음 8클럭의 지연시간 이후 매 클록마다 8개의 픽셀이 출력되므로 엄청난 데이터 연산이 요구되는 영상처리에 효율적인 방법으로 사용되어질 수 있다. 이 방법은

기존의 f/f 열을 이용한 방법과 흡사하나 8개의 데이터를 한꺼번에 처리하기 위하여 enable과 hold를 이용한 타이밍 컨트롤을 통하여 <그림 7>과 같이 설계되어 질 수 있다.

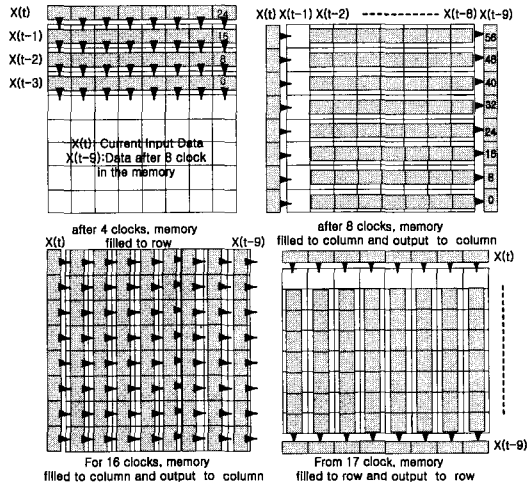


그림 7. 전치메모리의 구조
Fig. 7. The structure of transpose memory.

Ⅳ. 제안된 방법의 IDCT 칩 설계

지금까지 제시된 가변길이 다중비트 코딩을 이용한 PE와 전치 메모리를 이용하여 고속 처리용 IDCT Core를 설계하였다. 설계를 위하여 전체적인 구조를 C 모델로 검증하였다. 검증 방법을 IEEE STD 1180-1990의 사양에 따라 입력 12비트 출력 9비트의 난수로 생성하였으며 오차는 PE(Peak Error), PME(Peak Mean Error), PMSE(Peak Mean Square Error), OME (Overall Mean Error), OMSE(Overall Mean Square Error)를 각각 측정하였고 결과는 <표 4>와 같다.

설계된 Core는 전체적으로 8개의 픽셀을 동시에 처리하기 위하여 <그림 5>와 같은 PE를 4개 사용하였다. 그 결과 54MHz의 동작 주파수에서 400Mpixel/sec로 동작하였고 이는 HDTV를 위한 MPEG-2 MP@HL을 상회하는 연산 속도이다. <그림 8>은 사용된 IDCT의 Layout 결과이고 전체적인 사양은 <표 5>와 같다. <표 5>에서 보듯이 54MHz의 동작 주파수로부터 0.019ns 마다 8개의 픽셀이 나오므로 1초에 400Mpixel을 처리할수 있게 설계되었고 , 28 클럭 이후 지속적인 데이터 처리가 가능하다. 설계된 IDCT는 고속 연산이 필요한 HDTV의 core로 설계되어 그 기능을 검증하였다.

표 4. 설계된 IDCT Core의 정확도 측정
Table 4. the accuracy measurement of Implemented IDCT Core.(PE: Peak Error PME: Peak Mean Error PMSE: Peak Mean Square Error OME: Overall Mean Error OMSE: Overall Mean Square Error).

Input Pixel Range	PE(< 1)	PME (<0.015)	PMSE (<0.06)	OME (<0.0015)	OMSE (<0.02)
-256 255	-1 1	0.0066	0.02	0.00127	0.0133
255 -256	-1 1	0.0064	0.0192	0.00147	0.0132
-300 300	-1 1	0.0066	0.0166	0.00123	0.0126
300 -300	-1 1	0.0056	0.0168	0.00117	0.0125
-5 5	-1 1	0.0076	0.0139	0.00135	0.0053
5 -5	-1 1	0.0071	0.0147	0.00131	0.0053

표 5. 설계된 IDCT Core의 사양
Table 5. The specification of IDCT Core.

처리 블록 크기	8X8
동작 주파수	54MHz
Latencies	28 clocks
Throughput	400Mpixels/sec
전력소비	0.9W at 3.3V
공정	0.6㎛ triple metal CMOS

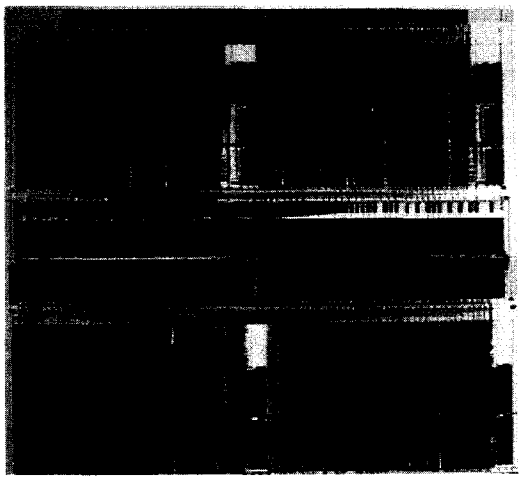


그림 8. IDCT의 레이아웃 결과
Fig. 8. The Figure of layout for IDCT Core.

VI. 결 론

본 논문에서는 제시된 가변길이 다중비트 코딩 알고리듬과 이를 적용하기 위한 DCT/IDCT 설계 그리고 실제 설계된 IDCT core의 결과에 대해 논하였다. 제시된 알고리듬은 상수 계수를 가지는 행렬 곱셈에 있어서 효율적인 알고리듬으로 검증되었고, 제시된 DCT/IDCT 설계에 있어서 속도 측면이나 면적 측면에 있어서도 효율성을 검증하였다. 설계된 IDCT Core의 경우 54MHz의 동작 주파수에 400Mpixels/sec의 높은 처리 속도를 보였고 그 크기는 24mm2 그리고, 3.3 V에서 0.9W의 전력을 소모하며 이는 IEEE STD 1180-1990의 사양에 맞게 설계되었으며 DCT/IDCT를 사용하는 HDTV 비디오 디코더에 적용하여 기능을 검증하였다.

참 고 문 헌

[1] A.D. Booth, "A signed binary multiplication technique," Quarterly J. Mechan. Appl. Math., Vol. IV, part 2, 1951.

[2] O.L. MacSorley, "High speed arithmetic in binary computers," Proc. IRE, Jan. 1961.

[3] H. Sam, and A. Gupta, "A generalized Multibit Coding of Twos Complement Binary Numbers and Its Proof with Application in Multiplier Implementation," IEEE Trans. Computers, Vol. 39, No. 8, Aug. 1990.

[4] S. Vassiliadis, E. M. Schwarz, and D. J. Hanrahan, "A general proof for overlapped multiple-bit scanning multiplications," IEEE Trans. Comput., Feb. 1989.

[5] IEEE Std 1180-1990 "IEEE Standard Specifications for the Implementations of 8x8 Inverse Discrete Cosine Transform," Dec 6, 1990.

[6] K Rao and P. Yip, Discrete Cosine Transform. Academic press, 1990.

[7] M.T. Sun, T.C. Chen, and A.M. Gottlieb, "VLSI implementaion of a 16x16 discret cosine transform", IEEE Trans. Circuits Syst., Vol. 35, No. 4, pp. 610~617, Apr. 1989.

[8] Stanley A White, "Applications of Distributed Arithmetic to Digital Signal Processing: A Tutorial Review," IEEE ASSP Magazine, July, 1989.

[9] J. R Choi, "A 400 Mpixels/sec IDCT for HDTV by Multibit Coding and Group Symmetry," IEEE International Solid State Circuits Conference, San Francisco, Feb. 1997.

[10] A. Habibi and P. A. Wintz, "Fast multipliers," IEEE Trans. Comput., pp. 153~157, Feb. 1970.

[11] C.S. Wallace, "A suggestion for a fast multiplier," IEEE Trans. Electron. Compu., Feb. 1964.

[12] A. Gupta, "A 50 ns 16 x 16 bit 2s Complement parallel multiplier," proc. ISELDECS, pp. 747~749, 1987.

[13] B. Parhami, Computer Arithmatic, Algorithms and Hardware Designs. New York: Oxford University Press, 2000.

[14] K. Hwang, Computer Arithmetic Principles, Architecture and Design. New York: wiley, 1979.

저 자 소 개

金 大 原(正會員)

1998년 2월 : 경북대학교 전자공학과 졸업. 2000년 2월 : 경북대학교 전자공학과 석사. 2000년 3월~현재 : 경북대학교 전자공학과 박사 과정. <주관심분야 : 통신칩설계 영상압축 및 워터마킹칩설계>

崔 峻 林(正會員)

1986년 2월 : 연세대학교 전기공학과 졸업. 1988년 8월 : 미국 Cornell 대학교 전자전기공학과 박사. 19991년 7월 : 미국 Minnesota 대학교 전자전기공학과 박사. 1987년 6월~1988년 5월 : 미국 National Nanofabrication Facility 연구원. 1988년 6월~1988년 5월 : 미국 Center For Microtechnology (MEIS) 연구원. 1991년 7월~1997년 2월 : LG전자기술원. 1997년 3월~현재 : 경북대학교 전자전기컴퓨터학부 조교수. <주관심분야 : 암호칩설계, 영상압축칩설계>