論 文
57-6-25

영상 압축 응용분야를 위한 DCT 아키텍처 개발

An Efficient DCT Architecture for Image Compression Applications

庾 盛 郁†
(Sungwook Yu)

Abstract - This paper presents an efficient architecture for 2^n-point DCT algorithm. The proposed approach makes use of the fact that, in most DCT applications, the scaling operation in the DCT unit can be eliminated and combined with the scaling operation in the quantizer unit. This important property is efficiently exploited with the CORDIC (COordinate Rotation DIgital Computer) algorithm to produce a regular architecture suitable for VLSI implementation. Although there have been several attempts to exploit CORDIC algorithm in developing DCT architectures, the proposed approach provides the most efficient way for scaled DCT applications by completely eliminating the scale factor compensation.

Key Words : DCT, CORDIC, Scaled DCT

1. 서 론

DCT (Discrete Cosine Transform) 알고리즘은 음성 압축, 영상 압축 등 많은 디지털 신호처리 응용분야에서 널리 사용되고 있으며 특히, 2-D DCT (2차원 DCT)는 JPEG, MPEG, H.264 등의 영상 압축 표준의 기본 알고리즘으로 채택되어 그 중요성이 매우 높다. [1-2] 그러나, (2차원) DCT는 많은 연산량을 필요로 하기 때문에 DCT의 고속 연산을 위한 알고리즘 및 아키텍처 개발에 대한 많은 연구가 있어 왔다 [3-11].

2차원 DCT를 계산하는 방법은 크게 row-column 분해 방법, 심박 구조 (systolic architecture), 직접 계산법 등으로 구분할 수 있다. row-column 분해 방법은 2차원 이미지 블록의 각 행에 1차원 DCT를 행하여 중간 데이터를 저장한 후, 그 중간 데이터의 각 열에 다시 1차원 DCT를 행하는 방법으로서 가장 널리 쓰이는 방법이다. 심박 구조는 기본 계산 요소인 computational element를 규칙적으로 나열하여 각 클럭마다 기본 연산 수행 및 중간 데이터 이동을 반복하는 구조로서 규칙적인 배열에 의해 구현은 용이하나 다른 방법에 비해 많은 수의 가산기와 승산기를 필요로 한다. 직접 계산법은 2차원 이미지 블록을 1차원으로 분해하지 않고 직접 처리하는 방식으로서 가장 작은 수의 가산기와 승산기를 요구하나 결선 및 내부 구조가 가장 복잡하여 VLSI 구현 면에서는 가장 불리한 방식이다.

앞서 언급한 바와 같이, DCT의 여러 응용 분야 중 가장 중요한 것은 영상 압축 응용 분야이며 이런 분야에서는 original DCT 출력 대신 scaled DCT를 계산하는 것이 유리하다 [5-8]. scaled DCT 출력은 original DCT 출력에 비해 상수배 만큼의 차이가 있으나 대개의 영상 압축 응용 분야에서는 이러한 차이를 추가 연산 없이 복구할 수 있다. 한편, scaled DCT는 original DCT보다는 훨씬 적은 연산량을 필요로 하기 때문에 영상 압축 응용 분야에서는 scaled DCT를 위한 알고리즘 및 아키텍처 개발이 더욱 중요하다고 할 수 있다.

CORDIC (COordinate Rotation DIgital Computer)은 삼각함수 계산, 벡터 회전 등 여러 분야에서 널리 쓰이는 매우 유용한 알고리즘이다. 특히, CORDIC 알고리즘은 가산기와 쉬프터 등의 간단한 구성 요소의 규칙적인 배열로 구현이 가능하기 때문에 VLSI 구현용으로 매우 적합하여 CORDIC을 이용한 DCT 아키텍처 개발에도 많은 연구가 있어 왔다 [9-11]. 그러나, 지금까지의 CORDIC 기반 DCT 아키텍처는 original DCT에 적합한 구조로서 보다 중요한 scaled DCT의 특성을 충분히 이용하지 못하고 있다. 이에 본 논문에서는 CORDIC 알고리즘과 scaled DCT 알고리즘을 매우 효율적으로 결합한 새로운 DCT 아키텍처를 제안하고자 한다.

본 논문의 구성은 다음과 같다. 우선 2절에서는 1차원 DCT의 간접 계산법을 이용하여 벡터 회전 연산을 분리하는 과정과 분리된 벡터 회전 연산을 (3-value) CORDIC 알고리즘을 이용하여 처리하는 과정을 설명한다. 3절에서는 제안된 방식을 기존의 CORDIC 기반 DCT 아키텍처와 비교하고 마지막으로 4절에서 본 논문의 요약 및 결론을 제시한다.

2. CORDIC을 이용한 DCT 계산 알고리즘

서론에서 언급한 바와 같이, DCT는 여러 영상 압축 응용 분야에서 널리 쓰이고 있으며 그림 1은 MPEG, JPEG 등에서 쓰이는 DCT 기반 부호기의 블록 다이어그램을 보여준다 [1-2]. 그림에서 2차원 DCT 블록은 8×8 또는 16×16의 입력

† 교신저자, 正會員 : 中央大學 電子電氣工學部 助敎授 · 工博
E-mail : sungwook@cau.ac.kr
接受日字 : 2008年 1月 10日
最終完了 : 2008年 3月 24日

데이터에 2차원 DCT를 행하는 블록이며 그 결과는 양자화기의 입력이 된다. 양자화기의 또 다른 입력은 8×8 또는 16×16개의 상수값을 저장한 테이블이며 그 테이블에 저장된 상수값들은 응용 분야에 따라 달라진다. 양자화기는 각각의 2차원 DCT 출력을 테이블에 저장된 상수로 나누어 준 후, 양자화하는 역할을 하며 엔트로피 부호기는 데이터의 통계적 특성을 이용하여 추가의 압축을 행한다. 양자화기의 역할이 DCT 블록의 각 데이터를 테이블에 저장된 상수값으로 나누어 주는 것이므로 (또는 테이블에 저장된 상수값의 역수를 곱하는 것으로 간주할 수도 있음) original DCT 대신 그 상수배에 해당하는 scaled DCT를 구해도 (테이블의 상수값을 그에 맞게 적절히 변경시키면) 무방함을 알 수 있다 [5-8]. 이러한 특성은 1차원 DCT와 row-column 분해 방법을 이용하여 2차원 DCT를 구현한 경우에도 여전히 성립하므로 본 논문은 1차원 (scaled) DCT의 효율적인 구현 방식에 초점을 맞춘다.

실수 데이터 $x_0(n),\ n=0,1,...,N-1$ 에 대한 N-포인트 1차원 DCT는 다음과 같이 주어진다.

$$X_0(k)=\sqrt{\frac{2}{N}}c(k)\sum_{n=0}^{N-1}x_0(n)\cos\frac{(2n+1)k\pi}{2N}\qquad 0\le k\le N-1 \tag{1}$$

식 (1)에서 scale factor 함수 $c(k)$는 다음과 같다.

$$\begin{cases}c(k)=1/\sqrt{2} & k=0\\ c(k)=1 & k=1,2,\cdots,N-1\end{cases} \tag{2}$$

식 (2)의 N-포인트 DCT는 다음과 같이 N-포인트 IDFT (또는 N-포인트 DFT)를 이용하여 간접적으로 구할 수 있다 [4]. 우선, 원래의 입력 데이터 $x_0(n)$ 을 다음과 같이 재배열하여 새로운 입력 데이터 $x(n)$ 을 얻는다.

$$\begin{cases}x(k)=x_0(2k)\\ x(N-1-k)=x_0(2k+1)\end{cases}\qquad k=0,1,\cdots(N/2)-1 \tag{3}$$

그림 1 DCT 기반의 부호화기
Fig. 1 DCT-based Encoder

그러면 식 (1)의 $X_0(k)$ 와 재배열된 N-포인트 데이터 $x(n)$ 의 N-포인트 IDFT

$$X(k)=\frac{1}{N}\sum_{n=0}^{N-1}x(n)\exp\left(j\frac{2\pi nk}{N}\right)\qquad 0\le k\le N-1 \tag{4}$$

사이에는 다음과 같은 관계식이 성립한다.

$$X_0(k)=\mathrm{Re}\left\{\sqrt{2N}c(k)\exp\left(j\frac{\pi k}{2N}\right)X(k)\right\}\qquad 0\le k\le N-1 \tag{5}$$

입력 데이터 $x_0(n)$ 이 실수일 경우, $X(0)$ 와 $X(N/2)$ 도 실수이고 $X(N-k)=X^*(k),\ k=1,2,\cdots,N/2-1$ 의 관계가 성립하므로

$$X_0(k)=\sqrt{N}X(0)\qquad k=0 \tag{6}$$

$$X_0(k)=\sqrt{N}X(N/2)\qquad k=N/2 \tag{7}$$

$$\begin{cases}X_0(k) &= \mathrm{Re}\left[\sqrt{2N}\exp\left(j\frac{\pi k}{2N}\right)X(k)\right]\\ X_0(N-k) &= \mathrm{Im}\left[\sqrt{2N}\exp\left(j\frac{\pi k}{2N}\right)X(k)\right]\end{cases}\quad k=1,2,\cdots,N/2-1 \tag{8}$$

original DCT의 경우와 scaled DCT의 경우 모두 상수 scale factor는 중요하지 않으므로 식 (6)-(8)에서 $\sqrt{N}$ 은 무시할 수 있다. 한편, scaled DCT의 경우, 입력 데이터의 변화에 무관한 scale factor는 양자화기의 테이블 상수값을 변화시켜 처리할 수 있으므로 역시 무시할 수 있다. 즉, scaled DCT의 경우, 식 (8)의 $\sqrt{2}$ 까지도 추가로 생략할 수 있다. 따라서, scaled DCT의 경우, $k=0$ 인 경우와 $k=N/2$ 인 경우에는 IDFT 출력 $X(k)$ 로부터 직접 $X_0(k)$ 를 구할 수 있고 $k=1,2,\cdots,N/2-1$ 인 경우에는 $X(k)$ 를 복소수 평면에서 $\pi k/(2N)$ 만큼 (반시계 방향으로) 회전시켜 얻어지는 복소수의 실수값과 허수값을 취함으로써 구할 수 있다. 즉, 1개의 N-포인트 IDFT와 $N/2-1$ 개의 벡터 회전을 통하여 N-포인트 DCT를 구할 수 있다.

식 (8)에 나타난 ($N/2-1$ 개의) 벡터 회전을 위하여 여러 가지 방법이 이용될 수 있으나 그 중 가장 효과적인 방법 중 하나가 CORDIC (COordinate Rotation DIgital Computer) 알고리즘을 이용하는 것이다 [12-13]. 벡터 회전에 사용되는 CORDIC 알고리즘의 rotation mode에서 회전각 θ 는 다음과 같이 ATR (Arc Tangent Radix)라 불리우는 기본 각들의 선형결합 형태로 나타낸다.

$$\theta=\sum_{i=0}^{M-1}\sigma_i\tan^{-1}\left(2^{-i}\right) \tag{9}$$

복소수 평면상의 점을 식 (9)에 주어진 각도 θ 만큼 반시계 방향으로 회전시키기 위해서는 다음과 같은 과정을 반복해야 하며 이 때, M이 클수록 최종 결과의 bit precision이 커지게 된다.

$$\begin{cases} x_{i+1} = x_i - \sigma_i 2^{-i} y_i \\ y_{i+1} = y_i + \sigma_i 2^{-i} x_i \end{cases} \qquad 0 \le i \le M-1 \tag{10}$$

그림 2는 식 (10)에 주어진 CORDIC iteration을 수행하는 CORDIC 프로세서의 구조를 보여 준다. 그림에서 알 수 있듯이 CORDIC 프로세서는 가산기와 hardwired 쉬프터가 매우 규칙적으로 배열되어 있는 구조로서 VLSI 구현에 매우 적합한 형태이다. 그러나 식 (10)에 주어진 각각의 iteration은 원하는 회전 효과 외에 절대값 증가의 효과를 함께 갖는 유사 회전 (pseudo rotation)이며 각각의 iteration은 다음의 k_i배 만큼 절대값을 증가시키게 된다.

$$k_i = \sqrt{\left(1+\sigma_i^2 2^{-2i}\right)} \qquad 0 \le i \le M-1 \tag{11}$$

따라서, 전체 M번의 iteration이 끝나게 되면 다음에 주어진 scale factor K배 만큼 절대값이 증가하게 된다.

$$K = \prod_{i=0}^{M-1} k_i \tag{12}$$

따라서, original DCT의 경우에는 유사 회전이 끝난 후, 이러한 scale factor를 처리하기 위한 후처리기 (post processor)가 필요하다. 이를 위해, iteration 결과를 식 (12)에 주어진 scale factor로 나누어서 (또는 scale factor의 역수를 곱해서) 처리하는 방식이 있으며 이는 추가의 승산기를 필요로 한다. 또 다른 방식으로서는 다음과 같은 scaling iteration을 이용하는 방법이 있다 [14].

$$\begin{cases} x_i = x_i\left(1+\gamma_i 2^{-i}\right) \\ y_i = y_i\left(1+\gamma_i 2^{-i}\right) \end{cases} \qquad 0 \le i \le M-1 \tag{13}$$

이 때, γ_i의 값은 다음의 조건을 만족하도록 +1, 0, -1 중에서 정해진다.

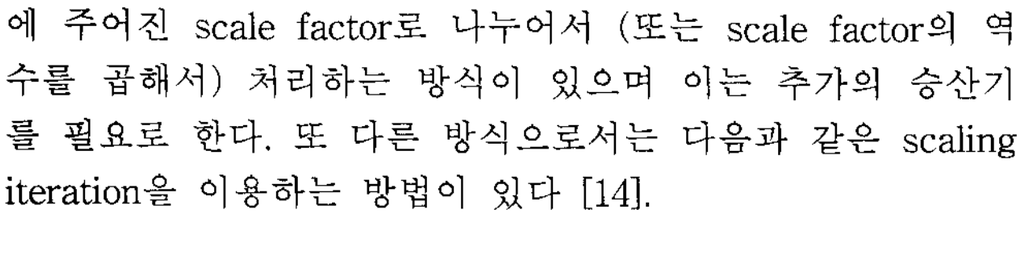


그림 2 벡터 회전을 위한 CORDIC 프로세서

Fig. 2 CORDIC Processor for Vector Rotation

$$K^{-1} \cong \prod_{i=0}^{M-1} \left(1+\gamma_i 2^{-i}\right) \tag{14}$$

한편, scaled DCT의 경우에는 식 (12)의 scale factor를 양자화기 내에서 처리할 수 있으므로 추가의 후처리기 없이도 시스템을 구현할 수 있다.

CORDIC 알고리즘은 식 (9)의 σ_i가 취할 수 있는 값의 종류에 따라 2-value CORDIC과 3-value CORDIC 알고리즘으로 나눌 수 있다. 2-value CORDIC 알고리즘의 경우, σ_i는 +1 또는 -1의 값을 갖게 되며 따라서 식 (10)에 주어진 iteration을 M번 수행해야 한다. 한편, 3-value CORDIC 알고리즘의 경우에는 σ_i가 +1, -1의 값 외에 0의 값을 가질 수도 있다. σ_i가 0의 값을 가질 경우, 회전이 필요하지 않으므로 전체 iteration 횟수는 M보다 작아지게 되어 하드웨어 복잡도를 줄일 수 있는 장점이 있다. 그러나 3-value CORDIC 알고리즘에서는 σ_i가 0일 때의 k_i의 값이 σ_i가 +1 또는 -1일 때의 k_i의 값과 달라지게 되므로 식 (12)에 주어진 scale factor K값 역시 회전각 θ에 따라 달라지게 된다. 따라서, CORDIC 알고리즘을 이용한 다른 DCT 아키텍처에서는 하드웨어 복잡도가 큰 2-value CORDIC 알고리즘을 사용하거나 또는 3-value CORDIC 알고리즘과 scale factor 후처리기를 동시에 사용해야 한다. 그러나 본 논문에 제안된 DCT 아키텍처에서는 CORDIC 프로세서를 효율적으로 이용하여 별도의 후처리기 없이 3-value CORDIC 알고리즘을 사용할 수 있으며 이에 대해서는 다음 절에서 자세히 설명한다.

3. DCT 아키텍처 비교

서론에서 설명한 바와 같이, CORDIC 알고리즘은 여러 종류의 DSP 알고리즘 구현에 널리 쓰여 왔으며 DCT 알고리즘 구현에도 몇몇 연구가 있어 왔다. 가령, 그림 3은 기존의 CORDIC 기반의 DCT 아키텍처 중 [9-11]에서 제안된 방식인데 이 방식은 다른 저자들에 의해 발표되었으나 본질적으로 같은 구조라고 할 수 있으며 기존의 CORDIC 기반 DCT 아키텍처 중 가장 효율적인 구조라고 할 수 있다.

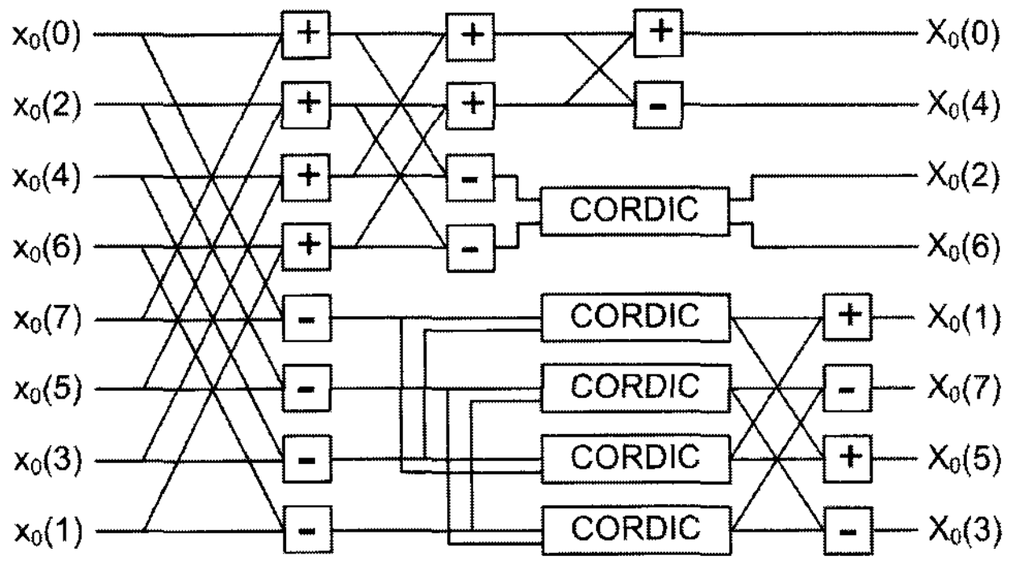


그림 3 기존 CORDIC 기반의 DCT 아키텍처

Fig. 3 Conventional CORDIC-based DCT Architecture

그림 4는 8-포인트 DCT 계산을 위한 제안된 DCT 아키텍처의 구조를 보여준다. 2절에서 설명한 바와 같이, 제안된 구조는 original (실수) 입력데이터를 재배열하는 블록, N-포인트 IDFT 블록, 그리고 벡터 회전 블록으로 구성된다. 입력 데이터 재배열 블록은 routing wire를 통해 처리가 가능하므로 추가의 하드웨어를 요구하지 않으며 이는 기존 방식에서도 마찬가지이다. 입력 데이터가 일반 복소수가 아닌 실수의 경우에는 N-포인트 IDFT 블록의 복잡도를 크게 줄일 수 있다. 그림 4에서는 실수 입력 데이터용 IDFT(DFT) 방식 중 가장 효율적인 방법의 하나인 Sorensen *et al.* [15]의 방법을 사용하고 있으며 그림에서의 승산기는 고정된 숫자인 $\sqrt{1/2}$로 곱하는 승산기를 의미한다. IDFT를 거친 데이터는 벡터 회전 블록을 통과하게 되는데, 식 (6)과 (7)에서 보인 바와 같이 $X(0)$와 $X(4)$는 벡터 회전을 위한 블록을 추가로 필요로 하지 않으며 나머지 데이터들은 식 (8)에서 보인 바와 같이 2개씩 짝을 지어 벡터 회전을 하게 되므로 3개의 CORDIC 프로세서가 필요하게 된다.

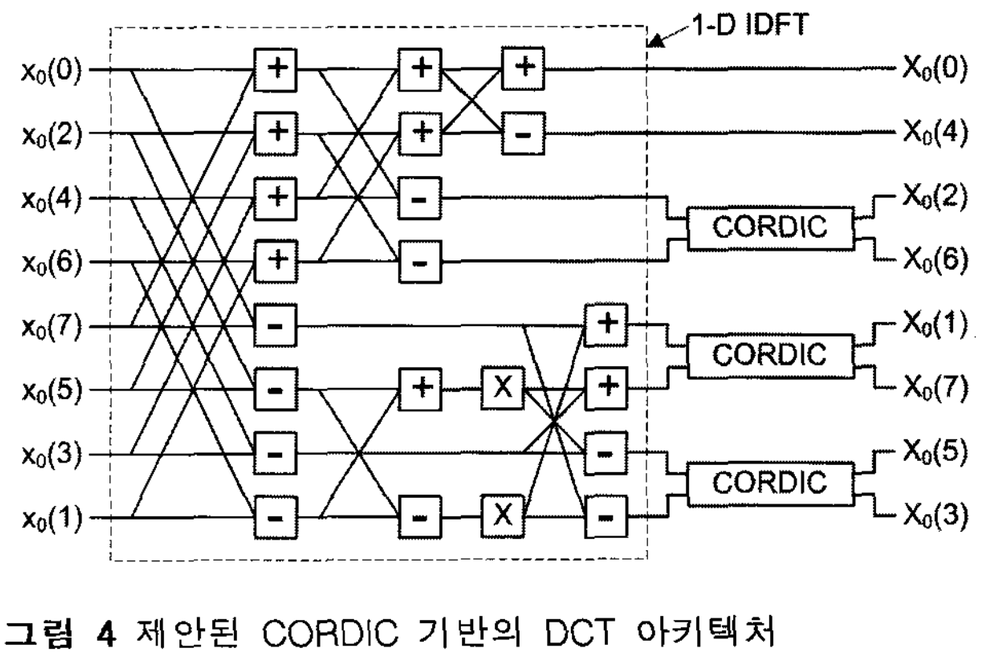


그림 4 제안된 CORDIC 기반의 DCT 아키텍처
Fig. 4 Proposed CORDIC-based DCT Architecture

표 2는 기존의 방식과 본 논문에서 제안된 방식을 비교하기 위해 original 8-포인트 DCT 계산을 위한 하드웨어 복잡도를 비교하였다. 표에서 M=12를 사용하였다. 3-Value CORDIC 알고리즘과 승산기 방식의 후처리기를 사용한 경우, 제안된 방식이 2개의 가산기를 더 사용하는 반면 기존 방식은 2개의 CORDIC 프로세서가 더 필요함을 볼 수 있다. 그림 2에서 볼 수 있듯이 하나의 CORDIC 프로세서는 여러 개의 가산기로 이루어져 있으므로 CORDIC 프로세서의 복잡도가 가산기의 복잡도보다 훨씬 크며 따라서 제안된 방식의 복잡도가 더 작음을 알 수 있다. 일반적으로 scaling iteration을 이용한 후처리기는 (고정된 상수로 곱하는) 승산기에 비해 복잡도가 더 크며 따라서 scaling iteration을 이용한 경우에도 제안된 방식의 하드웨어 복잡도가 낮다. 2-value CORDIC 알고리즘을 사용할 때 필요한 iteration 수는 대개 3-value CORDIC 알고리즘을 사용할 때 필요한 iteration 수와 scaling iteration 수의 합과 비슷하며 따라서 이 경우에도 제안된 방식이 더 효율적임을 알 수 있다.

서론에서 말한 바와 같이 현재 대부분의 DCT 응용분야에서는 original DCT보다 scaled DCT의 계산이 더욱 중요하며 표 3은 scaled 8-포인트 DCT 계산을 위한 하드웨어 복잡도를 비교하였다. 이 때 주목해야 할 점은 제안된 방식의 CORDIC 프로세서 출력은 추가의 연산을 거치지 않는 반면, 그림 3 하단부의 CORDIC 프로세서 출력은 추가의 덧셈, 뺄셈 연산을 거쳐야 한다는 점이다. 이는 scaled DCT의 관점에서 보면 매우 큰 의미를 가지며 그 이유는 다음과 같다. 우선, 2절에서 설명한 바와 같이 3-value CORDIC 알고리즘이 2-value CORDIC 알고리즘에 비해 필요한 iteration 수가 매우 작기 때문에 가능하면 3-value CORDIC 알고리즘을 사용하는 것이 유리하다. CORDIC 프로세서가 가장 후반부에 위치한 경우, 각각의 3-value CORDIC 프로세서의 scale factor는 서로 다르나 그 값들이 입력 데이터의 변화에 따라 바뀌지는 않는다. 따라서, 각각의 scale factor의 값을 양자화기의 테이블의 대응 위치에 반영하면 후처리기가 필요 없게 된다. 그러나 그림 3 하위 4개의 데이터 $X_0(1), X_0(7), X_0(5), X_0(3)$의 경우처럼 CORDIC 프로세서 출력이 추가의 연산을 거쳐야 하는 경우에는, 입력 데이터의 상대적 크기에 따라 scale factor가 달라지므로 2-value CORDIC 알고리즘을 사용하거나 또는 scale factor 보상을 위한 후처리기가 필요하게 된다. 단, 4개의 데이터 각각을 보상해 줄 필요 없이 각 쌍에서 상대적 크기만 보상해 주면 되므로 후처리기는 4개가 아닌 2개가 필요하게 된다. 표 3에서도 표 2의 경우처럼, 제안된 방식이 기존 방식에 비해 더 유리함을 알 수 있으며 제안된 방식은 2-value CORDIC 알고리즘을 쓸 필요가 없으므로 2-value CORDIC 알고리즘의 경우에 대한 분석은 생략하였다.

표 4는 scaled 16-포인트 DCT 계산을 위한 하드웨어 복잡도를 보여준다. 표에서 알 수 있듯이 본 논문에서 제안된 방식은 큰 사이즈의 DCT 계산에서 더욱 유리하며 그 이유는 다음과 같다. 우선, $N(=2^n)$-포인트 DCT 계산을 위해 필요한 CORDIC 프로세서의 수가 제안된 방식에서는 N의

표 2 Original 8-포인트 DCT 아키텍처 하드웨어 복잡도 비교
Table 2 Hardware Complexity for Original 8-Point DCT

	3-Value CORDIC 알고리즘							2-Value CORDIC 알고리즘		
	scaling iteration 방식의 후처리기				승산기 방식의 후처리기					
	가산기 수	승산기 수	CORDIC 프로세서 수	후처리기 수	가산기 수	승산기 수	CORDIC 프로세서 수	가산기 수	승산기 수	CORDIC 프로세서 수
기존방식	18	0	5	5	18	5	5	18	0	5
제안된 방식	20	2	3	3	20	5	3	20	2	3

표 3 Scaled 8-포인트 DCT 아키텍처 하드웨어 복잡도 비교
Table 3 Hardware Complexity for Scaled 8-Point DCT

	3-Value CORDIC 알고리즘과 scaling iteration 방식의 후처리기 사용				3-Value CORDIC 알고리즘과 승산기 방식의 후처리기 사용		
	가산기 수	승산기 수	CORDIC 프로세서 수	후처리기 수	가산기 수	승산기 수	CORDIC 프로세서 수
기존 방식	18	0	5	2	18	2	5
제안된 방식	20	2	3	0	20	2	3

표 4 Scaled 16-포인트 DCT 아키텍처 하드웨어 복잡도 비교
Table 4 Hardware Complexity for Scaled 16-Point DCT

	3-Value CORDIC 알고리즘과 scaling iteration 방식의 후처리기 사용				3-Value CORDIC 알고리즘과 승산기 방식의 후처리기 사용		
	가산기 수	승산기 수	CORDIC 프로세서 수	후처리기 수	가산기 수	승산기 수	CORDIC 프로세서 수
기존 방식	58	0	21	14	58	14	21
제안된 방식	60	10	7	0	60	10	7

증가에 대해 천천히 증가하는 데 비해 기존 CORDIC 기반의 DCT 아키텍처에서는 매우 빨리 증가한다 [9]. 두 번째로 기존 방식에서는 N이 커질수록 scale factor 보상을 요구하는 CORDIC 프로세서의 비율이 함께 커진다. 가령, 표 3의 8-포인트 scaled DCT에서는 40% (5개 중 2개)의 CORDIC 프로세서가 후처리기를 요구하나 표 4의 16-포인트 scaled DCT에서는 66.7% (21개 중 14개)의 CORDIC 프로세서가 후처리기를 요구함을 알 수 있다. 한편, 제안된 방식에서는 N의 값에 관계 없이 모든 CORDIC 프로세서가 후처리기를 요구하지 않으므로 N이 커질 수록 복잡도 차이가 더 커지게 된다.

4. 결 론

본 논문에서는 (DFT/IDFT를 이용한) DCT의 간접 계산법과 CORDIC 알고리즘을 효과적으로 결합시켜 새로운 DCT 아키텍처를 제안하였다. 제안된 방식은 기존의 CORDIC 기반 DCT 아키텍처와는 달리 CORDIC 프로세서를 DCT 블록의 제일 후반부에 위치시켜 후처리기 없이 3-Value CORDIC 알고리즘을 사용할 수 있는 구조이다. 따라서, 제안된 방식은 scaled DCT에서 더욱 효과적이며 특히, 큰 사이즈의 DCT 계산에서는 기존 방식에 비해 매우 작은 하드웨어 복잡도를 보인다. 또한 제안된 방식은 N-포인트 DFT/IDFT에 의한 간접 계산 방식을 이용하므로 지금까지 개발된 많은 DFT/IDFT 아키텍처를 그대로 이용할 수 있으며 적절한 컨트롤 로직과 MUX를 사용하면 N-포인트 DFT/IDFT 또는 N-포인트 DCT를 선택적으로 계산할 수 있도록 만들 수 있다는 장점이 있다.

감사의 글

본 연구는 2007년도 중앙대학교 학술연구비 지원에 의하여 수행되었습니다. 연구비 지원에 감사드립니다.

참 고 문 헌

[1] G. K. Wallace, "The JPEG Still Picture Compression Standard," Communications of the ACM, Vol. 34, pp. 30-44, 1991.
[2] D. Le Gall, "MPEG: A Video Compression Standard for Multimedia Applications," Communications of the ACM, Vol. 34, pp. 46-58, 1991.
[3] N. Ahmed, T. Natarajan and K. R. Rao, "Discrete Cosine Transform," IEEE Transactions on Computers, Vol. C-23, pp. 90-93, 1974.
[4] M. J. Narasimha and A. M. Peterson, "On the Computation of the Discrete Cosine Transform," IEEE Transactions on Communications, Vol. COM-26, pp. 934-936, 1978.
[5] Y. Arai, T. Agui and M. Nakajima, "A Fast DCT-SQ Scheme for Images," Transactions of the IEICE, Vol. E71, pp. 1095-1097, 1988.
[6] H. S. Hou, "Recursive Scaled DCT," Proceedings of the SPIE, Vol. 1567, pp. 402-412, 1991.
[7] E. Feig and S. Winograd, "Fast Algorithms for the Discrete Cosine Transform," IEEE Transactions on Signal Processing, Vol. 40, pp. 2174-2193, 1992.
[8] E. Feig, "A Fast Scaled-DCT Algorithm," Proceedings of the SPIE, Vol. 1244, pp. 2-13, 1990.
[9] E. P. Mariatos, D. E. Metafas, J. A. Hallas and C. E. Goutis, "A Fast DCT Processor, Based on Special Purpose CORDIC Rotators," IEEE International Symposium on Circuits and Systems, Vol. 4, pp. 271-274, 1994.
[10] F. Zhou and P. Kornerup, "High Speed DCT/IDCT Using a Pipelined CORDIC Algorithm," Proceedings of the 12th Symposium on Computer Arithmetic, pp.

180-187, 1995.

[11] T. Sung, Y. Shieh, C. Yu and H. Hsin, "High-Efficiency and Low-Power Architectures for 2-D DCT and IDCT Based on CORDIC Rotation," International Conference on Parallel and Distributed Computing, Applications and Technologies, pp. 191-196, 2006.

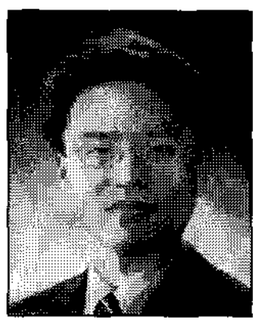
[12] J. E. Volder, "The CORDIC Trigonometric Computing Technique," IRE Transactions on Electronic Computers, Vol. EC-8, pp. 330-334, 1959.

[13] J. S. Walther, "A Unified Algorithm for Elementary Functions," Spring Joint Computer Conference Proceedings, Vol. 38, pp. 379-385, 1971.

[14] J. A. Lee and T. Lang, "Constant-Factor Redundant CORDIC for Angle Calculation and Rotation," IEEE Transactions on Computers, Vol. 41, pp. 1016-1025, 1992.

[15] H. V. Sorensen, D. L. Jones, M. T. Heideman and C. S. Burrus, "Real-Valued Fast Fourier Transform Algorithms," IEEE Transactions on Acoustics, Speech and Signal Processing, Vol. ASSP-35, pp. 849-863, 1987.

저 자 소 개



유 성 욱 (庾 盛 郁)

1992.2 서울대학교 전기공학과 졸업
1996.12 UT Austin ECE 석사
2000.5 UT Austin ECE 박사
2000.8 - 2004.2 Intel Corp.
2004.4 - 2005.2 삼성 시스템 LSI.
2005.3 - 현재 중앙대학교 전자전기공학부 조교수
Tel : (02) 820-5740
E-mail : sungwook@cau.ac.kr