論 文

모듈 생성 기법을 이용한 DCT/IDCT 코어 프로세서의 설계

正會員 黃 俊 夏* 正會員 韓 鐸 敦*

Design of DCT/IDCT Core Processor using Module Generator Technique

Joon Ha Hwang*, Tack Don Han* *Regular Members*

요 약

DCT(Discrete Cosine Transform)/IDCT(Inverse DCT)는 여러 DSP 분야와 영상압축 시스템에서 널리 사용되는 부호화 방식으로서 압축 및 복원 시스템에서 가장 많은 처리시간을 요하는 부분이다. 그러므로 이 부분의 성능을 향상시킴으로써 전체 영상 압축시스템의 성능을 향상시킬수 있다.

본 논문에서는 이러한 DCT/IDCT연산을 효율적으로 수행하기 위하여 모듈생성기법을 이용하여 하드웨어로 구성하였다. 설계한 DCT/IDCT 코어 프로세서는 부분합과 분산연산을 이용하여 비교적 적은 면적을 차지하며, 약간의 면적을 증가시킴으로써 DCT와 IDCT연산을 모두 수행한다. 또한 파이프라인 구조를 사용하여 고속으로 DCT/IDCT연산을 수행할 수 있으며, 적은 수의 반올림(rounding)단계를 거치므로 높은 정밀도로 연산을 수행한다. 그리고 모듈생성기법을 사용하여, 설계공정에 독립적이고 입력비트나 정밀도 등을 간단한 매개변수의 조정으로 변환시킬 수 있도록 설계하였다. 또한 구현한 코어프로세서는 CCITT권장안 H.261에 부합하는 정밀도로 연산을 수행한다.

ABSTRACT

DCT(Discrete Cosine Transform)/IDCT(Inverse DCT) is widely used in various image compression and decompression systems as well as in DSP(Digital Signal Processing) applications. Since DCT/IDCT is one of the most complicated part of the compression system, the performance of the system can be greatly enchanced by improving the speed of DCT/IDCT operation.

In this thesis, we designed a DCT/IDCT core processor using module generator technique. By utilizing the partial sum and DA(Distributed Arithmetic) techniques, the DCT/IDCT core processor is designed within small area. It is also designed to perform the IDCT(Inverse DCT) oper-

*延世大學校 電算科學科
論文番號 : 93-145

ation with little additional circuitry. The pipeline structure of the core processor enables the high performance, and the high accuracy of the DCT/IDCT operation is obtained by having fewer rounding stages. The proposed design is independent of design rules, and the number of the input bits and the accuracy of the internal calculation coa be easily adjusted due to the module generator technique. The accuracy of the processor satisfies the specifications in CCITT recommendation H. 261.

I. 서 론

최근에 디지탈 영상을 사용하는 응용분야가 점점 늘어나고 있는데 디지탈 방식으로 영상을 표현하면 영상의 처리등을 용이하게 할 수 있는 반면, 영상을 저장하는데 필요로 하는 비트(bit)수가 매우 많아지게 된다. 이러한 영상 정보를 효율적으로 사용하기 위하여는 저장, 처리, 전송하는 기술을 필요로 하게 된다.

영상 압축의 목적은 주로 영상의 전송과 저장에 있는데 디지탈 영상을 효율적으로 운용하기 위하여는 영상을 표현하는데 필요한 비트수를 줄이는 기술이 필요하다. 또한 여러 응용분야에서 실시간 처리를 필요로 하므로 이를 만족하기 위하여는 압축 시스템의 일부분, 또는 전체를 하드웨어(hardware)로 구현해야 할 필요가 있다.

본 논문에서는 JPEG표준의 영상 압축 시스템에서 가장 많은 시간이 소요되는 DCT/IDCT의 코어 프로세서를 설계하였다. 〈그림 1.1〉에 JPEG의 기본적인 압축과 복원 과정이 나타나 있다.

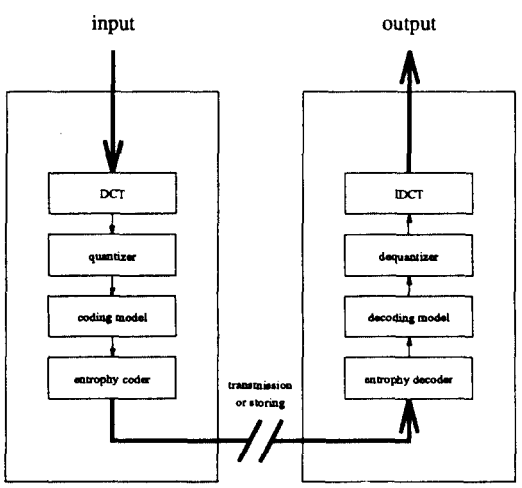

그림 1.1 JPEG의 기본적인 압축, 복원 과정

이 코어 프로세서는 모듈생성기(module generator) 기법을 사용하여 설계공정에 독립적으로 설계하였다. 또한 연산의 분리성(separability)을 이용하여 2회의 1-D DCT/IDCT를 수행함으로써 2-D DCT/IDCT의 결과를 얻는다. 설계된 코어 프로세서는 DCT 및 IDCT를 같이 수행할 수 있도록 설계되어 있으며, CCITT에서 규정하고 있는 정밀도 이상의 연산 정밀도로써 DCT/IDCT를 수행한다.

DCT를 구현하기 위한 기법은 여러가지가 존재하는데 본 논문에서는 분산 연산(distributed arithmetic)과 테이블 참조(table lookup)방식을 이용하였다. 이 기법을 사용하면 승산기(multiplier)대신에 비교적 적은 면적을 차지하는 참조 테이블(lookup table)을 사용하므로 좋은 단위성(modualrity)을 가지게 되므로 칩 면적을 감소시킬 수 있고 모듈생성기법으로 구현하기 용이하다.

본 연구에서는 부분합(partial sum)을 이용한 방식을 사용하였는데 부분합을 이용하지 않고 구현할 경우 가산기와 감산기 부분이 작아지고 ROM이 늘어나게 된다.

DCT를 구현하기 위하여 먼저 C 언어를 사용하여 DCT를 소프트웨어로 구현하였다. 이를 다시 멘토그래픽스 사(Mentor Graphics Corporation)의 GDT (Generator Development Tool)의 M 언어를 사용하여 각 부분및 전체 2-D DCT를 모델링하고 동작 시뮬레이션(behavioral simulation)을 하였고 시뮬레이션툴(simulation tool) Lsim을 사용하여 회로 동작을 검증하였다. 이 모델링을 기반으로 하여 모듈 생성기(module generator) 언어인 L 언어를 사용하여 레이아웃(layout)을 하고 Lsim ADEPT 모드(mode)에서 회로 동작을 검증한 결과 1.5μm CMOS공정에서 40MHz로 동작함을 확인하였다. 또한 설계규칙 오류검사(design rule check)를 위하여 Lrc를 사용하고 레이아웃 결과를 확인하기 위하여 Led를 사용하였다.

본 논문의 2장에서는 DCT 구현을 위하여 사용한 계산 알고리즘과 동작 방법에 대한 설명을 한다. 3장에서는 2장에서 설명한 DCT 계산 알고리즘을 적용하기 위한 전체적인 구성과 각 모듈의 구조에 대하여 기술한다. 4장에서는 코어 프로세서 구현에서 사용한 모듈생성기법과 전체적인 구성에 대하여 기술하고 이의 성능에 대한 분석을 하며 5장에서는 결론을 내린다.

Ⅱ. DCT/IDCT 연산방식 및 알고리즘

2.1 연구배경

처음 DCT를 발표하면서 N-point DCT계수를 2N-point FFT를 이용하여 계산하는 방식을 제안하였다[1]. 그러나 이 방식은 2Nlog2N번의 복소수 가산과 N(log2N+1)번의 복소수 승산의 많은 계산을 필요로 한다[2]. 이러한 점을 개선하기 위하여 효율적인 DCT연산을 위한 여러 알고리즘들이 제안되었는데[2,3,4] 이들은 주로 가산과 승산횟수를 줄이는데 목적을 두고 있다.

또한 DCT를 하드웨어로 구현하기에 적합한 여러 알고리즘들과 구조에 관한 연구들이 진행되었고 이를 바탕으로 하드웨어를 구성하였다[5,6,7,8,9,10,11,12].

DCT를 구현하기 위한 구조에는 승산기(multiplier)를 사용하여 벡터행렬승산(vector matrix multiplication)을 하는 방법과 나비형 구조(butterfly architecture)와 회전 연산기(rotator)를 사용하는 방법, 그리고 분산연산을 이용하여 미리 계산된 부분합(partial sum)의 값들을 저장한 ROM(Read Only Memory)과 누적기(accumulator)를 이용한 방식등이 있다.

본 논문에서는 채택한 방식인 ROM과 누적기를 이용한 방식은 적은 계산단계를 거치므로 정밀도가 높으며 규칙적인 구조를 갖고 있기 때문에 VLSI로 구현하기 적합하다. 또한 DCT의 입력으로서 JPEG 기본 사양에서 사용하는 8×8 블록(block)을 사용하고, 입력 데이타의 정확도(precision)는 8비트로 표현되어 -128에서 127까지의 값을 갖게되고 변환된 출력값은 -1024에서 1023의 범위에 있게 된다. 입력 영상의 화소값을 f(i, j)라 하고, 주파수 영역에서의 계수를 F(u, v)라 하면 다음의 식으로 변환을 나타낼 수 있다[17].

· Forward DCT of JPEG :

$$F(u,v)=(1/4)C(u)C(v)\sum_{i=0}^{7}\sum_{j=0}^{7} f(i,j)$$
$$\cos(2i+1)u\pi/16)\cos(2j+1)v\pi/16)$$

· Inverse DCT of JPEG :

$$f(i,j)=(1/4)\sum_{u=0}^{7}\sum_{v=0}^{7} C(u)C(v)F(u,v)$$
$$\cos(2i+1)u\pi/16)\cos(2j+1)v\pi/16)$$

where

$$C(x)=\begin{cases}1\sqrt{2} & \text{if } x=0\\ 1 & \text{otherwise}\end{cases}$$

2-D DCT 변환의 특성 중, 분리가능성(separability)를 이용하여, 2-D DCT를 1-D DCT로써 수행할 수 있다[19,20].

2.2 DCT/IDCT의 정의

입력 데이타가 x(m), m=0, 1,...,(N-1)일때 1-D DCT와 IDCT(Inverse Discrete Cosine Transform)는 각각 〈식 2.1〉, 〈식 2.2〉와 같이 정의된다[1].

$$X(k)=\sqrt{(2/N)}\,c(k)\sum_{m=0}^{N-1} x(m)\cos\frac{(2m+1)k\pi}{2N} \qquad (2.1)$$

$$k=0,1,2,\cdots,(N-1)$$

$$x(m)=\sqrt{(2/N)}\sum_{k=0}^{N-1} c(k)X(k)\cos\frac{(2m+1)k\pi}{2N} \qquad (2.2)$$

$$m=0,1,2,\cdots,(N-1)$$

$$\text{where } c(k)\equiv\begin{cases}1/\sqrt{2} & \text{if } k=0\\ 1 & \text{otherwise}\end{cases}$$

x(m)은 입력 화소값을 나타내고 X(k)는 변환된 DCT계수의 k번째 항을 나타낸다.

2.3 부분합과 분산연산을 이용한 DCT의 계산

DCT와 IDCT연산은 M.L. Liou[13]등에 의하여 각각 다음과 같은 행렬곱셈으로 나타낼 수 있음이 증명되었다.

$$X=Cx \qquad (2.3)$$

$$x=C^{t}X \qquad (2.4)$$

〈식 2.3〉, 〈식 2.4〉에서 x와 X는 각각 N×1 입력 벡터(input vector)와 N×1의 변환된 DCT계수를 나타낸다. C^t는 C의 전치(transpose)행렬을 나타내며 C는 N×N DCT계수 행렬로서 〈식 2.5〉는 N=8인 경우의 DCT계수 행렬을 나타낸 것이다.

$$C=\begin{bmatrix} C_4 & C_4 & C_4 & C_4 & C_4 & C_4 & C_4 & C_4 \\ C_1 & C_3 & C_5 & C_7 & -C_7 & -C_5 & -C_3 & -C_1 \\ C_2 & C_6 & -C_6 & -C_2 & -C_2 & -C_6 & C_6 & C_2 \\ C_3 & -C_7 & -C_1 & -C_5 & C_5 & C_1 & C_7 & -C_3 \\ C_4 & -C_4 & -C_4 & C_4 & C_4 & -C_4 & -C_4 & C_4 \\ C_5 & -C_1 & C_7 & C_3 & -C_3 & -C_7 & C_1 & -C_5 \\ C_6 & -C_2 & C_2 & -C_6 & -C_6 & C_2 & -C_2 & C_6 \\ C_7 & -C_5 & C_3 & -C_1 & C_1 & -C_3 & C_5 & -C_7 \end{bmatrix} \quad (2.5)$$

위의 식에서 C_i는 DCT 계수로서 다음과 같다.

$$C_i=\frac{1}{2}\cos\frac{i}{16}\pi$$

또한 〈식 2.3〉과 〈식 2.4〉에서 각각 〈식 2.6〉과 〈식 7〉를 유도할 수 있다.

$$\begin{bmatrix} X_0 \\ X_4 \\ X_2 \\ X_6 \\ X_1 \\ X_3 \\ X_5 \\ X_7 \end{bmatrix} = \begin{bmatrix} C_4 & C_4 & 0 & 0 & 0 & 0 & 0 & 0 \\ C_4 & -C_4 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & C_2 & C_6 & 0 & 0 & 0 & 0 \\ 0 & 0 & C_2 & -C_6 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & C_1 & C_3 & C_5 & C_7 \\ 0 & 0 & 0 & 0 & C_3 & -C_7 & -C_1 & -C_5 \\ 0 & 0 & 0 & 0 & C_5 & -C_1 & C_7 & C_3 \\ 0 & 0 & 0 & 0 & C_7 & -C_5 & C_3 & -C_1 \end{bmatrix}$$

$$\begin{bmatrix} x_0+x_7+x_3+x_4 \\ x_1+x_6+x_2+x_5 \\ x_0+x_7-x_3-x_4 \\ x_1+x_6-x_2-x_5 \\ x_0-x_7 \\ x_1-x_6 \\ x_2-x_5 \\ x_3-x_4 \end{bmatrix} \quad (2.6)$$

1/2 ·

$$\begin{bmatrix} x_0+x_7 \\ x_1+x_6 \\ x_2+x_5 \\ x_3+x_4 \\ x_0-x_7 \\ x_1-x_6 \\ x_2-x_5 \\ x_3-x_4 \end{bmatrix} \begin{bmatrix} C_4 & C_2 & C_4 & C_6 & 0 & 0 & 0 & 0 \\ C_4 & C_6 & -C_4 & -C_2 & 0 & 0 & 0 & 0 \\ C_4 & -C_6 & -C_4 & C_2 & 0 & 0 & 0 & 0 \\ C_4 & -C_2 & C_4 & -C_6 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & C_1 & C_3 & C_5 & C_7 \\ 0 & 0 & 0 & 0 & C_3 & -C_7 & -C_1 & -C_5 \\ 0 & 0 & 0 & 0 & C_5 & -C_1 & C_7 & C_3 \\ 0 & 0 & 0 & 0 & C_7 & -C_5 & C_3 & -C_1 \end{bmatrix} \begin{bmatrix} X_0 \\ X_2 \\ X_4 \\ X_6 \\ X_1 \\ X_3 \\ X_5 \\ X_7 \end{bmatrix} \quad (2.7)$$

〈식 2.6〉은 입력 벡터는 2단의 나비형 연산기로 구현할 수 있고 〈식 2.7〉의 변환된 벡터는 1단의 나비형 연산기로 구성할 수 있다. 〈식 2.6〉과 〈식 2.7〉의 행렬은 모두 2항과 4항의 내적만을 포함하고 있는데 2항의 내적만을 구한뒤 4항의 내적은 부분합을 이용하여 구할 수 있다. 〈식 2.7〉에서 X_2와 X_4의 위치를 바꾸면 〈식 2.6〉에서 필요한 12개의 2항의 내적은 〈식 2.7〉에 있는 16개의 2항의 내적에 모두 포함된다. 그러므로 이들 내적의 값들을 ROM으로 구현하고 가산기와 누적기를 사용하면 DCT와 IDCT를 모두 처리할 수 있다.

또한 〈식 2.3〉, 〈식 2.4〉과 같은 행렬 연산은 분산연산을 이용하여 계산하면 적은 하드웨어로 효율적인 회로를 구현할 수 있다[14]. 분산연산을 사용하는데에 있어 8×8블록과 16비트의 정밀도를 지원하도록 한 클럭에 21트씩, 즉 2BAAT(bit-at-a-time)를 사용한다.

Ⅲ. DCT/IDCT 코어 프로세서의 구조 및 동작

3.1 전체적인 구조

2장에서 설명한 바와 같이 2-D DCT/IDCT는 행과 열에 대하여 각각 1-D DCT/IDCT를 수행함으로써 2-D DCT/IDCT의 결과를 구하게 된다. 〈그림 3.1〉은 전체 2-D DCT/IDCT의 구조를 나타낸 것이다. DCT/IDCT를 지정하는 신호선(signal line)에 의하여 DCT 또는 IDCT를 수행하게 된다.

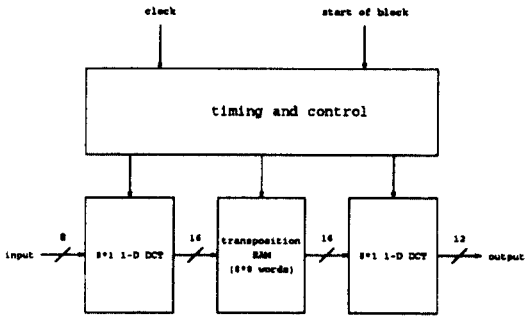

그림 3.1 2-D DCT/IDCT의 전체 구조

3.2 각 모듈의 구조

〈그림 3.2〉는 전체 2-D DCT/IDCT에서 1-D DCT/IDCT부분을 나타낸 것이다. 다음의 부절에서는 2-D DCT/IDCT연산에서 사용된 모듈들에 대한 설명과 함께 DCT/IDCT연산의 동작을 설명한다.

3.2.1 Q 레지스터

각각의 Q 레지스터는 병렬입력-병렬출력(parallel-in parallel-out) 레지스터로서 8개가 존재한다. 첫번째 레지스터는 입력 단자(input port)로부터, 그리고 나머지 7개의 Q 레지스터는 자신의 상단에 위치한 Q 레지스터로 부터 한 클럭에 하나의 화소값을 전달받아 8클럭 후에 8개의 화소값을 동시에 R 레지스터에 병렬적으로 전송한다.

3.2.2 R 레지스터

각각의 R 레지스터는 병렬입력-순차출력(parallel-in serial-out) 레지스터로서 입력 비트수 크기의 데이타를 입력받아 2비트씩 전송하는 역할을 한다. 전처리기에서의 가산과 감산으로 인하여 입력 비트수보다 많은 비트가 중간 계산단계에서 생성되는데 이를 효율적으로 처리하기 위하여 R 레지스터에서 미리 오버플로우가 발생하지 않도록 비트수를 늘려준다. 또한 부호확장(sign extension)을 위하여 R 레지스터에 입력되는 입력선 중에서 부호 비트가 전송되는 선을 R 레지스터에서 추가되는 상위 비트에 모두 연결함으로써 별도의 추가적인 모듈을 사용하지 않고 간단히 처리할 수 있다. 2비트 쉬프트연산은 하위 2비트를 출력선에 연결하고 R 레지스터에 저장되어 있는 화소값을 2비트씩 오른쪽으로 쉬프트하여 다시 R 레지스터에 피드백(feedback)하여 저장하는 방식으로 구현할 수 있다. 이러한 동작을 8클럭 동안 반복하고 다시 Q 레지스터로부터 새로운 화소값들을 병렬로 입력받는다.

3.2.3 전처리 나비형 연산기

DCT의 경우에는 R 레지스터에서 전송된 데이타를 〈식 2.6〉과 같은 입력 벡터의 데이타 형식에 맞추기 위하여 ROM에 전달되기 이전에 2단의 나비형 연산을 수행해야 한다. 첫번째 단에서는 〈식 2.6〉에서의 입력 벡터를 구하기 위하여 필요한 2개의 입력 화소값의 합과 차들을 구한다. 그리고 2번째 단에서는 첫번째 단에서 출력된 4개의 2항들의 합들을 입력받아 4항의 합과 차들을 계산한다. 이러한 2단의 나비형 연산기에서 계산된 값은 〈식 2.6〉의 입력벡터의 2비트씩으로서 ROM의 입력으로 연결된다.

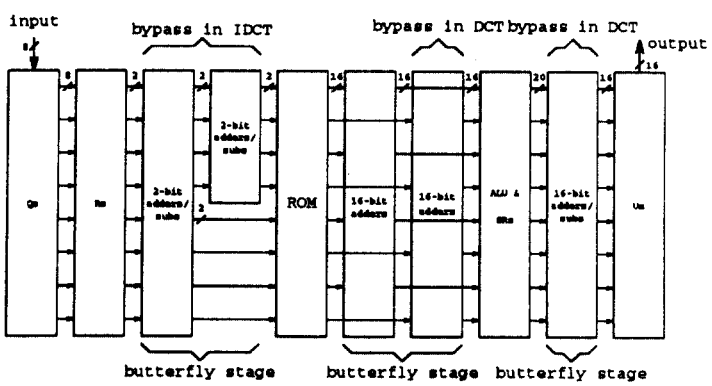

그림 3.2 1-D DCT/IDCT의 구조

3.2.4 2비트 가산기/감산기

전처리 단계의 나비형 연산기는 2비트의 가산기와 감산기들로 구성되어 있다. 가산기는 2개의 2비트 수를 입력받아 캐리 비트(carry bit)에 저장되어 있는 캐리와 더하여 2비트와 합을 출력하고 캐리를 캐리 비트에 저장한다. 감산의 경우에서도 비슷한 방법으로 바로우(borrow)와 계산을 하게 된다.

3.2.5 ROM 모듈

ROM모듈에서는 2비트씩 쉬프트되어 전처리된 8개의 입력 화소값들 또는 IDCT의 입력벡터를 전달받아 ROM에 저장된 이들의 내적의 값들을 병렬적으로 출력한다. ROM모듈은 모두 16개의 ROM으로 구성되어 있는데 이들 ROM은 각각 2개의 입력벡터에 대한 특정한 2항의 내적을 구하기 위하여 사용된다. ROM은 2개의 2비트 입력을 받으므로 ROM은 16워드를 갖게 된다.

3.2.6 후처리 나비형 연산기

후처리 단계에서는 ROM에서 출력되어 나온 2항들의 내적의 값들을 더하거나 빼서 입력데이타의 2비트에 대한 1-D DCT를 수행한 값을 계산한다.

ROM에서 출력된 2항들의 내적을 이용하여 DCT와 IDCT의 값을 구하게 되는데 DCT연산의 경우에는 X_0, X_2, X_4, X_6의 값은 ROM의 출력에서 직접 구할 수 있으므로 후처리 단계를 거치지 않는다. X_1, X_3, X_5, X_7의 경우는 2항의 내적 2개를 가산하여 구하게 되는데 후처리기의 첫번째 나비형 연산단계에서 이를 수행한다. IDCT의 경우에는 8개의 출력값의 모든 경우에 2항의 내적 2개씩을 가산하여 계산하여야 한다.

〈식 2.7〉에서 나타나 있듯이 IDCT의 경우에는 행렬을 곱한 결과가 두 화소값의 합 또는 차의 값의 1/2의 값을 갖게 하므로 이들 값의 합과 차로서 화소값을 구하게 된다. DCT의 경우에는 이러한 나비형 연산을 필요로 하지 않으므로 이 단계에서는 단지 순서만을 재정렬하는 역할을 한다.

3.2.7 누적기

누적기는 나비형 연산의 결과에서 나오는 출력값과 누적기에 저장되어 있는 값을 2비트 쉬프트한 값을 더한다. 누적기의 값을 2비트 쉬프트하는 이유는 나비형 연산에서 출력되는 값과 자리수를 정렬하기 위해서이다. 다음 클럭에서 다시 누적기의 값을 오른쪽으로 2비트 쉬프트하고 이를 나비형 연산에서 나오는 출력값과 더하여 누적기에 저장한다. 이러한 동작을 8클럭 동안 반복하여 16비트에 대한 계산값을 모두 더한 뒤 U레지스터에 결과를 보낸다. 누적기는 가산과 감산을 수행할 수 있는데 감산을 수행하는 경우는 마지막에 부호비트가 전송되는 경우이다. 누적기에서는 항상 2비트 쉬프트만을 수행하기 때문에 별도의 쉬프터가 필요없이 간단하게 2비트 쉬프트와 반올림을 수행할 수 있다. 누적기에서 나온 데이타는 후처리의 마지막 나비형 연산단계를 거쳐 U 레지스터로 전달된다.

3.2.8 U 레지스터

U 레지스터는 계산된 8개의 값들을 병렬적으로 입력받아 하나씩 순차적으로 출력하는 역할을 한다. 즉, 8클럭마다 병렬적으로 입력된 값들을 매 클럭마다 자신의 상단에 위치한 U 레지스터에 전송한다. 가장 위에 위치한 U 레지스터에서는 데이타를 출력하게 된다. 이 출력값들은 1-D DCT의 결과로서 전치 RAM이나 출력단자로 보내지게 된다.

3.2.9 전치 RAM

〈그림 3.1〉의 2-D DCT의 전체구조에서와 같이 2D-DCT는 2회의 1D-DCT로 계산하게 된다. 2개의 1D-DCT 모듈 중 첫번째 모듈에서 생성된 8×8 블록의 64워드의 중간 결과는 RAM에서 전치되어 다시 1D-DCT를 수행하게 된다. 이러한 전치를 수행하기 위하여 8×8 데이타 블록이 RAM에 행(row) 순서로 데이타가 저장되었으면 이 블록을 읽을 때에는 열(column)순서로 이를 읽고, 열 순서로 저장되었으면 행 순서로 데이타를 읽는다. 그리고 한 블록을 전부 저장했을 때에는 행과 열을 바꾸어 이번에는 열 순서로 저장하고 행 순서로 읽는다. 이런 식으로 반복함으로써 다른 데이타의 값을 변경하지 않으면서 행렬 전치를 수행할 수 있다.

데이타의 행과 열 입출력 순서를 변경하기 위하여 각 블록의 시작마다 6비트의 주소(address) 지정선의 상위 3비트와 하위 3비트를 교환(exchange)한다. 이러한 기능을 위하여 블록의 마지막 워드를 알아내기 위하여 6비트 계수기(counter), 데이타 선의 교환을 위한 MUX, 그리고 행 또는 열 순서의 상태를 저장할 수 있는 레지스터를 사용한다.

3.3 DCT/IDCT의 계산 정밀도

DCT 변환을 수행하는데 있어 승산(multiplication)과 가산(addition)을 줄이기 위한 여러가지 알고리즘이 제안되었는데 이러한 고속 알고리즘들은 대부분 고정 정밀도(fixed precision)의 실수 연산을 하게 된다. 이러한 연산에서 오는 반올림이나 버림으로 인하여 각각의 알고리즘을 약간의 오류가 발생하게 된다. 이러한 오류의 허용치에 대하여 CCITT 표준안에서는 구체적인 정의하고 있고 정밀도 검증 방법도 제시하고 있다[15].

C를 IDCT의 행렬 계수를 표현하는 비트의 수라 하고, I를 첫번째 DCT를 수행한 후의 중간 결과를 표현하는 비트 수라하면, 같은 구조로 선행되었던 연구[16]에서는 CCITT IDCT 표준에 부합하려면 C〉12, I〉4 또는 C〉13, I〉13으로 선택해야 함을 증명하였다. 본 연구에서는 C=13, I=16을 사용한다.

Ⅳ. DCT/IDCT 코어 프로세서의 구현 및 성능평가

3장에서 설명한 DCT연산방식은 구조가 간단하고 단위성(modualrity)이 우수하므로 VLSI로 구현하기가 용이하다. 본 연구에서는 이러한 구조를 가지는 DCT/IDCT 연산기를 CMOS VLSI 기술을 이용하여 모듈생성기법으로 설계하였다. 모듈생성기법을 사용하므로 특정한 반도체 기술에 의존하지 않으며 매개 변수등을 변화시킴으로써 간단하게 비트수 등을 조절할 수 있다.

DCT 코어 프로세서의 논리 설계는 Mentor Graphics에서 제공하는 GDT(Generator Developement Tool)의 M언어를 사용하여 수행하였으며 동작 검증은 GDT의 Lsim을 이용하였다. DCT 코어 프로세서는 GDT의 L언어로 설계하였고 Lsim ADEPT mode에서 검증하였다.

4.1 계층적인 구조

구현한 코어 프로세서는 모듈 생성기를 사용하여 구현하기 때문에 용이하게 계층적인 설계를 할 수 있다. 리프 셀 생성기들을 구성하고 이들을 이용하여 블럭 셀(block cell)들을 설계한다. 그리고 이 블럭 셀들을 이용하여 목적 셀(target cell)을 구성한다. 이러한 계층적인 구조는 하위 셀들의 호출(call)로 이루어지는데 하위 셀의 호출시에 매개 변수들을 사용함으로써 같은 하위 셀을 여러 상위 셀들에서 사용할 수 있다.

4.2 리프 셀(leaf cell) 생성기

리프 셀에서 사용하는 매개변수들은 셀에 따라 약간의 차이가 있으나 기본적으로 사용되는 매개변수들로 width, ratio, power, track등이 있다. Width는 트랜지스터의 게이트 채널(gate channel)의 폭을 나타내는 변수로서 이를 증가시킴으로써 지연시간을 단축시킬 수 있으나 면적이 증가된다. 본 연구에서는 최소 폭의 3배를 width로 사용하였다. Ratio는 풀업(pull-up)대 풀 다운(pull-down)의 트랜지스터의 게이트 채널의 폭의 비를 나타내는 변수로서 2를 기본으로 사용하고 있다. power는 전원선(VDD)와 접지선(GND)의 폭을 나타내는 변수이다. Track은 리프 셀 내부에서 배선을 위하여 사용하는 가로의 메탈1(metall) 트랙(track)의 수로서 기본적으로는 5트랙을 사용하고 있다. 〈그림 4.1〉은 본 연구에서 사용한 리프 셀 구성방식을 나타낸 것이다.

리프 셀 내부에서는 메탈2(metal2)를 사용하지 않고 메탈1과 폴리(poly)만을 사용하여 배선한다. 전원선과 접지선 방향으로 메탈1 선을 배치하고 이에 수직으로 폴리선을 배치하여 사용한다. 또한 전원선과 접지선은 리프 셀에서 생성하지 않고 전원선과 접지선에 연결할 수 있는 메탈2 노드(node)만을 생성한다. 입력선들은 좌측에서 들어오고 우측으로 출력선들이 나간다. 제어신호(control signal)들은 리프 셀의 상단과 하단에 생성하였는데 이는 상위 셀에서 배치와 배선을 용이하게 하기 위해서이다. 그리고 입력과 출력에는 각각 메탈2 노드를 생성하여 상위 셀에서 메탈2로 배선을 할 수 있도록 구성하였다.

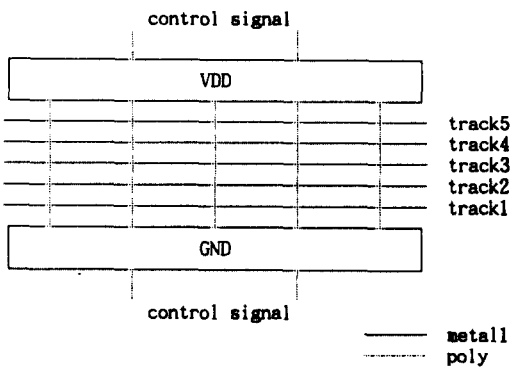

그림 4.1 리프 셀의 설계구조

4.3 블럭 셀(block cell) 생성기

블럭 셀들은 주로 1비트의 리프 셀들을 이용하여 여러비트를 연결함으로써 구성된다. 블럭 셀에서 리프 셀들을 호출하여 사용할 때에는 전원선과 접지선의 이격 거리를 줄이기 위하여 하나씩 교대로 상하 대칭으로 위치를 변경한 리프 셀들을 배치하고 2개의 리프 셀들이 공통으로 전원선과 접지선을 공유하도록 구성하였다. 〈그림 4.2〉에 이러한 구성이 나타나 있다.

〈그림 4.2〉에서 type 1은 셀을 위치를 변경하지 않은 셀이고 type 2는 상하 반대로 위치를 변경한 셀이다. 이러한 위치 변경을 위하여 'updn'이라는 매개변수를 사용한다. 또한 위치 변경으로 인하여 입력선과 출력선들의 위치가 반대로 되는 것을 방지하기 위하여 track을 지정할 때 updn 매개 변수를 사용하여 지정함으로써 입출력선의 위치를 일정하게 하였다.

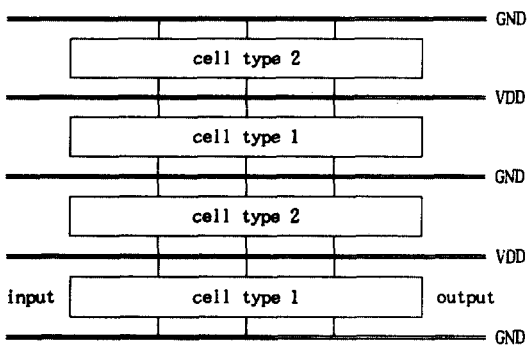

그림 4.2 블럭 셀의 배치 및 배선 구조

다음은 이러한 기법을 이용하여 track을 지정한 Q 레지스터의 L언어 프로그램의 일부분이다.

```
L::TECH ANY
#include "my_layout.macro"
CELL q(width=3.0, ratio=2.0, power=15,
track=8, cell=8, updn=0)
{
  CALL TECH_INCLUDE no_cell (FILE="lay-
  out. aux");
                    .
                    .
                    .
  M1M2 outr[num] AT(pipo[num].outdata. X,
  pipo[num].gnd[1]. Y +gnd2ncon+
  met2met*(track-2*num-1)*updn
  +num+1));
  /* num: the number of the output node
  to R reg. */
}
```

그림 4.3 Q 레지스터의 트랙 지정방식

이 예에서는 Q 레지스터의 R 레지스터로의 출력선을 생성시키기 위한 중간 노즈인 'outr'의 트랙 위치를 지정하는 경우를 보여준다. 매개 변수 updn은 일반 셀인 경우 0값을 사용하고 상하 위치를 변경시킬 때에는 1값을 사용한다. 또한 'track'은 트랙 번호를 나타내는 변수이다.

블럭 셀에서는 메탈1과 메탈2를 사용하여 배선을 하는데 수평 방향으로는 메탈2를 사용하고 수직방향으로는 메탈1을 사용하여 배선을 한다.

4.4 목적 셀 생성기

전체 DCT 코어 프로세서는 부분합이 저장된 ROM과 ROM 이전의 입력을 생성하기 위한 전처리 부분, 그리고 ROM에서 얻은 값으로 부터 결과를 계산하는 후처리 부분으로 구성되어 있다. 배선을 위하여 가로 방향으로 메탈2, 세로 방향으로는 메탈1을 사용하였다. 또한 본 프로세서는 2-phase non-overlapping clock을 사용한다. 〈그림 4.4〉에 코어 프로세서의 레이아웃(layout)이 나타나 있다.

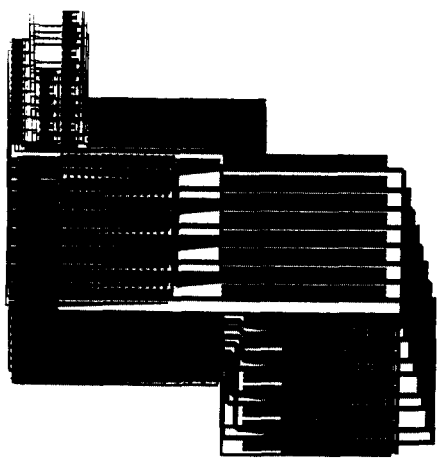

그림 4.4 DCT/IDCT 코어 프로세서의 전체 레이아웃

4.5 성능 분석

본 논문에서는 GDT사의 Lsim ADEPT mode에서 25℃ 5V의 환경에서 시뮬레이션을 수행하였고 3.0μm, 1,5μm, 1.0μm, 0.8μm 공정에서 설계규칙을 검증하였다. 〈표 4.1〉에 각 설계공정에서의 코어 프로세서의 크기가 나타나 있다.

표 4.1 각 설계공정에 따른 코어 프로세서의 크기

설계공정	크기
scmos (3.0μm)	15.4mm × 17.4mm
vs1500 (1.5μm)	10.1mm × 10.9mm
csp3 (1.2μm)	9.9mm × 11.2mm
csp3s (1.0μm)	5.9mm × 6.3mm
csp4 (0.8μm)	3.4mm × 7.0mm

본 연구의 DCT 코어 프로세서는 5단계의 파이프라인 구조로 이루어져 있는데 이들은 Q 레지스터, R 레지스터, 전처리 단계와 ROM 입력 사이의 레지스터, 누적기, 그리고 후처리 단계로 이루어져 있다. 이 중 가장 지연시간이 긴 부분인 누적기에서 24ns의 지연시간을 필요로 한다. 〈그림 4.4〉는 Lsim ADEPT mode로 누적기를 시뮬레이션한 결과가 나타나 있다. 이 코어 프로세서는 처음 데이타가 입력된뒤 처음 결과값이 나오기까지 97클럭 주기의 지연시간을 필요로 한다. 그러나 처음 출력값이 생성된 이후로는 매 클럭 주기마다 연속으로 하나의 출력값이 발생된다. 본 DCT 코어 프로세서는 40MHz에서 동작하므로 여러분야의 실시간 응용에 사용될 수 있다.

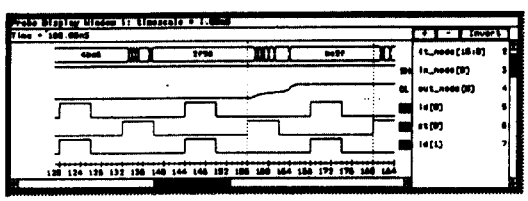

그림 4.5 1.5μm CMOS 공정 25℃ 환경에서 Lsim ADEPT mode로 수행한 누적기 시뮬레이션 결과

V. 결 론

DCT/IDCT 연산은 영상과 비디오 압축및 복원에서 가장 효율적인 부호화 방식의 하나로 많은 응용분야에서 널리 사용되고 있고 JPEG등의 표준기관에서도 DCT를 채택하고 있다.

본 논문에서는 이러한 DCT 연산의 실시간 처리등을 필요로 하는 여러 응용에서 사용될 수 있도록 VLSI CMOS기술을 이용하여 설계하였다. 또한 기존의 완전주문자방식(full custom)으로 회로를 구현하면 반도체 제작공정(technology)에 따라 설계규칙(design rule)이 다르므로 전체를 다시 디자인(design)하여야 하지만 모듈 생성기법을 사용하여 구현하면 설계규칙이 변하더라도 다시 레이아웃(layout)할 필요가 없으며, 매개변수(parameter)를 조정함으로써 간단히 비트수 등도 변화시킬 수 있는 장점이 있다.

설계된 DCT 코어 프로세서는 분산연산에 기초하여 부분합 방식과 참조 테이블을 사용함으로써 구조가 규칙적이고 적은 면적을 차지한다. 또한 부분합 방식을 이용함으로서 면적의 큰 증가없이 IDCT도 수행할 수 있다. 본 DCT 코어 프로세서는 전체가 파이프라인 구조로 설계되어 있어 빠른 속도로 연산을 수행할 수 있다. 또한 CCITT에서 규정하고 있는 정밀도 이상의 높은 정밀도로 DCT 연산을 수행한다.

본 DCT 코어 프로세서는 Mentor Graphics사의 GDT상에서 모델링, 동작 검증, 구현 및 회로 검증을 수행하였으며 이중메탈을 사용하는 3.0μm, 1.5μm, 1.3μm, 1.0μm, 그리고 0.8μm 설계 공정에서 설계 규칙을 검증하였다. 또한 1.5μm CMOS공정에서 Lsim ADEPT mode로 시뮬레이션한 결과 40MHz로 동작함을 확인하였다.

앞으로의 연구에서는 가장 지연시간이 큰 누적기 부분의 리플캐리 가산기(ripple carry adder)대신에 CSA(Carry Skip Adder)나 VBA(Variable Block Adder)등을 사용하여 100Mhz이상에서 동작할 수 있도록 연산 시간을 단축시키는 방법, 또는 파이프라인 단계를 추가하여 시간을 단축시키는 방법과 ROM의 크기와 나비형 연산기의 크기를 조정하여 전체 프로세서의 면적과 지연시간을 단축시키는데 대한 연구가 계속되어야 할 것이다.

참 고 문 헌

1. N. Ahmed, T. Natarajan, and K.R. Rao, "Discrete Cosine Transform," IEEE Trans. Commput., vol.C-23, pp.88-93, 1974.

2. W. Chen, C.H. Smith, and S.C. Fralick, "A fast computational algorithm for the discrete cosine transform," IEEE Trans. Commun., COM-25, pp.1004-1009, Sept. 1977.
3. B.G. Lee, "A new algorithm to compute the Discrete Cosine Transform," IEEE Trans. Accoust. Speech, Signal Processing, vol.ASSP-32, no.6, pp.1243-1245, Dec.1984.
4. S.C. Chan K.L. Ho, "A New Two-Dimensional Fast Cosine Transform Algorithm," IEEE Trans. Signal Processing, vol.39, pp.481-485, Feb. 1991.
5. M.H. Lee, "On Computing 2-D Systolic Algorithm for Discrete Cosine Transform," IEEE Trans. on Circuits and Systems, vol.37, no.10 pp.1321-1323, Oct.1990.
6. M.T. Sun, L. Wu, and M.L. Liou, "A Concurrent Architecture for VLSI Implementation of Discrete Cosine Transform," IEEE Trans. Circuits Syst., vol.CAS-34, pp.992-994, Aug. 1987.
7. M.T. Sun, T.C. Chen, and A.M. Gottlieb, "VLSI Implementation of a 16-by-16 Discrete Cosine Transform," IEEE Trans. Circuits Syst., vol.36, no.4, April. 1989.
8. M. Vetterli, A. Ligtenberg, "A Discrete Fourier-Cosine Transform Chip," IEEE J. of Selected Areas in Communications, vol. SAC-4, 1, Jan. 1986.
9. M. Maruyama, H. Uwabu, I. Iwasaki, H. Fujiwara, T. Sakaguchi, M.T. Sun, M.L. Liou, "VLSI Architecture and Implementation of a Multi-Function, Discrete Cosine Transform Processor," Proc. SPIE Symp. on Visual Commun. Image Proc., vol.1360, pp.410-417, Oct. 1990.
10. N.T. Cho, S.U. Lee, "DCT Algorithms for VLSI Parallel Implementation," IEEE Trans. Accoust. Speech, Signal Processing, vol.ASSP-38, pp.121-127, no.1, Jan. 1990.
11. Rovert M. Harlick, "A Storage Efficient Way to Implement the Discrete Cosine Transform," IEEE Trans. on Computer, Jul. 1976.
12. A. Peled and J.A. Bellisio, "A New Harbware Realization of Digital Filters," IEEE Trans. Accoust. Speech, Signal Processing, vol.ASSP-22, pp.456-462, Dec. 1974.
13. M.L. Liou, J.A. Bellisio, "VLSI Implementataion of Discrete Cosine Transform for Visual Communication," Proc. Int. Conf. on Commun. Tech., Bejing, China, Nov. 1987.
14. Stanley A. White, "Applications of Distributed Arithmetic to Digital Signal Processing : A Tutorial Riview," IEEE ASSP Magazine, vol. 6, no.3, July 1989.
15. International Standard DIS 10918-1 CCITT Recommendation T.81
16. Mismatch Errors of Matrix Multiplication IDCT, Doc. #364, CCITT SGXV Working Party XV/4, specialist Group on Coding for visual Telephony, Sept. 1988.
17. G. Wallace, W.B. Penennbaker, and J.L. Mitchell, JPEG Technical Specification(revision 5), JPEG-8-R5, Dec. 1989.
18. Leonardo Chiarigline, Status report of ISO MPEG, ISO/IEC JTC1/SC2/WG11, Sep. 1990.
19. Rafael C. Gonzalez, Paul Wintz, DIGITAL IMAGE PROCESSING : Second Edition, ADDISON-WESLEY PUBLISHING COMPANY, Nov. 1990.
20. Anil K. Jain, Fundermentals of DIGITAL IMAGE PROCESSING, Prentice-Hall International, Inc., 1989.

黃 俊 夏(Joon-Ha Hwang)정회원
1991년 : 연세대학교 이과대학 전산과학과 학사
1993년 : 연세대학교 이과대학 전산과학과 석사
※주관심분야 : DSP 칩 설계, ASIC 설계, 컴퓨터 구조

韓 鐸 敦(Tack-Don Han) 정회원
1978년 : 연세대학교 공과대학 전자공학과 학사
1983년 : Wanyne State University 컴퓨터공학 석사
1987년 : University of Massachusetts 컴퓨터공학 박사
1981년 : 금성전기 연구원
1987년 ~ 1989년 : Cleveland 주립대학 조교수
1989년 ~ 현재 : 연세대학교 전산과학과 부교수
※주관심분야 : 병렬처리 컴퓨터 구조 및 알고리즘 연구, Fault-Tolerant System 연구, ASIC 설계 및 VLSI 칩설계의 Area/Time tradeoffs 에관한 연구