하드웨어 기반의 H.264 정수 변환 및 양자화 구현

준회원 임 영 훈*, 정회원 정 용 진**

Hardware Implementation of Integer Transform and Quantization for H.264

Young-Hun Lim* *Associate Member*, Yong-Jin Jeong** *Regular Member*

요 약

본 논문에서는 H.264의 정수 변환 모듈, 양자화 모듈, 역양자화 모듈, 정수 역변환 모듈에 대한 하드웨어 구조를 제안한다. 새로운 동영상 압축기술인 H.264의 전체 구성 중에서 핵심 부분인 동영상 데이터의 영역 변환 및 양자화 기능들을 하드웨어로 설계할 수 있도록 알고리즘을 기술하고, 저전력 설계를 위하여 하드웨어 사이즈를 최소화하도록 구조를 정하였다. 구현된 전체 모듈들은 PCI 인터페이스를 통한 Altera APEX-II FPGA 구성과 삼성 STD130 0.18um CMOS Cell Library를 이용하여 각각 합성하고 검증하였다. 이렇게 검증된 구조의 성능은 ASIC으로 구현하였을 경우 최대 동작 주파수가 100MHz이며, QCIF의 사이즈 기준으로 초당 최대 1295 프레임의 계산을 수행할 수 있으며, 이는 하드웨어 기반의 H.264 실시간 부호화기를 설계하기에 적합한 구조임을 보여준다.

Key Words : H.264, MPEG4 AVC, Integer Transform, Quantization.

ABSTRACT

In this paper, we propose a new hardware architecture for integer transform, quantizer, inverse quantizer, and inverse integer transform of a new video coding standard H.264/JVT. We describe the algorithm and derive hardware architecture emphasizing the importance of area for low cost and low power consumption. The proposed architecture has been verified by PCI-interfaced emulation board using APEX-II Altera FPGA and also by ASIC synthesis using Samsung 0.18 um CMOS cell library. The ASIC synthesis result shows that the proposed hardware can operate at 100 MHz, processing more than 1,300 QCIF video frames per second. The hardware is going to be used as a core module when implementing a complete H.264 video encoder/decoder ASIC for real-time multimedia application.

Ⅰ. 서 론

최근 디지털 멀티미디어 네트워크는 광범위한 서비스를 제공하면서 급속히 발전되고 있다. 그 중에 디지털 동화상을 이용한 통신 서비스를 실현하기 위해서는 대용량의 회선을 이용하거나 아니면 데이터를 압축하지 않으면 안 된다. 지금까지 동화상 압축을 위한 표준으로MPEG-1[1], MPEG-2[2], MPEG-4[3], H.261[4], H.263[5] 등이 있으나, 최근 대두되고 있는 무선 디바이스와 같은 저비트율을 지향하는 개체에서는 효율적으로 사용하기에 부적합한 방식이므로 좀더 효율적인 압축방식을 요구하고 있다.

본 논문에서 다루게 될 H.264 혹은 MPEG-4 AVC (Advanced Video Coding)라고 불리는 이 기술은 다른 기술보다 네트워크 자원을 덜 쓰면서도 DVD수준의 동영상을 인터넷을 통해 전송할 수 있게끔 고안된

* 광운대학교 전자통신공학과 실시간구조 연구실(limyh14@explore.kw.ac.kr), ** 광운대학교 부교수(yjjeong@daisy.kw.ac.kr)
논문번호 : 030372-0825, 접수일자 : 2003년 8월 25일
※본 연구는 광운대학교 2003 교내학술연구 및 정보통신연구원, IDEC의 지원으로 이루어졌습니다.

기술 표준이다. ITU-T (International Telecommunication Union)와 MPEG (Moving Picture Experts Group)이 연합하여 구성된 JVT(Joint Video Team)는 이 기술을 차세대 동영상 비디오 압축기술로 표준관련 국제기구에서 인증을 받는다는 계획을 세우고 표준화 마무리를 진행하고 있다.

본 논문에서는 H.264[6]의 동화상 부호화 알고리즘의 핵심 부분인 정수 변환 (Integer Transform) 모듈, 양자화 (Quantization) 모듈, 역양자화 (Inverse Quantization) 모듈, 정수 역변환 (Inverse Integer Transform) 모듈을 구성하는 하드웨어 구조를 제안하고 많은 연산량이 소요되는 블록의 알고리즘을 최적화 시키고 구현하는 방법을 제안한다.

본 논문의 구성은 다음과 같다. 먼저 2장에서는 H.264의 정수 변환, 양자화, 역양자화 정수 역변환의 알고리즘 구조에 대해 살펴보고 3장에서는 각각의 모듈들을 제안된 알고리즘을 이용하여 최적화 하는 하드웨어 구현을 기술한다. 그리고 4장에서는 제안된 하드웨어 구조를 설계 검증을 하고 H.264 표준에서 제공된 소프트웨어 구조와 비교하여 그 결과를 분석한다. 마지막으로 5장에서 결론을 맺는다.

II. 알고리즘의 구조

본 장에서는 H.264의 부호화기 전체 구조 중에서 정수 변환, 양자화, 정수 역변환, 역양자화의 구조에 대하여 기술하고 하드웨어로 설계하기 적합한 구조로 변환하는 알고리즘을 기술한다.

1. H.264의 부호화기 구조

입력된 영상데이터의 각각의 마크로블록 (Macro block)은 인트라모드 (Intra Mode)와 인터모드(Inter Mode)로 부호화 되어진다. 인트라모드는 입력된 영상 데이터로부터 구성되어지며, 인터모드는 재구성된 기준 영상 (Reference Frame)을 이용하여 움직임 예측 (Motion Estimation)과 움직임 보정 (Motion Compensation)에 의하여 구성되어진다. 각 모드에 의해 생성된 값과 입력된 영상 데이터와의 차로 생성된 레지듀얼 (Residual) 마크로블록은 정수 변환과 양자화를 거쳐 양자화된 변환 계수로 변환되며 엔트로피코딩 (Entropy Coding)을 거치게 된다. 그와 동시에 생성된 양자화된 변환 계수는 다시 역양자화와 정수 역변환을 거치고 인트라모드 또는 인터모드로 형성된 값과 더하여진 후 필터를 거쳐 기준 영상으로 사용된다. 이러한 부호화 과정 중에서 아래 그림 1에 점선으로 표시된 것이 본 논문에서 구현된 정수 변환, 양자화, 역양자화, 정수 역변환 구조이다.

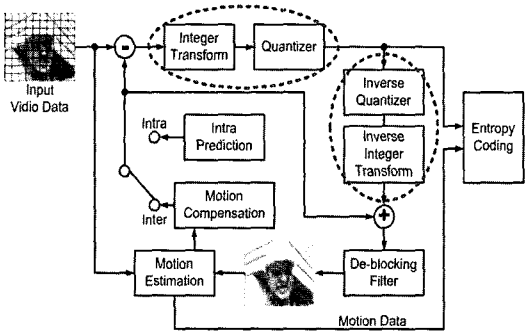

그림 1. H.264의 부호화기 구조

2. 정수 변환의 구조

MPEG-1, MPEG-2, MPEG-4, H.261, H.263과 같은 이전 동영상 압축 표준들은 8화소x8화소 DCT (Discrete Cosine Transform)의 단일 블록 크기를 기초로 하여 변환을 수행한다[8]. 그러나 H.264의 변환 방식은 변환기에 입력되는 데이터의 방식에 따라 세 가지 방식으로 변환이 이루어진다. 첫 번째 방식으로 인트라모드의 16화소x16화소 모드에서 예측되는 휘도 DC(Direct Current) 계수의 4화소x4화소 배열을 위한 변환이 있고, 두 번째 방식으로 채도 DC 계수의 2화소x2화소 배열을 위한 변환이 있으며, 마지막 세 번째 방식으로 위의 두 가지 변환 방식을 제외한 기본적인 휘도 AC (Alternating Current) 계수 또는 채도 AC 계수의 4화소x4화소 블록을 입력으로 하는 변환이 있다. 만약 추가적으로 "Adaptive Block Size Transform" 모드를 사용한다면 움직임 보정으로 선택되어진 사이즈(4x8, 8x4, 8x8, 16x8, etc)에 의해 변환 방식이 선택되어질 수 있다. 아래 그림 2는 하나의 마크로블록 안에서 변환 방식에 따라 입력되어지는 각 블록 변환의 진행 순서를 나타낸다[6].

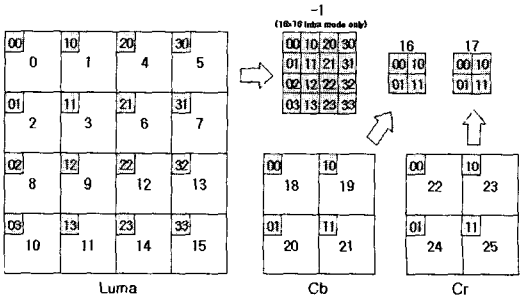

그림 2. 4화소x4화소의 변환 진행 순서

만약 마크로블록이 16화소x16화소 휘도모드 방식으로 변환된다면 휘도 블록의 각 4화소x4화소의 DC 계수로 만들어진 블록의 순서를 -1로 하여 제일 먼저 변환하게 된다. 그 다음부터는 순서대로 휘도 블록 데이터의 0에서 15까지 진행한다. 그리고 채도 DC 계수 값을 추출하여 만들어진 16, 17 블록은 2화소x2 화소 변환 방식을 사용하여 변화하고, 마지막으로 18-25번까지의 채도 블록 데이터를 변환하면서 하나의 마크로블록 변환이 끝나게 된다.

이러한 순서대로 진행되는 정수 변환은 기본적으로 DCT를 기반으로 하였으나 몇 가지 차이점을 가지고 있다. 첫 번째로 H.264에서 쓰이는 DCT는 정수 변환 방식이다. 즉, 모든 동작은 정수 연산으로 수행되어지며 부동소수점 연산으로 계산되어지는 일반적으로 쓰이는 DCT와는 다른 것이 특징이다. 두 번째로 역 정수 변환은 H.264에서 완전한 표현이 가능하며 만약 정확히 기술이 되어진다면 부호화하고 복호화하는 과정에서 데이터의 오차가 없다. 마지막 세 번째로 변환기 코어(core)중에서 하드웨어적으로 큰 면적을 차지하고 시간 지연이 많은 곱셈기와 같은 큰 모듈이 필요 없고 오직 덧셈기와 같은 상대적으로 적은 면적과 시간 지연이 적은 모듈들만으로도 구성이 가능하다.

영상 압축에 사용되는 변환 공식 중에서 일반적인 DCT의 변환공식은 아래와 값이 식(1)과 같고 H.264에서 쓰이는 정수 변환 공식은 아래와 같이 식(2)과 같다[7].

$$Y = AXA^{T}$$

$$= \begin{pmatrix} a & a & a & a \\ b & c & -c & -b \\ a & -a & -a & a \\ c & -b & b & -c \end{pmatrix} \begin{pmatrix} X \end{pmatrix} \begin{pmatrix} a & b & a & c \\ a & c & -a & -b \\ a & -c & -c & b \\ a & -b & a & -c \end{pmatrix}$$

$$\text{where} \quad a = \frac{1}{2},\ b = \sqrt{\frac{1}{2}}\cos\left(\frac{\pi}{8}\right),$$

$$c = \sqrt{\frac{1}{2}}\cos\left(\frac{3\pi}{8}\right). \qquad (1)$$

$$Y = (CTC^{T}) \otimes E$$

$$= \left(\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & d & -d & -1 \\ 1 & -1 & -1 & 1 \\ d & -1 & 1 & -d \end{pmatrix} \begin{pmatrix} X \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 & d \\ 1 & d & -1 & -1 \\ 1 & -d & -1 & 1 \\ 1 & -1 & 1 & -d \end{pmatrix} \right)$$

$$\otimes \begin{pmatrix} a^2 & ab & a^2 & ab \\ ab & b^2 & ab & b^2 \\ a^2 & ab & a^2 & ab \\ ab & b^2 & ab & b^2 \end{pmatrix}$$

$$\text{where} \quad a = \frac{1}{2},\ b = \sqrt{\frac{2}{5}},\ d = \frac{1}{2}. \qquad (2)$$

식 (2)에서 ⊗ (Scaling Operation)는 계산된 CTC^{T}의 행렬과 E의 행렬에서 각각 같은 위치의 값과의 곱셈을 나타낸다. H.264의 문맥 중에서 사용되어진 정수 변환은 일반적인 DCT의 수행 결과와 거의 동일한 결과를 나타내며 몇 가지의 중요한 장점들을 가지고 있다[8]. 정수 변환 공식 중에 CTC^{T}는 오직 정수 연산의 덧셈기, 뺄셈기, 쉬프트만으로 결과를 도출할 수 있다. 그러나 정수 변환 모듈 이후의 양자화 모듈에서는 ⊗의 계산을 위해 하나의 곱셈기를 요구한다.

3. 양자화의 구조

입력 신호를 유한한 개수로 근사화하는 것을 양자화라 하는데 이것은 비선형 연산이며 원신호의 완전한 복원이 불가능하다. 아래에 그림 3에서 보이는 것과 같이 선형의 입력 레벨을 근사화된 출력 레벨로 바꾸는 방식으로 양자화를 진행하게 되는데 실질적으로 압축의 양을 결정하게 된다. H.264는 스칼라 양자화 모듈을 가지고 있다. 즉, 나눗셈기를 사용하지 않고 부동소수점연산을 하지 않음을 의미한다.

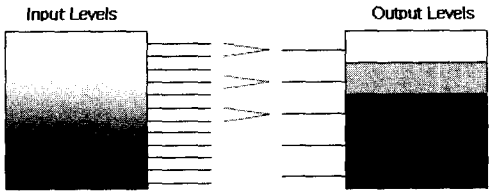

그림 3. 양자화의 기본 개념

양자화 모듈은 H.264에서 제공하는 QP (Qutization Parameter)에 의하여 인덱싱(Indexing) 되어있다. 모두 52레벨로 되어있으며 각각의 Qstep (Quantization Step)값을 가지고 있다. 이렇게 정해진 넓은 범위의 Qstep은 Bit-rate와 Quality 사이의 관계에서 정밀하고 유연하게 부호화를 할 수 있게 한다.

위와 같은 방식으로 구성되어지는 양자화 모듈은 정수 변환 모듈에서의 공식과 연계되어 아래 식 (3),(4),(5)와 같이 유도가 된다. 여기서 PF는 Position Factor이며, MF는 Multiplication Factor이다.

$$Z = round\left(\frac{Y}{Qstep}\right). \qquad (3)$$

$$Z = round\left(W \cdot \frac{PF}{Qstep}\right),\ W = CXC^{T}. \qquad (4)$$

$$Z = round\left(W \cdot \frac{MF}{2^{qbits}}\right),$$

$$\text{where} \quad qbits = 15 + floor(QP/6). \qquad (5)$$

최종적인 공식으로 아래 식(6)과 같이 유도가 된다.

$$Z=(W\cdot MF+f)\gg qbits. \quad (6)$$

위와 같이 유도되어진 공식을 살펴보면 정수 변환 모듈과 양자화 모듈이 서로 연계되어 하나의 곱셈기만으로도 정수 변환과 양자화를 수행할 수 있음을 알 수 있다.

4. 역양자화의 구조

"역양자화" 또는 "리-스케일링(Rescaling)"이라 불리며 양자화와는 반대로 유한한 개수로 근사화된 출력신호를 선형 레벨로 바꾸는 것이 역양자화이다. 이미 양자화 과정에서의 비선형 연산에 의하여 원신호의 완전한 복원이 불가능하다. 아래 식(7)은 역양자화의 기본 공식이다.

$$Y'=Z\cdot Qstep. \quad (7)$$

역양자화에서는 위 식과 같은 기본공식을 이용하여 역양자화 모듈에서는 H.264에서 제공되어지는 52 레벨의 QP를 가지고 VF (Pre-scaling Factor)를 만들어 놓고 계산에 사용한다. 위의 역양자화 기본 공식은 정수 역변환에서 사용되어지기 위한 W' 값으로 유도되어지기 위해서 아래 식 (8),(9)와 같이 유도 된다. 여기서 28은 라운딩 에러를 피하기 위하여 사용된 상수 스케일링 요소이다.

$$W'=Z\cdot Qstep\cdot PF\cdot 2^{8},\quad W'=C_i^T W C_i. \quad (8)$$

$$W'=Z\cdot VF\cdot 2^{floor(QP/6)}$$

$$\text{where}\quad VF=Qstep\cdot PF\cdot 2^{8}. \quad (9)$$

앞에서와 마찬가지로 최종적인 역양자화 공식과 같이 정수 역변환 모듈과 역양자화 모듈이 서로 연계되어 하나의 곱셈기만으로도 정수 역변환과 역양자화를 수행할 수 있음을 알 수 있다.

5. 정수 역변환의 구조

정수 역변환의 구조는 일반적인 DCT의 수행 결과와 거의 동일한 결과를 나타내는 정수 역변환 공식을 사용하여 구성된다. 정수 변환과 마찬가지로 모든 동작은 정수 연산으로 수행되어지며 부호화하고 복호화하는 과정에서 데이터의 오차가 없다. 변환기 코어(core)중에서 하드웨어적으로 큰 면적을 차지하고 시간 지연이 많은 곱셈기와 같은 큰 모듈 대신에 오직 덧셈기와 같은 상대적으로 적은 면적을 가진 모듈로 구성되어질 수 있는 장점들을 가지고 있다. H.264에서 쓰는 정수 역변환의 공식은 아래 식(10)과 같다.

$$X'=C_i^T(Y\otimes E_i)C_i=C_i^T W' C_i$$

$$=\begin{bmatrix}1 & 1 & 1 & 1/2\\ 1 & 1/2 & -1 & -1\\ 1 & -1/2 & -1 & 1\\ 1 & -1 & 1 & -1/2\end{bmatrix}\left(\begin{bmatrix}Y\end{bmatrix}\otimes\begin{bmatrix}a^2 & ab & a^2 & ab\\ ab & b^2 & ab & b^2\\ a^2 & ab & a^2 & ab\\ ab & b^2 & ab & b^2\end{bmatrix}\right)$$

$$\begin{bmatrix}1 & 1 & 1 & 1\\ 1 & 1/2 & -1/2 & -1\\ 1 & -1 & -1 & 1\\ 1/2 & -1 & 1 & -1/2\end{bmatrix}. \quad (10)$$

위 식과 같이 유도된 정수 역변환 공식은 역양자화에서 계산되어진 W' 을 이용하여 주파수 영역의 데이터인 Y 값을 공간 영역의 데이터로 변환할 수 있다. 이때 사용되어지는 연산은 정수 변환과 같이 16 비트 정수 연산의 덧셈기, 뺄셈기, 쉬프트만으로 결과를 도출할 수 있게 된다.

III. 코어 부분 설계

본 장에서는 H.264의 정수 변환, 양자화, 역양자화, 정수 역변환 알고리즘의 기본구조를 이용하여 하드웨어 구조로 변환 및 적용에 관해 기술한다. 먼저 각각의 하드웨어 구조를 살펴보기 전에 구현된 전체 모듈에서의 영상 데이터의 흐름을 알아본다.

PCI 인터페이스의 입력 레지스터로 입력된 영상 데이터는 정수 변환 모듈을 거쳐서 공간영역 데이터에서 주파수 영역 데이터로 변환이 된다. 이렇게 변환된 데이터는 미리 QP (Quantizer Parameter)를 입력으로 하여 모듈러 값을 생성하는 Simple Divider 모듈과 양자화 룩업테이블 (Look Up Table)을 통해 만들어진 MF (Multiplication Factor), PF (Position Factor)들을 요소로 하여 양자화를 거치게 된다[9].

이렇게 변환된 영상 데이터들은 H.264의 엔트로피 코딩을 위하여 AC Level과 AC Run 값을 생성하는데 쓰이게 된다. 그와 동시에 양자화를 거친 데이터들은 다음 영상의 기준 데이터로 쓰이기 위하여 역양자화를 거치게 되는데 양자화 계산과 마찬가지로 Simple Divider 모듈과 역양자화 룩업테이블을 통하여 만들어진 VF (Rescaling Factor)와 PF들을 요소로 역양자화를 거치게 되며 마지막으로 정수 역변환을 거쳐 다시 주파수 영역 데이터에서 공간 영역 데이터로 변환이 되어 PCI 출력 레지스터에 저장이 된다. 이러한 구현된 모듈 내에서의 영상 데이터의 흐

름을 아래 그림 4와 같이 표현하였다.

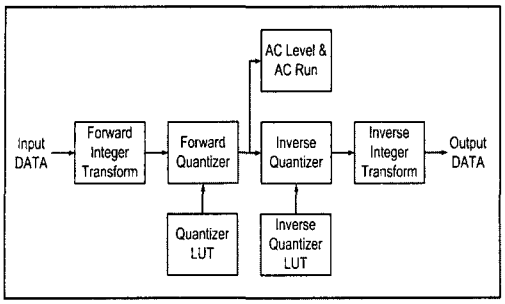

그림 4. 영상 데이터의 흐름도

1. 전체 모듈의 구조

본 논문을 위해 구현된 전체 모듈은 먼저 시뮬레이터를 이용하여 결과를 확인하였고 FPGA 검증 보드와 PCI 인터페이스를 이용하여 정수 변환 모듈, 양자화 모듈, 역 양자화 모듈, 역 정수 변환 모듈 등을 구현하고 결과 값을 비교 확인하였다. 아래에 그림 5는 구현된 전체 모듈의 블럭도이다.

전체 모듈을 구성하기 위해 먼저 PCI 인터페이스를 통하여 PCI 영상데이터 입력 레지스터에 4화소x4화소의 16비트 데이터 값들이 입력된다. 그 후에 입력된 영상 데이터들은 멀티플렉서(Multiplexer)를 통하여 정수 변환이 수행되어진다. 영상데이터의 변환 순서는 수평 정수 변환을 하고난 후 그 결과를 레지스터에 저장하고 그 후에 멀티플렉서 입력으로 수직 정수 변환을 할 수 있도록 데이터 값의 위치를 변경한 후 다시 DCT-1d 모듈을 통하여 수직 정수 변환을 한다. 이러한 두 번의 변환을 거쳐 복잡한 행렬 계산 없이 공간 영역의 영상 데이터들은 주파수 영역의 영상 데이터로 변환이 된다.

정수 변환에 의하여 생성된 결과는 양자화 모듈의 입력으로 들어가게 되는데 양자화 모듈의 입력으로 들어가기 위해서는 16비트 곱셈기를 이용한 계산이 필요하므로 64비트로 출력된 정수 변환 값들을 멀티플렉서를 통하여 16비트로 나누어 입력하게 된다. 그리고 양자화를 거치기 전에 QP를 입력으로 받는 Simple Divider 모듈의 모듈러 연산을 통하여 미리 양자화 룩업테이블과 역양자화 룩업테이블의 값을 생성하게 된다. 이렇게 생성된 값은 양자화 모듈에 입력되어 정수 변환 되어진 영상 데이터들과의 계산을 통하여 양자화하게 된다.

양자화되어진 데이터들은 VLC에서 이용되어 지기 위해 AC Level과 AC Run값으로 변환되어 레지스터에 저장되게 된다. 한편 양자화되어진 데이터들은 다음 영상의 기준 영상으로 만들어지기 위하여 멀티플렉서를 통하여 정수 역변환을 수행한다. 정수변환과 마찬가지로 먼저 수평 정수 역변환을 하고난 후 그 결과를 레지스터에 저장하게 된다. 그 후에 멀티플렉서입력으로 수직정수 역변환을 할 수 있도록 데이터 값의 위치를 변경한 후 다시 정수 IDCT-1d 모듈을 통하여 수직 정수 역변환을 한다. 이렇게 생성된 결과를 PCI 출력 레지스터에 저장함으로서 영상 데이터의 정수 변환, 양자화, 역양자화, 정수 역변환을 마치게 된다.

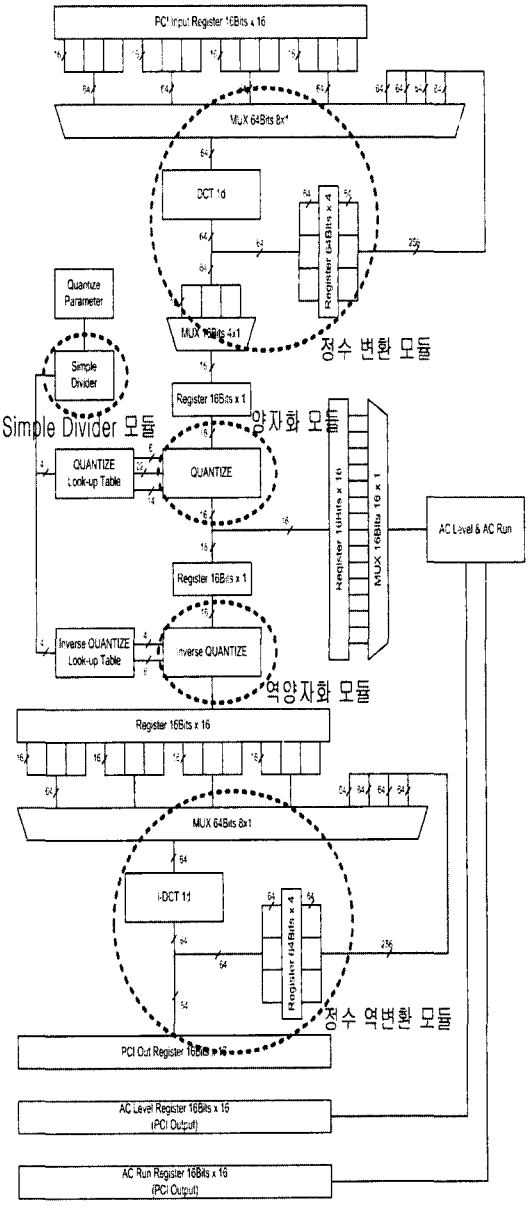

그림 5. 전체 모듈 블록도

2. 정수 변환 모듈의 구조

정수 변환을 하기 위해서는 4화소x4화소의 행렬계산식을 이용하여야 한다. 이와 같은 행렬 계산은 하드웨어적으로 구현하기 어렵고 곱셈기와 같은 면적이 크고 시간 지연이 많은 모듈이 사용되어야 한다. 그러나 행렬계산을 수평 성분과 수직 성분으로 나누어서 아래 그림 4와 같은 구조의 DCT-1d를 수평, 수직으로 두 번의 계산을 수행하면 덧셈기 8개만을 이용하여 행렬 계산식과 같은 결과 값을 얻을 수 있다. 그리고 DCT-1d를 구현하기 위하여 덧셈기, 뺄셈기, 쉬프트기를 이용하여 행렬 계산식으로 이루어진 정수 변환 모듈을 구현하였다. 아래에 그림 6은 정수 변환을 하기 위한 DCT-1d의 구조를 나타낸다.

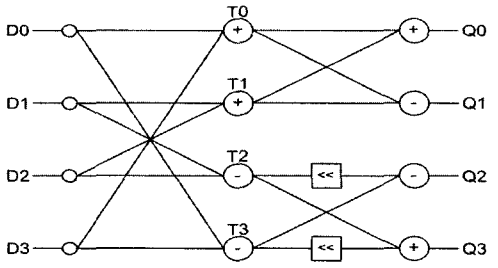

그림 6. DCT-1d의 구조

그러나 DCT-1d 모듈을 설계하여 정수 변환을 하려면 수평 정수 변환과 수직 정수 변환으로 나누어서 두 번 거쳐야 하고 이와 같은 과정을 구현하기 위해서는 수평 성분과 수직 성분을 선택할 수 있도록 멀티플렉서를 사용하여야 하며 계산 되어진 값들을 임시로 저장할 수 있도록 레지스터가 필요하게 된다. 이때 사용되어지는 멀티플렉서는 64비트 8입력 멀티플렉서가 사용되며 레지스터는 64비트 4개가 사용되어진다. 이런 방식으로 정수 변환 모듈을 구현하게 되면 동작하기 위한 클록수가 증가하게 되지만 클록 대비 게이트 사이즈가 줄어들게 된다. 아래에 그림 7은 정수 변환 또는 정수 역변환을 하기 위한 데이터 흐름도를 나타내었다.

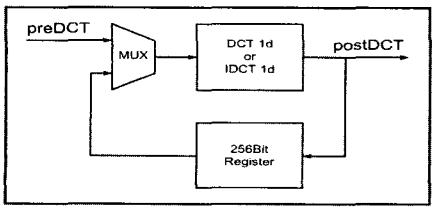

그림 7. DCT 또는 IDCT 모듈의 구조

3. 양자화 모듈의 구조

양자화를 하기 위해서는 양자화 모듈과 입력된 QP의 값을 이용하여 미리 계산되어진 양자화 룩업테이블이 사용되어진다. 양자화 모듈은 정의된 알고리즘에 의해 계산되어지기 위한 모듈이고 이 계산에는 PF, MF, qbits, f 등과 같은 요소들이 필요로 하며 이 중에서도 제일 계산 량이 많은 qbits 와 f 값들을 미리 계산하여 저장시켜 놓은 양자화 룩업테이블이 필요하다. 이 중에 양자화 룩업테이블은 필요로 하는 값을 선택하기 위하여 QP의 몫과 나머지 값을 계산할 수 있는 모듈러 연산이 필요로 하게 된다. 여기서 모듈러 연산은 복잡한 형태이기 때문에 모듈러 연산을 대체할 수 있는 Simple Divider 모듈을 구현하여 양자화 룩업테이블의 값을 선택하게 된다. 이렇게 만들어진 양자화 룩업테이블 값과 정수 변환된 데이터를 입력으로 하는 양자화 모듈을 구현하였다. 아래에 그림 8은 곱셈기, 덧셈기, 쉬프트기로 이루어진 양자화의 구조를 나타낸다.

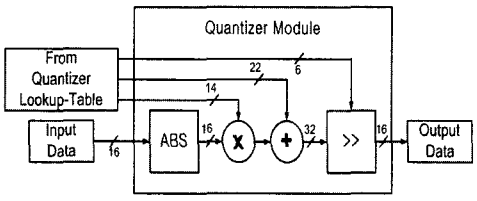

그림 8. 양자화 모듈의 구조

4. 정수 역변환 모듈의 구조

정수 역변환을 하기 위해서는 정수 변환과 마찬가지로 4화소x4화소의 행렬 계산식을 이용하여야한다. 정수 변환 모듈에서 쓰인 방식과 같이 IDCT-1d를 수평 성분과 수직 성분으로 나누어서 멀티플렉서를 통하여 두 번의 계산을 수행하면 덧셈기 8개만을 이용하여 행렬 계산식과 같은 결과 값을 얻을 수 있다. 그리고 IDCT-1d를 구현하기 위하여 덧셈기, 뺄셈기, 쉬프트기를 이용하여 행렬 계산식으로 이루어진 정수 역변환 모듈을 구현하였다. 아래에 그림 9는 정수 역변환을 하기 위한 IDCT-1d의 구조를 나타낸다.

정수 역변환을 수행하는 과정도 정수 변환과 마찬가지로 IDCT-1d 모듈을 설계하여 수평 정수 변환과 수직 정수 변환으로 나누어 두 번 거치게 된다. 이와 같은 과정을 구현하기 위해서는 수평 성분과 수직 성분을 선택할 수 있도록 멀티플렉서를 사용하여야 하며 계산 되어진 값들을 임시로 저장할 수 있는 레지스터가 필요하게 된다. 이때 사용되어지는 멀티플렉

서는 64비트 8입력의 멀티플렉서이며, 레지스터는 4개의 64비트 레지스터가 필요하게 된다. 이러한 방식으로 구현을 하게 되면 데이터를 처리하기 위한 클록수가 증가하지만 클록 대비 게이트 사이즈가 줄어들게 된다.

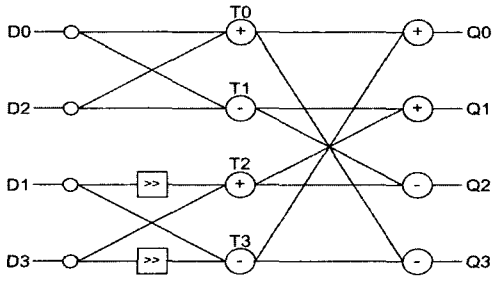

그림 9. IDCT-1d의 구조

5. 역양자화 모듈의 구조

역양자화를 하기 위해서는 양자화 모듈과 같이 역양자화 모듈과 미리 계산되어진 역양자화 룩업테이블이 사용되어진다. 역양자화 모듈은 정의된 알고리즘에 의해 계산되어지기 위한 모듈이고 이 계산에는 QP, PF, VF 등과 같은 요소들이 필요로 하며 이 중에서도 제일 계산 량이 많은 qbits 와 f 값들을 미리 계산하여 저장시켜 놓은 역양자화 룩업테이블이 필요하다. 이때 역양자화 룩업테이블과 양자화 룩업테이블의 값은 같기 때문에 공유하여 모듈을 사용하게 된다. 이렇게 만들어진 역양자화 룩업테이블 값과 부호비트, 양자화 데이터를 입력으로 하는 역양자화 모듈을 구현하였다. 아래에 그림 10은 곱셈기, 부호기, 쉬프트기로 이루어진 역양자화의 구조를 나타낸다.

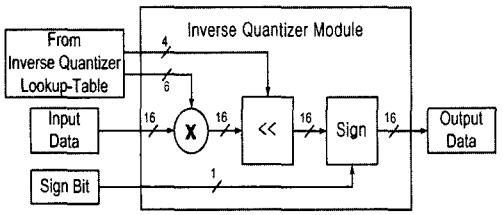

그림 10. 역양자화 모듈의 구조

6. Simple Divider의 구조

양자화와 역양자화의 구조를 앞장에서 유도한 알고리즘과 같이 간단하게 유도하기 위해서는 MF, VF, qbits, f 값들이 필요하게 되는데 이때 MF, VF와 같은 룩업테이블 값들과 qbits, f 값들을 도출하기 위해서는 QP의 몫과 나머지 값이 필요하며 이 값들을 생성하기 위해서는 룩업테이블을 이용하거나 모듈러 연산을 이용하여 값을 생성하는 모듈을 구현하여야한다. 그러나 룩업테이블을 이용하거나 일반적으로 복잡한 구조로 이루어진 모듈러 연산을 이용하여 구현하는 경우에는 많은 하드웨어 리소스가 필요하다. 양자화와 역양자화 룩업테이블에서 필요로 하는 값은 입력된 QP의 3비트의 몫과 4비트의 나머지 값이므로 모듈러 연산 없이 필요한 값을 생성하는 Simple Divider 모듈을 구현하여 하드웨어 리소스를 줄였다. 아래 그림 11은 비교기, 덧셈기로 이루어진 Simple Divider의 구조를 나타낸다.

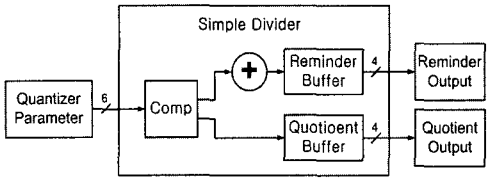

그림 11. Simple Divider 모듈의 구조

Ⅳ. 설계 검증 및 성능 분석

본 장에서는 제안된 하드웨어 구조로 구현한 전체 모듈들에 대하여 동작을 기술하고 설계 검증 및 성능 분석을 한다. 먼저 구현된 전체 모듈의 동작 클록수를 먼저 살펴보면, 정수 변환 모듈은 정수 역변환 모듈과 마찬가지로 8개의 클록을 사용하여 데이터를 처리할 수 있다. 그러나 20개의 클록을 사용하여 정수 변화 모듈을 구성하여 양자화 모듈에 들어가는 데이터의 타이밍과 한 클록 차이로 일치시키게 되면 전체적으로 대략 64개의 클록을 이용하여 구현할 수 있는 전체 모듈을 총 30개의 클록만으로 구현할 수 있다. 아래 그림 12는 하나의 4화소x4화소의 기본 블록이 정수 변환 및 양자화되기 위해 구현된 전체 모듈을 거치는 클록의 수를 나타내었다.

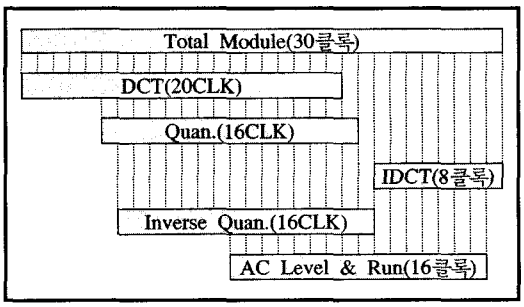

그림 12. 기본 블록의 구현 클록의 수

구현된 모듈들의 설계 검증을 위해 두 가지 방법을 사용하여 결과 값을 분석하였다. 첫번째로는 Altera 사의 APEX20KE600EBC 디바이스와 PLX-Tech사의 PLX9050 디바이스를 이용하여 PCI 인터페이스를 구성하고 구현한 모듈을 테스트 할 수 있는 프로그램을 작성하여 FPGA 환경에서 검증하였다. 이렇게 FPGA로 구현 시 결과 값에 따른 최대 동작 주파수는 28MHz이며 대략 21%의 로직 리소스가 사용되어진다.

두번째로는 Samsung STD130 0.18um CMOS Cell Library와 Synopsys사의 Design Analyzer를 이용하여 합성 결과 값을 분석하였다. 구현된 전체 모듈을 ASIC Library를 이용하여 합성한 결과, 측정된 최장 지연 경로(Critical Path)는 9.82ns이며 최대 동작 주파수는 약 100MHz가 된다.

QCIF (176화소 x 144화소) 크기의 영상을 기준으로 성능을 분석하여 보면 하나의 프레임은 99개의 마크로블록으로 이루어져 있고 하나의 마크로블록은 26개의 블록으로 이루어져 있다. 그리고 하나의 블록이 실행되는 속도는 위에서 나타낸 바와 같이 100MHz의 30클록이다. 이러한 성능에 의하여 1초당 정수 변환, 양자화, 역양자화, 정수 역변환을 모두 수행하는 초당 최대 프레임 수는 대략 1295 프레임이 된다. 이것을 식으로 나타내어 보면 아래 식 (11)과 같다.

$$\frac{1}{99\times 26\times 30\times 0.01\times 10^{-6}} \simeq 1295.0 \text{ [F/Sec]}. \quad (11)$$

또한 CIF (352화소x288화소) 크기의 영상을 기준으로 성능을 분석하여 보면 대략 초당 320 프레임의 변환 성능을 보인다. H.264 부호화기의 전체 모듈을 구현하기 위해서는 움직임 측정 모듈 등과 같은 복잡성이 큰 모듈에 더 의존적이겠지만 위와 같은 성능을 바탕으로 정수 변환 모듈 및 양자화 모듈 등을 구현함에 있어서 성능의 저하 없이 하드웨어 리소스를 줄일 수 있게 된다.

전체 모듈을 구현 후 PCI 인터페이스 모듈을 제외하여 측정한 게이트 사이즈는 약 30K게이트이다. 전체 모듈 중에서 면적을 크게 차지하는 부분은 16비트 곱셈기와 덧셈기로 이루어진 양자화 모듈이며 역양자화 모듈에서 필요한 VF와 비교하여 상대적으로 큰 MF를 필요하기 때문에 역양자화 모듈의 약 두 배의 게이트수가 필요함을 알 수 있다. 아래 표 1은 Samsung STD130 0.18um 공정을 이용하여 구현된 모듈의 성능을 나타내었다.

표 1. 구현된 모듈 성능 (STD130 0.18um공정)

모듈	Cycle per Frame[클록]	Gate	Critical Path[ns]
정수 변환	20	1311.7	3.7
양자화	16	3989.2	9.36
정수 역변환	8	1001.3	3.28
역양자화	16	1610.9	5.98
AC Level & Run	16	410.3	0.32
Simple Divider	-	105.0	1.32

성능 분석에 앞서 본 논문의 제작 시점에 H.264의 하드웨어 구현에 대한 비교 및 분석 자료가 없기에 JVT의 H.264 표준 알고리즘을 범용 프로세서에서 구현한 소프트웨어인 JM 6.0의 구조를 비교 대상으로 선정하였다. 그리고 JM 6.0의 정수 변환, 정수 역변환, 양자화, 역양자화 구조들과 본 논문에서 하드웨어로 구현된 결과를 아래에 표 2와 같이 성능 비교표로 나타내었다[10].

표 2. JM 6.0의 구조와 제안된 모듈의 성능 비교

	JM6.0 S/W 구조	본 논문의 H/W 구조	비고
정수 변환	덧셈 16회	덧셈기 8개 (덧셈연산 16회)	MUX 사용
양자화	곱셈 16회	곱셈기 1개 (덧셈연산 16회)	-
역양자화	곱셈 16회	곱셈기 1개 (덧셈연산 16회)	-
정수 역변환	덧셈 16회	덧셈기 8개 (덧셈연산 16회)	MUX 사용
전체 모듈에서 필요한 나눗셈 횟수	2	Simple Divider 사용	-

표에서 알 수 있듯이 본 논문에서 제안된 모듈에서는 동작에 필요한 연산 횟수는 똑같지만 멀티플렉서를 사용하여 기본적으로 정수 변환 모듈과 정수 역변

환 모듈의 구조를 수평 성분과 수직 성분으로 나누어 두 번 반복하여 동작하도록 구현하고 덧셈기의 개수를 반으로 줄여 하드웨어 리소스를 적게 쓸 수 있도록 구성하였다.

양자화 구조와 역양자화 구조에서는 기본적으로 제안된 구조와 JM 6.0의 구조를 비교하여 보면 동작을 위한 곱셈의 연산 횟수가 같다. 그러나 정수 변환, 정수 역변환과는 달리 멀티플렉서를 사용하지 않고 정수 변환에서 16개로 분리된 행렬 요소를 그대로 양자화와 역양자화에서 사용하여 하드웨어 리소스를 줄였다. 또한 양자화, 역양자화 계산을 하기 위해 나눗셈 또는 모듈러 연산이 필요한 JM 6.0의 구조보다 나눗셈 또는 모듈러 연산 없이 간단한 덧셈기와 비교기로 양자화와 역양자화 계산에 필요한 입력 값을 생성하는 Simple Divider를 사용하는 제안된 구조가 전체적인 하드웨어 리소스를 줄이는 측면에서 우수한 결과를 나타냄을 알수 있다.

V. 결 론

본 논문에서는 H.264의 핵심 모듈 중에서도 정수 변환 모듈, 양자화 모듈, 역양자화 모듈, 정수 역변환에 대한 알고리즘의 구조를 기술하고 하드웨어로 제안된 모듈을 구현하였다. 각각의 정수 변환 모듈 및 양자화 모듈들의 병렬 처리 사용 시에 중복되어지는 구조들을 저전력 설계를 위해 제안된 하드웨어 구조를 이용하여 구현하여 게이트 사이즈를 줄였다. 그리고 양자화 구조와 역양자화 구조에서 복잡하고 하드웨어 리소스를 많이 차지하는 모듈들을 모듈러 연산과 같은 기능을 수행하는 Simple Divider 모듈, 양자화 룩업테이블, 역양자화 룩업테이블을 이용하여 적은 하드웨어 리소스를 차지하도록 설계하였다.

이러한 제안된 하드웨어 구조로 구현하기 위해서 정수 변환을 구현 시 동작을 위해 필요한 클록수가 12클록이 증가하는 것과 같이 일부 모듈에서는 클록수가 증가한다. 그러나 각 모듈에 입력되어지는 데이터의 타이밍을 조절하여 전체적으로 64개의 클록을 이용하여 구현되어야하는 전체 모듈들을 총 30개의 클록만으로 동작할 수 있도록 구현하였고 곱셈기, 덧셈기와 같은 큰 면적을 차지하는 모듈들의 개수를 줄여 전체적으로 하드웨어 사이즈의 최소화와 전력소모의 최소화를 꾀하였다. 이렇게 제안된 전체 모듈은 게이트 사이즈와 동작 클럭을 동시에 줄일 수 있기 때문에 H.264의 정수 변환, 양자화, 역양자화, 정수 역변환을 하드웨어로 설계하기에 우수한 구조라고 할 것이다. 현재는 전체 H.264의 인코더 부분 중에서 움직임 측정, 움직임 보정, 엔트로피코딩의 각 모듈들을 설계 중에 있으며 향후 전체 구조를 VLSI로 구현할 예정이다.

참 고 문 헌

[1] ISO/IEC 11172-1/2/3: "Information technology coding of moving picture and associated audio for digital storage media at up to about 1.5 Mb/s," International Standard, 1993.

[2] ISO/IEC 13818-1/2/3: "Information technology generic coding of moving picture and associated audio", Draft International Standard, Mar. 1994.

[3] ISO/IEC 13522-1/2/3 : "Information technology coded representation of multimedia information objects," Draft International Standard, Nov. 1993.

[4] ITU-T Rec. H.261 : "Video codec for audio visual services at px64 kb/s", Mar. 1993.

[5] ITU-T Rec. H.263 : "Video coding for low bitrate communication," May 1996.

[6] ITU-T Rec. H.264/ISO/IEC 11496-10, "Advanced Video Coding," Final Committee Draft, Document JVT-F100, Dec. 2002.

[7] A. Hallapuro and M. Karczewicz, "Low complexity transform and quantization - Part 1 : Basic Implementation," JVT document JVT-B038, Feb. 2001.

[8] A. Puri and T. Chen, *Multimedia Systems, Standards, and Networks*, Marcel Dekker, Inc., 2000.

[9] 김환용, 김형균, 고영욱, "ROM Table 구조를 갖는 DCT 계수 인코더의 설계," 공업기술개발연구지, Vol. 17, pp. 199-207, 1997.

[10] JVT H.264 Reference Software version 6.0, ftp://ftp.imtc-files.org/jvt-experts/, 2002.

임 영 훈(Young-Hun Lim) 준회원

2001년 1월 : 대진대학교 전자공학과 졸업
2001년 1월~2003년 2월 : 동서전자 기업부설연구소 연구원
2003년 2월~현재 : 광운대학교 전자공학과 석사과정

<주관심분야> 영상처리, 제어 시스템 설계, ASIC 설계

정 용 진(Yong-Jin Jeong) 정회원

1983 2월 : 서울대학교 제어계측공학과 졸업
1983년 3월~1989년 8월 : 한국전자통신연구원
1991년 5월 : 미국 UMASS 전자전산공학과 석사
1995년 2월 : 미국 UMASS 전자전산공학과 박사
1995년 4월~1999년 2월 : 삼성전자 반도체 수석 연구원
1999년 3월~현재 : 광운대학교 전자공학부 부교수

<주관심분야> 컴퓨터 연산 알고리즘, ASIC 설계, 무선 통신, 정보보호