하드웨어 기반의 H.264/JVT 변환 및 양자화 구현

임영훈, 정용진
광운대학교 전자통신공학과
e-mail : *limyh14@explore.kw.ac.kr, yjjeong@daisy.kw.ac.kr*

Hardware Implementation of Transform and Quantization for H.264/JVT

Young-hun Lim, Yong-Jin Jeong
Depart. of Electronics & Communications Engineering
Kwangwoon University

Abstract

In this paper, we propose a new hardware architecture for integer transform, quantizer operation of a new video coding standard H.264/JVT. We describe the algorithm to 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. 서론

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. 지금까지의 동영상 압축을 위한 표준으로는 MPEG-1, MPEG-2, MPEG-4[1], H.261, H.263[2] 등이 있으나, 최근 대두되고 있는 무선 디바이스와 같은 저비트율을 지향하는 개체에서는 효율적으로 사용하기에 부적합한 방식이므로 더욱 효율적인 압축방식을 요구하고 있다.

본 논문에서는 H.264[3]의 동화상 부호화 알고리즘의 핵심 부분인 정수 변환 (Integer Transform) 모듈, 양자화 (Quantization) 모듈을 구성하는 하드웨어 구조를 제안하고 많은 연산량이 소요되는 블록의 알고리즘을 최적화시키고 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 최적화하는 하드웨어 구현을 기술한다. 그리고 4장에서는 제안된 하드웨어 구조를 각각의 방법의 의하여 설계 검증을 한 후 그 결과를 분석한다. 마지막으로 5장에서 본 논문에 대한 결론을 맺는다.

Ⅱ. 알고리즘의 구조

본 장에서는 H.264의 정수 변환, 양자화의 구조에 대하여 기술하고 하드웨어로 설계하기 적합한 구조로 변환하는 알고리즘을 기술한다.

본 논문은 광운대학교 정보통신연구원 및 IDEC의 툴 지원으로 이루어졌습니다.

가. 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[4]. 그러나 H.264의 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 변환식은 아래와 같이 식(1)과 같고 H.264에서 쓰이는 정수 변환식은 아래와 같이 식(2)과 같다[5].

$$Y = AXA^T = \begin{pmatrix} a & a & a & a \\ b & c & -c & -b \\ a & -a & -a & a \\ c & -b & b & -c \end{pmatrix} X \begin{pmatrix} a & b & a & c \\ a & c & -a & -b \\ a & -c & -c & b \\ a & -b & a & -c \end{pmatrix}$$

$$\text{where } a = \frac{1}{2},\ b = \sqrt{\frac{1}{2}}\cos\left(\frac{\pi}{8}\right),\ c = \sqrt{\frac{1}{2}}\cos\left(\frac{3\pi}{8}\right). \quad (1)$$

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

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

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

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

식 (2)에서 ⊗(Scaling Operation)는 계산된 CTC^T의 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. 정수 변환 공식 중에 CTC^T는 오직 정수 연산의 덧셈기, 뺄셈기, 쉬프트만으로 결과를 도출할 수 있다. 그러나 정수 변환 모듈 이후의 양자화 모듈에서는 ⊗의 계산을 위해 하나의 곱셈기를 요구한다.

나. 양자화의 구조

입력 신호를 유한한 개수로 근사화하는 것을 양자화라 하는데 이것은 비선형 연산이며 원신호의 완전한 복원이 불가능하다. 아래에 그림 1에서 보이는 것과 같이 선형의 입력 레벨을 근사화된 출력 레벨로 바꾸는 방식으로 양자화를 진행하게 되는데 실질적으로 압축의 양을 결정하게 된다.

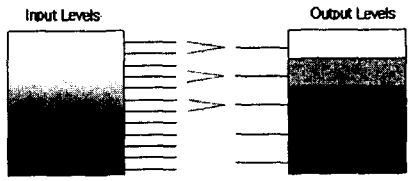

그림 1. 양자화의 기본 개념

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

이러한 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.

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

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

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

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

최종적인 공식으로 아래 식(6)과 같이 유도가 된다.

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

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

Ⅲ. 코어 부분 설계

본 장에서는 H.264의 정수 변환, 양자화 알고리즘의 기본구조를 이용하여 하드웨어 구조로 변환 및 적용에 관해 기술한다.

가. 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 2는 정수 변환을 하기 위한 DCT-1d의 구조를 나타낸다.

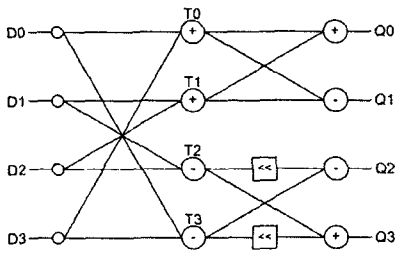

그림 2. DCT-1d의 구조

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.

나. 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. 아래 그림 3는 곱셈기, 덧셈기, 쉬프트기로 이루어진 양자화의 구조를 나타낸다.

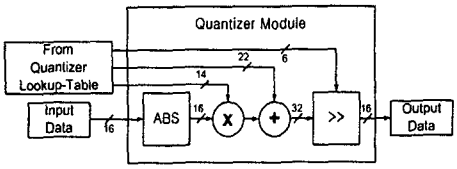

그림 3. 양자화 모듈의 구조

바. 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 구현하여야 한다. 그러나 룩업테이블을 이용하거나 복잡한 구조로 이루어진 모듈러 연산을 이용하여 구현하는 경우에는 많은 하드웨어 리소스가 필요하다. 양자화 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 4는 비교기, 덧셈기로 이루어진 Simple Divider의 구조를 나타낸다.

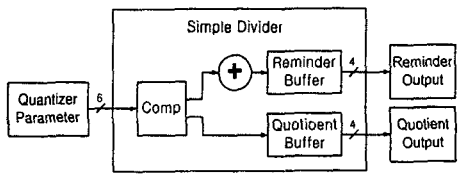

그림 4. Simple Divider 모듈의 구조

Ⅳ. 설계 검증 및 성능 분석

제안된 구조를 이용하여 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 있

다. 아래 그림 5은 하나의 4화소x4화소의 기본 블록이 정수 변환 및 양자화되기 위해 구현된 전체 모듈을 거치는 클록의 수를 나타내었다.

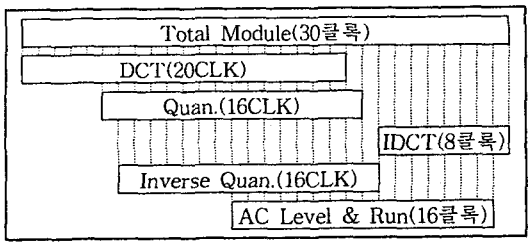

그림 5. 기본 블록의 구현 클록의 수

구현된 모듈들의 설계 검증을 위해 두 가지 방법을 사용하여 결과 값을 분석하였다. 첫번째로는 Altera사의 APEX20KE600EBC 디바이스와 PCI 인터페이스를 이용하여 구현한 모듈을 테스트 할 수 있는 프로그램을 작성하고 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 Library와 Synopsys사의 Design Analyzer를 이용하여 합성 결과를 분석하였다. 구현된 전체 모듈을 ASIC Library를 이용하여 합성한 결과, 측정된 최장 지연 경로는 9.82ns이며 최대 동작 주파수는 약 100MHz가 된다.

QCIF (176화소x144화소) 크기의 영상을 기준으로 성능을 분석하여 보면 1초당 정수 변환, 양자화, 역양자화, 정수 역변환을 모두 수행하는 초당 최대 프레임 수는 대략 1300 프레임이 된다. H.264 부호화기의 전체 모듈을 구현하기 위해서는 움직임 측정 모듈 등과 같은 복잡성이 큰 모듈에 더 의존적이겠지만 위와 같은 성능을 바탕으로 정수 변환 및 양자화 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. 표 1은 Samsung STD130 0.18um 공정을 이용하여 구현된 모듈의 성능을 나타내었다.

표 1. 구현된 모듈 성능 (STD130 0.18um공정)

모듈	Cycle per Block[clock]	Gates	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

V. 결론

본 논문에서는 H.264의 핵심 모듈 중에서도 정수 변환 모듈, 양자화 모듈에 대한 알고리즘의 구조를 기술하고 하드웨어로 제안된 모듈을 구현하였다. 저전력 설계를 위해 중복되어지는 계산들을 제안된 하드웨어 구조를 이용하여 구현한 결과 클럭 대비 게이트 사이즈를 줄였다. 그리고 양자화 구조에서 복잡하고 하드웨어 리소스를 많이 차지하는 모듈들을 모듈러 연산과 같은 기능을 수행하는 Simple Divider 모듈, 양자화 룩업테이블을 이용하여 적은 하드웨어 리소스를 차지하도록 설계하였다.

이러한 제안된 하드웨어 구조로 구현하기 위해서 정수 변환을 구현 시 동작을 위해 필요한 클록수가 12 클록이 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, 양자화를 하드웨어로 설계하기에 우수한 구조라고 할 것이다.

참고문헌

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

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

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

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

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