H.264/AVC용 Intra coding의 변환 및 양자화 모듈의 VHDL 구현

최덕영, 손승일

한신대학교 정보통신학과

VHDL Implementation of Transform and Quantization Intra Coding for H.264/AVC

Dug Young Choi, Seung Il Sonh

Dept. of Information and Communication HanShin University

E-mail : coolduck2@hs.ac.kr, saisonh@hs.ac.kr

요 약

디지털 비디오 압축기술은 멀티미디어 응용분야의 핵심으로 현재 빠르게 보급되어 최근에는 디지털 비디오 압축 관련 국제 표준안 중 MPEG-4와 H.264가 발표되었다. 유연성이 좋은 MPEG-4와 달리 H.264는 비디오 프레임의 효율적인 압축과 신뢰성을 강조 한다. 특히 H.264의 압축 기술은 카메라폰이나 DMB등의 작은 크기의 영상에서 고품질의 영상을 보다 효율적으로 제공 한다.

이에 본 논문은 현존하는 다른 비디오 코딩 표준과 비교할 때 코딩 효율이 기준의 두 배인 새로운 비디오 코딩 표준 H.264/AVC에서 사용하는, 변환 및 양자화를 연구하고 이를 기존의 정지영상 표준안인 JPEG나 JPEG 2000과 비교 분석하여 H.264/AVC의 공간적 압축인 인트라 코딩이 더 좋은 효과를 나타낸다는 것을 검증한 후 이를 토대로 하드웨어 설계언어인 VHDL언어를 이용하여 설계하고 FPGA칩인 XCV1000E에 다운로드 하여 칩 레벨의 시뮬레이션을 수행하여 설계된 변환 및 양자화 모듈을 검증하였다. 설계된 변환 및 양자화 모듈은 DMB 및 핸드폰 카메라와 같이 작은 정지 영상 압축에 응용이 가능하다.

1. 서론

현대에 있어서 영상정보는 아주 큰 비중을 차지하고 있다. 따라서 이러한 영상정보를 얼마나 빨리 그리고 많이 압축 시킬 수 있느냐가 핵심적인 관건이다. 이에 많은 분야에서 영상을 압축 시키는 방법들이 연구되어 지고 있다. 그중에서 정지 영상의 압축에는 많은 표준안들이 존재한다. JPEG나 JPEG 2000, MPEG, H.263등이 정지 영상 압축을 사용하는데 최근 발표된 H.264/AVC의 인트라 코딩은 기존의 압축 방식보다 월등히 좋은 효과를 보여주고 있다[1].

H.264/AVC의 인트라 코딩은 한 화면을 블록으로 나누어 첫 블록을 제외한 나머지 블록은 바로 전 블록이나 이웃하는 윗부분에 있는 블록을 사용하여 9개의 인트라 예측을 수행하고 인트라 예측을 수행 후에는 모드 값과 차이 값이 나오는데 여기서 차이 값을 변환 및 양자화 후에 압축을 수행한다.

2. 변환

인트라 예측 후 나온 차이 값은 JPEG나 MPEG에서는 DCT를 사용한다. 식 1은 4*4 행렬에서 사용하는 DCT 계수 값을 나타내는 행렬식이다.

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

$$a = \frac{1}{2}, b = \sqrt{\frac{1}{2}} cos(\frac{\pi}{8}), c = \sqrt{\frac{1}{2}} cos(\frac{3\pi}{8})$$

[식1]

하지만 H.264에서는 식1과 같은 DCT 변환을 사용하지 않고 식2와 같은 2D 변환을 사용한다. 식2는 DCT의 행렬 곱셈을 등가의 형태로 인수분해 한 것이며, CXC^T는 2D 변환의 핵심이고 E는 스케일링 계수이다. 상수 a와 b는 DCT와 동일하고 d는 c/b이다. 여기서 변환을 보다 간략하게 하기위해 d는 0.5로 근사화한다[2].

$$Y = CXC^T \otimes E$$

$$= \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & d & -d & -1 \\ 1 & -1 & -1 & 1 \\ d & -1 & 1 & -d \end{bmatrix} [X] \begin{bmatrix} 1 & 1 & 1 & d \\ 1 & d & -1 & -1 \\ 1 & -d & -1 & 1 \\ 1 & -1 & 1 & -d \end{bmatrix} \otimes \begin{bmatrix} a^2 & ab & a^2 & ab \\ ab & b^2 & ab & b^2 \\ a^2 & ab & a^2 & ab \\ ab & b^2 & ab & b^2 \end{bmatrix}$$

[식2]

H.264 코덱에서 근사화 변환은 DCT와 거의 동

일한 압축 성능을 나타내며, 여러가지 중요한 장점을 가진다. 변환의 핵심 부분인 CXC^T는 덧셈, 뺄셈 그리고 쉬프트만을 사용하여 정수 연산을 수행할 수 있다. 입력이 ±255의 범위 내에서 존재하므로 16-비트 연산을 변환 과정에서 사용할 수 있다. 이와 같은 중요한 장점은 하드웨어 구현 시 연산량을 적게하여 구현을 용이하게 한다[1][2].

3. 양자화

H.264는 스칼라 양자화를 사용한다. 순방향 및 역방향 양자화의 방법은 나눗셈 또는 소수점 연산을 사용하지 않고 post-스케일링과 pre-스케일링을 사용한다. 식3은 기본적인 순방향 양자화식이다.

$$Z_{ij} = \partial(W_{ij}/Qstep)\,(\partial = \text{반올림}) \quad [식3]$$
$$(W_{ij} = CXC^T)$$

여기서 Qstep은 양자화 스텝 사이즈이고, Z_{ij}는 양자화된 계수이다. H.264에서는 52개의 Qstep 값을 지원하며 양자화 파라미터 QP에 의해 지시된다. 표 1은 H.264에서 제공하는 양자화 스텝 사이즈 값들이다. 아래 표에서 보듯이 Qstep은 QP가 6 증가할 때 마다 두 배가 된다. 하지만 H.264에서는 식3과 같은 양자화를 사용하지 않는다. 이는 양자화 과정 전에 변화 과정에서 post-스케일링 계수 값을 수행해 주어야 하기 때문에 식4와 같은 식을 사용하여 양자화를 수행한다[3].

<표 1> H.264 코덱의 양자화 스텝 사이즈

QP	0	1	2	3	4	5	6	7	8	9	10	11	12	...
qStep	0.625	0.6875	0.8125	0.875	1	1.125	1.25	1.375	1.625	1.75	2	2.25	2.5	...
QP	...	18	...	24	...	30	...	36	...	42	...	48	...	51
qStep	...	8	...	16	...	32	...	64	...	128	...	160	...	224

$$Z_{ij} = \partial(W_{ij} \times \frac{PF}{Qstep}) \quad [식4]$$
$$(W_{ij} = CXC^T),\ (PF = \text{스케일 계수 } E)$$

식4와 같은 양자화는 고정 소수점을 사용하기 때문에 하드웨어 구현 시 용이하지 못하다. 따라서 연산을 더 간단히 하기 위해 계수(PF/Qstep)은 참조 소프트웨어에서 MF에 의한 곱셈과 오른쪽 쉬프트로 구현하여 나눗셈을 사용하지 않으므로써 하드웨어 구현을 용이하게 한다. H.264는 표 2와 같이 MF 계수 또한 지원해주고 있다. 식5는 MF를 사용한 양자화 과정이다[4].

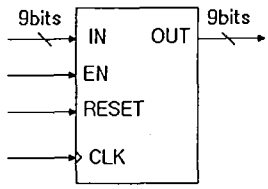
$$Z_{ij} = \partial(W_{ij} \times \frac{MF}{2^{qbits}})$$
$$\frac{MF}{2^{qbits}} = \frac{PF}{Qstep} \quad [식5]$$
$$qbits = 15 + floor(QP/6)$$

<표 2> 곱셈 계수 MF

QP	Positions (0,0),(2,0), (2,2),(0,2)	Positions (1,1),(1,3), (3,1),(3,3)	Other Positions
0	13107	5243	8066
1	11916	4660	7490
2	10082	4194	6554
3	9362	3647	5825
4	8192	3355	5243
5	7282	2893	4559

4. 변환 및 양자화 설계

4.1 입출력 신호도



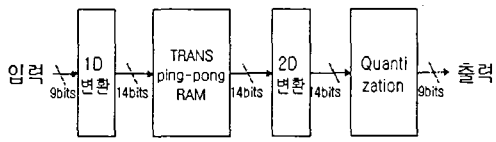
[그림 1] 입출력 신호도

그림 1은 입출력 신호도를 보여준다. RESET가 셋팅 되고 EN 신호가 1로 인가되면 9비트를 입력받아 44클럭 지연 후 결과 값이 9비트로 출력된다.

4.2 변환 및 양자화 전체 블록도

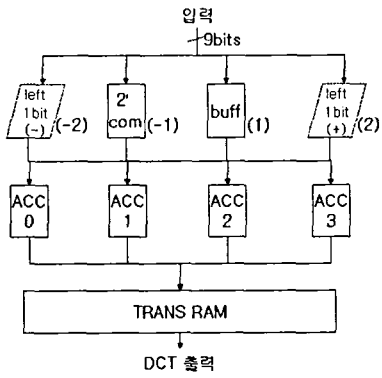
그림 2와 같이 전체 블록도를 보면 다음과 같은 세 개의 블록으로 나누어진다.

1) 1D 변환는 DCT와 같은 역할을 한다.
2) 핑퐁 램은 첫 블록의 데이터만 지연이 생긴 후 다음 블록부터는 순차적으로 결과 값이 나오게 한다.
3) 양자화는 입력된 14비트의 데이터를 받아 MF 계수와 곱셈 연산 후 반올림을 하고 상위 9비트만을 출력으로 보낸다.



[그림 2] 변환 및 양자화 전체 블록도

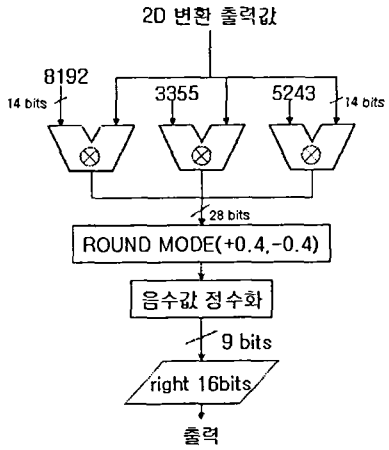
4.3 1D 변환 블록



[그림 3] 1D 변환 변형 블록도

그림 3은 1D 변환의 블록도를 나타낸다. 입력 값으로 ±255가 올수 있으므로 9비트를 받아 결과 값은 15비트로 확장을 시켜서 다음 과정으로 보낸다.

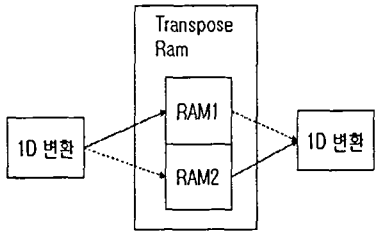
4.4 양자화 블록도



[그림 4] 양자화 블록도

그림 4는 양자화 블록도를 보여준다. 2D 변환 후 출력 값은 곱셈과 쉬프트 연산만을 사용하기 위하여 MF 계수를 곱한다. 그 다음 Round Mode에서 양수는 +0.4를 음수는 -0.4를 더해준다. 이는 양자화를 실행한 후 0에 가까운 값들을 만들기 위해서다. Round Mode 후에는 음수값 정수화 부분을 수행하는데 여기서 한 가지 주의 할 점은 Round Mode 후에 음수 값중 -0.4를 더하여도 -1이 되지 않은 값은 0이 되어야 하므로 이러한 경우를 체크해 주어야 한다는 것이다. Round Mode는 정수 계산을 하여 나온 정수 값이지만 실제적인 값은 28비트 중에서 하위 16비트를 제외한 다음 값부터 9비트의 값이 실제적인 데이터 값이므로 실수형 이라고 생각 할 수 있다. 그리고 끝으로 오른쪽으로 16비트 쉬프트를 실행해 주는데 16비트 쉬프트를 통하여 정수 값을 얻을 수 있다.

4.5 Transpose RAM



[그림 5] 핑퐁램을 이용한 변환 전체 블록도

1D 변환이 실행된 후 출력된 값은 Transpose Unit에 저장된 후 열과 행이 전치된 값이 출력되어 다시 2D 변환으로 입력된다. 여기서 Transpose RAM은 핑퐁 램을 사용하는데 그림 5에서와 같이 1D 변환 후 점선이 아닌 화살표 방향과 같이 RAM1에 데이터를 저장시키는 동안 RAM2에 데이터가 저장 되어 있다면 그 데이터는 2D 변환으로 데이터를 내보낸다. 이렇게 데이터를 2D 변환으로 보내면 처음 1D 변환 블록을 처리 할 때만 지연이 생기고 다음 블록부터는 연속적으로 출력 값이 나오게 된다.

4.6 타이밍과 컨트롤

<표 3> ACC선택 타이밍 테이블

	acc0	acc1	acc2	acc3
T0	1	2	1	1
T1	1	1	-1	-2
T2	1	-1	-1	2
T3	1	-2	1	-1

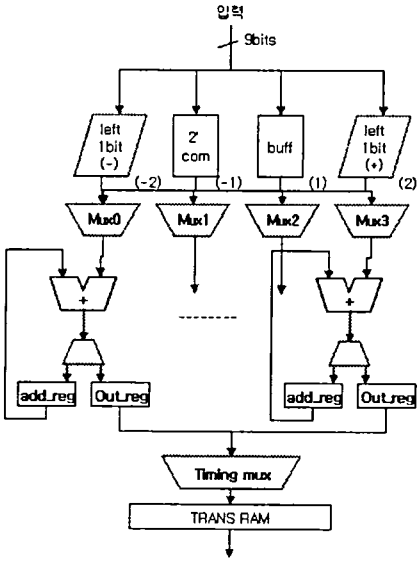
<표 4> MF 연산 타이밍 테이블

	MUL1	MUL2	MUL3
T0	O	X	X
T1	X	X	O
T2	O	X	X
T3	X	X	O
T4	X	X	O
T5	X	O	X
T6	X	X	O
T7	X	O	X

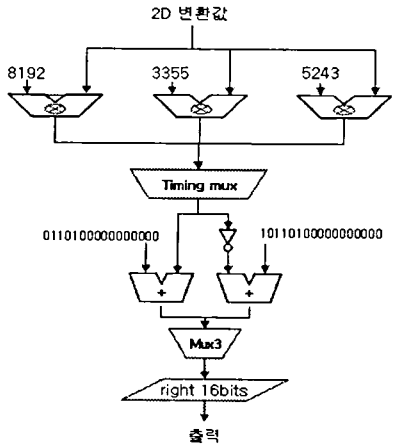
표 3테이블에 있는 값들은 그림 3에 있는 변환의 계수 값들이다.

표 4테이블에 있는 MUL은 MF 계수가 곱해지는 곱셈기를 나타낸다.

4.7 변환 및 양자화 전체 구성도



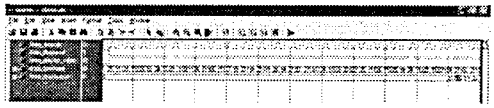
[그림 6] 변환 전체 구성도



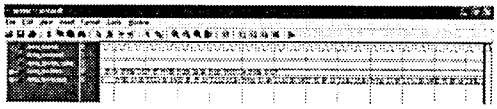
[그림 7] 양자화 전체 구성도

5. 출력 파형

그림 8은 입력 값의 파형으로써 리셋이 세팅되면 16개씩 블록단위로 영상 값이 계속 입력된다. 그림 9는 출력 값의 파형으로써 16개씩 블록단위의 영상의 값이 입력된 되면 44 클럭의 지연이 생긴 후 연속적으로 변환 및 양자화 수행이 된 값이 출력된다.



[그림 8] 입력값 파형



[그림 9] 출력값 파형

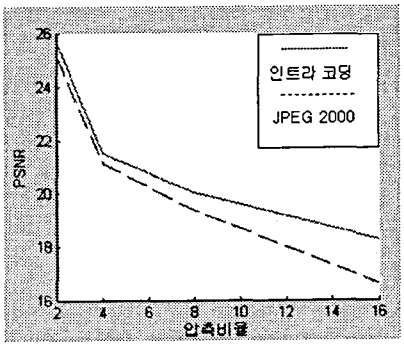
6. 설계 결과

설계된 변환 및 양자화 모듈은 4*4의 블록 영상 값 9비트를 입력받아 1D 변환을 수행 한 후 14비트의 출력 값을 출력한다. 이 출력 값을 16 클럭 동안 핑퐁 램에 저장한다. 16개의 값이 저장되면 핑퐁 램에서 4*4의 값의 전치 행렬을 수행하여 다시 2D 변환을 수행하는데 이때 수행하는 2D 변환은 처음 수행하였던 것과 같은 모듈을 사용한다. 2D 변환 후에는 14비트의 영상 값을 양자화 모듈로 보내어 양자화 처리를 수행한 후 9비트의 출력 값을 얻는다. 본 논문에서 수행하는 변환 및 양자화는 나눗셈과 소수점 연산을 하지 않고 쉬프트를 사용하여 출력된 값을 얻음으로써 인트라 코딩의 변환 및 양자화를 수행하였다.

[그림 10] 352*288 qcif 입력 영상

[그림 11] 352*288 qcif 출력 영상

[그림 12] 352*288 qcif jpeg2000으로 압축한 영상



[그림 13] 인트라 코딩과 JPEG 2000의 PSNR

이와 같은 방법으로 설계를 하여 그림 10을 입력 영상으로 나온 출력 값을 다시 복호화 시키면 그림 11과 같은 영상을 얻을 수 있다. 그림 12는 JPEG 2000 프로그램을 사용하여 압축한 영상이다. 그림 13은 인트라 코딩과 JPEG 2000 프로그램을 사용하여 압축 시킨 영상의 PSNR을 매트랩을 이용하여 구한 것이다. 그림에서 빨간색 선은 인트라 코딩을 의미하며 파란색 점선은 JPEG 2000의 PSNR을 의미한다[5][6][7].

7. 결론

영상 압축에 관한 연구는 계속적으로 이루어지고 있다. 그중에서도 최근 발표한 **H.264/AVC** 관한 연구들은 더욱더 활발이 연구되고 있다.

본 논문은 영상 압축 표준안인 **H.264/AVC** 연구하고 설계함으로써 기존에 존재하는 JPEG나 더 나아가 JPEG 2000 보다 더 좋은 정지 영상 압축 표준안을 제시하였으며 이를 하드웨어 설계언어인 VHDL언어를 이용하여 설계하고 FPGA칩인 XCV1000E에 다운로드 하여 칩 레벨의 시뮬레이션을 수행하여 설계된 변환 및 양자화 모듈을 검증하였다. 설계된 변환 및 양자화 모듈은 DMB 및 핸드폰 카메라와 같이 작은 정지 영상 압축에 응용이 가능하다.

8. 참고문헌

[1] 박경세, "디지털 멀티미디어 방송 기술 및 서비스", 커뮤니케이션북스

[2] Lain E.G. Richardson, "H.264 and MPEG-4", 홍릉과학출판사, 2004.9

[3] H. S. Malvar er al.,"Low-Complexity Transform and Quantization in H.264/AVC", IEEE Rrans. on Circ> and Sys> for Video Technology, Col. 13, No. 7, pp.598-603, July 2003.

[4] Joint Video Team of ISO/IEC MPEG & ITU-T VCEG, "JVT-G050r1", 2003.

[5] Detlev Marpe, Valeri George, Hans L.Cycon, and Kai U. Barthel, "Performance evaluation of Motion_JPEG 2000 in comparison with H.264/AVC operared in pure intra coding mode", 2003.

[6] John Wiley · Sons, "Video CODEC Design", WILEY, 2002.

[7] http://iphome.hhi.de/index.php