한국항행학회 논문지 제17권 제1호 2013년 2월 (JKONI 17(1): 69-79, Feb. 2013)

병행설계를 이용한 H.264/AVC의 DCT 및 CAVLC 하드웨어 구현

Hardware Implementation of DCT and CAVLC for H.264/AVC based on Co-design

왕덕상*, 서석용**, 고형화**0

Duck-Sang Wang*, Seok-Yong Seo** and Hyung-Hwa Ko**0

요 약

본 논문에서는 H.264/AVC의 부호기를 제작하기 위하여 DCT(Discrete Cosine Transform) 부호화와 엔트로피 부호화인 CAVLC(Context-Adaptive Variable Length Coding)를 하드웨어 IP로 설계하고 나머지 부분은 소프트웨어로 설계하는 병행설계(Co-Design)방법을 이용하였다. DCT 및 Hadamard 변환의 처리속도를 개선하기 위하여 Shift table을 제안하여 기존의 방식보다 16(%)정도 빠른 연산이 가능했다. 설계된 IP들은 Xilinx ML410보드의 Virtex-4 FX60 FPGA에 다운로드하여 MicroBlaze CPU를 이용하여 H.264/AVC의 참조 소프트웨어인 JM13.2와 연동이 가능하도록 설계하였다. 검증을 위해 각 IP에 대한 기능 시뮬레이션을 ModelSim을 이용하여 수행하였다. 마지막으로 실제 FPGA에 포팅하여 정상 동작여부를 확인하였다. 실험 결과 MicroBlaze를 이용한 S/W 연산시와 비교하여 H/W를 이용할 경우 DCT는 약 16배, CAVLC는 약 10배 빠른 처리 속도를 나타내었다. 본 연구는 H.264시스템의 H/W와 S/W의 병행설계에 관한 것이지만, 개발에 사용한 방법은 다른 임베디드 시스템 개발에도 유용하게 사용할 수 있다.

Abstract

In this paper, DCT(Discrete Cosine Transform) and CAVLC(Context Adaptive Variable Length Coding) are co-designed as hardware IP with software operation of the other modules in H.264/AVC codec. In order to increase the operation speed, a new method using SHIFT table is proposed. As a result, enhancement of about 16(%) in the operation speed is obtained. Designed Hardware IPs are downloaded into Virtex-4 FX60 FPGA in the ML-410 development board and H.264/AVC encoding is performed with Microblaze CPU implemented in FPGA. Software modules are developed from JM13.2 to make C code. In order to verify the designed Hardware IPs, Modelsim program is used for functional simulation. As a result that all Hardware IPs and software modules are downloaded into the FPGA, improvement of processing speed about multiples of 16 in case of DCT hardware IP and multiples of 10 in case of CAVLC compared with software-only processing. Although this paper deals with co-design of H/W and S/W for H.264, it can be utilized for the other embedded system design.

Key words : H.264/AVC , DCT , CAVLC

I. 서 론

2003년 새로운 차세대 비디오 압축 표준인 H.264/AVC가 ITU-T의 VCEG(Video Coding Expert Group)과 ISO/IEC의 MPEG(Moving Picture Expert Group)이 함께 구성한 JVT(Joint Video Team)에 의해

* 엘지전자 카사업부(LG Electronics. co. Ltd.)

** 광운대학교 전자통신공학과(Dept. of Electronics and Communication Eng., Kwang-Woon University)

· 제1저자 (First Author) : 왕덕상(Duck-Sang Wang)

0 교신저자 (Corresponding Author) : 고형화(Hyung-Hwa Ko, tel: +82-2-940-5137, email : hhkoh@kw.ac.kr)

· 접수일자 : 2013년 1월 22일 · 심사(수정)일자 : 2013년 1월 25일 (수정일자 : 2013년 2월 20일) · 게재일자 : 2013년 2월 28일

http://dx.doi.org/10.12673/jkoni.2013.17.01.069

제정되었다[1][2]. 이 표준은 ISO/IEC와 ITU-T에서 각각 사용되는 명칭인 MPEG-4 part 10 Advanced Video Coding (ISO/IEC 14496-10)과 ITU-T 권고명 H.264를 근거로 하여 H.264/AVC라고 불린다. H.264/AVC에는 이전 방식의 부호화 표준들(MPEG-2, H.264, MPEG-4 등)에는 없는 새로운 부호화 방식인 화면 내 예측 부호화(Intra Prediction), 다양한 크기의 블록 사용, 4x4 DCT, 다중 참조 영상 참조, 향상된 루프 필터와 엔트로피 부호화 등을 이용하여 이전 방식들보다 높은 압축 효율을 가지게 된다[3]. 이런 높은 압축 효율을 바탕으로 각종 저장매체, 모바일 화상 통신, 인터넷, 휴대 이동방송, HD-DVD등 다양한 분야에서 이용되고 있다[4].

H.264/AVC는 5프레임의 영상을 32화소 움직임 추정영역을 사용하여 메인 프로파일로 부호화 한 경우 MPEG-2보다 저비트율에서 약 45～65(%), 고비트율에서 약 25～45(%)까지 비트율 감소 효과 있다고 알려져 있다[5]. 그러나 이러한 높은 압축률과 더불어 부호기의 복잡도와 연산량이 많아지게 되었다. 이에 따라 저사양 PC 및 휴대용 멀티미디어 기기에서는 동영상을 원활하게 재생하기가 힘들다. 이에 따라 고화질의 영상을 재생하기 위하여 하드웨어로 제작된 코덱을 스마트폰, 휴대용 멀티미디어기기 등에 탑재하여 원활한 영상 재생을 할 수 있도록 하고 있다. 이에 따라 H.264/AVC의 하드웨어 IP확보의 중요성이 증가하고 있으며 내장형 프로세서, 메모리, 주변장치 등 기타 로직들을 하나의 칩으로 제작하는 SoC/IP의 개발이 중요한 연구 과제이다. 하지만, 실제 ASIC으로 구현 할 경우, 테스트 단계에 이르기까지 많은 시간과 비용이 소모된다. 특히 전체적으로 하드웨어와 소프트웨어 부분을 별도로 설계하여야 하는데, 설계 공간을 탐색하여 최적화된 분할을 찾는 것은 매우 어려운 문제이며, 한번 ASIC으로 구현되면 추후 수정이 불가능하다. 이러한 단점을 보완하기 위해서 초기 설계 단계에서부터 하드웨어와 소프트웨어를 동시에 개발하는 병행설계 방법이 각광받고 있다.

현재 H.264/AVC 하드웨어 코덱을 실제로 구현 방법은 대부분 ASIC 하드웨어 혹은 대규모 병렬 DSP 방식을 이용한 것으로써 소프트웨어적인 접근이 아니라 하드웨어적인 접근이다. 이러한 하드웨어적 접근 방법들은 소프트웨어에 비해서 확장성이 떨어지며, 개발에 많은 시간과 비용이 필요하고, 추후 수정이 불가능하다. 따라서 하드웨어의 빠른 연산과 소프트웨어의 넓은 확장성을 가지는 하드웨어와 소프트웨어의 병행설계가 반드시 필요하다.

본 논문에서는 ImpulseC CoDeveloper 프로그램을 사용하여 소프트웨어와 하드웨어를 개발 단계부터 동시에 개발하는 병행설계 방법을 이용한 임베디드용 H.264/ AVC 엔코더 시스템 중 DCT/Hadamard 부호화, CAVLC 엔트로피 부호화 모듈을 하드웨어로 설계 및 구현하였다.

본 논문의 구성은 2장에서 H.264/AVC 표준방식에 대한 개요와 설계한 IP에 대한 이론적 설명을 하였으며, 3장에서는 설계한 IP의 설계 방법과 제안한 알고리즘에 대해 기술하였다. 4장에서는 실험 및 결과를 제시하였고, 5장에서 결론을 이끌어 내었다.

Ⅱ. H.264/AVC의 기술적 내용

2-1 주파수 영역에서의 보간

H.264/AVC 표준은 기존의 비디오 부호화 방식들과 마찬가지로, 이미 부호화된 이전 화면으로부터 움직임을 추정하여 예측신호를 구성하고, 예측오류 신호를 이산 코사인 변환(DCT) 부호화하게 된다. 압축 성능을 높이기 위해 기존의 MPEG 방식에 비해 4×4 크기기의 블록 단위 까지 움직임 보상과 DCT/Hadamard 변환을 수행하며, 화질의 향상을 위해 DCT 변환을 이용하는 영상부호화 방식의 결점이었던 블록 경계의 왜곡을 억제하는 Deblocking 필터를 사용한다. DCT 양자화한 후 신호를 엔트로피 부호화할 때, 주위의 정보를 기반으로 부호화 테이블을 바꾸는 방식으로 영상의 성질에 맞게 부호화하여 압축 성능을 높인다. H.264/AVC 방식은 현재 전 세계에서 차세대 동영상 압축 기술로 디지털 방송, 휴대전화 등의 동영상 전송 및 응용 분야에 사용되고 있다.

2-2 DCT 부호화와 양자화

H.264/AVC에서는 움직임 예측 혹은 인트라 예측을 통해 얻어진 차분 데이터에 4×4 화소단위 직교변환을 수행한다. 기존의 정지영상 및 동영상 압축에서는 8×8 화소단위 DCT변환을 사용하는 것에 비해 4×4 변환을 사용하여 얻는 장점은 첫째, 4×4변환이 8×8변환보다 처리하는 데이터 수가 적고 유효자리수가 적기 때문에 구현이 용이하고, 둘째, 4×4 화소단위의 화면 내 예측과 4×4 화소단위의 움직임보상에서 부호화의 최소 단위가 4×4화소이기 때문에 직교변환의 4×4 화소단위와 일치한다.

H.264/AVC DCT 변환은 곱셈, 덧셈, 뺄셈, 쉬프트 연산만을 사용하는 정수형 DCT를 사용한다. H.264/AVC의 정수형 DCT를 사용하게 되면 실수연산에 의해 발생되는 문제점인 많은 연산량과 변환 전후의 데이터 값의 불일치 문제를 해결할 수 있다. 이러한 문제점을 해결함으로써 H.264/AVC의 DCT변환은 빠른 연산과 DCT변환 전후의 데이터가 일치하도록 설계되어 있다. DCT 변환식은 식(1)과 같다.

$$Y = A \times A^T = \begin{pmatrix} a & a & a & a \\ b & c & -c & -b \\ a & -a & -a & a \\ c & -b & b & -c \end{pmatrix} \times \begin{pmatrix} a & b & a & c \\ a & c & -a & -b \\ a & -c & -a & b \\ a & -b & a & -a \end{pmatrix} \quad (1)$$

여기서,

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

H.264/AVC는 HDTV와 같은 고해상도의 영상을 높은 효율로 엔코딩 하기 위해 8×8화소단위 변환도 도입되어있는데 이는 FRExt(Fidelity Range Extensions)을 통하여 사용이 가능하도록 되어 있다. H.264/AVC에서는 정수 DCT 이외에도 계산량 감소를 위해 Hadamard 변환이 도입되었다. Hadamard 변환은 덧셈과 뺄셈연산만을 사용하는 직교변환이다.

2-3 개선된 정수형 DCT 알고리즘 제안

본 논문에서는 H.264/AVC에서 사용되는 양자화 Table을 Shift 연산만으로 처리하는 방법을 제안한다. 식(2)는 H.264/AVC의 정수형 DCT 연산식이다. 양자화 계수와 DCT계수를 이용해 만든 테이블 값 MF를 곱하여 DCT 연산과 양자화가 완료된다.

$$|Z_{ij}| = (|W_{ij}| \cdot MF + F) \gg qbit \quad (2)$$

여기서, W_{ij}는 DCT변환된 행렬의 원소이며, F는 offset값이다.

본 논문에서는 H.264/AVC의 정수형 DCT 연산에서 사용된 MF의 값을 근사화하여 표현할 수 있는 테이블들을 생성한다. 이 테이블을 생성하기 위해서 식(3)을 이용하여 SHIFT1, SHIFT2, SHIFT3 값들을 구한다. 얻어진 SHIFT된 값들은 각각의 테이블에 저장되고 저장된 값들을 이용하여 식(4)의 연산을 행한다.

$$MF \cong 2^{SHIFT1} + 2^{SHIFT2} + 2^{SHIFT3} \quad (3)$$

$$|Z_{ij}| = (|W_{ij}| \ll 2^{SHIFT1} + |W_{ij}| \ll 2^{SHIFT2} + F) \gg qbit \quad (4)$$

표1은 MF값을 이용해 얻어진 SHIFT 테이블을 보여준다. 이렇게 생성된 SHIFT 테이블 값을 이용하여 DCT 양자화 연산시 기존의 MF값을 이용한 곱셈연산을 하지 않고 쉬프트 연산만을 하도록 하는 구조로 설계하였다. 식(4)를 이용하여 쉬프트 연산을 3번 수행하도록 하고, 성능 비교를 위하여 $MF \cong 2^{SHIFT1} + 2^{SHIFT2}$ 식을 이용하여 쉬프트 연산을 2번 실행하였다.

표 1. SHIFT Table 생성
Table 1. Generation of SHIFT Table

MF		SHIFT1		SHIFT2		SHIFT3	
13,107	12,222	13	13	12	12	9	0
16,777	11,916	14	13	8	11	7	10

2-4 CAVLC 엔트로피 부호화

H.264/AVC는 두 종류의 엔트로피 부호화를 포함하는데 문맥 기반 적응적 가변길이 부호화(Context-based Adaptive Variable Length Coding, CAVLC)와 문맥기반 적응적 이진 산술 부호화

(Context-based Adaptive Binary Arithmetic Coding, CABAC)이 있다. 메인 프로파일에는 CAVLC와 CABAC 모두 포함되어 있으며, 베이스라인 프로파일에는 CAVLC만이 포함되어 있다.

CAVLC는 부호화될 파라미터가 4×4 또는 2×2 DCT 변환 계수들의 차분 값을 부호화하는데 사용되는 방법으로 양자화된 블록에 대한 몇 가지 특성들의 장점들을 다음과 같이 얻을 수 있다.

① 예측, 변환, 양자화를 거치면 블록들은 일반적으로 0을 많이 포함하게 된다. CAVLC는 연속적인 0들을 간결하게 표현하기 위해 Run-level 부호화를 이용한다.
② 지그재그 스캔이후 0이 아닌 가장 큰 계수들은 종종 연속된 ±1이고 CAVLC 는 간결한 방법으로 고주파 ±1계수들의 개수를 알 수 있다.
③ 이웃 블록 내에 있는 0이 아닌 계수들의 개수는 look-up 테이블을 이용하여 부호화되고 look-up 테이블의 선택은 이웃하는 블록 내에 있는 0의 계수들의 개수에 따라 변화한다.
④ 0이 아닌 계수들의 크기는 재배치된 배열의 시작(DC 계수 근처)에서 더욱 크고 고주파 쪽으로 갈수록 더 작아지는 경향이 있다. CAVLC는 이러한 특징을 이용하여 최근에 부호화된 레벨의 크기에 따라 레벨 파라미터를 위한 VLC look-up 테이블을 선택한다.

표2는 CAVLC의 부호화 요소 내용이며, 그림 1은 CAVLC를 이용한 크기의 4×4 크기의 엔트로피 부호화를 나타내고 있다. 먼저 4×4크기의 DCT 데이터의 차분 값을 Zigzag Scan을 통하여 1차원으로 변환한다. 변환된 1차원 데이터를 이용하여 2진수로 이루어진 비트열을 생성한다.

표 2. CAVCL의 부호화 요소
Table 2. Encoding elements of CAVLC.

Element	Description
Coeff_token	0이 아닌 계수들의 개수(Total Coeff)와 Trailing Ones를 부호화(블록당 하나)
Trailing_ones_sign_flag	Trailing One 값의 부호(Trailing One 마다하나)
Level_prefix	0이 아닌 계수에 대한 부호의 첫 번째 부분(Trailing Ones를 제외하고 계수마다 하나)
Level_suffix	0이 아닌 계수에 대한 부호의 두 번째 부분(항상 존재 하지는 않음)
Total_zeros	첫 번째 0이 아닌 계수 이후에 존재하는 전체 0의 개수를 부호화(Zigzag scan 순서로서 블록당 하나)
Run_before	각각의 0이 아닌 계수 앞에 오는 0의 개수를 부호화(Zigzag의 반대 순서)

CAVLC 엔트로피 부호화 방법은 다음과 같다.

① 계수들의 개수와 TrailingOnes(±1)를 이용하여 Coeff_token의 테이블 값에서 해당 하는 비트열을 입력한다.
② TrailingOnes의 sign(±)값을 입력한다.
③ 0이 아닌 계수에 대한 Level_prefix와 Level_subfix를 이용하여 Level의 테이블에서 해당하는 비트열을 입력한다.
④ 1차원 데이터에서 마지막 연속적인 0이 나오기 전까지의 0의 개수를 이용하 여 Total zero의 테이블에서 해당하는 비트열을 입력한다.
⑤ 0이 아닌 계수의 역순으로부터 남은 0의 개수와 연속된 0의 개수를 이용하여 Run_before의 테이블에서 해당하는 비트열을 입력한다.

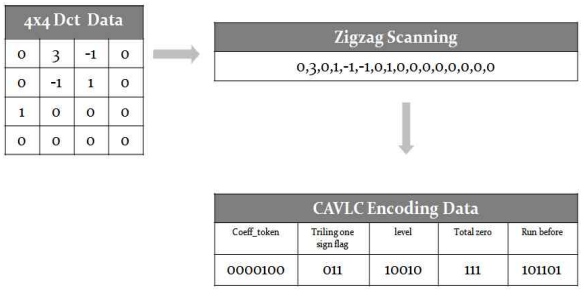

그림 1. CAVLC 부호화
Fig. 1. CAVLC Encoding.

Ⅲ. DCT 변환부 및 CAVLC 부호화부 하드웨어 설계

3-1 병행 설계 기법

하드웨어와 소프트웨어를 동시에 설계하는 병행설계(Co-Design)는 설계하고자 하는 시스템의 연산량이나 제어의 편리성 등에 따라 소프트웨어와 하드웨어를 동시에 설계하는 방식이다. 병행설계에는 ImpulseC CoDeveloper 프로그램을 사용하였다. ImpulseC 프로그램은 S/W와 H/W를 동시에 개발하는 병행설계(Co-Design) 연동 모델을 설계하는데 효율적인 툴로서, 초기 단계부터 하드웨어와 소프트웨어를 동시에 개발, 검증하고 하드웨어로 처리할 부분과 소프트웨어로 처리할 부분을 최적으로 분할하여 가격대 성능비를 향상 시킬 수 있으며 하드웨어와 소프트웨어의 설계 시간, 설계비용 등을 감소시킬 수 있다. 전체를 C언어로 개발하고 그 중에 하드웨어 부분은 VHDL, Verilog 언어로 자동으로 변환해 준다. 이때 가장 먼저 고려해야할 부분은 소프트웨어와 하드웨어의 분할 기준이다. 분할 기준은 큰 계산량을 필요로 하지 않으면서 복잡한 연산을 수행해야 하는 경우에는 소프트웨어로 구현하고, 큰 계산량, 규칙적인 연산, 반복적으로 사용되는 연산, 병렬적으로 처리될 수 있는 연산은 하드웨어로 구현한다. 다음으로 고려해야할 부분은 하드웨어와 소프트웨어를 연결하는 인터페이스이다. 인터페이스 부분은 하드웨어와 소프트웨어간의 데이터 전달을 담담하게 되며, 효율적인 인터페이스 블록을 설계해야 하드웨어와 소프트웨어간의 오버헤드(Overhead)를 줄일 수 있다. 하드웨어 블록과 소프트웨어 블록간의 인터페이스 방법은 메모리를 공유하는 방법, FIFO를 사용하는 방법, Handshaking 프로토콜을 이용하는 방법 등이 있다[6][7].

본 논문에서는 H.264/AVC 모듈중에서 DCT/Hadamard 부호화와 CAVLC 엔트로피 부호화는 하드웨어로 제작하였고, 기타 다른 모듈은 MicroBlaze를 이용하여 소프트웨어로 처리할 수 있도록 설계하였다. 그림 2는 설계한 IP를 탑재한 FPGA의 전체 구조이다. DCT 변환과 CAVLC 엔트로피 변환은 H/W로 처리하고 이를 제외한 H.264/AVC의 다른 모듈은 MicroBlaze를 이용한 S/W연산으로 처리 하도록 설계 하였다. 설계한 H/W IP는 Modelsim을 통하여 시뮬레이션 검증을 한 후 Xilinx ISE/XPS를 사용하여 H/W를 IP로 추가 한 후, Synthesis와 Implementation을 수행하였다. XPS를 이용하여 Bit 파일이 생성되면 Xilinx사의 개발용 보드인 ML410보드의 Virtex-4 FX60 FPGA에 다운로드 하였다.

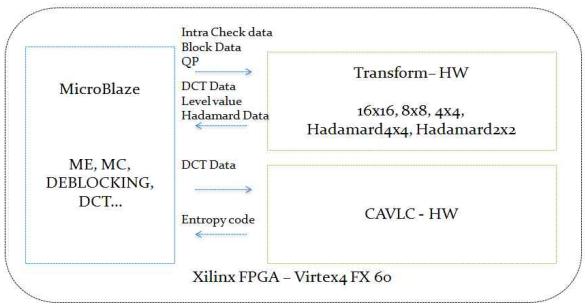

그림 2 구현된 시스템의 구조
Fig. 2. Structure of Implemented System.

3-2 변환 부호부 H/W 모듈 설계

H.264/AVC의 변환부호부는 4×4 DCT 변환을 기본으로 하고 충실도 확장규격(FREXt) 적용 시 고해상도 영상의 원활한 부호화를 위하여 8×8 DCT 변환을 지원한다. 그렇기 때문에 DCT H/W 설계시 4×4 DCT 변환과 8×8 DCT 변환 모두를 지원할 수 있게 설계하였으며, 휘도와 색차영상을 구분하여 휘도신호일 때 Hadamard변환을 수행하도록 설계하였다. 또

한 인트라모드의 16×16 영상의 경우 16×16 크기의 영상을 4×4 크기의 영상으로 분할 한 후, 4×4 DCT 연산을 수행 할 수 있도록 설계하였다. 그림 3은 설계한 4×4 DCT와 Hadamard H/W의 내부 구조를 나타낸 것이다. 여기서 4×4 DCT 대신에 8×8 DCT로 대체하고 Hadamard 변환부를 제거하여 8×8 DCT를 함께 구현하였다.

S/W 모듈과 H/W모듈간의 데이타 전송은 32비트 스트림을 통해 이루어진다. 입력 받은 영상 데이터는 H/W모듈의 32비트 크기의 메모리 블록에 저장되고, 입력된 영상에 대하여 4×4 DCT 변환을 수행하게 된다. 4×4 DCT 변환이 완료되면 변환된 데이터 값을 32비트 스트림을 통하여 S/W 모듈로 전송하게 되고, 이미지의 종류가 휘도일 경우 Hadamard 변환을 수행하고 채도일 경우 바로 양자화 과정을 수행하게 된다. 양자화 수행 시 앞서 제안한 방법을 이용

하여 수행하게 되고 Shift 연산에 사용될 데이터 값은 사전에 8비트 크기의 look-up 테이블에 저장시켜 놓고 이를 이용하여 양자화를 수행하게 된다. 양자화 완료 후 H/W 모듈은 DCT 및 Hadamard 변환한 후 양자화 된 데이터를 32비트 스트림을 통하여 S/W 모듈로 보낸다.

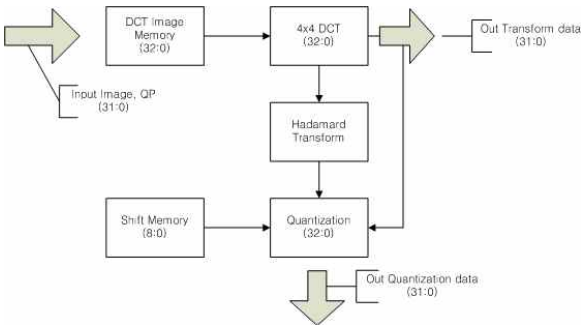

그림 3. 4×4 DCT H/W의 내부 구조
Fig. 3. Internal Structure of 4×4 DCT H/W.

3-3 CAVLC H/W 모듈 설계

CAVLC는 지그재그 스캔한 4×4 DCT의 차분 값을 엔트로피 부호화하는 모듈로서 CABAC보다는 데이터 압축률이 낮지만 빠른 연산이 가능하다는 장점이 있기 때문에 H.264/AVC의 베이스라인 프로파일과 메인 프로파일 모두에서 사용된다. 4×4 DCT 데이터를 지그재그 스캔을 통하여 1차원 배열로 재구성 한 후 Coeff_token, 부호비트, level, totalzero, runbefore에 해당하는 값들을 메모리에 저장된 각각의 테이블 값을 이용하여 부호화하게 된다.

그림 4는 설계한 CAVLC H/W 모듈의 내부 구조를 나타낸 것이다. 32비트의 스트림을 통하여 양자화된 4×4 DCT 변환의 차분 값과 문맥기반에 관련한 파라미터 값들을 입력 받은 후 데이터의 0과 0이 아닌 값들에 대한 개수를 구하게 된다. 이후 0이 아닌 계수의 개수와 절대 값 1의 개수를 이용하여 Coeff_token table에서 해당하는 값을 가져오게 된다. 그리고 절대 값 1의 계수에 대한 Sign(±)을 입력하고 0이 아닌 계수에 대해서 Level table을 이용하여 해당하는 코드 값을 가져온다. 마지막에 연속적으로 0 값이 나오는 부분을 제외한 영역에서 0값의 개수를 구하여 Total zero table에서 해당하는 값을 가져오고 마지막으로 0이 아닌 계수의 위치와 주위의 값들에 따라 Run_before table에서 값을 가져와 CAVLC 코드를 생성한다.

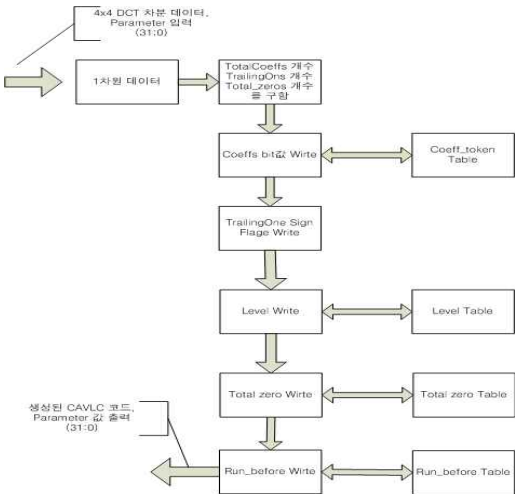

그림 4. CAVLC H/W의 내부 구조
Fig. 4. Internal Structure of CAVLC H/W.

Ⅳ. 실험 및 결과

4-1 실험 방법

설계된 IP의 검증은 Modelsim 시뮬레이터를 이용하여 기능 시뮬레이션을 실시하였고, Xilinx 사의 ISE 8.2 와 EDK 8.2 Tool을 사용하여 H/W 모듈과 MicroBlaze를 합성하고, S/W 모듈과 H/W 모듈의 연동 실험을 통해 정상 동작 여부을 확인하였다. 실제 검증에 이용된 H/W 시스템은 Xilinx사의 ML410 설계 테스트 보드이며 Virtex-4 FX60 FPGA 칩을 사용하고 있다. 또한 FPGA 내부에 생성된 MicroBlaze Core를 이용하여 75MHz 주파수로 동작시켰다.

성능 평가는 하드웨어 구조와 동일한 동작 구조를 가지는 JM13.2의 소프트웨어 모델을 구현하여 MicroBlaze를 이용한 소프트웨어로 처리했을 때 걸리는 시간과 하드웨어로 처리 했을 때 걸리는 시간을 비교하였다.

MicroBlaze Core는 내부 타이머가 존재 하지 않기 때문에 OPB-Timer를 이용하여 처리 속도를 측정하였다. OPB-Timer는 제작한 IP가 동작될 때 클럭을 측정하는 타이머로써 단위는 ticks이고, ticks에 프로세서의 동작 주기를 곱하면 동작 시간이 된다[8]. 공정한 시간 측정을 위하여 10번 반복 수행 후 평균 처리 속도를 구하였다.

4-2 실험 결과

제안한 방법에 대한 성능 평가는 H/W의 성능 평가를 위하여 기존의 4×4 DCT, CAVLC H/W와 게이트 사용량 및 연산 클럭수를 비교하였다. 병행설계을 이용한 임베디드 시스템의 성능평가를 위하여 8×8 DCT와 양자화 연산만을 실행하는 H/W 모듈을 제작하여 H.264/AVC에서 사용하는 8×8 정수형 DCT와의 연산처리 시간을 비교하였다. JM13.2를 이용하여 CAVLC와 CABAC 엔트로피 부호화 시 PSNR과 전체 비트 수를 비교 분석하였다.

(1) 제안한 DCT 방법의 연산시간

표 3은 Xilinx ML410 보드를 이용하여 실제 Virtex-4 FX60 FPGA 칩에서 기존의 8×8 정수형 DCT 변환과 제안한 방법을 소프트웨어적으로 100번 수행하였을 때의 연산 처리 시간을 비교한 것이다. 실험 결과 기존의 방법에 비하여 16(%)정도 빠른 연산이 가능함을 확인하였다.

표 3. DCT 연산 시간 비교
Table 3. Comparison of processing time for DCT

	기존 정수형 DCT 방법	제안한 Shift 방법
소요시간 (tick)	79,055	66,254

표 4. S/W 모듈과 H/W 모듈의 연산 시간 비교
Table 4. Comparison of processing time with S/W and H/W module.(단위:Tick)

		S/W	H/W	개선비
연산 속도	DCT	71×10^9	7×10^9	10.4배
	CAVLC	12×10^9	0.7×10^9	15.5배

표4에는 구현한 H/W 모듈의 성능을 보였다. ML410보드 상에서 소프트웨어적으로 구현한 모듈과 하드웨어 모듈의 연산 시간을 비교하였다. DCT의 경우 약 10.4(배), CAVLC의 경우 약 15.5(배) 빠른 것을 확인하였다.

(2) PSNR과 비트율

표 5에는 H.264/AVC의 정수형 DCT와 제안한 Shift 방법을 이용하여 DCT를 하였을 때의 PSNR과 비트율을 비교한 것이다. JM13.2를 이용하여 PC상

표 5. 제안한 DCT에 따른 CAVLC 엔트로피 부호화시 평균 PSNR과 비트율
Table 5. Average PSNR and Bit Rate using CAVLC entropy coding for proposed DCT.

영 상	Original		Shift 2회		Shift 3회	
	PSNR (dB)	Total Bit	PSNR (dB)	Total Bit	PSNR (dB)	Total Bit
bus	34.47	799,072	34.65	817,047	34.46	796,444
city	35.16	256,992	35.28	261,091	35.15	255,561
soccer	35.93	470,304	36.03	476,643	35.86	465,555
crew	36.40	527,688	36.47	539,724	36.38	527,003
football	35.81	1,201,568	36.06	1,232,377	35.79	1,198,930
foreman	36.81	303,552	36.91	311,335	36.77	302,855
harbour	34.02	769,144	34.17	782,446	34.01	769,236
ice	39.14	356,384	39.36	361,885	39.16	357,600
mobile	33.68	820,568	33.85	845,162	33.68	822,377
akiyo	39.39	56,928	39.55	58,078	39.31	57,122
테스트 영상 : QCIF - 58 Frame 전체						

에서 10개의 QCIF(176×144) 영상(Bus, City, Soccer, Crew, Footboll, Foreman, Harbour, Ice, Mobile, Akiyo)을 58 프레임으로 CAVLC로 부호화하였을 때를 비교한 것이다. 엔코딩시 파라미터 값은 JM13.2의 기본 설정으로 설정하였다. 실험 결과 Shift 연산을 3회 실시하였을 경우 원영상과 거의 유사한 성능이 나타남을 확인할 수 있었다. Shift 연산을 2회 실시하였을 경우에는 PSNR이 약간 떨어지고 비트량은 증가하는 trade-off가 발생하는 것을 확인 할 수 있었다.

표 6은 10개의 영상을 CAVLC와 CABAC 방법으로 각각 부호화 하였을 때 평균값을 나타낸 것이다. 실험 결과 Shift 연산 3회 이용하였을 경우에는 CAVLC 엔트로피 부호화시 원영상과 비교하여 PSNR도 약간 개선되고 비트량은 0.17(%) 정도 증가하는 것을 확인하였다. CABAC 엔트로피 부호화시 PSNR은 유지되고 비트량이 0.24(%) 감소하는 것을 볼 수 있다. 반면에 Shift 연산을 2번 이용할 경우 CAVLC 엔트로피 부호화시에는 PSNR이 0.15(dB) 낮아지고 비트량은 2.17(%) 감소하였다. CABAC 엔트로피 부호화를 이용하였을 때 원영상보다 PSNR이 0.1(dB) 낮아지고 비트량은 1.9(%) 정도 감소하였다. 원래의 방법과 성능이 개선이 이루어지면서도 표3에서 보인 것과 같이 연산 속도는 16(%) 정도의 빨라지는 것으로 나타났다.

본 논문의 연구결과를 기존의 논문과 비교해 보면, 참고문헌[8][9]의 경우 PSNR이 0.066(dB) 감소하고 비트율은 2.31(bps) 증가 하지만 모드결정 연산 시 61.2(%) 빠른 연산을 수행할 수 있다. 또한 DED(Domain Edge Direction) 방법[10]의 경우 PSNR이 0.17(dB) 감소하고 비트율이 4.23(bps) 증가하지만 모드결정시 73.78(%)의 연산 속도 감소가 이루어진다. 기존 방법들은 PSNR 감소와 비트율의 증가가 발생하는 반면 제안한 방법은 Shift 연산 3회 실행 시 PSNR 감소와 비트율 증가 없이 DCT 연산 시간을 16(%) 감소시킬 수 있다. 또한 Shift 연산 2회 실행 시

표 6. 엔트로피 부호화에 따른 평균 PSNR과 비트율 비교
Table 6. Comparison of average PSNR and Total Bit according to entropy coding

	Original		Shift 2회		Shift 3회	
	PSNR (dB)	Total Bit	PSNR (dB)	Total Bit	PSNR (dB)	Total Bit
CAVLC	36.08	556,220	35.93	544,141	36.11	557,182
CABAC	36.09	522,014	36.00	511,961	36.09	520,740

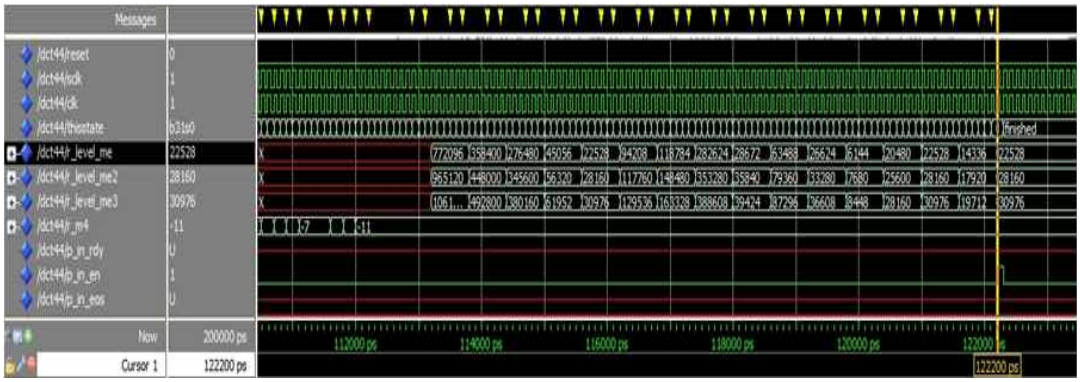

그림 5. 16×16 DCT 시뮬레이션 결과
Fig. 5. 16×16 DCT Simulation result.

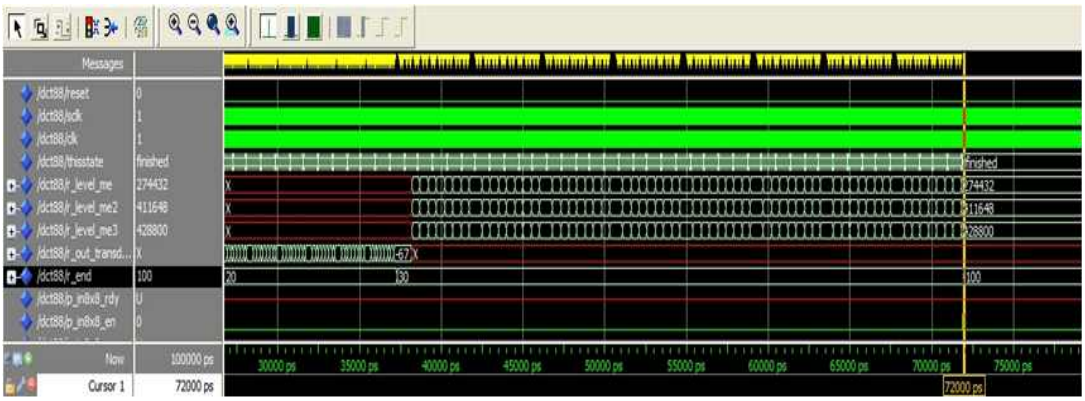

그림 6. 8×8 DCT 시뮬레이션 결과
Fig. 6. 8×8 DCT Simulation result.

PSNR의 0.1～0.15(dB) 감소로 비트량이 약 2(%) 정도 줄어들게 되므로 효율적이라 할 수 있다.

(3) H/W 모듈의 성능 평가

단독설계 방법으로 제작된 하드웨어와의 성능 비교를 위하여 전체 로직의 크기 및 연산시 소요되는 클럭을 비교 분석하였다. 병행설계 방법으로 제작한 4×4 DCT H/W 모듈과 기존에 단독설계 방법으로 제작된 4×4 DCT H/W 모듈의[11] 로직 크기 및 소요 클럭을 비교하였다. 비교 대상으로 사용한 기존에 제작된 4×4 DCT H/W의 경우 변환 및 역변환을 수

행할 수 있도록 제작되어 있으며 PCI인터페이스 부분을 제외한 로직의 게이트 사용량은 30K이다. 또한 한번의 4×4 DCT와 양자화 연산을 수행하는데 21 클럭이 소요되는 것으로 제시되었다.

반면에 본 논문에서 병행설계를 이용한 4×4 DCT H/W 설계시 면적을 최적화할 경우, 총 게이트 사용량은 스트림 인터페이스를 포함하여 158K가 사용되었고 한번의 4×4 DCT와 양자화 연산을 수행하는데 198클럭이 소요되었다. 이것은 본 논문에서 병행설계를 하기위해 사용한 Impulse C 프로그램이 유한상태기계(FSM: Finite State Machine)에 기반을 두어 하드웨어를 자동으로 생성하는 구조이기 때문에 최적화에 한계가 있고, 하드웨어와 소프트웨어의 인터페이스를 위한 소요 로직이 증가하기 때문이다. 그러나, 절대적인 게이트량이 최근의 ASIC 칩 용량에 비하여 매우 작은 것을 알 수 있다. 또한, 제작한 4×4 DCT 하드웨어의 성능 향상을 위하여 속도를 최적화할 경우 게이트의 사용량은 183K이며 연산에 필요한 클럭은 121클럭이 소요되었다. 한편 병행처리 방법으로 제작한 CLVLC H/W는 전체 게이트 사용량이 583K이고 한 Codeword를 연산하는데 100～200클럭 사이가 소모되며 임의의 데이터 10회 연산시 필요한 평균 클럭은 157클럭이다.

그림 5는 인트라 모드에서 16×16 영상을 4×4 크기

로 나누어 4×4 DCT 연산 시 ModelSim 시뮬레이션 결과이다. 시뮬레이션 시 클록은 100(ps)로 설정하였고 입력 영상은 메모리에 저장시켜 놓은 값을 이용하였다. 시뮬레이션 결과 Visual Studio 2008로 실행하여 얻은 결과 값과 동일한 데이터가 출력되는 것을 확인하였고, 16×16 크기의 영상을 DCT 변환하는데는 122.2(ns)의 처리시간이 소요되었다. 그림 6은 FREXt 프로파일에서 사용되는 8×8 DCT 연산의 ModelSim 시뮬레이션 결과이다. 8×8 DCT 연산을 하는데 걸리는 시간은 72(ns)가 얻어졌다.

Ⅴ. 결 론

본 논문에서는 하드웨어와 소프트웨어를 동시에 설계하는 병행설계(Co-Design)를 이용한 임베디드 시스템에서 사용 가능한 H.264/AVC 엔코더를 설계하는데 있어서 DCT 변환부와 CAVLC 엔트로피 부호화부를 하드웨어로 설계하였다.

DCT 변환부를 설계함에 있어서는 H.264/AVC에서 제시한 변환식에 의한 DCT계수를 사용하지 않고 $MF \cong 2^{SHIFT1} + 2^{SHIFT2} + 2^{SHIFT3}$ 를 이용한 근사화된 수치와 곱셈 연산이 아닌 Shift 연산을 이용하여 DCT 연산을 수행하였다. 실험 결과 H/W로 제작시 제안한 방법이 H.264/AVC에서 사용하는 기존의 정수형 DCT보다 16(%) 정도 빠른 연산이 가능하였고 FPGA 사용량도 감소하였다. 또한 3번의 Shift 연산을 행하였을 때 정수형 DCT와 거의 같은 PSNR과 비트율을 얻을 수 있었다 Shift 연산을 2번

행하였을 때는 PSNR 0.1~0.15(dB) 감소하지만 비트량은 약 2(%) 감소되었다.

ML410 보드를 이용한 실제 FPGA에서 동작 실험 결과 개발된 DCT 하드웨어 IP는 MicroBlaze를 이용한 소프트웨어 연산에 비해 약 10.4(배), CAVLC 하드웨어 IP는 소프트웨어 연산보다 약 15.5(배) 빠른 속도로 연산이 가능하였다.

본 논문에서 구현된 하드웨어는 기존의 단일 설계 방법에 의한 결과와 비교하면 로직의 면적이 더 많이 요구되는 것을 나타내었으나, 이는 설계에 사용한 Impuse C 프로그램이 FSM(유한상태머신)방식에 기반하기 때문이다. 그러나, 설계에 소요되는 시간이 짧고 투입되는 비용이 적게 들어 하드웨어와 소프트웨어 병행설계를 통해 멀티미디어 시스템을 빨리 개발하는 경우에 유용할 것으로 생각된다.

휴대용 멀티미디어 기기의 보급과 함께 고압축, 고화질의 영상을 원활히 재생할 수 있는 하드웨어 코덱의 IP 확보가 중요시 되고 있는 요즘 본 연구는 병행설계를 이용한 H.264/AVC 하드웨어 코덱을 제작하는데 있어서 기본 기반을 제공할 것으로 생각한다

.

감사의 글

본 논문은 2011년도 교내학술연구비 지원에 의해 연구되었음.

Reference

[1] JVT G050r1, "Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification(ITU-T Rec, H.264/ISO/IEC 14496-10 AVC)," May 2003

[2] Thomas Wiegand, Gary J. Sullivan, Gisle Bjontegaard, and Ajay Luthra, “Overview of the H.264/AVC Video Coding Standard,” *IEEE Trans. on Circuits and System for Video Technology,* Vol. 13, No.7, pp.560~576, July 2003.

[3] Iain E.G. Richardson, “H.264 and MPEG-4 Video Compression,” *The Robert Gordon University, Aderdeen*, 2004.

[4] Je-Chang Jeong, “H.264/AVC TextBook”, *HongRung Publishing Co*. 2005.

[5] Thomas Wiegand, Heiko Schwarz, Anthony Joch, and Faouzi Kossentini, “Rate-Constrained Coder Control and Comparison of Video Coding Standard,” *IEEE Transactions on Circuits and Systems for Video Technology*, Vol.13, pp.688~703, July 2003

[6] Jun-Mo Jeong, “HW/SW co-design of H.264/AVC Decoder using ARM-Excalibur ,” *J. KAIS*, Vol.10, No.7, pp1480~1483, 2009.

[7] Seong-Mo Park, Sukho Lee, Kyoungseon Shin, Jea-Jin Lee, Moo-Kyong Caung, Jun-Young Lee, and Nak-Woong Eum, "A Low Power Design of H.264 Codec Based on Hardware and Software Co-design," *J. KICS*, vol.25, No.12, pp.10-18, 2008

[8] Jhing-Fa Wang, Jia-Ching Wang, Jang-Ting Chen, An-Chao Tsai, and Anand Paul, "A Novel Fast Algorithm for Intra Mode Decision in H.264/AVC Encoders," *ISCAS2006*, pp.3498～3501, July 2006.
[9] Seong-Whan Lee, Young-Min Kim, and Sung Woo Choi, "Fast Scen Charnge Detection using Direct Feature extraction from MPEG Compressed Videos," *IEEE Trans. On Multimedia* Vol.2, No. 4. pp.240～254. Dec 2000
[10] Byeongdu La, Minyoung Eom, and Yoonsik Choe, "Fast Intra Mode Decision for H.264/AVC by Using the Approximation of DCT Coefficient," *IEEK Trans. On SP*, Vol.44, No.3, pp.23～32, May 2007.
[11] Young-Hun Lim and Yong-Jin Jeong, "Hardware Implementation of Integer Transform and Quantization for H.264," *J. KICS*, Vol.28, No.12C, 2003.

왕 덕 상 (Duck-Sang Wang)

2009년 2월 : 홍익대학교 컴퓨터정보통신공학과(공학사)
2011년 2월 : 광운대학교 전자통신공학과(공학석사)
2011년 10월~현재 : LG전자 Car사업부 근무

관심분야 : H.264 , H/W-S/W co-desing,

서 석 용 (Seok-Yong Seo)

1996년 2월 : 관동대학교 전자통신공학과(공학사)
2000년 8월 : 광운대학교 전자통신공학과(공학석사)
2012년 2월 : 광운대학교 전자통신공학과(공학박사)
관심분야 : JBIG2, H.264, H/W-S/W co-desing, Watermarking

고 형 화 (Hyung-Hwa Ko)

1979년 2월 : 서울대학교 전자공학과 (공학사)
1982년 2월 : 서울대학교 전자공학과 (공학석사)
1989년 2월 : 서울대학교 전자공학과 (공학박사)
1985년 3월~현재 : 광운대학교 전자통신공학과 교수

관심분야 : H.264, JBIG2, H/W-S/W co-desing, HEVC, Watermarking,