H.264용 Context-Based Adaptive Variable Length Coder(CAVLC) 설계

이홍식* · 서기범**

*우송대학교

A design of Context-Based Adaptive Variable Length Coder For H.264

Hong-Sic Lee* · Kibum Suh**

*Electronic Dept. Graduate School, Woosong University

E-mail : candols@wsu.ac.kr

요 약

본 논문에서는 AMBA 기반으로 사용될 수 있는 H.264용 CAVLC모듈의 새로운 구조와 설계를 하였다. 설계된 모듈은 Annex B.1 의 long-start code방식과 RTP 방식을 지원하며, 한 매크로 블록당 최대 420 cycle내에 동작한다. 제안된 구조를 검증하기위하여 JM 8.5부터 reference C를 개발하였으며, reference C로부터 test vector를 추출하여 개발된 회로를 검증하였다.
제안된 회로는 54MHz clock에서 동작하며, 합성결과 hynix 0.35 um TLM 공정에 14096 gate크기이다.

ABSTRACT

This paper propose an novel CAVLC architcture for H.264 and designed the CAVLC module which can be used in AMBA based design. This designed module can be operated in 420 cycle for one-macroblock and support both long-start code method using Annex B.1 and RTP. To verify the CAVLC architecture, we developed the reference C from JM8.5 and verified the our developed hardware using test vector generated by reference C. The designed circuit can be operated in 54MHz clock system, and has 14096 gate counts using Hynix 0.35 um TLM process.

키워드

CAVLC, H.264, RTP, NAL, entropy coder

Ⅰ. 서 론

H.264는 비디오 부호화에 해당하는 기능과 저장매체에 저장하거나 통신망을 통한 전송을 위한 기능을 구분하기 위하여 부호화기의 구조를 비디오 부호화 계층(Video Coding Layer)과 네트워크 적응 계층(Network Abstraction Layer)으로 분리하였다. 이런 특징으로 H.264는 회선 기반 망이나 패킷 기반 망을 통하여 효율적으로 데이터를 전송할 수 있도록 NAL Unit(Network Abstraction Layer Unit)이라는 구조화된 형태로 미디어 데이터를 저장한다[1].

부호화 계층에서 부호화한 비디오 비트스트림 데이터는 네트워크 적응 계층에서 슬라이스 단위로 나누어져서 각각 NAL Unit으로 저장하며, 나누어진 NAL Unit들은 패킷 기반망과 회선 기반망을 통하여 수신측으로 전송된다. NAL Unit을 통하여 수신측으로 전송하는 방법은 표준으로 지정해 놓지 않아서 전송 방법에는 제한이 없다. 일반적으로 패킷 기반 망에서 각각의 NAL Unit 들은 독립적인 RTP(Real-Time Transport Protocol) 패킷에 담아서 부호화기에서 부호화된 순서대로 전송한다.

본 논문에서는 베이스 프로파일 레벨 3규격의 H.264 코덱용 ASIC의 CAVLC 모듈의 하드웨어를 설계하였다. 그림 1과 같이 부호화 과정의 최종 단계는 Entropy Coding이다. Entropy Coding은 심볼의 출현 빈도에 의한 확률에 근거하여 최적의 코드워드를 배정하는 통계학적 예측에 기초하고 있으며 H.264에는 정보원 심볼의 발생확률에 따라 서로 다른 길이의 코드를 부여하는

VLC(Variable Length Coding)와 수학적 인코딩을 사용하는 CABAC(Context-Based Adaptive Binary Arithmetic Coding) 두 가지 형태의 Entropy Coding이 채택되어 있다. 본 논문에서는 모든 심볼들을 UVLC(Universal VLC) 테이블로 처리하는 부분과 4x4 양자화 된 계수 처리에 특화되어 설계된 CAVLC(Context-Based Adaptive Variable Length Coding)로 나누어 설계 하였다.

제안된 CAVLC모듈의 기능 및 성능에 관련된 주요 특징은 NAL Unit 과 Annex B.1의 스트림 생성을 지원하며 매크로블록보다 상위의 syntax에 대한 비트스트림을 생성할 수 있다. 이때 매크로블록 상위정보는 AMBA 인터페이스를 통하여 공급된다. 또한 Coeff_token를 생성하기 위한 인접블록에 대한 정보는 내부에서 발생시키며, 기존의 논문[4]에 비해 작은 메모리크기를 가진다.

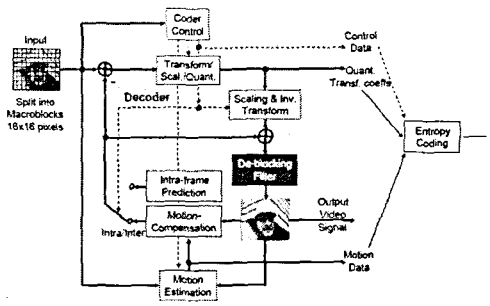

그림 1. H.264 Encoder 구조

II. CAVLC(Context-Based Adaptive Variable Length Coding)

H.264에서는 하나의 매크로블록 단위로 엔트로피 부호화가 수행된다. 하나의 매크로블록은 그림 2와 같이 4x4 블록이 25개, 2x2 블록이 2개의 블록으로 구성된다.

각 블록은 그림 2와 같이 미리 정의된 순서에 따라 부호화된다. 매크로블록의 부호화모드가 I16MB mode일 때는 4x4 Luma DC(Direct Current)블록(block -1)이 부호화되고, 4x4 Luma AC(Alternate Current) 블록(block 0 - 15)이 부호화된다. 그 다음으로 색도 DC(block 16 - 17)이 부호화 되고, 색도 AC(block 18 - 25)가 부호화된다. 매크로 블록의 부호화 모드가 I16MB가 아닐 때는 4x4 Luma DC 블록의 부호화가 생략된다.

CAVLC는 지그재그 스캔된 4x4 block과 2x2 block의 변환 계수들의 오차 블록을 부호화하는데 사용되는 방법이다. CAVLC는 양자화 된 4x4 block의 여러 가지 특징을 이용하기 위해 만들어졌다.[3]

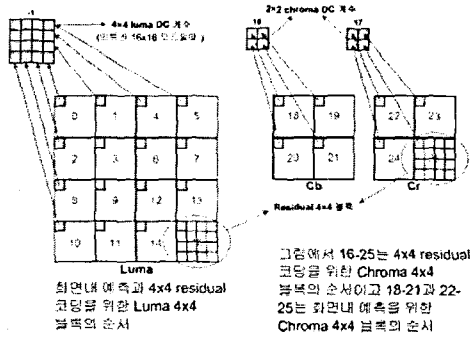

그림 2. Macroblock의 Encoding 순서

1. 예측, 변환 그리고 양자화 이후에 block들은 일반적으로 대부분 0을 포함한다. CAVLC는 연이어 있는 0을 간결하게 표현하기 위해 run-level 코딩을 사용한다.
2. 지그재그 스캔 이후에 0이 아닌 가장 큰 계수들은 대개 ±1이고 CAVLC는 고주파 성분의 ±1 계수('Trailing Ones')의 개수를 간결한 방법으로 알려준다.
3. 인접한 블록의 0이 아닌 계수들의 개수는 상관관계가 있다. 계수들의 개수는 look-up 테이블은 인접한 블록 내의 0이 아닌 계수들의 개수에 따라 선택된다.
4. 0이 아닌 계수들의 레벨(크기)은 재배치된 배열의 시작점(DC 계수 주변)에서 보다 큰 경향이 있고 고주파 쪽으로 갈수록 작다. CAVLC는 이러한 특징을 이용하여 최근에 부호화된 레벨의 크기에 따라 레벨 파라미터를 위한 VLC look-up 테이블의 선택을 달리한다.

변환 계수 블록의 CAVLC 인코딩은 다음과 같은 순서로 진행된다.

1. 계수의 개수와 TrailingOnes를 부호화한다.
2. 각 TrailingOnes의 부호를 부호화한다.
3. 나머지 0이 아닌 계수들의 레벨을 부호화한다.
4. 마지막 계수 이전의 전체 0의 개수를 부호화한다.
5. 각 0의 run을 부호화한다.

III. CAVLC의 Encoder 구조

CAVLC의 지그재그 스캔에 의한 변환은 이전 단계인 TQ/IQIT(Transform Quantization/Inverse Transform Inverse Transform)[2]에서 변환 되어 그림 3과 같이 SRAM에 저장되어진다.

각 메모리의 크기는 휘도신호의 블록의 경우 한 블록당 끝을 나타내는 0을 포함하여 최대 17개를 가질 수 있고, 한 매크로 블록당 16개의 블록으로 구성되어 있기 때문에 17x16=272가 된다.

색도신호의 블록의 경우 DC계수가 없으므로, 블록당 최대 16개의 계수 값을 가진다. 따라서, 색도신호의 AC계수는 16 x 8(Cb, Cr의 8개 블록) = 128 이 되어 Coefficient AC block의 address수는 400개이다. Coefficient DC block은 block number 0번에 대한 17과 block number 16과 17에 대해 각각 5씩 총 27이 소요되어 총 address는 427이다.

각 cycle당 20bit가 소요되며 상위 4bits는 Run값이 저장되어지고, 하위 16bits는 Level값이 저장되어지며 계수는 처음부터 연속적으로 채워진다.

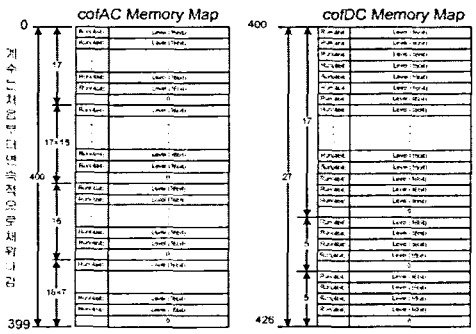

그림 3. CAVLC Memory MAP

그림 4는 CAVLC Encoder의 구조를 나타내고 있다. 이것은 nC_SRAM, Stream_buffer와 3개의 ROM table, Run-Level FIFO, Exp-Golomb table, Calcul_4x4_block, Multiplexing, Packing, Controller로 구성된다.

TQ/IQIT[2]에서 지그재그 스캐닝 되어 SRAM에 저장된 데이터는 MUX를 통하여 AC/DC Coeff_data를 구분하여 Calculator_For 4x4block으로 입력되어, 첫 번째 VLC인 Coeff_token은 0이 아닌 계수들의 전체 개수(TotalCoeffs)와 trailing ±1값의 개수(TrailingOnes)를 인코딩한다.

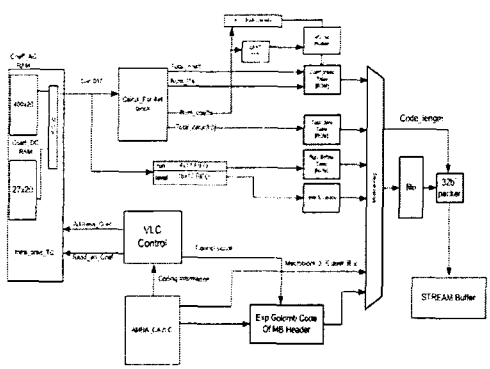

그림 4. CAVLC Encoder Architecture

그림 5는 본 논문에서 제안된 nC_SRAM memory 구조를 보이고 있다.(D1기준)

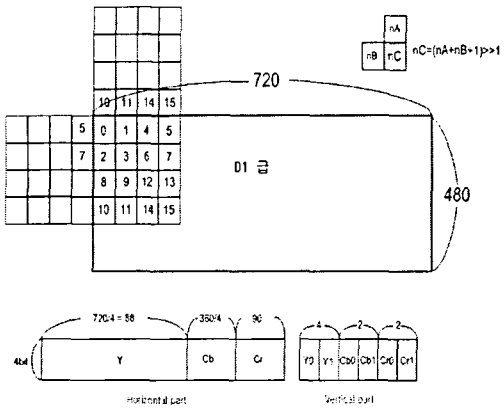

그림 5. nC SRAM의 Memory 구조

[4]에서는 Horizontal part와 Vertical part부분을 모두 memory로 처리한 반면 우리가 제안한 구조에서는 Horizontal part는 memory로 구현하고 Vertical part는 레지스터로 구현하여 memory의 크기를 감소하였다. 여기서, D1 기준으로 필요한 Memory 크기는, 그림 5에 보이는 것과 같이, (720/4 + 360/4 + 360/4)*4 = 1440 bit가 필요하며, 이것을 byte로 환산하면 180 byte가 된다. 표 1에서는 [4]가 제안한 구조의 nC_SRAM memory 크기와 우리가 제안한 nC_SRAM memory 크기를 비교하고 있다.

표1. CAVLC Encoder에 요구되는 memory size

	Lee[4]	Our method
HD(1920x1080)	1306 byte	480 byte
D1(720x480)	490 byte	180 byte
CIF(352x288)	240 byte	88 byte
QCIF(176x144)	120 byte	44 byte

TotalCoeffs는 0~15까지의 값을 가지며 TrailingOnes는 0~3까지의 값을 가진다. 3개 이상의 Trailing ±1이 존재하면 마지막 3개만 '특별한 경우'로 취급되고 나머지는 일반 계수들처럼 코딩된다. Coeff_token을 인코딩 하는데 테이블[1]을 사용하며 테이블은 이전에 코딩된 왼쪽과 위쪽의 블록에 존재하는 0이 아닌 계수들의 개수에 의해 선택되며 그림 6과 같이 결정된다.

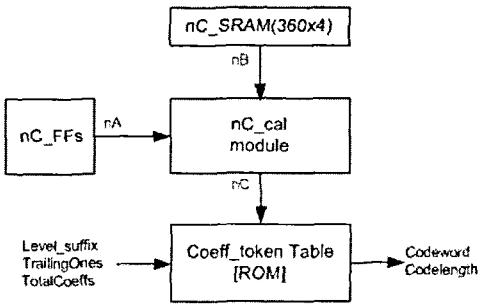

그림 6. Total_Coeffs module의 구조

위쪽과 왼쪽의 블록 nB와 nA가 모두 유효하면 nC = round((nA+nB)/2)이고, 위쪽만 유효하다면 nC = nB이고, 왼쪽만 유효하다면 nC = nA이고, 모두 유효하지 않다면 nC = 0이다.

Horizontal part의 SRAM과 Vertical part의 레지스터에서 이전의 코딩된 값을 계산한 nC값에 의해 VLC가 결정되어 지며 VLC값에 의하여 VLC table[1]이 결정되는데 적은 값의 TotalCoeff에는 특히 짧은 코드의 테이블을 할당하고 큰 값의 TotalCoeffs에 특히 긴 코드의 테이블을 할당함으로써 긴 코드 값의 길이를 줄여준다.

각 ROM에서 발생한 codeword와 codelength는 Multiplexer에 의해 선택되어져 최종 codeword의 값은 32bits로 packing 되어 진다.

IV. 실험결과

Mentor graphics 사의 modelsim을 이용하여 다음과 같은 결과 파형을 얻었다. 이 결과는 최대 420 cycle안에 CAVLC의 동작이 처리되고 있음을 보인다.

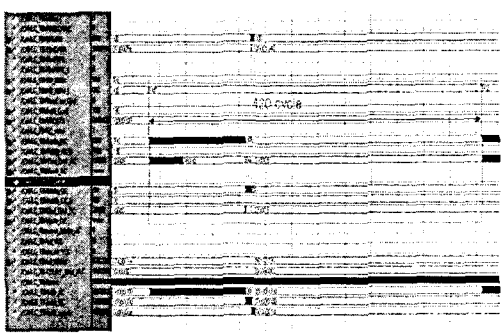

그림 7. 시뮬레이션 파형

V. 결 론

본 논문에서는 H.264 Encoder중 Entropy coding(CAVLC)을 하드웨어로 구현하였고, 그 과정 및 결과를 제시하였다. CAVLC 각 구성요소들은 nC_RAM, ROM table, Packer, Exp-golomb, VLC control, run-level FIFO, Calcul_4x4_block으로 구성되었다. 구현된 하드웨어 모듈은 54MHz에서 동작주파수를 가지며 최대 420 cycle안에 동작한다. Mentor graphics사의 modelsim을 이용하여 시뮬레이션 하였고 Synopsys Design compiler을 통해 합성결과 hynix 0.35um TLM 공정 사용시 54MHz의 동작스피드에서 14096gate 크기의 결과를 얻을 수 있었다.

참고문헌

[1] 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, Docs. JVT-G050r1

[2] Dae-Hyoung Lee "An Efficient Hardware Architecture of TQ/IQIT module for H.264 Encoder" IT-SoC Conference, Oct. 2004

[3] Lain E.G. Richardson "H.264 and MPEG-4 VIDEO COMPRESSION" John Wiley & Sons, Ltd. , 2003

[4] Dae-Joon Lee "Hardware Implementation for CAVLC Encoder For MPEG4 AVC" IT-SoC Conference, Oct. 2004