ISSN: 1226-7244 (Print)
ISSN: 2288-243X (Online)
논문번호 13-03-19
132

Journal of IKEEE.Vol.17,No.3,352~359,September 2013
http://dx.doi.org/10.7471/ikeee.2013.17.3.352

코드 테이블 패턴 분석을 통한 저비용 H.264/AVC 엔트로피 코딩 유닛 설계

Design of Low Cost H.264/AVC Entropy Coding Unit Using Code Table Pattern Analysis

송 세 현*, 김 기 철*★

Sehyun Song*, Kichul Kim*★

Abstract

This paper proposes an entropy coding unit for H.264/AVC baseline profile. Entropy coding requires code tables for macroblock encoding. There are patterns in codewords of each code tables. In this paper, the patterns between codewords are analyzed to reduce the hardware cost. The entropy coding unit consists of Exp-Golomb unit and CAVLC unit. The Exp-Golomb unit can process five code types in a single unit. It can perform Exp-Golomb processing using only two adders. While typical CAVLC units use various code tables which require large amounts of resources, the sizes of the tables are reduced to about 40% or less of typical CAVLC units using relationships between table elements in the proposed CAVLC unit. After the Exp-Golomb unit and the CAVLC unit generate code values, the entropy unit uses a small size shifter for bit-stream generation while typical methods are barrel shifters.

요 약

본 논문에서는 H.264/AVC 베이스라인 프로파일을 위한 엔트로피 코딩 유닛을 제안한다. 엔트로피 코딩에서는 매크로블록 부호화를 위해 코드 테이블들을 사용한다. 이 코드 테이블에 있는 코드워드들 간에는 일정한 패턴이 존재하는데, 본 논문에서는 코드워드의 패턴을 분석하여 하드웨어 비용을 줄였다. 제안하는 엔트로피 코딩 유닛은 Exp-Golomb 유닛과 CAVLC 유닛으로 구성된다. Exp-Golomb 유닛은 하나의 유닛에서 5가지 코드타입을 처리하고, 코드 패턴을 이용하여 두 개의 덧셈기만으로 복잡한 Exp-Golomb 연산을 처리 할 수 있다. 기존의 CAVLC 유닛은 큰 사이즈의 코드 테이블을 필요로 한다. 본 논문에서는 테이블에 차분 코드워드들을 저장하여 기존 방식 대비 테이블 크기를 40% 이하로 줄인다. 엔트로피 코딩 유닛은 생성된 코드워드들이 MSB에 코드값 0이 분포되는 패턴을 이용하여 쉬프터 사이즈를 대폭 줄였다.

Key words : H.264/AVC, encoder, entropy coding, CAVLC, Exp-Golomb

Ⅰ. 서론

모바일 기기의 발달은 개인이 다양한 비디오 콘텐츠를 생산하고 배포하게 만들었다. 이는 모바일 기기에서 효율적인 비디오 인코딩이 가능해야 함을 의미한다. 개인이 소비자 입장에 머물러 있을 때에는 비디오 디코더를 더욱 중요하게 생각하였기 때문에 인코더 연구가 상대적으로 부족하였다. 특히, H.264/AVC 비디오 코덱은 현재 가장 많이 사용되는 비디오 압축 방법으로 모바일 기기에 적합한 하드웨어 설계 방법을 필요로 한다. H.264/AVC 비디오 코덱은 화면 간 움직임 예측기, 화면 내 움직임 예측기, 정수 변환, 양자화, 역양자화, 디블록킹 필터, 엔트로피 코딩과정으로 이루어진다. 엔트로피 코딩은 화면

* Dept. of Electrical and Computer Engineering, University of Seoul
shsong02@gmail.com, 02-6490-5690
★ Corresponding author
※ This work was supported by the 2012 sabbatical year research grant of the University of Seoul.
Manuscript received Sep. 6, 2013; revised Sep. 12, 2013; accepted Sep. 12. 2013

간 움직임 예측 과정 또는 화면 내 움직임 예측 과정에서 압축된 블록 영상을 부호화 하는 기술로서 다른 연산 과정과는 다른 방식의 연산을 필요로 한다. 부호화 과정은 처리 영상 블록에 따라 연산 횟수가 변하고 생성되는 코드워드 (codeword)의 길이도 다르다. 또한 큰 사이즈의 코드 테이블을 필요로 하므로 이를 최적화 시키는 방법이 필요하다.

본 논문에서는 모바일 기기에서 사용가능한 엔트로피 코딩 유닛을 제안한다. H.264/AVC 인코더는 베이스라인 (baseline), 익스텐디드 (extended), 메인 (main), 하이 (high) 프로파일로 나누어진다[1]. 본 논문에서는 베이스라인 프로파일로 full HD 영상 처리를 목표로 한다. 베이스라인 프로파일의 엔트로피 코딩은 Exp-Golomb 연산과 CAVLC (context-based adaptive variable length coding) 연산으로 구성된다. Exp-Golomb 연산은 비디오 스트림의 SPS (sequence parameter set) RBSP, PPS (picture parameter set) RBSP, 슬라이스 헤더, 매크로블록 헤더 등을 압축하기 위해 사용되고, CAVLC 연산은 양자화 된 계수들을 부호화하는데 사용된다. CAVLC 연산은 압축 효율을 증대화 시키기 위해 양자화 된 계수들의 길이와 레벨정보를 따로 분리하여 부호화하는 방법을 사용하고 있으며, 이전블록의 정보를 다음 블록의 부호화 테이블 선택 기준으로 사용한다. 본 논문에서 제안하는 엔트로피 코딩 유닛은 다음과 같은 세 가지를 목표로 한다. 첫 번째, bit-stream 생성기의 쉬프터 크기를 줄인다. 생성되는 코드워드들은 서로 다른 길이를 갖고 있으며, 그 범위가 매우 넓다. 이는 큰 사이즈의 쉬프터를 요구하게 되는데, 이를 줄일 수 있는 방법을 연구한다. 두 번째, Exp-Golomb 연산에서 사용되는 모든 코드타입을 지원하는 연산 유닛을 개발한다. Exp-Golomb에는 5가지 코드타입이 존재하고 이에 따라 처리 방법이 다르다. 이를 하나의 통합 유닛으로 처리한다. 세 번째, CAVLC 연산에 사용되는 코드 테이블을 최적화 한다. CAVLC는 다양한 종류의 코드 테이블을 필요로 하는데 이를 줄이는 방법을 연구한다.

논문의 구성은 다음과 같다. 본론 1은 엔트로피 코딩 유닛을 제안하고 매크로블록을 처리하기 위한 동작 과정을 설명한다. 본론 2는 새로운 Exp-Golomb 유닛을 제안한다. 본론 3은 새로운 CAVLC 유닛을 제안한다. 마지막으로 실험 결과를 통해 본 논문의 엔트로피 코딩 유닛의 유용함을 증명한다.

II. 본론

2.1 엔트로피 코딩 유닛

엔트로피 코딩은 가변 길이 코딩을 사용하기 때문에 코드워드의 길이가 일정하지 않다. 생성되는 코드워드는 연속적으로 붙여져 패킷 화 되어져야 한다. 기존 엔트로피 코딩 유닛들은 큰 사이즈의 쉬프터를 이용하여 이 문제를 해결하려고 한다. 하지만 이와 같은 방법은 모바일 기기와 같이 자원이 한정된 곳에서 부담이 될 수 있다. 본 논문에서 제안하는 엔트로피 코딩 유닛은 쉬프터의 사이즈를 줄이기 위해 새로운 코드워드 표현법을 사용하고, 코딩에 사용되는 테이블 사이즈를 줄이기 위해 노력한다. 본 논문의 엔트로피 코딩 유닛은 그림 1과 같이 Exp-Golomb 유닛, CAVLC 유닛, bit-stream 생성기, NAL 생성기, 메인 컨트롤러로 구성된다.

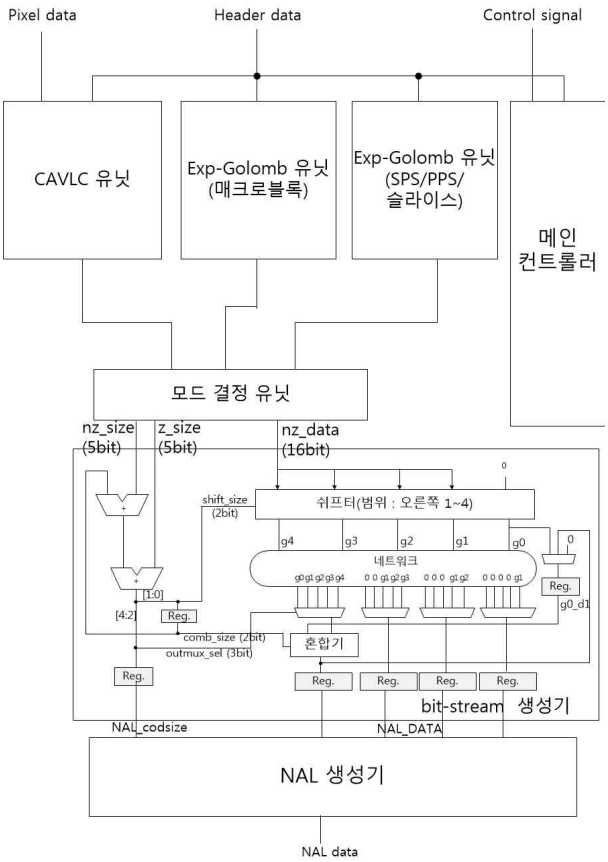

Fig. 1. Entropy coding unit
그림 1. 엔트로피 코딩 유닛

엔트로피 코딩 유닛의 동작 과정은 그림 2와 같다. 먼저, 메인 컨트롤러는 H.264/AVC 비디오 인코더가 시작되면 PPS RBSP와 SPS RBSP를 bit-stream 생성기에 전송한다. PPS와 SPS의 파라미터들은 시스템 요구사항에 따라 미리 정해진 값들이다. 별도의 Exp-Golomb 연산을 수행하지 않고 미리 코딩된 값

을 ROM에서 출력한다. 이 후 첫 번째 매크로블록이 전송되면 슬라이스 헤더를 부호화 하여 bit-stream 생성기로 전달한다. 슬라이스 헤더 역시 일부 파라미터를 제외하고는 미리 정해진 값들이다. 대부분의 값들은 미리 계산 되어져 ROM에서 전송되며, 일부 파라미터들은 실시간 계산되어 전송 시 대체된다. 슬라이스 헤더의 부호화가 종료되면 (sliceh_end=1), 매크로블록 헤더 부호화를 시작 한다. 매크로블록 헤더 부호화가 종료되면 (mbh_end=1). 양자 화된 계수들을 부호화 한다. 휘도 블록의 부호화가 종료되면 색차 블록에 대해 수행한다. 매크로블록에 대한 모든 부호화가 종료되면 End_MB 단계에서 엔트로피 코딩 유닛을 초기화하고 다음 매크로블록이 입력될 때까지 Wait 상태에서 대기한다. 두 번째 매크로블록이 입력될 때에는 슬라이스 헤더 생성 과정을 거치지 않고 매크로블록 생성을 진행한다. 하나의 슬라이스에는 복수개의 매크로블록이 존재한다. 메인 컨트롤러는 시스템 시작 전에 목표 해상도에 따라 프레임 당 처리해야 하는 매크로블록 수를 정해 놓는다. 프레임 단위의 매크로블록들을 처리하면 (mb_last=1), 다음 매크로블록이 발생(mb_start=1)할 때, 다시 슬라이스 헤더를 부호화 한다. 이와 같은 과정은 처리하고자 하는 프레임 수만큼 반복된다.

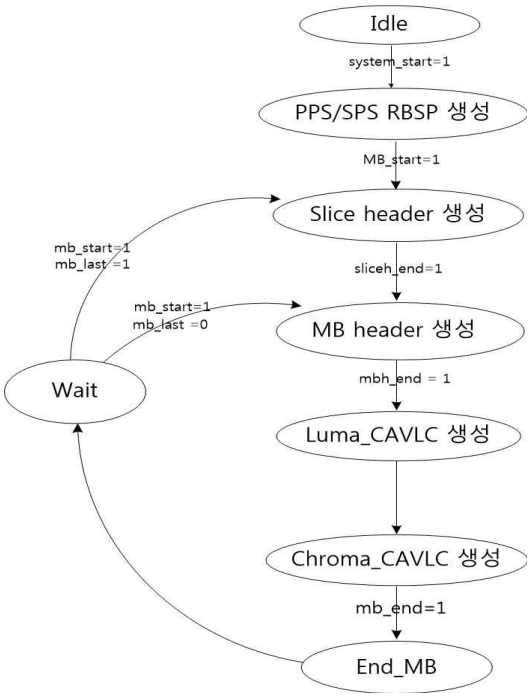

Fig. 2. Timing diagram of entropy coding
그림 2. 엔트로피 코딩의 타이밍도

Exp-Golomb 유닛과 CAVLC 유닛의 출력 신호는 nz_size, z_size, nz_data로 구성된다. 각 유닛으로부터 생성된 코드워드는 서로 다른 코드 길이를 가지며 MSB 쪽에 코드값 0이 분포되어 있거나 전부 0이 아닌 값으로 구성된다. 그러므로 코드의 0이 아닌 값만을 버스로 전송하고 코드값 0의 길이만 전송하게 되면 버스 대역폭을 줄일 수 있다. z_size는 코드값 0의 길이를 뜻하고, nz_size, nz_data는 각각 0이 아닌 코드의 길이와 값을 나타낸다. nz_data는 16비트 데이터 버스를 사용한다.

Table 1. Processes of bit-stream generator
표 1. bit-stream 생성기 연산 과정

	1	2	3	4	5
codewords	01	1	0_ 0011	000_ 1111	0_ 0011
nz_size	1	1	2	4	2
z_size	1	0	3	3	3
shift_size	2	3	0	3	0
comb_size	0	2	3	0	3
outmux_sel	0	0	2	1	2
g4	0000	0000	0000	0000	0000
g3	0000	0000	0000	0000	0000
g2	0000	0000	0000	0000	0000
g1	0000	0000	0011	0001	0011
g0	0100	0010	0000	1110	0000
g0_d1	·	0100	0110	0000	1110
NAL0	·	0000	0110	0110	0001
NAL1	·	0000	0000	0011	0000
NAL2	·	0000	0000	0000	0000
NAL3	·	0000	0000	0000	0000
NAL_size	·	0	0	2	1

그림 1의 bit-stream 생성기는 기존 엔트로피 코딩 유닛들보다 작은 사이즈로 설계되고, 2단계의 쉬프터로 구성된다. 미세조정은 첫 번째 오른쪽 쉬프터에서 처리되고, 큰 단위 조정은 네트워크 이후의 MUX들을 이용한다. 표 1은 bit-stream 생성기의 동작 과정을 나타낸 것이다. 예를 들어 01, 1, 00011, 00011111, 00011 값을 순차적으로 처리하고 싶을 때, 입력값 (nz_size, z_size, nz_data)는 (1, 1, 0..01), (1, 0, 0...01), (2, 3, 0...011), (4, 3, 0...01111), (2, 3, 0...011) 순으로 입력된다. 첫 번째 코드워드는 4비트 이하이기 때문에 NAL 유닛으로 출력되지 않고 g0_d1에 0100으로 저장된다. 두 번째 코드워드는 3 비트 오른쪽 쉬프트 되어 혼합기로 전달된다. comb_size가 2인 경우에는 g0_d1의 상위 2비트와 첫 번째 MUX의 출

력 값의 하위 2비트를 조합하여 4비트를 만든다. 이 때, MUX 출력 값은 g0, 0, 0, 0 이다. MUX 출력 값은 mux_sel 신호에 의해 결정된다. 세 번째 데이터는 쉬프트 되지 않고 네트워크로 전달되며, comb_size=3과 outmux_sel=2에 의해 0110_0011_0000_0000이 출력된다. 이 때, NAL_size는 출력되는 NAL 데이터의 유효 비트를 의미한다. NAL_size=2인 경우에는 상위 8비트만이 유효함을 의미한다. NAL 생성기는 출력되는 비트를 스택에 쌓아두고 일정 패킷단위로 나눠서 외부에 전달한다.

2.2 Exp-Golomb 유닛 설계

Exp-Golomb은 SPS RBSP, PPS RBSP, 슬라이스 헤더, 매크로블록 헤더를 압축하기 위해 사용된다. 표 2은 Exp-Golomb의 코드워드를 나타낸다.

Table 2. Codewords of the Exp-Golomb
표 2. Exp-Golomb의 코드워드

code_num	codewords
0	1
1-2	$01X_0$
3-6	$001X_1X_0$
7-14	$0001X_2X_1X_0$
15-30	$00001X_3X_2X_1X_0$
31-62	$000001X_4X_3X_2X_1X_0$
....	....

코드워드는 <Mzeros><1><INFO>와 같은 형태를 갖는다. <Mzeros>은 M개의 zero 코드를 갖는다. <1>은 <Mzeros>와 <INFO>의 구분자 역할을 하며, 코드값 1을 갖는다. <INFO>는 식 (1)과 같은 값을 갖는다.

$$M = \text{floor}(\log_2[\text{code\_num}+1]) \quad (1)$$
$$\text{INFO} = \text{code\_num} + 1 - 2^M$$

또한 Exp_Golomb은 부호화하기 위한 파라미터의 종류에 따라 u(n), ue(v), se(v), me(v), te(v)와 같은 5가지 코드타입을 갖는다. 코드타입은 서로 다른 방식으로 code_num을 생성한다. ue(v)은 파라미터 값이 직접 code_num이 된다. se(v)는 파라미터가 부호를 가질 때 사용된다. 예를 들어 파라미터 0 ,1, -1, 2, -2, 3 순으로 code_num은 0, 1, 2, 3, 4, 5이 된다. me(v)는 파라미터에 따라 정의되어 있는 테이블을 이용하여 code_num을 생성한다. te(v)는 파라미터가 1보다 클 경우 직접 code_num이 되지만 1일 경우 0으로 변경된다. 마지막으로 u(n)은 직접 코드워드를 입력하는 방법이다. 코드타입의 v는 생성된 코드워드가 가변 길이임을 의미하고, n은 고정된 길이를 의미한다. 본 논문에서 제안하는 Exp-Golomb은 5가지 코드타입을 하나의 유닛에서 지원하며, 간단한 연산 유닛으로 Exp-Golomb의 코드워드를 연산 한다.

그림 3는 본 논문에서 제안하는 Exp-Golomb 유닛으로 파라미터와 코드타입을 입력 받는다. 2 스테이지로 이루어진 Exp-Golomb 유닛은 z_size, nz_size, nz_data를 최종 출력으로 한다. z_size는 [Mzeros]를 나타낸다. nz_size는 [1]를 포함한 [INF0]의 길이를 나타낸다. nz_value는 [1]를 포함한 [INF0]의 값을 나타낸다.

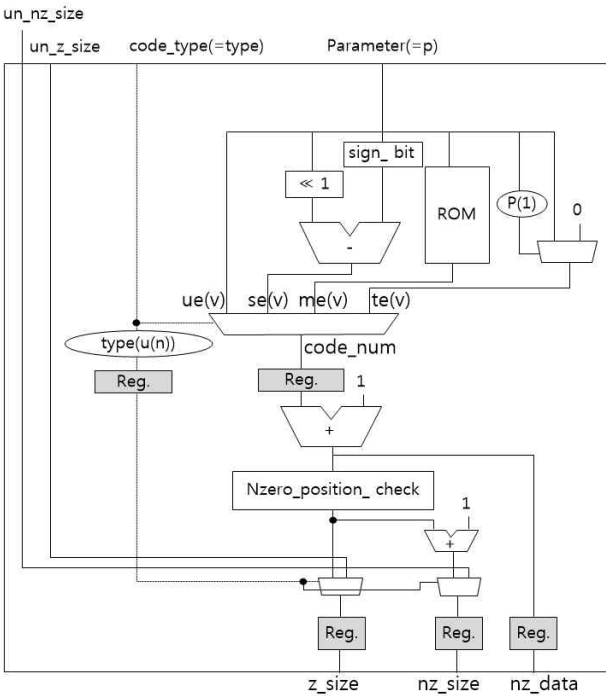

Fig. 3. Exp-Golomb unit
그림 3. Exp-Golomb 유닛

첫 번째 스테이지는 파라미터를 code_num으로 변경하고 코드타입에 따라 선택된 값을 레지스터에 저장한다. se(v)를 계산하는 과정에서의 sign_bit는 파라미터의 부호가 음수이면 1값을 갖고, 양수이면 0값을 갖는다. type(u(n))는 코드타입이 u(n)이면 1값을 갖고, 나머지일 경우는 0값을 갖는다. p(1)은 파라미터 값이 1이 아닌 경우 0 값을 가진다. 두 번째 스테이지의 nzero_position_check는 code_num의 첫 번째 0이 아닌 코드값의 위치를 나타낸다. 예를 들어

code_num=9이면 nzero_posion_check는 3이 되고, z_size=3, nz_size=4, nz_value= “1010”이 된다. 만약 code_type이 u(n)이면, 직접 입력된 un_z_size, un_nz_size가 각각 z_size, nz_size가 되고 code_num이 nz_data가 된다. 본 논문의 Exp-Golomb 유닛은 그림 2와 같이 간단한 연산 유닛으로 식 (1)과 같은 복잡한 연산식을 처리하여 자원을 절약하였다.

2.3 CAVLC 유닛 설계

CAVLC는 4x4 블록의 양자화 계수를 부호화하기 위해 사용된다. 큰 사이즈의 코딩 테이블을 필요로 하고, 생성되는 코드워드의 수가 일정하기 않기 때문에 최적화된 하드웨어 설계가 어렵다. 본 논문의 로 줄이고, 생성되는 코드워드 수의 가변 범위를 줄

Table 3. Table optimization method for Coeff_token
표 3. Coeff_token을 위한 테이블 최적화 방법

T_O	T_C	0≤nC<2	residual_code	code_size	value
0	0	1	1	1	1
0	1	000101	000100	5	4
0	2	00000111	00000010	2	2
0	3	0_00000111	0_00000000	1	0
0	4	00_00000111	00_00000000	1	0
0	5	000_00000111	000_00000000	1	0
0	6	00000_00001111	00000_00001000	2	4
0	7	00000_00001011	00000_00000100	0	-4
0	8	00000_00001000	00000_00000011	0	-3
0	9	000000_00001111	000000_00000111	1	-7
0	10	000000_00001011	000000_00000100	0	-4
0	11	0000000_00001111	0000000_00000100	1	4
0	12	0000000_00001011	0000000_00000100	0	4
0	13	00000000_00001111	00000000_00000100	1	4
0	14	00000000_00001011	00000000_00000100	0	-4
0	15	00000000_00000111	00000000_00000100	0	-4
0	16	00000000_00000100	00000000_00000011	0	-3

CAVLC는 테이블 사이즈를 기존 사이즈 대비 40% 임으로 하드웨어 설계에 용이하도록 한다.

CAVLC는 Coeff_token, Trailing_ones_sign_flag, Level_prefix, Level_suffix, Total_zeros, Run_before와 같은 요소를 가지며 나열된 순으로 부호화되어야 한다. Coeff_token은 TotalCoeff, TrailingOnes, nC에 의해 다른 코드 테이블을 사용한다. TotalCoeff는 정수값 0이 아닌 계수들의 수를 나타내고, 0부터 16의 범위를 갖는다. TrailingOnes는 스캔 순서에서 초기 값들이 정수값 ±1일 경우 그 수를 나타내고, 0부터 3의 범위를 갖는다. nC는 주변 블록들에서 정수값 0이 아닌 계수들의 수를 나타낸다. 테이블을 통해 생성된 코드워드는 최대 16비트를 갖게 되는데, 기존방식으로 테이블을 설계하게 되면 총 16 x 308 비트의 사이즈를 필요로 한다. 표3은 테이블 사이즈를 줄이기 위한 변환 방법이다.

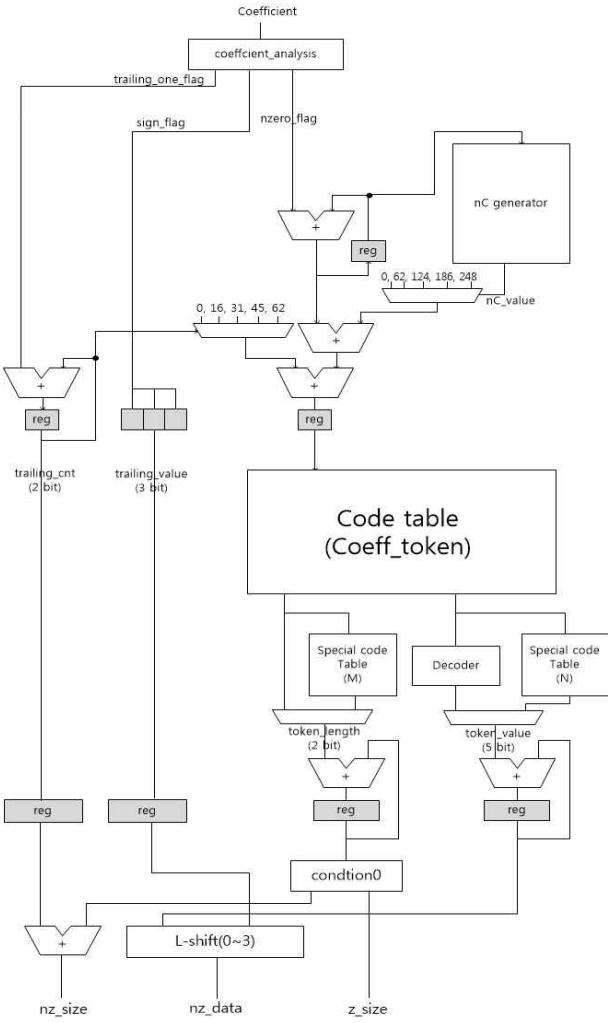

Fig. 4. Coeff_token code generator
그림 4. Coeff_token 코드 생성기

표 3은 TrailingOnes (=T_O)가 0인 경우, TotalCoeff (=T_C)가 증가할 때의 코드워드들이다. 표에서 residual_code와 code_size는 이전 T_C의 코드워드에 비해 달라진 값과 길이를 나타낸 것이며, value는 0이 아닌 코드값을 정수로 표현한 것이다. 실제 테이블에는 code_size와 value값만 입력된다. 모든 Coeff_token 테이블에 이와 같은 방법을 적용하면 code_size의 종류는 {0, 1, 2, M}으로 한정되어 2비트로 표현된다. 전체 코드워드의 10%가 M에 해당하며, 별도의 특수 코드 테이블에서 관리한다. 테이블의 value는 {-7, -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5, 7, 8, N}으로 한정되며, 순서대로 부호화되어 4비트로 표현된다. 전체 코드워드의 4%가 N에 해당되며, 특수 코드 테이블에서 관리한다. 본 논문에서 사용한 코드 최적화 방법은 추가 코드표를 감안하더라도 기존 테이블에 비해 사이즈를 40% 이하로 줄일 수 있다.

그림 4는 표 3을 구현한 것이다. Coeff_token 코드 생성기의 입력은 4x4 블록의 양자화 계수들이다. coefficient_analysis에 의해 생성된 trailing_ones_flag, sign_flag, nzero_flag는 각각 TrailingOnes의 여부, 계수의 부호값, 0이 아닌 계수의 여부를 나타낸다. 코드 테이블 선택에 필요한 nC는 계산 전에 결정되기 때문에 미리 nC에 대한 테이블을 선택한다. TrailingOnes는 스캔 순서 초기에 결정되기 때문에 nC와 TrailingOnes를 테이블의 기본 주소로 사용하여 증가되는 TotalCoeff에 따라 목표 코드워드에 접근해 간다. TotalCoeff에 의해 선택된 테이블의 출력 값인 code_size와 value는 차분값들이기 때문에 누적기에서 누적되어 실제 값으로 변환된다. 별도로 trailing_ones_flag가 생성될 때, sign_flag를 저장하여 Trailing_ones_sign_flag를 결정한다. bit-stream 생성기의 연산 횟수를 줄이기 위해 token_value를 TrailingOnes 만큼 왼쪽 쉬프트 시켜 Trailing_ones_sign_flag를 조합하여 출력한다. Coeff_token 코드 생성기의 출력 값은 Exp-Golomb 유닛과 동일하게 z_size, nz_size, nz_data이다. 원본 코드 테이블에서 코드워드들의 0이 아닌 코드값 길이는 5 비트를 넘지 않는다[1]. 그림 3의 condition0은 code_size가 5비트 이상이면 nz_size를 5로 고정시키고, z_size를 증가 시킨다. 5 이하인 경우에는 z_size를 0으로 출력한다.

그림 5는 CAVLC 유닛이다. 매크로블록 메모리로부터 4x4 블록의 양자화 계수를 zig-zag 순으로 읽어서 각 연산 유닛에 공급한다. 이때, 정수값 0이 아닌 계수는 FIFO 레지스터에 저장해두고 처리해야할 계수의 수를 기억해 둔다. 스캔이 마무리 되면 2사이클의 딜레이 후에 Coeff_token + trailing_ones_sign_flag 결과가 Coeff_token 코드 생성기에서 출력된다. 동시에 Level값을 하나씩 처리하고 Coeff_token + trailing_ones_sign_flag 이후에 딜레이 없이 바로 출력한다. Level은 suffixLength에 따라 다른 테이블을 사용하는데 본 논문의 CAVLC는 테이블 규칙을 이용하여 로직회로만으로 처리한다. 모든 Level값이 처리되면 Total_zeros를 출력한다.

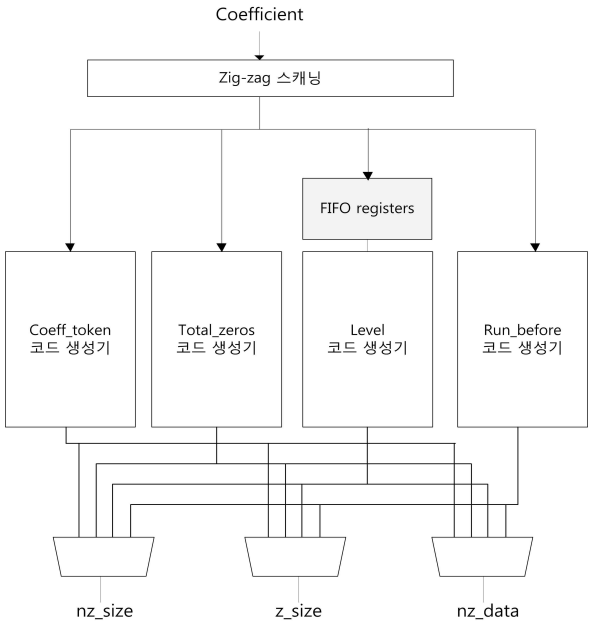

Fig. 5. CAVLC unit
그림 5. CAVLC 유닛

Total_zeros 생성기 역시 Coeff_token 생성기와 유사한 방법을 사용하여 테이블 사이즈를 줄인다. 마지막으로 Run_before를 출력한다. Run_before는 Level 수와 위치에 따라 값과 연산 횟수가 달라진다. 하지만 입력값의 경우의 수가 한정되고 출력 값의 크기가 크지 않음을 이용하여 4회의 고정 횟수로 모든 경우의 run_before를 생성할 수 있다. Level 횟수가 적은 경우에는 연산시간의 손해가 발생할 수 있지만 고정 연산시간으로 인해 4x4 블록을 연속적으로 처리할 때, 파이프라인 처리가 가능하게 한다. CAVLC 연산 시간은 16 사이클의 스캔시간, 초기 2사이클의 딜레이 시간, 4 사이클의 고정시간과 Level 처리에 소요되는 가변시간으로 구성된다. Level 수가 증가할수록 연산 시간은 비례적으로 증가한다. 매크로블록을 처리할 경우 연속적인 4x4 블록을 파이프라인으로 처리 가능

하기 때문에 스캔시간 16 사이클은 대부분 중첩되어 초기 1회를 제외하고는 무시할 수 있다.

III. 실험결과

본 논문에서 제안하는 엔트로피 코딩 유닛은 Exp-Golomb 유닛을 이용하여 SPS RBSP, PPS RBSP, 슬라이스 헤더, 매크로블록 헤더들을 부호화 시키고, CAVLC 유닛을 이용하여 양자화 계수들을 부호화 시킨다. 생성된 코드워드들은 bit-stream 생성기와 NAL 생성기로 패킷화 되어 외부로 전송된다. 본 논문의 엔트로피 코딩은 실제 인코더에 사용가능 도록 모든 과정을 완벽하게 지원한다.

Table 4. Comparison of the entropy coding units
표 4. 엔트로피 코딩 유닛의 성능 비교표

	[2]	[3]	[4]	Proposed
Technology	0.18 um	0.18 um	0.18 um	90 nm
Gate count	15,864	16,385	11,700	13,600
Clock frequency	100	81	158	150
Target format (MHz)	FHD	FHD	FHD	FHD
연산 지원 여부	EXPG+ CAVLC	CAVLC	CAVLC	EXPG+ CAVLC

본 논문에서는 엔트로피 코딩 유닛의 크기를 예측하기 위해 TSMC 90nm CMOS 1P9M 공정에서 150MHz로 동작 시켰으며, 약 13,600 gates로 합성되었다[5]. 표 4는 엔트로피 코딩 유닛들의 성능을 비교한 것이다. 표에서 확인 할 수 있듯이 본 논문의 엔트로피 코딩 유닛은 상대적으로 적은 리소스를 필요로 하며, 성능에서도 뒤처지지 않음을 알 수 있다. 또한 Exp-Golomb (EXPG)와 CAVLC를 모두 만족한다.

IV. 결론

본 논문에서 제안하는 엔트로피 코딩 유닛은 모바일 기기에 적합하도록 저비용 설계를 위해 노력하였다. 첫째, 자원의 많은 부분을 차지하는 bit-stream 생성기에서 배럴 쉬프터를 사용하지 않고 2 스테이지로 구성된 작은 쉬프터를 사용하였다. 둘째, Exp-Golomb 연산 시에 하나의 유닛에서 5가지 코드 타입을 처리 하였으며, 두 개의 덧셈기만으로 Exp-Golomb 연산을 만족시켰다. 셋째, CAVLC 유닛에서 사용되는 테이블들을 기존 대비 40%로 줄였다. 본 논문에서 제안하는 엔트로피 코딩 유닛은 TSMC 90nm 공정에서 150 MHz로 동작하며, 약 13,600 gates로 설계되어 모바일 기기사용에 적합함을 보여준다.

References

[1] Joint Video Team of ITU-T and ISO/IEC JTC 1, “Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification (ITU-T Rec. H.264 | ISO/IEC 14496-10 AVC),” *Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG*, JVT-G050, March 2003.

[2] Feng-Min Huang; Lei, S. -F, "High performance and low cost entropy encoder for H.264 AVC baseline entropy coding," *Communications, Circuits and Systems*, 2008. ICCCAS 2008. International Conference on , vol., no., pp.675,678, 25-27 May 2008

[3] J. G. Woo, W. J. Lee, J. S. Kim, “VLSI Design of H.264/AVC CAVLC encoder for HDTV Application,” Journal of IEEK SD, vol. 47, no. 7, pp. 45-53, 2007.

[4] D. J. Lee, Y. J. Jeong, “VLSI architecture design of CAVLC entropy encoder/decoder for H.264/AVC,” Journal of KICS, vol. 30, no. 5C, pp. 371-381, May, 2005.

[5] Synopsys Corp., "Synopsys 90nm Generic Library for Teaching IC Design,“ http://www.synopsys.com/Community/UniversityProgram/Pages/Library.aspx, 2013.

BIOGRAPHY

Sehyun Song (Student Member)

2006 : BS degree in Electrical and Computer Engineering, University of Seoul.
2008 : MS degree in Electrical and Computer Engineering, University of Seoul.
2008~ : Working toward the PhD degree in Electrical and Computer Engineering, University of Seoul.

Kichul Kim (Member)

1982 : BS degree in Electrical Engineering, Seoul National University.
1984 : MS degree in Electrical Engineering, Seoul National University.
1991 : PhD degree in Electrical Engineering, University of Southern California.
1984~1994 : Senior Research Engineer, Electronics and Telecommunications Research Institute (ETRI).
1994~ : Professor, in Electrical and Computer Engineering, University of Seoul.