논문 2010-05-10

H.264/AVC 동영상 압축 표준에서 Coeff_token 부호화를 위한 효율적임 메모리 구조 설계

(Design of Efficient Memory Architecture for Coeff_Token Encoding in H.264/AVC Video Coding Standard)

문 용 호*, 박 경 춘, 하 석 운
(Yong Ho Moon, Kyoung Choon Park, Seok Wun Ha)

Abstract : In this paper, we propose an efficient memory architecture for coeff_token encoding in H.264/AVC standard. The VLCTs used to encode the coeff_token syntax element are implemented with the memory. In general, the size of memory must be reduced because it affects the cost and operation speed of the system. Based on the analysis for the codewords in VLCTs, new memory architecture is designed in this paper. The proposed memory architecture results in about 24% memory saving, compared to the conventional memory architecture.

Keywords : H.264/AVC, Coeff_token, Encoder, Memory, Variable-length code table

1. 서 론

오늘날 다양한 멀티미디어 응용 서비스들이 출현, 제공되고 있다. 이러한 서비스들은 동영상 압축 기술에 기반하여 개발되어 왔으며 보다 더 향상된 서비스 제공을 위해서는 우수한 성능을 지닌 압축 표준의 개발이 요구된다. 2003년 5월에 새롭게 제정된 H.264/AVC 동영상 압축 표준은 기존 표준들에 비하여 뛰어난 성능을 지니고 있으며 현재 다양한 멀티미디어 응용 분야에 적용되고 있다[1].

H.264/AVC에서는 보다 향상된 압축 성능을 제공하기 위하여 다양한 부호화 기법들이 새롭게 채택되었다. 그 중에서도 CAVLC (Context-based Adaptive Variable Length Coding) 방식은 PMP, PDA와 같은 이동 동영상 단말 서비스 및 화상 회의를 주 응용분야로 하는 Baseline 프로파일에서 양자화된 정수 변환 계수들을 부호화하기 위하여 도입되었다. CAVLC 부호화 방식에서는 4x4 잔여 블록에서 얻어지는 양자화된 변환 계수들을 Coeff_token, Sign of T1s, Level, Total_zeros, Run_before라는 5가지 구문 요소들로 표현한 후 각각의 구문요소들을 압축한다. 이때 Coeff_token, Total_zeros, Run_before 구문요소들은 미리 정의된 가변길이 부호어 테이블(VLCT)들에 기반하여 부호화가 수행된다. 따라서 CAVCL 부호화 방식을 실제 구현하기 위해서는 VLCT들을 저장하기 위한 메모리 공간이 요구된다.

일반적으로 시스템의 설계 시 요구되는 메모리는 하드웨어의 비용을 증가시키고 시스템의 동작 속도를 감소시키는 요인 중 하나이다. 특히, 동영상 부호화를 소프트웨어로 구현할 경우에 있어서 캐쉬 메모리의 효과를 극대화하기 위해서는 VLCT에 대한 메모리 크기를 최대한 감소시키는 것이 필요하다. 이러한 이유로 효과적인 시스템 구현을 위해서는 메모리의 참조 횟수를 줄이는 동시에 메모리 공간을 최소화하는 것이 필요하다. Coeff_token 구문요소의 경우 1회의 메모리 참조만으로 부호화가 이루어진다. 따라서 VLCT의 구현에 있어서 최소의 메모리 공간을 차지하는 효과적인 메모리 구조가 설계되어야 한다. 그런데 지금까지 CAVLC 부호화부의 구현에 관한 연구가 활발히 수행되지 못하였

* 교신저자(Corresponding Author)
논문접수 : 2010. 02. 01., 수정일 : 2010. 03. 16.,
채택확정 : 2010. 05. 10.
문용호, 하석운: 경상대학교 정보과학과
박경춘: (주)에어로매스터
※ 본 연구는 중소기업청에서 주관하는 구매조건부 신제품개발사업(과제번호:S1065385)의 연구비 지원에 의하여 수행되었음.

다. 현재까지 CAVLC 복호화부의 구현에 있어서 하드웨어의 복잡성 감소[2], 저 전력 복원[3,4], 효율적인 메모리 참조[5], 고속 처리[6,7]등에 관한 연구들이 수행되어 왔다. 따라서 Coeff_token 부호화에서 요구되는 메모리를 최소화하기 위한 연구는 진행되지 못하였다.

본 논문에서는 Coeff_token 구문요소의 부호화 시에 요구되는 VLCT를 효과적으로 저장할 수 있는 새로운 메모리 구조를 제안한다. VLCT내의 부호어들에 대한 특징 분석을 통하여 보다 효율적인 메모리 구조를 설계한다. 본 논문에서 설계된 메모리 구조는 기존 메모리 구조에 비하여 24% 정도의 메모리가 절약되는 장점을 지닌다.

II. Coeff_token 부호화

H.264/AVC Baseline 프로파일에서 4x4 블록에 서의 양자화된 변환 계수들은 지그재그 순으로 재배열된 후 5개의 구문 요소들에 의해 부호화된다.

-Coeff_token: '0'이 아닌 계수의 총 개수(Tc)와 고주파 성분에서 크기가 1인 계수의 수(T1s)

-Sign of T1s: 역 지그재그 순서에서의 T1s 부호

-Level: T1s을 제외한 '0'이 아닌 계수의 크기

-Total_zeros: 지그재그 순서에서 DC와 '0'이 아닌 마지막 계수간의 존재하는 크기가 '0'인 계수들의 총 개수

-Run-before: 역 지그재그 순으로 각각의 '0'이 아닌 계수 앞에 존재하는 '0'인 계수들의 수

Coeff_token 구문요소를 부호화하기 위해서는 1개의 고정길이 부호어 테이블과 3개의 VLCT가 사용된다. 표 1은 Coeff_token 구문요소에 대한 VLCT들로서 테이블의 선택은 이전에 복원된 이웃 블록의 Tc값으로부터 이루어진다. 표 1에서 부호어가 특정 Tc와 T1s의 쌍에 대응됨을 알 수 있다. 일반적으로 부호어는 불규칙하고 복잡한 구조를 지니고 있기 때문에 VLCT는 메모리에 의하여 구현된다.

표 1의 VLCT에 존재하는 부호어들은 1비트에서 16 비트까지의 비트열들로 구성된다. 그리고 각 비트열은 1에서 15까지의 값을 나타낸다. 따라서 표 1의 부호어들을 Coeff_token 부호화에 사용하기 위해서는 부호어의 비트열 길이(LEN)를 저장하는 메모리(LEN_M)와 비트열의 값(VAL)을 저장하는 메모리(VAL_M)가 필요하다[8].

표 1. Coeff_token에 대한 3개의 VLCT.
Table 1. Three VLCTs for coeff_token element.

Tc \ T1s	0	1	2	3
0	1	-	-	-
1	000101	01	-	-
2	00000111	000100	001	-
3	000000111	00000110	0000101	00011
,,,,	,,,,	,,,,	,,,,	,,,,
16	0000000000000100	0000000000000110	0000000000000101	0000000000001000

(a) VLCT0

Tc \ T1s	0	1	2	3
0	11	-	-	-
1	001011	10	-	-
2	000111	00111	011	-
3	0000111	001010	001001	0101
,,,,	,,,,	,,,,	,,,,	,,,,
16	0000000000000111	0000000000000110	0000000000000101	0000000000000100

(b) VLCT1

Tc \ T1s	0	1	2	3
0	1111	-	-	-
1	001111	1110	-	-
2	001011	01111	1101	-
3	001000	01100	01110	1100
,,,,	,,,,	,,,,	,,,,	,,,,
16	0000000001	00000000100	0000000011	0000000010

(c) VLCT2

그리고 LEN과 VAL는 각각 4비트씩의 메모리 공간을 차지한다. 따라서 표 1의 VLCT들을 메모리에 저장하기 위해서는 테이블 당 62개의 부호어에 대하여 LEN과 VAL 각각에 0.5바이트가 할당되어

총 186(=62x 0.5x3+ 62x0.5x3)바이트가 필요하다.

메모리에 저장된 LEN과 VAL을 참조하기 위한 메모리 주소(add)는 Tc와 T1s를 이용하여 식 (1)과 같이 정의된다.

표 2. VLCT0에 대한 기존 메모리 구조.
Table 2. Conventional memory architecture of VLCT0.

add	LEN_M		VAL_M	
	idx=1	idx=0	idx=1	idx=0
0	2	1	1	1
1	5	3	3	1
2	6	6	4	5
,,,,	,,,,	,,,,	,,,,	,,,,
30	13	13	4	6

$$add = codenum >> 1$$
$$where,\ codenum = (1-j)\cdot cn + j\cdot (cn - 1 - (cn >> 6) << 1) \quad (1)$$
$$cn = (Tc << 2) - 3 \cdot T1s$$
$$j = (cn + 4) >> 6$$
$$idx = codenum \% 2 \quad (2)$$

표 2는 VLCT0에 대한 기존 메모리 구조를 보여준다. 표 2에서 idx는 8비트 메모리내 상, 하위 4비트를 지시하는 것으로 식(2)에 의하여 손쉽게 얻어진다.

표 2에서 알 수 있듯이 1개의 메모리 주소는 2개의 부호어를 지시하며 idx에 의하여 최종 LEN과 VAL가 결정된다. 표 2와 같은 메모리 구조에 기초한 기존 Coeff_token 부호화는 다음과 같은 과정을 거쳐 수행된다.

-1단계:입력된 Tc와 T1s값으로부터 add와 idx를 계산한다.

-2단계:add를 이용하여 LEN_M, VAL_M을 참조하여 부호어의 LEN와 VAL을 읽어온다.

-3단계:idx값에 따라 최종 LEN와 VAL를 결정한 후 이들을 비트열 구성부로 전송한다.

이해를 돕기 위하여 VLCT0에서 Tc=2, T1s=1인 경우를 살펴보자. 이 경우, 식(1)과 식(2)에 의하여 cn=5, j=0, add=2, idx=1이다. 따라서 표 2의 메모리 구조로부터 LEN=6, VAL=4가 얻어진다. 비트열 구성부는 얻어진 LEN와 VAL에 따라 표 1의 부호어 즉, "000100"비트열을 출력한다. 그림 2는 기존 메모리 구조에 기초한 Coeff_token 부호화부의 구조도를 나타낸다[8].

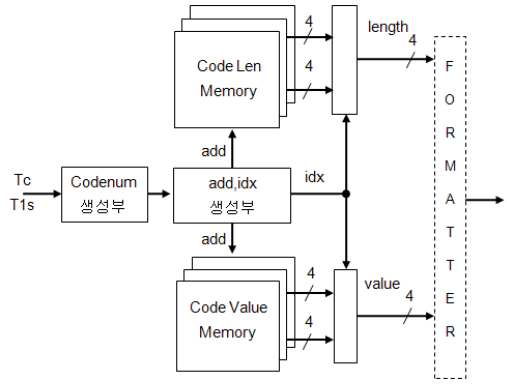

그림. 1 기존 메모리 구조에 기초한 Coeff_token 부호화 구조도

Fig. 1 Architecture of coeff_token encoding based on the conventional memory architecture

III. 새로운 메모리 구조 및 Coeff_token 부호화 방식

1. 새로운 메모리 구조 설계

1.1 기존 VLCT의 특징 분석

앞 절에서 알아본 바와 같이 Coeff_token 부호화에서는 LEN_M과 VAL_M이라는 2가지 메모리를 참조하는 것이 핵심이다. 이때 메모리 구조 및 참조는 본질적으로 Tc와 T1s에 의하여 결정된다. 그러므로 효율적인 새로운 메모리 구조를 설계하기 위해서는 Tc, T1s, 그리고 기존 부호어간에 어떤 관계가 존재하는 지를 분석하는 것이 필수적이다.

식(1)에서 알 수 있듯이 Codenum은 Tc와 T1s로부터 생성되는 파라메터로서 표 1의 부호어들을 1차원으로 나열한다. Codenum을 적용하여 LEN과 VAL을 각각 1차원으로 나열해 보면 LEN과 Codenum간에 상관관계가 존재함을 알 수 있다. 표 3은 이를 보다 더 명확하게 파악하기 위해서 LEN을 Codenum에 따라 정리한 것이다.

표 3으로부터 Codenum이 증가할수록 LEN 역시 증가함을 쉽게 확인할 수 있다. 그리고 8의 배수인 Codenum들 사이에 존재하는 LEN값들의 경우, 그 크기가 완만하게 변화함을 볼 수 있다. 한 예로

Codenum=8~15사이의 구간에 존재하는 VLCT0의 LEN값은 7~9의 범위를 지닌다. 만약 7을 이 구간에서의 기본값으로 설정한다면 구간 내 LEN은 0,1, 2의 값들 중 하나로 표현될 수 있다. 이 같은 사실은 2비트만으로 LEN을 표현할 수 있으며 아울러 메모리 공간이 감소될 수 있음을 의미한다. 표 3에서 괄호안의 값은 주어진 구간에 대한 기본값을 나타낸다. 표 3의 기본값에 대하여 대응하는 모든 LEN이 0~3의 값을 지님을 알 수 있다. 이러한 사실을 활용한다면 효율적인 메모리 구조가 새롭게 설계될 수 있을 것이다. 그러나 이것은 기본값이 주어져 있다는 가정에서 성립되어진다. 따라서 기본값을 어떻게 얻을 것인가가 해결되어야 한다.

1.2 메모리 구조 설계

앞의 설명에서 알 수 있듯이 Tc와 T1s는 Coeff_token 부호화에서 있어서 유일하게 활용할 수 있는 독립 변수이다. 따라서 기본값 역시 이들 변수에 의하여 생성되어야 한다. 표 3에서 기본값들과 이에 대응하는 Codenum들을 살펴보면 Code num이 증가할수록 기본값이 1 혹은 2만큼 증가함을 알 수 있다. 이것은 기본값을 Codenum의 함수로 모델링함으로써 손쉽게 얻을 수 있다는 것을 의미한다.

본 논문에서는 다음의 수식들과 같이 기본값을 모델링한다.

$$sel = (codenum < 8)\ ?\ 0 : 1 \tag{3}$$

$$k1 = (sel = 1)\ ?\ codenum >> 3\ :\ codenum\%16 + 1 \tag{4}$$

$$k2 = (1 - sel) << 2 + ((k1 + 2) >> 2) \cdot sel \tag{5}$$

$$s = (k1 <= 4)\ ?\ 0 : 1 \tag{6}$$

표 4는 Codenum값에 대한 식(3)~식(6)의 변수에 대한 값들을 보여준다. 표 4를 참고하여 식(7)에서 얻어지는 base0와 표 3의 VLCT0에 존재하는 기본값을 비교하면 서로 동일함을 알 수 있다.

$$base0 = (sel = 0)\ ?\ a : b$$
$$where,\ a = (s = 0)\ ?\ k1 : k2$$
$$b = (s = 0)\ ?\ (k1 << 1) + 5 : k2 + 13 \tag{7}$$

한편 식(8)은 3개의 VLCT들에 대한 기본값의 생성을 수식화한 것이다.

$$base = \begin{cases} base0 & \text{for } VLCT0 \\ base0 - 3 \cdot sel & \text{for } VLCT1 \\ base0 - (4 + k2) \cdot sel & \text{for } VLCT2 \end{cases} \tag{8}$$

표 3. LEN과 Codenum간의 상관 관계.
Table 3. Relationship of LEN and codenum.

code num	VL CT0	VL CT1	VL CT2	code num	VL CT0	VL CT1	VL CT2
0	1	2	4	**32**	**13 (13)**	**11 (10)**	**8 (8)**
1	2	2	4	33	14	11	8
2	3	3	4	34	14	12	8
3	**5 (4)**	**4 (4)**	**4 (4)**	35	14	11	8
4	6	6	6	36	14	12	8
5	6	5	5	37	14	12	8
6	7	6	5	38	14	12	8
7	6	4	4	39	14	12	8
8	**8 (7)**	**6 (4)**	**6 (3)**	**40**	**14 (14)**	**12 (11)**	**9 (9)**
9	8	6	5	41	15	12	9
10	8	6	5	42	15	13	9
11	7	5	4	43	15	13	9
12	9	7	6	44	15	12	9
13	9	6	5	45	15	13	9
14	9	7	5	46	15	13	9
15	8	6	4	47	15	13	10
16	**10 (9)**	**8 (6)**	**7 (4)**	**48**	**15 (15)**	**13 (12)**	**9 (9)**
17	10	7	5	49	15	13	9
18	10	8	6	50	16	13	10
19	9	6	4	51	16	13	10
20	11	8	7	52	16	13	10
21	11	8	6	53	16	14	10
22	11	9	6	54	16	14	10
23	10	7	5	55	16	14	10
24	**13 (11)**	**9 (8)**	**7 (6)**	**56**	**16 (15)**	**13 (12)**	**10 (9)**
25	13	9	6	57	16	14	10
26	13	11	7	58	16	14	10
27	11	9	6	59	16	14	10
28	13	11	7	60	16	14	10
29	13	11	7	61	16	14	10
30	13	11	7				
31	13	11	7				

기존의 4비트 LEN은 식(8)의 의하여 생성되는 기본값을 이용하여 다음과 같이 표현될 수 있다.

$$LEN = base + offset \tag{9}$$

식(9)의 offset는 기존의 LEN과 기본값 간의 차이를 나타내며 새롭게 설계될 메모리에 저장되는 할 정보이다. 이때 각각의 offset에는 2비트의 저장 공간이 할당된다. 왜냐하면 offset은 0~3의 범위를 지니기 때문이다. 그런데 일반적으로 메모리는 8비트 단위로 저장, 참조된다. 따라서 4개의 offset들에 하나의 메모리 주소가 할당되어야 한다. 식(10)은 새로운 메모리 구조에서 필요한 메모리 주소 Add의 생성을 보여준다. 그리고 식(11)은 동일 메모리 주소 내에 존재하는 4개의 offset들을 식별하기 위한 지시자 Idx의 생성을 나타낸다.

표 4. Codenum에 대한 sel, k1, k2, s 변수값.
Table 4. Values of sel, k1, k2, and s according to the codenum.

code num	sel	k1	k2	s	code num	sel	k1	k2	s
0	0	1	4	0	32	1	4	1	0
1	0	2	4	0	33	1	4	1	0
2	0	3	4	0	34	1	4	1	0
3	0	4	4	0	35	1	4	1	0
4	0	5	4	1	36	1	4	1	0
5	0	6	4	1	37	1	4	1	0
6	0	7	4	1	38	1	4	1	0
7	0	8	4	1	39	1	4	1	0
8	1	1	0	0	40	1	5	1	1
9	1	1	0	0	41	1	5	1	1
10	1	1	0	0	42	1	5	1	1
11	1	1	0	0	43	1	5	1	1
12	1	1	0	0	44	1	5	1	1
13	1	1	0	0	45	1	5	1	1
14	1	1	0	0	46	1	5	1	1
15	1	1	0	0	47	1	5	1	1
16	1	2	1	0	48	1	6	2	1
17	1	2	1	0	49	1	6	2	1
18	1	2	1	0	50	1	6	2	1
19	1	2	1	0	51	1	6	2	1
20	1	2	1	0	52	1	6	2	1
21	1	2	1	0	53	1	6	2	1
22	1	2	1	0	54	1	6	2	1
23	1	2	1	0	55	1	6	2	1
24	1	3	1	0	56	1	7	2	1
25	1	3	1	0	57	1	7	2	1
26	1	3	1	0	58	1	7	2	1
27	1	3	1	0	59	1	7	2	1
28	1	3	1	0	60	1	7	2	1
29	1	3	1	0	61	1	7	2	1
30	1	3	1	0					
31	1	3	1	0					

$$Add = add >> 1 \quad (10)$$

$$Idx = codenum\%4 \quad (11)$$

본 논문에서는 식(10)과 식(11)에 기초하여 기존 LEN대신에 offset을 저장하는 새로운 메모리 구조를 제안한다. 표 5는 제안하는 메모리 구조이다. 표 5에서 알 수 있듯이 제안하는 메모리 구조는 48(=16x1x3)바이트의 메모리 공간을 차지한다.

표 5.제안하는 새로운 LEN_M 구조.
Table 5. The proposed LEN_M structure.

MEM / Idx / Add	VLCT0 offset				VLCT1 offset				VLCT2 offset			
	0	1	2	3	0	1	2	3	0	1	2	3
0	0	0	0	1	1	0	0	0	3	2	1	0
1	2	2	3	2	2	1	2	0	2	1	1	0
2	1	1	1	0	2	2	2	1	3	2	2	1
3	2	2	2	1	3	2	3	2	3	2	2	1
4	1	1	1	0	2	1	2	0	3	1	2	0
5	2	2	2	1	2	2	3	1	3	2	2	1
6	2	2	2	0	1	1	3	1	1	0	1	0
7	2	2	2	2	3	3	3	3	1	1	1	1
8	0	1	1	1	1	1	2	1	0	0	0	0
9	1	1	1	1	2	2	2	2	0	0	0	0
10	0	1	1	1	1	1	2	2	0	0	0	0
11	1	1	1	1	1	2	2	2	0	0	0	1
12	0	0	1	1	1	1	1	1	0	0	1	1
13	1	1	1	1	1	2	2	2	1	1	1	1
14	1	1	1	1	1	2	2	2	1	1	1	1
15	1	1	-	-	1	2	-	-	1	1	-	-

2. Coeff_token 부호화 방식

표 5의 메모리 구조를 기반으로 한 Coeff_token 부호화는 다음과 같이 수행된다.

-1단계: 입력된 Tc와 T1s값으로부터 Codenum, base, Add, add, Idx, 그리고 idx를 계산한다.

-2단계: add, idx를 이용하여 VAL_M을 참조하고 VAL 정보를 읽어온다.

-3단계: Add, Idx를 이용하여 제안하는 메모리로 부터 부호어의 offset 길이 정보를 읽어와 서식(10)과 같이 최종 LEN을 구한다.

-4단계: 얻어진 LEN과 VAL를 비트열 구성부로 전송한다.

이해를 돕기 위하여 VLCT1에서 Tc=5, T1s=2인 경우를 살펴보자. 이 경우, Codenum=14이고 식(10)과식(11)에 의하여 add=7, Add=3, Idx=2이다. 표 5의 메모리 구조로부터 offset=3이 얻어진다. 그리고 식(7)과 식(8)에 의하여 base0=7, base=4가 계산된다. 따라서 식(9)에 의하여 LEN=4+3=7이 얻어진다. 이것은 표 3에 표시된 대응하는 부호어의 길이와 일치한다. 그림 2는 제안하는 메모리 구조를 이용한 Coeff_token 부호화부의 구조도이다.

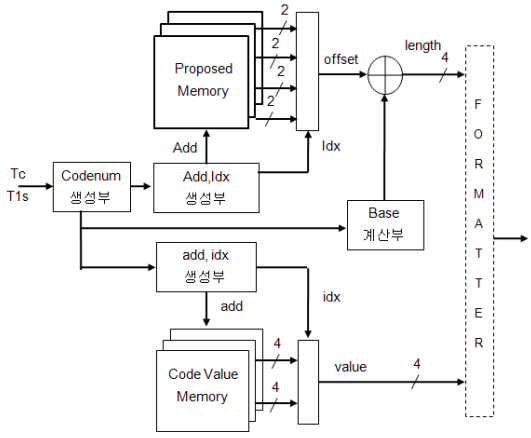

그림. 2 제안하는 메모리 구조에 기반한 Coeff_token 부호화 구조도

Fig. 2 Architecture of coeff_token encoding based on the proposed memory architecture

IV. 성능 분석

본 논문에서는 제안한 메모리 구조의 효과를 객관적으로 확인하기 위하여 Verilog HDL로 설계, 검증된 그림 1과 그림 2의 coeff_token부호화기를 0.13um공정에서 ARM Artisan의 Sage-X로 구현하였다. 표 6은 구현된 각 부호화기에 대한 면적과 Gate수를 정리한 것이다.

표에서 부호화기 전체 면적의 95%이상이 메모리가 차지함으로 알 수 있다. 그리고 제안한 메모리의 면적이 기존 부호화기의 메모리 면적에 비하여 약 24%정도 감소되었음을 보여준다. 이것은 제안한 메모리 구조의 효율성으로 인하여 메모리가 절약되었기 때문이다. 또한 전체 부호화기의 면적에 있어서도 제안한 부호화기의 면적이 22%정도 감소되었다. 이것은 제안하는 메모리 구조의 구동을 위하여 새롭게 도입된 식(3)~식(11)으로 인한 하드웨어의 오버헤드가 크지 않음을 의미한다.

표 6. 각 부호화기에서의 면적과 Gate 수

Table 6. Area and the number of gates for each coeff_token encoder.

	기존 부호화기	제안 부호화기
메모리 면적	75,656.5	57,418.0
전체 면적	77,282.6	60,172.8
총 Gate 수	15,177	11,817

V. 결 론

본 논문에서는 Coeff_token 구문요소의 부호화시에 요구되는 VLCT를 효과적으로 저장할 수 있는 효율적인 메모리 구조를 제안하였다. VLCT내의 부호어들에 대한 특징 분석을 통하여 기존 부호어의 길이 정보를 이론적으로 모델링하고 이를 기반으로 새로운 메모리 구조를 설계하였다. 설계된 메모리 구조는 기존 메모리 구조에 비하여 24% 정도의 메모리가 절약되는 장점을 지닌다. 이것은 CAVLC 부호화에서 사용하는 전체 메모리(=365바이트)에 대하여 약 %12의 메모리 감소 이득에 해당한다.

참 고 문 헌

[1] T. Wiegand, G. J. Sullivan, G. Bjøntegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," IEEE Trans. Circuits and Syst. Video Technol., Vol. 13, No.7, pp. 560-576, July, 2003.

[2] H. Chang, C. Lin, and J. Guo, "A novel low-cost high-performance VLSI architecture for MPEG-4 AVC/H.264 CAVLC decoding", IEEE International Symposium on Circuits and Systems, pp. 6110-6113, May, 2005.

[3] H. Huang and C. Fan, "Architecture design of low-power and low-cost CAVLC decoder for H.264/AVC", IEEE Asia Pacific Conference on Circuits and Systems, pp. 1336-1339, 2008.

[4] H.Y. Lin, Y. H. Lu, B.D. Liu and J.F. Yang, "Low power design of H.264 CAVLC

Decoder", IEEE International Symposium on Circuits and Systems, pp. 2689-2692, 2006.
[5] Y. H. Moon, "An advanced total_zeros decoding method based on new memory architecture in block. H.264/AVC CAVLC", IEEE Trans. on Circuits and Systems for Video Technology. Vol. 18, No.9, pp. 1312-1317, Sep. 2008.
[6] G. G. Lee et al, "Low complexity and high throughput VLSI architecture for AVC/H.264 CAVLC decoding", IEEE International Symposium on Circuits and Systems, pp. 1229-1232, May, 2009.
[7] 문 전학, 이 성수, "효율적인 H.264/AVC 엔트로피 복호기 설계", 전자공학회논문지, 제44권, SD편12호, pp. 1147-1152, 2007.
[8] 이 대준, 정 용진 "H.264/AVC를 위한 CAVLC 엔트로피 부/복호화기의 VLSI설계", 한국통신학회논문지, pp. 371-381, 30권, 5C호, 2005.

저 자 소 개

문 용 호(Yong Ho Moon)

1992년 2월 : 부산대학교 전자공학과 학사.
1994년 2월 : 부산대학교 전자공학과 석사.
1998년 8월 : 부산대학교 전자공학과 박사.
1998년 8월~2001년 8월 : 삼성전자 DM 연구소 책임연구원.
현재, 경상대학교 정보과학과 조교수.
관심분야 : 영상 처리, 동영상 압축, 임베디드 시스템, SoC.
Email : yhmoon5@gnu.ac.kr

박 경 춘(Kyoung Choon Park)

1998년 : 동서대학교 환경공학과 학사.
현재, (주)에어로매스터 S/W 개발팀 차장.
관심분야 : 임베디드 소프트웨어, Avionics.
Email : gilsion@amc21.co.kr

하 석 운(Seok Wun Ha)

1979년 2월 : 부산대학교 전자공학과 학사.
1985년 2월 : 부산대학교 전자공학과 석사.
1995년 2월 : 부산대학교 전자공학과 박사.
2002년 : UC Riverside 방문교수
현재, 경상대학교 정보과학과 교수.
관심분야 : 신호처리, 임베디드 시스템, 컴퓨터 비젼.
Email : swha@gnu.ac.kr