논문 2008-45SP-5-5

효율적인 H.264/AVC CAVLC 복호화기 구현을 위한 고속 Coeff_token 복원 방식

(A Fast Coeff_token Decoding Method for Efficient Implimentation of H.264/AVC CAVLC Decoder)

문 용 호*, 박 태 희**

(Yong-Ho Moon and Tae-Hee Park)

요 약

본 논문에서는 기존 VLCT의 재구성에 기반한 고속 Coeff_token 복호화 방식을 제안한다. 기존의 개선 방식은 여전히 많은 메모리 액세스를 필요로 하기 때문에 고속 복원 및 저 전력 소비를 요하는 PMP, DMB, DVB-H와 같은 멀티미디어 서비스에 적합하지 않다. 본 논문에서는 Coeff_token 복원에 사용되는 기존 부호어들의 구조 분석을 통하여 새로운 부호어 구조를 정의하고 이들을 효율적으로 저장하는 메모리 구조를 제시한다. 모의실험은 제안 방식이 기존 방식보다 최소 10%에서 최대 57% 정도 메모리 액세스 이득을 지니고 있음을 보인다. 이것은 제안 방식이 복원 영상의 성능과 화질의 저하없이 복원시에 요구되는 저 전력 소비 및 고속 처리 제약을 해결할 수 있음을 의미한다.

Abstract

In this paper, we propose a fast coeff_token decoding method based on the re-constructed VLCT. Since the conventional decoding method is still based on large memory accesses, it is not suitable for the multimedia services such as PMP, DMB, DVB-H where fast decoding and low power consumption are required. Based on the analysis for the codeword structure, new structure of the codeword and the corresponding memory architecture are developed in this paper. The simulation results show that the proposed algorithm achieves memory access saving from 10% to 57%, compared to the conventional decoding method. This means that the issues of low power consumption and high speed decoding can be resolved without video quality and coding efficiency degradation

Keywords: Coeff_token, CAVLC, H.264/AVC, VLCT

I. 서 론

2005년 ITU-T/ISO/IEC Joint Video Team에서 발표한 H.264/AVC 동영상 압축 표준[1]은 MPEG-2 및 MPEG-4를 대체하는 새로운 표준으로서 세계 각국에서 신 개념 멀티미디어 서비스의 기본 기술로서 채택하고 있다. 그러나 H.264/AVC 표준에서 새롭게 도입된 다양한 기능들은 부호화뿐만 아니라 복호화에서도 상당한 계산량 증가를 가져왔다[2,3]. 이러한 복호화기의 과도한 계산량은 H.264/AVC 기반의 멀티미디어 서비스 및 제품 개발에 큰 문제점이 되고 있다. 따라서 최근에 H.264/AVC 복호화기의 효율적인 구현에 대한 관심이 크게 증가하고 있다.

H.264/AVC Baseline Profile은 PMP, PDA, DVB-H와 같은 이동 동영상 단말 서비스를 주 응용 분야로 하고 있다. H.264/AVC Baseline Profile에서 양자화 된 DCT 계수들은 CAVLC(Context-based Adaptive Variable Length Coding) 복원 방식에 의하여 복호화된다. CAVLC 복호화부에서는 입력 비트열로부터 Coeff_token, Sign of T1s, Level, Total_zeros, Run_

* 정회원, 경상대학교 컴퓨터과학부 정보과학전공
(Dept. of Informatics, GyeongSang Nat'l Univ.)
** 정회원, 동명대학교 메카트로닉스공학과
(Dept. of Mechatronics Eng., TongMyong Univ.)
접수일자: 2008년2월1일, 수정완료일: 2008년7월31일

before 구문 요소(syntax element)들이 순차적으로 복원됨으로써 4x4 블록의 양자화된 DCT 계수들이 구해진다. 그런데 이때 소모되는 계산량은 전체 복호화 계산량의 30% 정도를 차지한다고 알려져 있다. 따라서 경쟁력 있는 H.264/AVC Baseline Profile 복호화기 개발을 위해서는 CAVLC 복원부의 효율적인 구현이 무엇보다 중요하다.

지금까지 하드웨어 비용과 파워 소모를 감소시킬 수 있는 몇몇 구현 기술들을 이용하여 CAVLC 복호화기를 VLSI로 설계하는 연구들이 수행되었다[4-6]. 그러나 이 연구들에서는 H.264/AVC 표준에서 제시된 가변길이 부호어 테이블(VLCT)을 기반으로 한 복원 방식이 사용되기 때문에 많은 메모리 엑세스가 요구된다. 최근에 FSM(Finite State Machine) 기법을 이용하여 VLCT를 이용하지 않고 Run_before 요소를 복원하는 새로운 복호화 방식이 제안되었다[7]. 그리고 1회의 메모리 엑세스만으로 Total_zeros구문 요소를 복원하는 고속 Total_zeros 복원 방식이 개발되었다[8]. 또한 산술 복원과 새로운 VLCT를 이용한 복원 방식으로 구성된 개선된 Coeff_token 복원 방식도 제안되었다[9]. 이러한 연구들은 CAVLC 복호화기의 성능 향상에 크게 기여하였다.

그러나 개선된 Coeff_token 방식[9]은 여전히 VLCT 기반의 가변길이 복원을 수행하고 있다. 일반적으로, VLCT를 이용한 복원은 비트열과 일치하는 부호어를 발견할 때까지 재귀적으로 메모리를 엑세스한다. 그러므로 VLCT 기반의 복원화기에서는 다수의 메모리 엑세스가 발생한다. 메모리 엑세스가 시스템의 파워 소모 및 동작 속도 감소에 큰 영향을 끼친다는 것은 잘 알려진 사실이다. 특히, 비디오폰이나 DMB와 같은 멀티미디어 서비스에 있어서 이 같은 요소들은 매우 중요한 고려 사항이다. 따라서 메모리 엑세스를 보다 더 감소시키기 위하여 향상된 Coeff_token 복원 방식의 개발이 필요하다. 본 논문에서는 기존 VLCT의 재구성을 통한 고속 Coeff_token 복호화 방식을 제안한다. Coeff_token 복원에 사용되는 기존 부호어들의 특징 분석을 통하여 새로운 부호어 구조를 정의하고 이들을 효율적으로 저장하는 메모리 구조를 제시한다. 모의실험은 개선된 복원 방식[9]에 비하여 제안 방식에서 최대 57%의 메모리 엑세스 감소가 이루어짐을 보여준다.

II. 기존 Coeff_token 복원 방식

1. 표준 Coeff_token 복원 방식[1]

H.264/AVC Baseline Profile에서 4x4 블록에 대한 양자화된 계수들은 지그재그 순으로 재배열된 후 5가의 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 계수간에 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 구문 요소에 대한 VLCT들로서 테이블의 선택은 이전에 복원된 이웃 블록의 Tc값으로부터 이루어진다. 표 1에서 부호어가 특정 Tc와 T1s의 쌍에 대응됨을 알 수 있다. 일반적으로 부호어는 불규칙적이고 복잡한 구조를 지니고 있기 때문에 VLCT는 메모리에 의하여 구현된다.

2. 개선된 Coeff_token 복원 방식[9]

VLCT를 이용한 복원 방식에서 메모리 엑세스 횟수를 감소시키기 위해서는 일반적으로 다음 사항들에 대한 해결책이 제시되어야 한다.

- 메모리 엑세스에 의한 복원을 어떻게 산술 연산으로 대체할 것인가 ?
- 메모리 엑세스에 의한 복원에 있어서 메모리에 저장할 부호어를 어떻게 구성할 것인가 ?
- 메모리 엑세스에 의한 복원에 있어서 어떻게 부호어를 메모리에 효과적으로 할당할 것인가 ?

표 1의 VLCT에 대한 면밀한 분석은 다음과 같은 특징들을 제시하고 있다.

1) 표 1의 부호어는 기본적으로 EG-Code 구조를 지니고 있다. 최초 비트 1이 나타나기 전까지 존재하는 비트 0의 개수를 nZ, 최초 비트 1이후에 존재하는 정보 비트열을 InFo라 할 때, 표 1에서 nZ는 0~14, InFo는 0~3의 범위를 지닌다.
2) nZ가 0 또는 1인 부호어들의 Tc와 T1s는 nZ와 InFo을 이용하여 산술적으로 표현할 수 있다.

표 1. Coeff_token 복원에 사용되는 3개의 VLCT.
Table 1. Three VLCTs used for Coeff_token decoding.

(a) VLCT0 (0≤nC<2)

Tc \ T1s	0	1	2	3
0	1		-	-
1	000101	01	-	-
2	00000111	000100	001	-
3	000000111	00000110	0000101	00011
4	0000000111	000000110	00000101	000011
5	00000000111	000000110	000000101	0000100
,,,,	,,,,	,,,,	,,,,	,,,,
15	0000000000000111	0000000000001010	0000000000001001	0000000000001100
16	0000000000000100	0000000000000110	0000000000000101	0000000000001000

(b) VLCT1 (2≤nC<4)

Tc \ T1s	0	1	2	3
0	11	-	-	-
1	001011	10	-	-
2	000111	00111	011	-
3	0000111	001010	001001	0101
4	00000111	000110	000101	0100
5	00000100	0000110	0000101	00110
,,,,	,,,,	,,,,	,,,,	,,,,
15	0000000000001001	0000000000001000	0000000000001010	0000000000000001
16	0000000000000111	0000000000000110	0000000000000101	0000000000000100

(c) VLCT2 (4≤nC<8)

Tc \ T1s	0	1	2	3
0	11		-	-
1	001011	10	-	-
2	000111	00111	011	-
3	0000111	001010	001001	0101
4	00000111	000110	000101	0100
5	00000100	0000110	0000101	00110
,,,,	,,,,	,,,,	,,,,	,,,,
15	0000000000001001	0000000000001000	0000000000001010	0000000000000001
16	0000000000000111	0000000000000110	0000000000000101	0000000000000100

3) 좌 상단에서 우 하단으로 갈수록 부호어의 비트 길이는 길어진다.
4) 표 1의 부호어들은 동일한 nZ를 지니는 부호어들로 분류할 수 있다. 따라서 nZ가 주어질 경우, 복호화 과정에서 요구되는 검색 과정은 동일한 nZ를 지니는 특정 그룹의 부호어들로 제한할 수 있다.
5) 동일 그룹내의 부호어들은 InFo에 의해서 식별되어진다. 그러므로 nZ가 구해진 경우, 기존 검색 과정은 비트 열과 일치하는 부호어의 InFo를 구하는 것과 동일하다.

이 같은 특징들을 기초로 개선된 Coeff_token복원 방식에서는 앞의 3가지 사항들에 대한 해결책을 제시하고 있다. 특징 1)과 2)는 표 1에서 밑줄로 표시된 부호어들의 Tc와 T1s가 아래의 수식으로 표현될 수 있음을 보여준다.

$$\begin{aligned} T1s &= nZ && for\ VLCT\ 0 \\ &= nZ << 1 + (1 - InFo[2]) && for\ VLCT\ 1 \\ &= 3 - InFo[2] \cdot InFo[1:0] && for\ VLCT\ 2 \end{aligned} \quad (1)$$

$$\begin{aligned} Tc &= T1s && for\ VLCT0 \\ &= T1s + nZ \cdot (1 - InFo[0]) \cdot (1 - InFo[1]) \\ & && for\ VLCT1 \\ &= T1s + (1 - InFo[2]) << 2 - InFo[1:0] \cdot \\ & \quad (1 - InFo[2]) && for\ VLCT2 \end{aligned} \quad (2)$$

개선된 복원 방식에서는 식 (1)과 식 (2)를 이용하여 산술 복원이 수행된다.

산술 복원이 불가능한 부호어들에 대하여 개선된 복원 방식에서는 새로운 부호어 구조를 정의하고 이를 효율적으로 저장하는 메모리 구조를 설계하여 메모리 엑세스 횟수를 감소시켰다. 개선된 복원 방식에서는 특징 1)과 3)를 토대로 (Tc, T1s)쌍과 codenum간의 대응 관계를 유도하였다. 식 (3)과 식(4)는 이를 나타낸다.

$$codenum = (Tc << 2) - 3 \cdot T1s \quad (3)$$

$$\begin{aligned} T1s &= codenum\ \%4 \\ Tc &= (codenum\ + 3 \cdot T1s) >> 2 \end{aligned} \quad (4)$$

그리고 이를 바탕으로 codenum을 특정 부호어의 메모리 주소로 설정하였다. 식 (4)로부터 Tc와 T1s가 메모리 주소로부터 쉽게 얻어짐을 알 수 있다.

표 2. TP0와 TP1의 정의.
Table 2. The definitions of TP0 and TP1.

*InFo*의 길이	TP1	TP0	TP1의 의미
0	11	-	*InFo*가 존재하지 않음
1	00	01	부호어의 *InFo*가 동일
2		10	
3		11	
1/2	01	00/01	부호어의 *InFo* 길이가 1 또는 2
2/3	10	00/01	부호어의 *InFo* 길이가 2 또는 3

또한 개선된 복원 방식에서는 앞에서 서술한 특징 3), 4)와 수식 (3), (4)를 기초로 하여 기존 VLCT를 MBASE와 MCODE라는 테이블로 재구성하고 새로운 모리 구조를 제시하였다. 각 그룹에 대하여 [iniNum] [TP1]의 형식으로 정의된 새로운 유형의 데이터가 MBASE에 저장된다. 여기서 6비트로 구성된 iniNum은 특정 그룹에 포함되는 부호어들에 대한 초기 MCODE 주소를 나타낸다. 그리고 TP1은 InFo의 기본 유형 정보를 제공하는 것으로 그 의미와 정의는 표 2와 같다. 표 2에서 TP1=11인 경우는 InFo가 존재하지 않은 경우를 의미하는 것으로 MCODE를 엑세스하지 않고 식(4)로부터 Tc와 T1s가 결정된다. 특징 5)에서 언급한 바와 같이 특정 그룹내에서는 기존 탐색 과정을 InFo 비교로 대체할 수 있다. 이러한 이유로 MCODE는 [TP0][InFo] [INC]의 형식으로 정의된 부호어들로 구성된다. 여기서 3 비트 길이의 INC는 다음에 검색할 메모리 주소에 대한 정보를 지니고 있으며 2 비트의 TP0는 TP1과 함께 InFo의 길이에 대한 정보를 제공한다. 표 2는 TP0와 TP1의 의미를 보여주고 있다.

개선된 Coeff_token의 복원 방식의 구체적인 복원 과정은 다음과 같다. 아래 과정들은 개선된 복원 방식에서 불필요한 메모리 엑세스가 제거됨을 잘 보여준다.

Step 1) 입력되는 비트열 버퍼로부터 nZ를 결정한다.
Step 2) 식 (1)과 식 (2)를 이용하여 산술 복원을 수행하고 이것이 불가능할 경우 nZ을 주소로 하여 MBASE를 엑세스한다.
Step 3) TP1이 11인 경우 식(4)를 이용하여 T1s와 Tc를 구한다.
Step 4) TP1이 11이 아닐 경우, iniNum을 주소로 하여 MCODE를 엑세스한다.
Step 5) MCODE의 INC, TP0을 이용하여 특정 그룹 내에서 비트열과 일치하는 InFo를 탐색하고 그 때의 메모리 주소를 구한다.
Step 6) 메모리 주소를 식(4)에 대입하여 T1s와 Tc를 구한다.
Step 7) TP0와 TP1으로부터 입력 비트열 버퍼의 포인터를 보정한다.

그러나 개선된 Coeff_token 방식에서는 초기 MCODE 주소를 얻기 위하여 MBASE가 항상 엑세스되며 주어진 입력 비트열에 대하여 여전히 반복적인 검색이 수행되는 문제를 지니고 있다.

III. 제안하는 고속 Coeff_token 복원 방식

개선된 Coeff_token 복원 방식은 기존 표준 복원 방식에서 요구되는 메모리 엑세스 횟수를 크게 감소시켰다. 그러나 이 방식은 주어진 입력 비트열에 대하여 여전히 반복적인 검색을 수행하고 있으며 초기 MCODE 주소를 얻기 위하여 MBASE를 항상 엑세스하는 문제를 지니고 있다. 이 문제에 대한 가장 이상적인 해결책은 아마도 단 1회의 메모리 엑세스를 통하여 Tc와 T1s를 복원하는 것일 것이다. 그런데 이를 위해서는 Tc와 T1s가 메모리에 직접 저장되어져야 하고 또한 주어지는 입력 비트열로부터 메모리 주소가 직접 결정되어져야만 한다.

앞에서 언급한 특징 1)에 따르면 입력 비트열은 n_Z와 InFo로 표현할 수 있다. 따라서 n_Z와 InFo간의 관계를 살펴보는 것이 필요하다. 표 3은 표 1의 부호어들에 대하여 nZ값을 지니는 부호어의 총 개수와 InFo의 비트 길이를 정리한 것이다. 표 3으로부터 동일한 nZ를 지니는 보호어들에 있어서 n_Code와 L_InFo간에는 매우 큰 상관도가 존재한다는 것을 알 수 있다. 특히, 밑줄 친 n_Code와 L_InFo 값들 간에는 식 (5)의 관계를 유도할 수 있다.

$$n\_Code = 1 << (L\_InFo) \tag{5}$$

이때 L_InFo는 2 또는 3의 고정된 값을 지닌다.

이러한 사실들은 밑줄 친 n_Code와 L_InFo를 지니는 부호어의 경우 입력 비트열로부터 메모리 주소를 직접 결정할 수 있는 아이디어를 제공한다. 만약 n_Z와 InFo를 각각 기준 주소(Base Address)와 오프셋 주소(Offset Address) 로 간주한다면 대응하는 부호어의 Tc

표 3. 동일한 n_Z를 지니는 부호어 개수(n_Code)와 InFo의 비트 길이(L_InFo).

Table 3. The number of Codewords (n_Code) and the length of InFo (L_InFo) having the same n_Z.

n_Z	VLCT 0		VLCT 1		VLCT 2	
	n_Code	L_InFo	n_Code	L_InFo	n_Code	L_InFo
0	1	0	2	1	8	3
1	1	0	3	1-2	8	3
2	1	0	6	2-3	8	3
3	3	1-2	4	2	8	3
4	3	1-2	4	2	8	3
5	4	2	4	2	8	3
6	4	2	4	2	7	2-3
7	4	2	8	3	4	2
8	4	2	8	3	2	1
9	8	3	8	3	1	0
10	8	3	6	2-3	-	-
11	8	3	4	2	-	-
12	8	3	1	0	-	-
13	4	2	-	-	-	-
14	1	0	-	-	-	-

와 T1s는 주어진 메모리 주소에 따라 저장될 수 있다. 본 논문에서는 이러한 아이디어를 바탕으로 새로운 복호화 테이블을 제안한다. 표 4는 제안된 테이블의 메모리 구조를 보여준다. 표 4에서 n_Z-InFo 열은 메모리 구조에 대한 손쉬운 이해를 위하여 삽입되었다. 표 3에서 알 수 있듯이 밑줄이 없는 n_Code와 L_InFo에 대응하는 기존 부호어에 대해서는 제안하는 메모리 구조가 적용되지 않는다. 따라서 Coeff_token 복원에 있어서 제안하는 메모리 구조를 적용할 수 있는 지를 판단할 수 있는 기준이 필요하다. 제안하는 메모리 구조가 이용될 수 있는 최초의 n_Z값과 마지막 n_Z값을 F_nZ와 L_nZ라 할 때 아래의 수식들이 표 3으로부터 유도된다.

$$F\_nZ = 5 - vLc << 1$$
$$where \quad vLc = 0 \quad for\ VLCT\,0$$
$$= 1 \quad for\ VLCT\,1$$
$$= 2 \quad for\ VLCT\,2 \qquad (6)$$

$$L\_nZ = F\_nZ - (vLc + vLc >> 1) \qquad (7)$$

한편, 표 4로부터 n_Z가 주어진 경우 InFo 비트 길이는 동일함을 알 수 있다. 그러나 서로 다른 n_Z값에

표 4. 제안하는 메모리 구조.

Table 4. The proposed memory architecture.

Add	MEM 0		n_Z	MEM 1		n_Z	MEM 2		n_Z
	Tc	T1s	-InFo	Tc	T1s	-InFo	Tc	T1s	-InFo
0	6	3	5-00d	7	3	3-00d	5	1	1-000
1	4	2	5-01d	4	2	3-01d	5	2	1-001
2	3	1	5-10d	4	1	3-10d	4	1	1-010
3	2	0	5-11d	2	0	3-11d	4	2	1-011
4	7	3	6-00d	8	3	4-00d	3	1	1-100
5	5	2	6-01d	5	2	4-01d	8	3	1-101
6	4	1	6-10d	5	1	4-10d	3	2	1-110
7	3	0	6-11d	3	0	4-11d	2	1	1-111
8	8	3	7-00d	5	0	5-00d	3	0	2-000
9	6	2	7-01d	6	2	5-01d	7	2	2-001
10	5	1	7-10d	6	1	5-10d	7	1	2-010
11	4	0	7-11d	4	0	5-11d	2	0	2-011
12	9	3	8-00d	9	3	6-00d	9	3	2-100
13	7	2	8-01d	7	2	6-01d	6	2	2-101
14	6	1	8-10d	7	1	6-10d	6	1	2-110
15	5	0	8-11d	6	0	6-11d	1	0	2-111
16	8	0	9-000	11	3	7-000	7	2	3-000
17	9	2	9-001	9	2	7-001	2	0	3-001
18	8	1	9-010	9	1	7-010	6	2	3-010
19	7	0	9-011	8	0	7-011	1	0	3-011
20	10	3	9-100	10	3	7-100	6	0	3-100
21	8	2	9-101	8	2	7-101	5	0	3-101
22	7	1	9-110	8	1	7-110	8	2	3-110
23	6	0	9-111	7	0	7-111	4	0	3-111
24	12	3	10-000	11	0	8-000	12	3	4-000
25	9	2	10-001	11	2	8-001	11	2	4-001
26	10	1	10-010	11	1	8-010	10	1	4-010
27	10	0	10-011	10	0	8-011	9	0	4-011
28	11	3	10-100	12	3	8-100	11	3	4-100
29	8	2	10-101	10	2	8-101	10	2	4-101
30	9	1	10-110	10	1	8-110	9	1	4-110
31	9	0	10-111	9	0	8-111	8	0	4-111
32	14	3	11-000	14	3	9-000	12	0	5-000
33	13	2	11-001	13	2	9-001	13	2	5-001
34	12	1	11-010	13	1	9-010	12	1	5-010
35	12	0	11-011	13	0	9-011	11	0	5-011
36	13	3	11-100	13	3	9-100	13	3	5-100
37	12	2	11-101	12	2	9-101	12	2	5-101
38	11	1	11-110	12	1	9-110	11	1	5-110
39	11	0	11-111	12	0	9-111	10	0	5-111
40	16	3	12-000	-		-	-	-	-
41	15	2	12-001	-		-	-	-	-
42	15	1	12-010	-		-	-	-	-
43	14	0	12-011	-		-	-	-	-
44	15	3	12-100	-		-	-	-	-
45	14	2	12-101	-		-	-	-	-
46	14	1	12-110	-		-	-	-	-
47	13	0	12-111	-		-	-	-	-

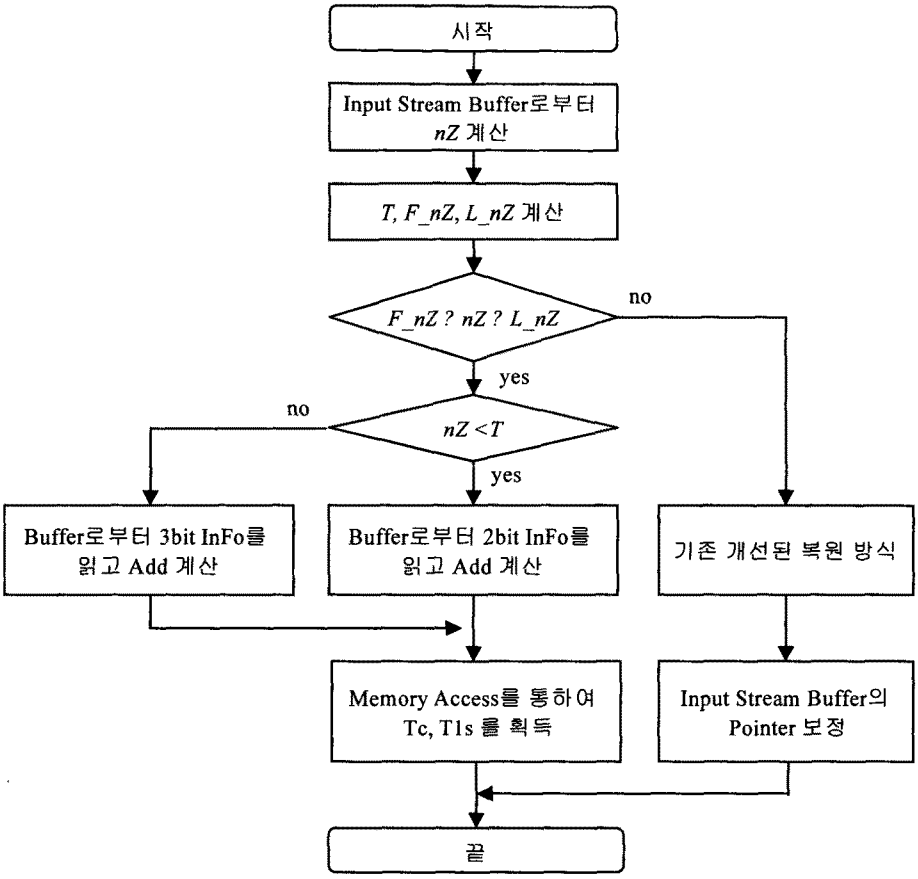


그림 1. 제안하는 복원 방식의 순서도
Fig. 1. Flow chart of the proposed decoding method.

있어서는 L_InFo는 동일한 값을 지니지 않는다. 따라서 nZ를 구한 후 입력 비트열로부터 추가로 읽어야 하는 비트수, 즉 L_InFo가 얼마인지 결정할 수 있는 방법이 요구된다. 이것은 다음과 같이 간단하게 표현할 수 있다.

$$T = 9 - (vLc << vLc) \tag{8}$$

$$\begin{aligned} L\_InFo &= 2 \quad for\ nZ < T \\ &= 3 \quad for\ otherwise \end{aligned} \tag{9}$$

식 (9)는 입력 비트 열 버퍼의 포인터를 보정할 필요가 없음을 보여준다.

그리고 n_Z가 주어지고 L_InFo에 의하여 입력 비트열로부터 InFo를 읽어오면 메모리 주소 Add는 식 (10)과 식 (11)에 의하여 계산된다.

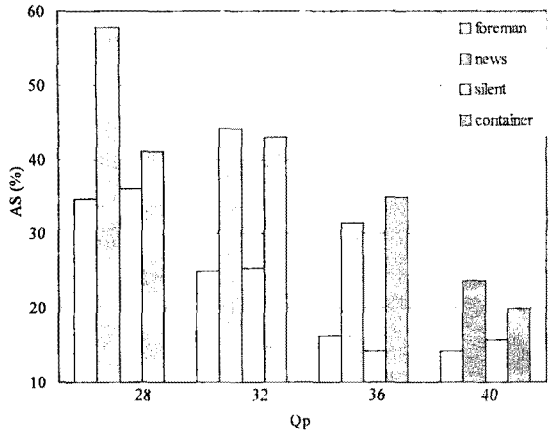
$$\begin{aligned} B\_Add &= (nZ - F\_nZ) << 2 \quad for\ nZ < T \\ &= (1 - vLc >> 1) << 4 + (nZ - T) << 8 \\ &\qquad for\ otherwise \end{aligned} \tag{10}$$

$$\begin{aligned} Add &= B\_Add + InFo[2:1] \quad for\ nZ < T \\ &= B\_Add + InFo[2:0] \quad for\ otherwise \end{aligned} \tag{11}$$

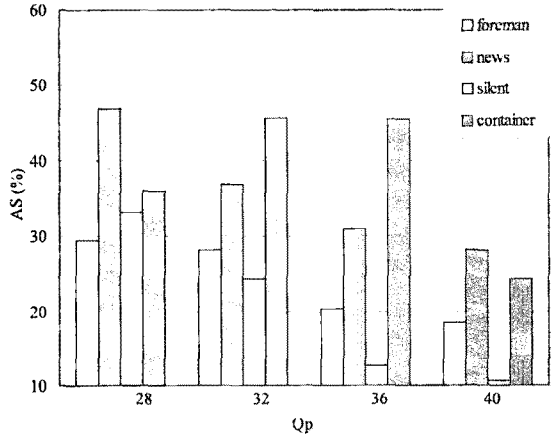
식 (11)을 표 4의 제안된 메모리 구조에 적용해 보면 검색 과정 없이 주어진 n_Z 와 InFo에 의하여 Tc와 T1s가 항상 얻어진다. 따라서 기존 개선 방식[9]에서 도입된 iniNum과 INC는 제안된 메모리 구조에는 불필요하다는 것을 알 수 있다. 이상의 사실을 기초로 본 논문에서는 고속 Coeff_token 복원 방식을 제안한다. 그림 1은 제안하는 복원 알고리즘의 순서도이다.

Ⅳ. 모의 실험 및 고찰

본 논문에서는 QCIF(176x144) 테스트 영상을 사용하여 제안 방식을 검증하였다. 탐색 범위는 16으로 설정하였고 V장의 예측 프레임을 적용하였다. R-D 최적화 기법은 압축된 테스트 영상에 대해 적용하였다. 그 외 다른 조건들은 H.264/AVC Baseline Profile에 기반을 두고 있다.



(a) 10Hz



(b) 30Hz

그림 2. 제안하는 복원 방식에서의 메모리 엑세스 이득
Fig. 2. The Memory Access Gain. achieved by the proposed decoding method

표 5. 4x4 블록당 평균 메모리 엑세스
Table 5. Averaged number of memory accesses for a 4x4 block.

Qp \ Seq.	Foreman	News	Silent	Container
28	0.47	0.40	0.49	0.30
32	0.36	0.42	0.38	0.27
36	0.28	0.37	0.27	0.27
40	0.19	0.27	0.20	0.25

다양한 QP에 대해서 기존 개선된 Coeff_token 복원 방식[9]과 제안 방식을 비교함으로써 그 성능을 검증하였다. 그림 2는 이를 보여준다. 그림 2에서 AS는 기존 복원 방식에 대한 제안된 방식에서의 메모리 엑세스 이득을 의미한다. 그림 2로부터 제안 방식에서 최소 10%에서 최대 57%의 메모리 엑세스 이득이 얻어짐을 알 수 있다.

보다 더 명확한 성능 검증을 위해서 제안 방식에서의 평균 메모리 액세스를 조사하였다. 표 5는 10Hz의 테스트 영상들에 있어서 4x4 블록당 평균 메모리 엑세스를 보여준다. 표 5로부터, 평균 메모리 액세스가 0.5이하임을 알 수 있다. 이것은 제안 방식에서 메모리 액세스가 크게 감소됨을 의미한다. 따라서 제안 방식은 복원 영상의 성능과 화질의 저하없이 복호화기의 파워 소모 증가 및 동작 속도 저하를 방지할 수 있는 잇점을 지닌다.

V. 결 론

Coeff_token 복원에 대한 기존의 개선 방식은 여전히 많은 메모리 엑세스를 필요로 하기 때문에 PMP, DMB, DVB-H와 같은 멀티미디어 서비스에 있어서 파워 소모 및 동작 속도 개선에 제약이 되었다. 이를 해결하기 위하여 본 논문에서는 고속 Coeff_token 복원을 위한 새로운 VLCT와 효율적인 메모리 구조를 제안하였다. 기존 부호어의 특징을 기반으로 새로운 부호어 구조 및 이를 효율적으로 저장하는 메모리 구조를 설계하였다. 모의실험 결과는 제안 방식이 기존 방식보다 최소 10%에서 최대 57% 정도 메모리 액세스 이득을 지니고 있음을 보였다. 이것은 제안 방식이 복원 영상의 성능과 화질의 저하없이 복호화기의 파워 소모 및 동작 속도 제약을 제거할 수 있음을 의미한다. 또한 제안 방식에서 사용되는 부호어는 1바이트로 구성되기 때문에 불필요한 메모리 낭비를 가져오지 않는 장점을 지닌다.

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] M. Horowitz, A. Joch, F. Kossentini, and A. Hallapuro, "H.264/AVC baseline profile decoder complexity analysis," IEEE Trans. Circuits and Syst. Video Technol., vol. 13, no.7 pp.704-716, July 2003.

[3] V. Lappalainen, A. Hallapuro, and T. D. Hämäläinen, "Complexity of optimized H.26L video decoder," IEEE Trans. Circuits and Syst.

Video Technol., vol. 13, no.7 pp.717-725, July 2003.
[4] Q. Peng and J. Jing, "H.264 Codec system-on-chip design and verification,"5th IEEE International Conference on ASIC, vol. 2, pp. 922-925, Oct. 2003.
[5] W. Di, G. Wen, H. Mingzeng, and J. Zhenzhou, "A VLSI architecture design of CAVLC decoder," 5th IEEE International Conference on ASIC, vol. 2, pp. 962-965, Oct. 2003.
[6] H. - C. Chang, C.-C. Lin, and J-I. Guo, "A novel low-cost high-performance VLSI architecture for MPEG-4 AVC/H.264 CAVLC decoding," IEEE International Symposium on Circuits and Systems (ISCAS), pp. 6110-6113, 2005.
[7] Y. H. Moon, G. Y. Kim, and J. H. Kim, "An efficient decoding of CAVLC in H.264/AVC video coding standard," IEEE Trans. on Consumer Electronics, vol. 51, no. 3, pp. 933-938, Aug. 2005.
[8] Y. H. Moon, "An advanced Total_zeros decoding method based on new memory architecture in H.264/AVC CAVLC," IEEE Trans. on Circuits and Systems for Video Technology. (Accepted and to be published in 2008)
[9] Y. H. Moon, "A new coeff_token decoding method with efficient memory access in H.264/AVC video coding standard," IEEE Trans. on Circuits and Systems for Video Technology, vol. 17, no. 6, pp. 729-736, Jun. 2007.

저 자 소 개

문 용 호(정회원)
1992년 부산대학교 전자공학과 졸업(공학사).
1994년 부산대학교 일반대학원 전자공학과 졸업 (공학석사).
1998년 부산대학교 일반대학원 전자공학과 졸업 (공학박사)
현재 경상대학교 정보과학전공 조교수
<주관심분야 : 동영상압축, 영상처리, SoC>

박 태 희(정회원)
1994년 부경대학교 정보통신 공학과 졸업(공학사).
1996년 부경대학교 전자공학과 졸업(공학석사).
1999년 부산대학교 전자공학과 박사 수료.
현재 동명대학교 메카트로닉스공학과 전임강사
<주관심분야 : 영상압축, 얼굴애니메이션>