Bit-wise comparison에 기초한 Huffman decoding 기법

정 종 훈, 김 병 일, 장 태 규, *장 홍 엽
중앙대학교 전자전기공학부. *삼성전자

Huffman decoding method based on bit-wise comparison

J. H. Jeong, B. I. Kim, T. G. Chang, *H. Y. Jang
School of Electronical and Electrical Engineering, Chung-Ang University
*Samsung Electronics
tgchang@jupiter.cie.cau.ac.kr

요약문

본 논문에서는 효율적인 허프만 디코딩을 수행할 수 있도록 하기 위하여 Bit-wise comparison 방법을 제시하였다. 이 방법은 허프만 코딩 원리인 이진트리 구성에 기초하여 허프만 테이블을 재구성 함으로서 디코딩 시간의 단축 및 알고리즘의 간소화를 가져오도록 하였고, 이를 토대로 MPEG-2 AAC 디코더의 허프만 디코딩 부분에 적용함으로써 성능검증을 수행하였다.

I. 서론

허프만 코딩 방식은 오디오, 영상, 기타 다양한 분야에 널리 사용되는 알고리즘으로서, 이는 데이터의 통계적인 출현 빈도수를 고려하여, 데이터들의 발생 확률에 따라 서로 다른 길이의 부호를 할당함으로써 정보의 손실 없이 압축을 실시하는 방법이다. 본 논문에서는 기존의 데이터의 발생빈도에 따라 정렬된 테이블을 이용하여 디코딩을 실시하는 순차 검색방식에서의 문제점을 제시하고, 이를 개선하기 위하여 이진트리의 구성원리에 따른 bit-wise comparison 방식을 제안하고, 이에 대한 성능 분석을 위하여 MPEG-2 AAC의 허프만 디코딩 과정에 적용함으로써 알고리즘에 대한 성능 분석을 실시하였다.

II. Bit-wise comparison에 기초한 Huffman decoder의 구성

2.1 순차 검색방식의 허프만 테이블 구성

허프만 부호화를 실시하기 위해서는 데이터들의 출현 빈도수를 고려하여 적합한 테이블을 구성하고 이를 이용하여 압축을 실시한다. 압축을 위한 코드워드를 생성하는 과정은 그림 1과 같이 이진트리를 구성하고, 이에 따라 표 1과 같이 허프만 부호화를 위한 테이블을 구성하게 된다.

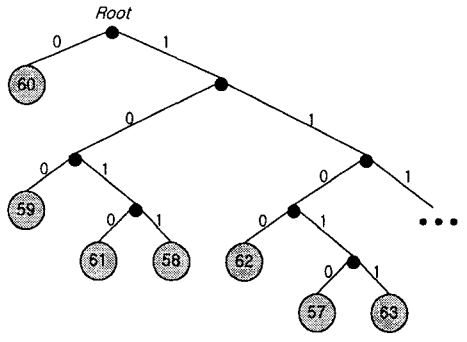

그림 1. 허프만 부호화를 위한 이진트리 구성의 예 (MPEG-2 AAC scalefactor)

표 1. 허프만 디코딩 테이블의 예

value	codeword length	codeword	
		hexa	binary
60	1	0	0
59	3	4	100
61	4	10	1010
58	4	11	1011
62	4	12	1100
57	5	26	11010
63	5	27	11011
...	...	...	...

허프만 복호화 과정은 압축된 codeword들의 집합인 bitstream으로 부터 일치되는 codeword들을 검색한뒤 이에 해당되는 value를 돌려주는 과정이다. 이 과정에서의 문제점은 bitstream내부에 해당 codeword에 대한 길이 정보가 포함되어 있지 않기 때문에 그림 2와 같이 압축 테이블상의 codeword의 길이 정보에 따라 해당되는 양 만큼의 bit들을 읽어들인뒤 일치하는 codeword를 순차적으로 검색하는 방법을 실시하여야 한다. 순차검색의 문제점은 최소 검색시간 대비 최대 검색시간의 비가 매우 클뿐 아니라, 데이터들의 패턴에 따라 검색시간의 변화가 매우 커지는 경우가 발생할 수 있다. 순차 검색방법을 이용시 검색시간은 아래의 식 1과 같다.

$$f(n) = O(n) \qquad - (1)$$

```
int HuffDecode()
{
    length      = *pLength
    codeword= getbits(length)
    while(codeword != *pCodeword)
    {
        pLength++
        pCodeword++
        pIndex++
        diff    = *pCodeword - length
        length = *pLength
        codeword <<= diff
        codeword |= getbits(diff)
    }
    return *pIndex
}
```

그림 2. 순차검색 방식을 이용한 허프만 복호화의 예

2.2 Bit-wise comparison에 기초한 허프만 테이블 구성

일반적으로 평균 및 최대 검색시간 면에서 최상의 성능을 가지는 검색방식은 이진트리를 이용한 binary-search 방식이다. 하지만 binary-tree를 이용한 검색방식의 우수성에도 불구하고 널리 사용되지 못하는 이유는 검색 테이블 구성의 어려움과, 이진 트리를 구성하기 위한 알고리즘의 난이성, 노드간의 연결을 위해 소요되는 추가적인 메모리 소요 등의 문제점이다.

본 논문에서는 binary-search방식 사용시 제기되는 문제점을 해결하기 위한 bit-wise comparison 허프만 디코딩 방식을 제안 하였다. Bit-wise comparison 방식에 의한 허프만 디코딩을 위하여 테이블을 구성하는 과정은 그림 3 에 보인 바과같이 이진트리의 각 검색단계별로 노드들을 그룹화 하고 이들을 1차원 배열상에 연속적으로 정렬하는 방식이다.

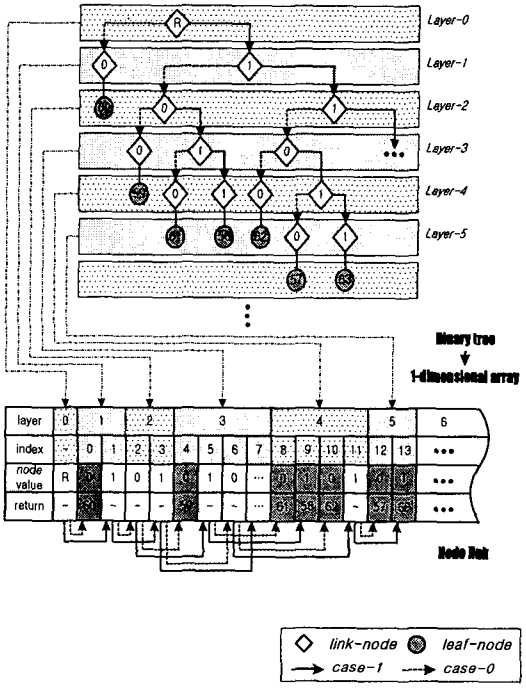

그림 3. bit-wise comparison을 위한 허프만 디코딩 테이블 재구성의 예 (MPEG-2 AAC scalefactor)

허프만 코딩을 위한 이진트리 구성시 한 노드에서 다음 검색단계의 노드로 이동시 발생가능한 경우의 수는 "0"과 "1"이다. 본 논문에서는 1차원 배열상에서 "0"의 경우 이동할 배열의 위치 바로 다음 위치에 "1"의 경우의 데이터를 할당함으로서 "0"일 경우 이동할 주소를 기준으로 "1"일 경우 주소를 하나 증가시킴으로써 다음 노드로의 이동을 가능토록 하였다.

이와 같은 방식을 이용함으로써 생성한 노드의 연결의 예는 표 3과 같다.

표 3. bit-wise comparison을 위한 노드들의 연결

index	case-0	case-1	offset ():case-1	return value
0	-	-	-	60
1	2	3	1+(1)	-
2	4	5	2+(1)	-
3	6	7	3+(1)	-
4	-	-	-	59
5	8	9	3+(1)	-
6	10	11	4+(1)	
7	...	...	...	...
8	-	-	-	61
9	-	-	-	58
10	-	-	-	62
11	12	13	1+(1)	
12	-	-	-	57
13	-	-	-	63
...	...	...	...	...

표 3 상의 offset은 배열상의 현재 위치를 기준으로 "case-0"일 경우 이동할 노드의 상대주소를 가지고 있다. 이를 통하여 절대주소 방식 사용시 요구되는 root 노드의 주소를 기억할 필요 없이 현재위치 기준 offset 만큼의 주소 이동으로 노드 이동이 가능한 장점이 있다. 또한 절대주소 사용시에 비하여 주소 기록을 위한 메모리 공간이 절약되는 이점이 있다. 그림 3에서 나타낸 것과 같이 허프만 코딩을 위한 이진트리의 노드는 검색단계의 중간에 위치하는 link-node와 최종적인 결과값을 나타내는 leaf-node의 두가지 종류가 있다. 이들은 서로 배타적으로 발생하므로 허프만 디코딩 테이블 구성시 link-node에서 필요로 하는 다음 노드 주소값은 음수로, leaf-node에서 필요로 하는 결과값은 양수로 할당함으로써 소요 메모리의 공간을 절약할 수 있도록 한다. 이와 같은 방식에 의하여 최종적으로 생성된 허프만 디코딩 테이블은 표 4와 같고, 허프만 디코딩을 실시하는 과정은 그림 4와 같고, 검색에 소요되는 시간은 수식 2와 같다.

$$f(n) = O(\log_2^n) \qquad - (2)$$

```
int HuffDecode()
{
    pTbl      =  pTbl + getbits(1)
    while(1)
    {
        if(*pTbl >= 0)    return *Value
        pTbl   = pTbl - (*pTbl) + getbits(1)
    }
}
```

그림 4. bit-wise comparison에 의한 허프만 디코더

Ⅲ. 성능 평가

3.1 평균 / 최대 검색횟수 분석

순차 검색방식에 의한 허프만 디코더에 비하여 bit-wise comparison 방식에 의한 허프만 디코더의 성능을 분석하기 위하여 본 논문에서는 MPEG-2 AAC 디코더의 허프만 디코더 부분을 사용함으로써 평균 및 최대 디코딩 시간을 비교 / 분석하였다.

표 4. 순차 검색방식에 의한 허프만 코드의 검색횟수

코드북 번호	검색단계 / 테이블크기	평균 검색횟수	최대 검색횟수
0	19/121	5.09	46
1	16/81	12.50	81
2	16/81	18.61	81
3	16/81	6.46	76
4	16/81	11.42	80
5	16/81	4.51	80
6	16/81	14.04	80
7	16/64	3.29	47
8	16/64	11.13	64
9	16/169	5.09	113
10	16/169	33.60	167
11	16/289	65.02	289

표 5. bit-wise comparison방식에 의한 허프만 코드의 검색횟수

코드북 번호	검색단계 / 테이블크기	평균 검색횟수	최대 검색횟수
0	19/121	3.91	13
1	16/81	4.51	11
2	16/81	5.39	9
3	16/81	3.77	15
4	16/81	4.77	12
5	16/81	3.08	13
6	16/81	5.17	11
7	16/64	2.64	10
8	16/64	4.81	10
9	16/169	2.94	12
10	16/169	6.51	12
11	16/289	7.41	12

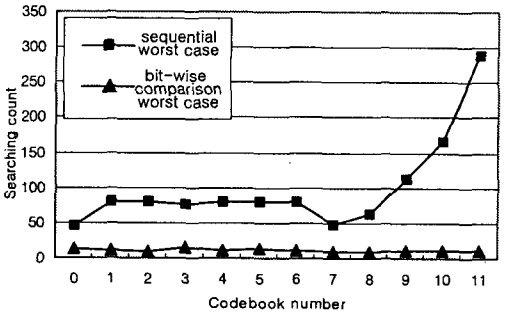

그림 5. 최대 검색횟수 비교

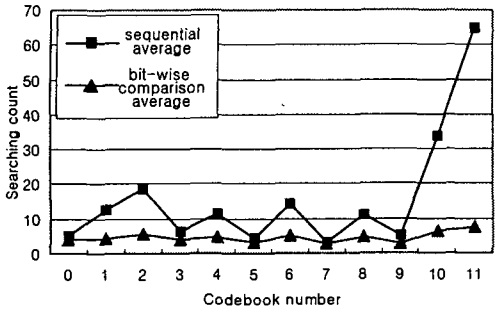

그림 6. 평균 검색횟수 비교

3.2 디코딩 시간 분석

본 논문에서 제안한 bit-wise comparison 에 기초한 허프만 디코딩 방법은 디코딩 검색 횟수의 감소뿐만 아니라 디코딩 알고리즘의 간소화로 인하여 디코딩 시간에 있어 성능 향상을 가져올 수 있도록 하였다. 이에 대한 분석을 위하여 Microsoft Visual C++에서 제공하는 성능 분석 도구인 Profiler를 사용하여 허프만 디코딩을 수행하는데 소요되는 시간에 대한 비교를 실시하였다. 테스트 파일로는 MPEG 웹사이트에서 제공하는 MPEG-2 AAC 성능 평가를 위한 test bitstream을 사용하였다.

표 6. MPEG-2 AAC의 허프만 디코더 수행시간 비교

곡명	Sequential search (msec)	Bit-wise comparison (msec)	비고
L1_44100.aac	80	35.9	122% ↑
L2_44100.aac	79.9	30.2	164% ↑
M1_32k.aac	73.1	27.1	169% ↑
M1_44k.aac	69.9	31.1	124% ↑
SIN2_32000.aac	36.3	18.8	93% ↑
SIN2_44100.aac	52.4	28	87% ↑

순차 검색에 의한 허프만 디코더의 실행결과에 비하여 bit-wise comparison 방식에 의한 허프만 디코더의 디코딩 시간 측면에서 약 134%의 성능 향상을 가져올 수 있었다.

Ⅳ. 결론

본 논문에서는 데이터 압축의 한 방법으로서 널리 사용되는 허프만 디코딩을 효율적으로 수행할 수 있는 Bit-wise comparison 기법에 의한 허프만 디코딩 알고리즘을 제시하였다. 이 방법은 기존 순차 검색에 의한 허프만 디코딩 방식에 비하여 bitnary-search에 기반한 검색방식의 장점인 평균/최대 디코딩 시간의 감소, 디코딩 시간 변화율의 감소, bit-wise 방식을 적용함으로써 이동할 노트의 판단을 위한 조건문의 제거로 인한 알고리즘 및 디코딩 시간의 감소, 상대번지의 사용 및 부호에 의한 노드 종류의 판단에 의한 효과적인 메모리 사용의 장점을 가져올 수 있다. 본 논문에서 제안한 Bit-wise comparison 방법을 사용함으로써 데이터 압축의 한 방법으로써 널리 사용되는 허프만 디코더의 성능 개선에 기여할 수 있을 것으로 사료된다.

참고문헌

[1] A.Gersho, "Advances in speech and audio compression", proc. IEEE, vol. 82, 99. 901-918, June 1994 Digital Speech

[2] Nick B.Body and Donald G.Bailey, "Efficient representation and decoding of static huffman code tables in a very low bit rate environment", IEEE, 1998

[3] Maja Bystrom and Susanna Kaiser, "Soft decoding of variable-length codes", IEEE, 2000

[4] Jae Ho Jeon, Young Seo Park, and Hyun Wook Park. "A fast variable-length decoder using separation", IEEE Transactions on circuit and systems for video technology, vol. 10, No. 5, August 2000

[6] ISO/IEC JTC1/SC29/WG11 N1650 "IS 13818-7 (MPEG-2 Advanced Audio Coding, AAC)"