論 文

한쪽으로 기운 허프만 트리에서의 효율적인 허프만 복호 기법

正會員 金 炳 漢* 正會員 林 鍾 錫**

An Efficient Decoding Technique for Huffman Code Using Tilted Huffman Trees

Byoung Han Kim*, Chong Suck Rim** *Regular Members*

要 約

영상 압축을 위하여 제시된 JPEG, MPEG 등과 같은 표준에서는 영상 자료의 허프만 부호화를 위하여 한쪽으로 기운 허프만 트리를 사용하고 있다. 본 논문에서는 이러한 허프만 트리에 대하여 효율적인 새로운 허프만 부호 복원방법을 제안한다. 제안한 허프만 부호 복원방법에서는 입력된 비트열로부터 미리 결정된 수 만큼의 연속된 비트를 반복적으로 취하여 이를 토대로 심볼 정보가 저장된 테이블을 참조함으로써 부호화된 심볼을 복원한다. 이러한 복원방법은 주어진 허프만 트리가 한쪽으로 기울어진 경우에 복원을 위한 테이블의 크기를 작게 할 수 있으며 실제로 본 논문에서는 이의 상한값을 제시한다. 또한, 이 방법은 한 클럭에 여러 비트를 동시에 처리하기 때문에 실시간 동작이 가능하며 이에 대한 평가 결과를 보인다.

ABSTRACT

The tilted Huffman trees are used in JPEG and MPEG image compression standards for Huffman coding. In this paper we propose a new decoding technique for Huffman code which is efficient if the given Huffman tree is tilted. In the proposed decoding technique for Huffman code, symbols are decoded by repeatedly obtaining the predefined number of consecutive bits and accessing symbol tables based on the obtained bits. We show that the size of the symbol table can be small if the Huffman tree is tilted. Specifically, we show an upper bound on the size in this paper. Since the proposed method processes multiple bits at each clock, it can be used for real time processing. We show such evaluation results.

I. 서 론

허프만 부호화 기법은 발생 확률이 높은 자료에는 상대적으로 적은 비트수의 부호를 할당하고 낮은 발생 확률을 갖는 자료에는 많은 비트수의 부호를 할당하여 부호의 평균 비트수를 최소화하는 방법이다[5, 6]. 허프만 부호화 기법은 정보의 손실없이 자료를 압축하는 방법 중에서 가장 우수한 방법으로 알려져 있고 따라서 JPEG, MPEG 등과 같은 영상 자료 압축

*三星電管 情報製品 研究所
Samsung Display Devices, Data Products Division R&D Center
**西江大學校 電子計算學科
Dept. of Computer Science, Sogang University
論文番號 : 93- 195

을 위하여 제시된 표준안에서 공통적으로 이를 채택하고 있다[3,4].

허프만 부호화 기법을 사용하여 자료를 부호화 하는 것은 발생 가능한 모든 심볼에 대한 허프만 부호를 미리 테이블에 저장하고 하나의 심볼을 부호화 할 때 구성한 테이블을 참조함으로써 해결할 수 있다[12]. 그러나 허프만 부호화된 자료를 복원하는 일은 자료의 각 심볼에 부여된 부호의 비트수가 다르기 때문에 부호간의 경계를 찾기가 곤란하여 간단하지 않다. 그런데 최근 VLSI 기술이 발달하여 정지 영상 압축 및 복원용으로 제안된 JPEG 표준을 이동 영상 압축 및 복원에 사용하는 경우도 있어[11] 실시간에 허프만 부호화된 자료를 복원할 필요가 있다. 여기서 실시간 처리란 디지탈 TV 국제 표준안인 CCIR 601에서 제시한 해상도와 초당 처리할 영상의 갯수를 만족하기 위하여 필요한 27 Mpixels/sec 이상의 처리율을 의미한다[3].

허프만 부호를 복원하는 기법에는 입력 비트열을 한 비트씩 처리하여 자료를 복원하는 순차적 기법과[2,12] 심볼에 부여된 부호의 최대 비트수 만큼을 입력 비트열로부터 취하여 자료를 복원하는 병렬적 기법이 있다[1,7,11,14,15]. 여기서 순차적 기법은 하나의 심볼을 복원하는데 그 심볼에 부여된 부호의 비트수 만큼의 단위시간이 필요하므로 실시간 처리에는 적당하지 않으며 단위시간 당 하나의 심볼을 복원할 수 있는 병렬적 기법이 실시간 처리에 보다 유리하다.

병렬적 기법을 사용하는 방법으로 최근 Sun 등이 발표한 복원기를 들 수 있다[7,14]. 이 방법에서는 허프만 부호에 관한 정보를 PLA에 미리 저장하고 바렐 쉬프트(Barrel Shifter)를 사용하여 입력 비트열로부터 매 클럭마다 부호의 최대 길이 만큼의 비트를 취하여 PLA에 저장된 심볼을 출력한다. 이 방법은 PLA를 사용하기 때문에 그 구조가 비교적 간단하고 정확히 한 클럭에 하나의 심볼을 출력한다는 장점이 있다. 그러나 PLA를 사용할 경우 그 내용을 변경할 수 없기 때문에 각 영상마다 서로 다른 허프만 부호를 사용할 수 있는 JPEG 표준에는 이 방법을 사용하기가 적당하지 않다.

허프만 부호가 가변적일 경우에는 복원기가 가지고 있는 허프만 부호에 관한 정보를 쉽게 수정할 수 있어야 한다. 이를 위하여 복원기의 저장 매체로써 RAM 또는 CAM(Content Addressable Memory) 등을 사용하는 방법이 있는데 일반적으로 많은 양의 하드웨어가 필요하다[1,15]. 그런데 Luthi 등이 최근 발표한 복원방법에서는 허프만 부호에 관한 정보를 RAM에 저장하고 Sun 등의 방법과 마찬가지로 입력 비트열로부터 부호의 최대 길이 만큼의 비트를 취하여 이를 RAM 접근을 위한 주소로 사용한다[11]. JPEG 표준에서의 각 허프만 부호는 그 부호의 앞 부분에 연속적으로 그 값이 '1'인 비트들이 존재하는데 이들을 '선행 비트'라고 하고 같은 부호의 나머지 비트들을 '후행 비트'라고 하면 후행 비트의 갯수는 많아야 7 비트이다[11]. Luthi 등은 이러한 사실을 이용하여 선행 비트의 가능한 갯수에 따라 RAM을 분류하고 후행 비트를 주소로 하여 이들 RAM을 접근함으로써 필요한 RAM의 크기를 작게 하였다. 그러나 이러한 방법을 사용하여도 필요한 RAM의 크기는 2,720×12 비트로서 여전히 많은 양의 메모리를 필요로 한다.

한편, Lin과 Messerschmit는 기존의 순차적 복원방법을 확장하여 보다 빠른 처리율을 갖는 복원방법을 연구하였다[8,9,10]. 그들의 방법에서는 입력된 비트열로부터 매 클럭당 $k \geq 1$ 비트를 입력받는 FSM(Finite State Machine)을 구현하여 부호를 복원한다. Parhi도 제안한 바 있는[13] 이러한 방법은 한 비트씩 입력받아 부호를 복원하는 순차적 방법에 비하여 k배의 처리율 향상을 기대할 수 있으나 FSM의 상태수가 k의 증가에 따라서 2의 지수배로 증가하므로 그 실현 가능성은 희박하다.

본 논문에서 제안하는 허프만 부호 복원방법에서는 Lin과 Messerschmit 그리고 Parhi 등의 방법과 유사하게 매 클럭당 한 개 이상의 비트를 입력 받아 허프만 부호를 복원한다. 즉, 입력된 비트열로부터 미리 결정된 수 만큼의 연속된 비트를 반복적으로 취하여 이를 토대로 심볼 정보가 저장된 테이블을 참조함으로써 복원을 수행한다. 그런데 우리는 복원에 필요한 메모리의 양을 줄이기 위하여 특별히 고안된 테이블을 사용하는데, 주어진 허프만 트리가 한쪽으로 기울어진 경우 이 테이블의 크기는 실제 구현에 적당한 어떤 상한값을 가짐을 보인다. 실제로 제안한 복원방법을 JPEG 표준에 적용할 경우 Luthi 등의 방법[11]에 비하여 약 60% 정도의 메모리를 절감할 수 있다. 또한 제안한 허프만 부호 복원방법은 허프만 부호 정보를 저장하는 매체로 RAM을 사용하기 때문에 여러 다양한 허프만 부호를 아무런 하드웨어의 수정없이 복원할 수 있다는 장점이 있다.

본 논문은 서론을 포함하여 모두 다섯 장으로 구성된다. 다음 2장에서는 본 논문에서 필요한 트리

(tree)에 관한 기본 용어를 정의한다. 3 장에서는 제안한 허프만 부호 복원방법을 설명하며 특히 이 방법에서 사용하는 테이블 크기의 상한값을 보인다. 그리고 4 장에서는 제안한 복원방법을 JPEG 표준에 적용하였을 경우에 필요한 하드웨어의 크기 그리고 실시간 처리 가능성 등을 검토한 결과를 기술하며 마지막 5 장에서 결론을 내린다.

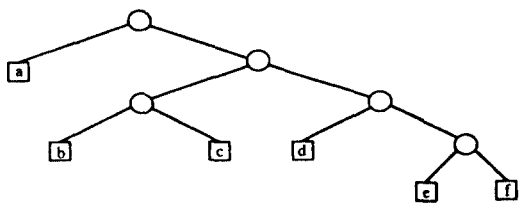
Ⅱ. 기본 정의

$T=(V, E)$를 루트(root)가 r인 트리라고 하자. 여기서 V와 E는 각각 노드(node)와 에지(edge)의 집합이다. 트리 T에서 각 노드 v의 레벨(level)은 $l(v)$로 표시하며 다음과 같이 정의한다. 먼저 루트의 레벨을 1로 하고 만약 V의 어떤 노드 v의 레벨이 i이면 v의 모든 자식 노드(child node)들의 레벨은 $i+1$로 한다. 트리 T의 높이는 T의 모든 노드의 레벨 중 최대값으로 정의한다. 트리 T의 각 노드가 많아야 두 개의 자식 노드들을 가지고 있으면 T를 2진 트리(binary tree)라고 부르고, 만약 정확히 두 개의 자식 노드를 가지고 있으면 편의상 그들에 순서를 부여하여 그 하나를 왼쪽 자식 노드, 다른 하나를 오른쪽 자식 노드라고 부른다. 그리고 하나 이상의 자식 노드를 가진 노드를 중간 노드(internal node), 자식 노드를 가지지 않은 노드를 잎 노드(leaf node)라고 부른다. T의 임의의 서로 다른 두 개의 노드 v와 w에 대해서 만약 v에서 w까지 하나의 경로(path)가 존재하고 $l(v)>l(w)$이면 w를 v의 자손(descendant)이라고 부른다. 또한, T의 임의의 노드 v를 루트로 하고 v의 모든 자손들로 구성된 트리를 T의 부트리(subtree)라 하고 $T(v)$로 나타낸다.

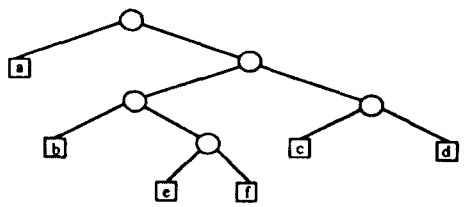
트리 T를 2진 트리라고 하자. 여기서, T의 모든 중간 노드는 정확히 두 개의 자식 노드를 갖는다고 가정한다(실제로 앞으로 설명하는 허프만 트리는 이러한 조건을 만족하도록 구성된다[5]). T의 모든 노드들을 하나의 평면에 배치하되 임의의 노드 v에 대하여 그 왼쪽 자식 노드는 v의 왼쪽 아래에 그리고, 오른쪽 자식 노드는 v의 오른쪽 아래에 배치한다고 하자. 이때, 가장 왼쪽에 놓여진 잎 노드의 레벨을 T의 왼쪽 높이라고 하고 가장 오른쪽에 놓여진 잎 노드의 레벨을 T의 오른쪽 높이라고 정의한다. 그리고, 임의의 노드 v를 루트로 하는 T의 부트리 $T(v)$에서 v의 레벨을 1로하고 v의 각 후손들의 레벨을 이에 준하여 다시 정의하면 $T(v)$의 왼쪽 높이와 오른쪽 높이는 T에서와 마찬가지로 정의된다.

2진 트리 T의 모든 각 중간 노드 v에 대해서 그의 왼쪽 자식 노드를 루트로 한 부트리의 오른쪽 높이가 v의 오른쪽 자식 노드를 루트로 한 부트리의 왼쪽 높이보다 작거나 같으면 T는 오른쪽으로 기울었다고 하고 이를 오른쪽으로 기운 트리라고 부른다. 그림 1에 이러한 트리의 예를 보인다. 즉, 그림 1(a)는 오른쪽으로 기운 트리지만 그림 1(b)는 오른쪽으로 기운 트리가 아니다. 그리고, 왼쪽으로 기운 트리도 오른쪽으로 기운 트리와 마찬가지로 비슷하게 정의된다.

S개의 심볼이 있고 이들 각각에 대한 허프만 부호가 정의되어 있다고 하자. 이들 심볼과 허프만 부호와의 관계는 다음과 같은 형태의 2진 트리 T로 나타낼 수 있는데 이것을 허프만 트리라고 한다. 먼저, T는 S개의 잎 노드를 가지고 있으며 이들은 각각 하나의 심볼에 유일하게 대응된다. 또한 루트를 포함한 T의 임의의 중간 노드 v에 대하여 v와 그의 왼쪽 자식 노드를 연결하는 에지에 라벨 0이 그리고 v와 그의 오른쪽 자식 노드를 연결하는 에지에 라벨 1이 부여되어 있으며(반대로 왼쪽 자식 노드에 연결된 에지에는 라벨 1이 그리고 오른쪽 자식 노드에 연결된 에지에는 라벨 0이 부여될 수도 있다) T의 루트에서 하나



(a) 오른쪽으로 기운 트리.
(a) A rightist tree.



(b) 어느쪽으로도 기울지 않은 트리.
(b) A tree which is neither rightist nor leftist.

그림 1. 오른쪽으로 기운 트리의 예
Fig. 1. An example of the rightist tree.

의 잎 노드 s까지의 경로에 존재하는 에지의 라벨들을 루트에서 가까운 것 부터 l_1, l_2, ..., l_d라고 하면 이로부터 얻은 비트열 $l_1 l_2 \ldots l_d$는 s에 대응되는 심볼의 허프만 부호와 동일하다. 주어진 심볼들에 대한 허프만 부호가 정해지면 이에 대응하는 허프만 트리는 유일하게 존재하며 역으로 허프만 트리가 주어졌다고 하면 이에 대한 허프만 부호가 유일하게 존재한다.

Ⅲ. 새로운 허프만 부호 복원방원

1. 기본동작원리

S개의 심볼에 대한 허프만 부호가 정의되어 있고 그 중 가장 긴 부호의 비트수를 K라고 하자. 본 논문에서 제안한 허프만 부호 복원방법에서는 먼저 양의 정수로 구성된 튜플 $B=(k_1, k_2, \ldots, k_n)$을 $\sum_{i=1}^{n} k_i = K$가 되도록 정의한다. 튜플 B는 허프만 부호화된 비트열로부터 매 클럭당 처리하는 비트수를 의미하는데 이의 용도는 앞으로 우리의 복원방법을 설명함에 따라 점차 밝혀질 것이다.

다음, 각 허프만 부호에 대응하는 심볼들을 하나의 테이블에 저장한다. 이 테이블은 앞으로 허프만 복원 테이블이라고 부르며 HT로 표기한다. HT의 i번째 항은 $HT[i]$로 나타낸다. HT에는 심볼 외에도 우리의 방법에 의하여 부호를 복원하는데 필요한 정보가 기록되어 있으며 이에 대한 내용 역시 차차 설명하기로 한다. 또한 테이블 HT를 읽기 위하여 변수 BA를 정의하는데 이곳에는 HT를 읽기 위한 기본주소가 기록되어 있다. 즉, HT는 매 클럭마다 입력되는 비트열을 오프셋으로 하고 BA에 이를 더한 값을 주소로 하여 참조된다.

어떤 자료를 허프만 부호화한 비트열을 b_1, b_2, ... 라고 하자. 이러한 비트열에 대한 복원을 수행하기 위하여 먼저 기본주소 BA를 0으로 한다. 다음, 튜플 B에 근거하여 b_1 부터 b_{k_1} 까지의 비트열을 입력받아 이를 2진수 BB로 하고 BA에 BB를 더하여 HT를 읽기 위한 주소 h를 만들어 낸다. 즉, $BB=(b_1 b_2 \cdots b_{k_1})_2$ 이며, $h=BA+BB$이다.

$HT[h]$의 내용은 두 가지가 있을 수 있다. 이의 설명을 위하여 비트 b_1 부터 시작하는 허프만 부호를 C_1이라고 하고 이의 길이(비트수)를 $|C_1|$으로 표시하자. 그리고 C_1 다음에 이어지는 허프만 부호를 C_2, C_3, ... 라고 하고 이들의 길이 역시 C_1과 마찬가지로 표시하자. 만약 $|C_1| \le k_1$이면 $HT[h]$에는 C_1으로 부호화된 심볼이 저장되어 있어서 이를 출력한다. 또한, $|C_1|$을 $HT[h]$에 아울러 저장하여 다음 허프만 부호 C_2를 복원할 때 비트 $b_{|C_1|+1}$ 부터 그 복원을 시작할 수 있도록 한다.

한편, $|C_1| > k_1$이면 C_1에 해당하는 비트열이 b_{k_1} 이후에도 계속됨을 의미하며 따라서 비트 b_{k_1+1} 부터 $b_{k_1+k_2}$ 까지 k_2 비트를 다음 클럭에서 입력받아 C_1에 대한 복원을 계속한다. 이를 위하여 $HT[h]$에는 다음 k_2 비트를 처리할 때 필요한 기본주소가 기록되어 있고 이는 BA에 저장된다.

$|C_1| > k_1$ 인 경우에 대한 설명을 계속한다. 앞에서 설명한 바와 같이 BA에는 새로운 기본주소가 저장되어 있으며 이에 새로 입력받은 k_2 비트를 더하여 HT를 읽기 위한 주소 h를 만들어 낸다. 즉, $h=BA+(b_{k_1+1} b_{k_1+2} \cdots b_{k_1+k_2})_2$이다. 만약 $|C_1| \le k_1+k_2$이면 앞에서 설명한 것과 마찬가지로 $HT[h]$에는 C_1으로 부호화된 심볼이 저장되어 있어 이를 출력하고 역시 $HT[h]$에 저장되어 있는 부호의 경계에 대한 정보 $|C_1|-k_1$을 이용하여 다음 클럭에 비트 $b_{|C_1|+1}$부터 시작하는 허프만 부호 C_2의 복원을 시작할 수 있도록 한다.

그러나 $|C_1| > k_1+k_2$라면 비트 $b_{k_1+k_2+1}$부터 다시 k_3 비트를 다음 클럭에서 입력받아 C_1에 대한 복원을 계속하고($HT[h]$에는 이에 사용할 새로운 기본주소가 저장되어 있다), 이 과정을 C_1에 대응하는 비트열이 끝날 때 까지 튜플 B에 정의된 숫자에 근거하여 k_4, k_5, ... 비트를 매 클럭마다 입력받아 같은 과정을 반복한다. 그런데 $|C_1| \le K$이므로 이 과정은 많아야 n번 반복되고 따라서 우리의 복원방법은 n 개의 클럭 내에 하나의 허프만 부호를 복원한다.

이제 테이블 HT의 내용을 아직 완전하지는 않지만 보다 확실히 정의한다. 입력 비트열을 b_1, $b_2 \cdots$ 라고 하고 지금까지 b_1 부터 b_{p-1}까지의 비트열을 처리하였으며 현재 허프만 부호 C_i의 복원을 위하여 b_p 부터 k_i 비트($1 \le i \le n$)를 입력 받았다고 가정하면 $h=BA+(b_p b_{p+1} \cdots b_{p+k_i-1})_2$가 된다. C_i의 마지막 비트를 b_{p+q-1}이라고 하자. 그러면 현재 C_i을 복원중이므로 $q \ge 1$이며 $HT[h]$에는 q의 값에 따라 다음과 같은 내용이 저장되어 있다.

1. $q > k_i$인 경우 C_i에 대응하는 비트열이 b_{p+k_i-1} 이후에도 계속되므로 $HT[h]$에는 다음 클럭에서의 작업을 위한 기본주소가 저장되어 있다.
2. $q \le k_i$인 경우 C_i에 대응하는 비트열이 b_{p+k_i} 이전

에 끝나므로 $HT[h]$에는 C_j로 부호화된 심볼과 q가 저장되어 있어 그 심볼을 출력하고 입력 비트열의 b_{p+q}부터 k_1, k_2, ... 비트씩 입력받아 C_j 다음에 이어지는 허프만 부호를 복원할 수 있도록 한다.

2. 허프만복원 테이블의 최적화

기존의 순차적 허프만 부호 복원방법은 허프만 트리의 루트로부터 출발하여 한 클럭에 한 개의 노드를 입력 비트의 값에 따라 선택하여 잎 노드에 도착하는 것과 동일하다[2,12]. 이와는 달리 3.1절에서 기술한 허프만 부호 복원방법은 주어진 튜플 $B=(k_1, k_2, ..., k_n)$에 따라 한 클럭에 여러개의 노드를 동시에 선택함으로써 n 클럭 이내에 허프만 트리의 해당 잎 노드에 도달하는 것과 같다(여기서 3.1 절에서 정의한 바와 같이 K를 허프만 부호의 최대 길이라고 할 때 $\sum_{i=1}^{n} k_i = K$이다). 이를 위하여 구성하여야 할 허프만 복원 테이블 HT의 크기는 허프만 트리가 임의의 형태일 경우에 그 상한치를 예측하기 어려워 $O(2^K)$라고 할 수 있다. 그러나 본 절에서는 HT에 중복되어 존재하는 정보를 제거함으로써 HT의 크기를 축소하는 방법을 설명한다. 물론 이 경우 HT의 내용과 복원방법에 있어서 3.1 절에서 설명한 것과 다소 차이가 있는데 이 역시 본 절에서 자세히 설명한다.

T를 루트가 r인 허프만 트리라고 하자. 튜플 $B=(k_1, k_2, ..., k_n)$가 주어졌을 때 루트 r과 임의의 $1 \le j \le n-1$에 대하여 그 레벨이 $\sum_{j=1}^{i} k_j+1$인 모든 중간 노드들을 중간 복원 노드라고 한다. 또한, 레벨이 $\sum_{j=1}^{i} k_j+1$인 중간 복원 노드 v를 루트로 하고 v의 후손 중 그 레벨 l이 $\sum_{j=1}^{i} k_j+1 < l \le \sum_{j=1}^{i+1} k_j+1$을 만족하는 모든 노드들로 구성된 트리($T$의 부그래프)를 부분 복원 트리라고 부르며 이를 $TP(v)$로 표시한다. 마찬가지로 또 하나의 중간 복원 노드인 r을 루트로 하고 그 레벨 l이 $1 < l \le k_1+1$을 만족하는 r의 모든 후손들로 구성된 트리 역시 부분 복원 트리이다. 예를 들어 그림 2에 하나의 허프만 트리에 대하여 $B=(3, 3)$일 경우에 정의되는 중간 복원 노드와 부분 복원 트리들을 보인다.

S개의 심볼을 부호화하기 위한 높이가 $K+1$인 허프만 트리 T와 튜플 $B=(k_1, k_2, ..., k_n)$가 주어져 있다고 하자. 여기서 $\sum_{i=1}^{n} k_i = K$이다. 앞의 3.1 절에서 설명한 복원방법에서 사용하는 허프만 복원 테이블 HT는 튜플 B에 의하여 정의된 T의 부분 복원 트리와 밀접한 관계에 있다. 즉, 현재 k_i, $1 \le i \le n$, 비트를 입력받아 HT를 참조한다면 이는 $i=1$인 경우 부분 복원 트리 $TP(v=r)$를 그리고 $i \ne 1$인 경우 레벨이 $\sum_{j=1}^{i-1} k_j+1$인 어떤 중간 복원 노드 v를 루트로 한 부분 복원 트리 $TP(v)$를 탐색하여 T의 잎 노드 또는 레벨이 $\sum_{j=1}^{i} k_j+1$인 다른 중간 복원 노드에 도달하는 것과 동일하다.

이러한 관찰로부터 우리는 HT의 각 항은 T의 부분 복원 트리 중 하나를 탐색하는데 사용된다는 것을 알 수 있다. 그런데 HT의 어떤 항이 $TP(v)$를 탐색하는데 사용된다면 이는 $TP(v)$외의 다른 부분 복원 트리의 탐색에는 사용되지 않는다. 따라서 $TP(v)$를 탐색

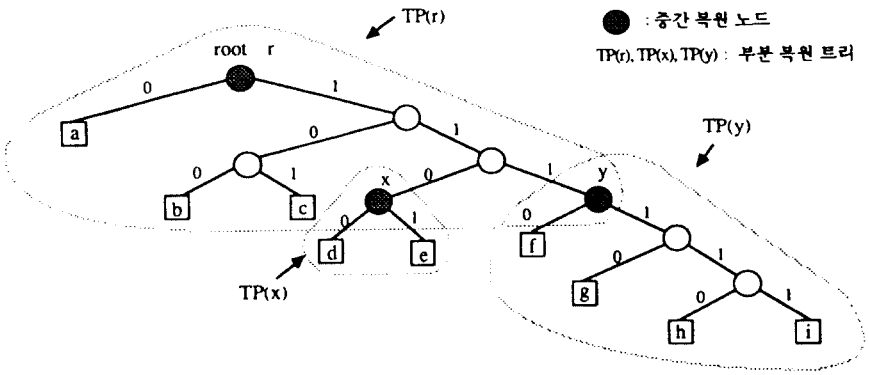


그림 2. 중간 복원 노드와 부분 복원 트리의 예. 여기서 $B=(3, 3)$이다.

Fig. 2. An example of intermediate decoding nodes and partial decoding trees, where $B=(3, 3)$.

하는데 사용되는 HT의 항들을 HT의 $TP(v)$에 대한 부분 복원 테이블이라고 하면 HT는 T의 부분 복원 트리들을 탐색하기 위한 부분 복원 테이블들로 구성되어 있다고 할 수 있다. 부분 복원 테이블의 예를 그림 3에 보인다. 그림 3은 그림 2의 허프만 트리에 대한 허프만 복원 테이블을 보인 것이다. 그림 2의 허프만 트리가 세 개의 부분 복원 트리로 구성되어 있으므로 그림 3의 허프만 복원 테이블도 세 개의 부분 복원 테이블로 구성되어 있다. 그림 3에서 보인 바와 같이 앞으로 부분 복원 트리 $TP(v)$에 대한 부분 복원 테이블을 HT^v로 표기한다.

그림 3에 보인 허프만 복원 테이블 HT에는 많은 항들의 내용이 서로 동일하다. 예를 들어 HT^x에는 심볼 d와 e가 각각 네 번 포함되어 있으며 이러한 사실은 HT의 크기를 크게하는 요인이 된다. 실제로 HT^x를 참조하기 위한 주소는 입력 비트열의 현재 처리할 비트 b_p 부터 연속된 세 비트 b_p, b_{p+1}, b_{p+2}를 입력받아 구성한 오프셋$(b_pb_{p+1}b_{p+2})_2$를 HT^x의 기본주소에 더하여 만들어진다. 여기서 우리는 HT^x로부터 읽은 내용은 오직 비트 b_p에 의해서만 결정된다는 사실을 쉽게 알 수 있다. 본 절의 나머지 부분에서는 이러한 사실을 이용하여 허프만 복원 테이블의 크기를 축소하는 방법을 설명하고 이에 따라 앞의 3.1 절에서 설명한 복원방법에 추가로 보완하여야 할 사항을 설명한다.

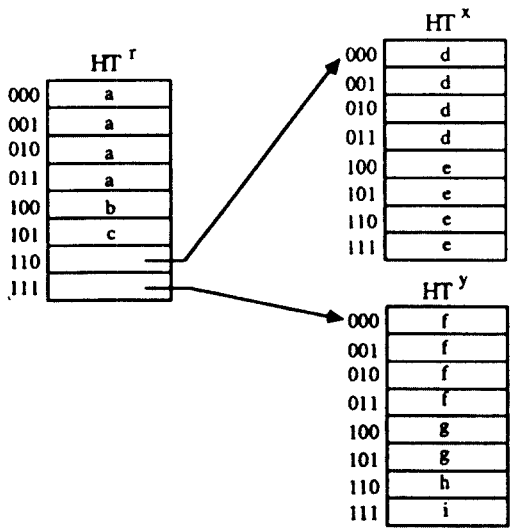


그림 3. 그림 2에 대한 허프만 복원 테이블.
Fig. 3. Huffman decoding table for the tree in Fig. 2.

임의의 부분 복원 트리 $TP(v)$에 대한 부분 복원 테이블 HT^v에서 그 내용이 같은 항들은 연속적으로 존재하며 그들의 갯수는 항상 2의 지수배이다. HT^v에 존재하는 서로 같은 내용을 갖는 항들 중에서 그 수가 가장 작은 항들의 갯수를 2^m이라고 하자. 여기서 m은 HT^v의 중복 정도라고 부르는 음이 아닌 정수인데 특히, $m=0$ 이면 그 테이블에 존재하는 항 중 그 내용이 유일한 항이 최소한 하나 존재하는 경우이다. 만약 $T(v)$의 높이가 k_i+1이라면 HT^v는 k_i 비트를 입력받았을 때 참조되고 이 중 처음 k_i-m 비트만이 HT^v로부터 읽을 내용을 결정하게 된다. 따라서, 그림 4에 몇개의 예를 보인 바와 같이 HT^v를 2^{k_i-m} 개의 항을 갖는 테이블로 축소하고 입력된 k_i 비트의 비트열 중에서 처음 k_i-m 비트만으로 축소된 테이블을 참조하도록 한다면 전체 허프만 복원 테이블의 크기를 작게 할 수 있다. 그런데, 앞으로 편의상 이렇게 축소

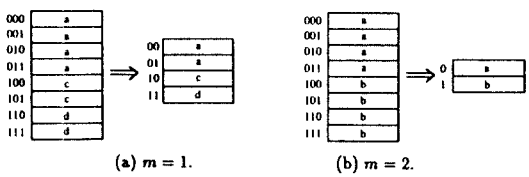


그림 4. 중복 정도에 따른 부분 복원 테이블의 축소($k_i=3$ 일 경우).
Fig. 4. Reduction of partial decoding tables by duplication factors($k_i=3$).

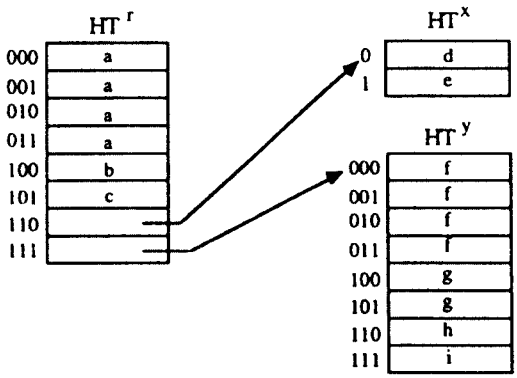


그림 5. 그림 3에 보인 허프만 복원 테이블의 축소.
Fig. 5. Reduction of the Huffman decoding table shown in Fig. 3.

된 부분 복원 테이블과 이들로 구성된 허프만 복원 테이블을 축소되기 전의 테이블과 마찬가지로 각각 HT'와 HT로 표기한다. 그림 5에 그림 3에 보인 허프만 복원 테이블을 축소하여 새로 구성한 결과를 보인다.

지금까지 설명한 테이블 축소방법을 사용하여 허프만 복원 테이블 HT를 재구성한다면 3.1 절에서 설명한 HT의 내용과 HT를 참조하기 위한 주소를 계산하는 방법 등이 약간 달라진다. 가장 큰 차이는 HT를 참조하기 위하여 정의된 변수 BA 외에도 현재 접근하고자 하는 부분 복원 테이블의 중복 정도 m을 저장하기 위한 변수가 추가로 필요하다. 편의상 이 변수를 m으로 표기하자. 복원을 처음 시작할 때와 현재 하나의 심볼을 복원하였을 경우에 다음 클럭에서 필요한 m 값은 0이다. 그러나 현재 처리중인 허프만 부호의 복원이 완료되지 못하고 다음 클럭에서도 계속해야 할 경우에 필요한 m 값은 0이 아닐 수도 있으며 이 값은 현재 읽은 HT의 항에 미리 저장되어 있다.

높이가 $K+1$인 임의의 허프만 트리에 대하여 $\sum_{i=1}^{n} k_i = K$인 튜플 $B=(k_1, k_2, \ldots, k_n)$가 주어졌다고 하자. 입력 비트열을 $b_1, b_2, \ldots$ 라고 하고 지금까지 b_1부터 b_{p-1}까지의 비트열을 처리하였으며 현재 허프만 부호 C_l의 복원을 위하여 b_p 부터 k_i, $1 \le i \le n$, 비트를 입력 받았다고 가정하자. 이때 HT를 참조하기 위한 주소 h는 다음과 같이 계산된다.

$$h = BA + (((b_p b_{p+1} \cdots b_{p+k_i-1})_2 \wedge (\underbrace{1 \cdots 10 \cdots 0}_{m})_2)/2^{k_i - m})$$

여기서 $\wedge$는 논리곱 AND를 의미한다. 그리고 C_l의 마지막 비트를 b_{p+q-1}이라고 하면 q의 값에 따라 $HT[h]$에는 다음과 같은 값들이 저장되어 있다.

1. $q > k_i$이면 다음 클럭에서 참조할 HT의 기본주소와 중복 정도 m.
2. $q \le k_i$이면 C_l로 부호화된 심볼과 q.

지금까지 설명한 축소된 형태의 허프만 복원 테이블은 주어진 허프만 트리로부터 쉽게 구성할 수 있으며[16], 부호 복원을 시작하기 전에 허프만 부호 복원기를 제어하는 CPU가 이 과정을 수행한다. 마지막으로 복원 도중에 읽은 HT의 내용이 위의 1항에 해당하는지 또는 2항에 해당되는지를 구분하기 위하여 HT의 각 항에 이를 위한 플래그 비트를 추가로 마련한다.

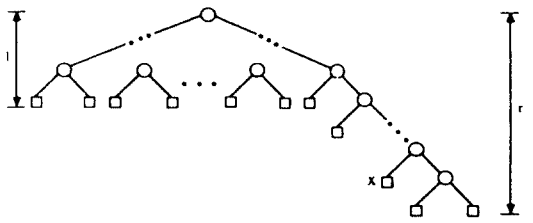


그림 6. 가장 많은 중복항이 존재하기 위한 부분 복원 트리의 형태.

Fig. 6. A partial decoding tree with largest dupplicated terms.

3. 허프만 복원 테이블의 크기 예측

앞의 3.2절에서 설명한 방법으로 허프만 복원 테이블을 축소하여도 임의로 주어진 모든 허프만 트리에 대하여 필요한 테이블의 크기를 예측하기는 곤란하다. 그러나 본 절에서는 주어진 허프만 트리가 오른쪽으로 기운 경우 필요한 테이블 크기의 상한값을 보인다. 이 결과는 트리가 왼쪽으로 기운 경우에도 마찬가지로 성립한다.

높이가 $K+1$인 오른쪽으로 기운 트리 T를 허프만 트리라고 하고 $\sum_{i=1}^{n} k_i = K$인 튜플 $B=(k_1, k_2, \ldots, k_n)$가 주어졌다고 하자. 임의의 부분 복원 트리 $TP(v)$에 대한 부분 복원 테이블 HT'에는 여전히 두 개 이상의 항이 서로 같은 내용을 가질 수 있는데 이 중 하나를 제외한 나머지 항들을 중복 항이라고 한다. 이러한 중복 항에는 항상 심볼이 저장되어 있고 이는 기억 장소의 낭비를 초래한다. 다음의 보조정리에서는 이러한 중복 항의 갯수에 대한 상한값을 보인다.

보조정리 1. 임의의 중간 복원 노드 v를 루트로 한 부분 복원 트리 $TP(v)$의 오른쪽 높이를 r, 왼쪽 높이를 l이라고 하자. 그러면 $TP(v)$에 대한 축소된 부분 복원 테이블 HT'에 존재하는 중복 항의 갯수는 많아야 $2^{r-1} - 2^{r-l} + l - r$이다.

증명. $TP(v)$를 포함하는 허프만 트리 T가 오른쪽으로 기운 트리이기 때문에 $TP(v)$ 역시 오른쪽으로 기운 트리이다. 따라서, HT'에 가장 많은 중복 항이 존재하기 위해서는 그림 6에 그 한 예를 보인 바와 같이 $TP(v)$의 형태는 다음과 같은 조건을 만족하여야 한다. 여기서 각 노드들의 레벨은 트리 $TP(v)$에서 결정된 레벨이다.

1. 레벨이 l보다 작은 모든 노드들은 각각 두 개의 자식 노드를 갖는다.

2. 레벨이 l보다 크거나 같은 모든 노드들의 경우, 같은 레벨에서 가장 오른쪽에 위치하는 노드는 두 개의 자식 노드를 갖고 나머지 노드들은 자식 노드를 갖지 않는다.

$TP(v)$에서 그 레벨 s가 r보다 작은 잎 노드에 대응하는 심볼은 HT^v의 2^{r-s} 개의 항에 나타나며, 따라서 이 심볼에 대한 $2^{r-s}-1$ 개의 중복 항이 존재한다. 예를 들어 그림 6의 심볼 x에 대응하는 잎 노드의 레벨은 $r-1$이므로 HT^v에 1 개의 중복 항이 존재한다. 이러한 사실로부터 앞에서 정의한 형태의 트리 $TP(v)$에 대한 $HT(v)$의 모든 중복 항의 갯수 M은 다음과 같다.

$$M=(2^{r-l}-1)(2^{l-1}-1)+\sum_{i=l+1}^{r-1}(2^{r-i}-1)$$
$$=2^{r-1}-2^{l-1}+l-r$$

끝으로, 만약 위에서 정의한 형태의 트리 $TP(v)$에서 레벨이 r이 아닌 어떤 잎 노드를 선택하여 이를 중간 노드로 바꾸고 두 개의 잎 노드를 그 중간 노드에 연결하여 새로운 부분 복원 트리 $TP'(v)$를 생성한다면($TP'(v)$는 오른쪽으로 기운 트리가 되어야 한다) $TP'(v)$에 대응하는 부분 복원 테이블의 중복 항의 갯수는 HT^v의 그것 보다 상대적으로 작다는 것을 쉽게 알 수 있다. 따라서, 오른쪽 높이와 왼쪽 높이가 각각 r이고 l인 부분 복원 트리 $TP(v)$에 대한 축소된 부분 복원 테이블 HT^v에 존재하는 중복 항의 갯수는 많아야 $2^{r-1}-2^{l-1}+l-r$이다. □

오른쪽으로 기운 허프만 트리 T와 튜플 B가 주어졌을 때 어떤 동일한 레벨을 갖는 중간 복원 노드들을 T에 위치한 순서에 따라서 왼쪽부터 오른쪽으로 v_1, v_2, ..., v_m 이라고 하자. 그리고 이러한 노드들을 루트로 한 부분 복원 트리 $TP(v_1)$, $TP(v_2)$, ..., $TP(v_m)$ 중에서 그 높이가 가장 큰 트리의 높이를 d라고 하자. 그러면 다음의 보조정리가 성립한다.

보조정리 2. $TP(v_1)$, $TP(v_2)$, ..., $TP(v_m)$에 대한 축소된 부분 복원 테이블 HT^{v_1}, HT^{v_2}, ..., HT^{v_m}에 존재하는 모든 중복 항의 갯수는 많아야 $2^{d-1}-d$이다.

증명. 우리는 이 보조정리를 귀납법으로 증명한다. 먼저 임의의 허프만 트리는 모든 중간 노드가 두개의 자식 노드를 갖으므로 $d=2$일 경우에는 $TP(v_1)$, $TP(v_2)$, ..., $TP(v_m)$ 모두 완전 2진 트리이고 따라서 중복 항이 존재하지 않는다. 그리고 $2^{2-1}-2=0$이므로 이 보조정리는 성립한다.

이제 k 보다 작은 모든 d에 대하여 이 보조정리가 성립한다고 가정하고 $d=k$일 경우를 고려하자. 트리 T는 오른쪽으로 기울었으므로 $i<j$인 모든 i,j에 대하여 $TP(v_i)$의 높이는 $TP(v_j)$의 높이보다 작거나 같다. $TP(v_1)$, $TP(v_2)$, ..., $TP(v_m)$ 중에서 그 왼쪽 높이가 오른쪽 높이 보다 작은 트리가 최소한 하나는 존재한다고 가정하고 이중 T에서 가장 오른쪽에 위치하는 부분 복원 트리를 $TP(v_p)$라고 하자. 그러면 당연히 $TP(v_p)$의 오른쪽 높이는 $d(=k)$이고 왼쪽 높이 l은 d 보다 작다. 만약 이러한 부분 복원 트리가 없다면 모든 부분 복원 테이블에 중복 항이 존재하지 않으므로 이 보조정리는 당연히 성립한다.

만약 $TP(v_p)$가 위의 조건을 만족하는 유일한 부분 복원 트리라면 보조정리 1에 의하여 중복 항의 갯수는 많아야 $2^{k-1}-2^{l-1}+l-k$이다. 그런데 l은 2 보다 크거나 같고 이러한 모든 l에 대하여 $-2^{l-1}+l\le 0$이므로 $2^{k-1}-2^{l-1}+l-k\le 2^{k-1}-k$이고 따라서 보조정리는 성립한다.

한편 $TP(v_p)$ 외에도 위의 조건을 만족하는 부분 복원 트리가 더 존재한다면 이들은 가정에 의하여 모두 $TP(v_p)$의 왼쪽에 존재하고 따라서 그들의 높이는 모두 l보다 작거나 같다. 그런데 $l<k$ 이므로 이러한 부분 복원 트리에 대한 부분 복원 테이블에 존재하는 모든 중복 항의 갯수는 많아야 $2^{l-1}-l$이다. 이것과 $TP(v_p)$에 대한 부분 복원 테이블에 존재하는 중복 항의 갯수를 합하면

$$2^{l-1}-l+2^{k-1}-2^{l-1}+l-k=2^{k-1}-k$$

보다 작거나 같다. □

허프만 복원 테이블의 크기를 예측하기 위하여 다음으로 생각하여야 할 사항은 테이블을 참조하기 위하여 필요한 기본주소를 저장하고 있는 항들의 갯수이다. 그런데 이것은 해당 허프만 트리에서 루트를 제외한 중간 복원 노드의 갯수와 동일함을 쉽게 알 수 있다. 따라서 우리는 허프만 트리에서 중간 복원 노드의 갯수의 상한 값을 구함으로써 기본주소를 저장하고 있는 항들의 갯수를 예측한다. 아래의 보조정리에서 이를 보인다.

보조정리 3. S개의 심볼에 대하여 높이가 $K+1$인 오른쪽으로 기운 허프만 트리 T와 $\sum_{i=1}^{n}k_i=K$인 튜플

$B=(k_1, k_2, \ldots, k_n)$가 주어졌다고 하자. T에 존재하는 루트가 아닌 모든 중간 복원 노드들의 갯수는 많아야

$$\lceil \frac{S}{2} \rceil + \sum_{m=1}^{n-2} \lceil \underbrace{\lceil \cdots \lceil}_{n-m} \frac{S}{2} \rceil \cdot \frac{1}{2^{k_{n-1}}} \rceil \cdot \frac{1}{2^{k_{n-2}}} \rceil \cdots \frac{1}{2^{k_{m+1}}} \rceil$$

이다.

증명. 트리 T에서 레벨이 $p_{n-1}=\sum_{i=1}^{n-1} k_i+1$ 보다 큰 잎 노드들이 많을수록 중간 복원 노드의 갯수가 많다는 것을 쉽게 알 수 있다. 따라서 실제 존재 가능성을 떠나서 T의 모든 잎 노드들의 레벨이 p_{n-1}보다 크다고 가정하자. 레벨이 p_{n-1}인 중간 복원 노드들은 각각 최소한 두 개의 잎 노드들을 그의 후손으로 가지고 있으므로 이들의 갯수 x_{n-1}은 많아야 $\lceil S/2 \rceil$개이다.

레벨이 $p_{n-2}=\sum_{i=1}^{n-2} k_i+1$인 중간 복원 노드들의 갯수를 알아보자. 만약 v가 레벨이 p_{n-2}인 하나의 중간 복원 노드라면 부분 복원 트리 $TP(v)$의 높이는 $k_{n-1}+1$이고 따라서 $TP(v)$는 $2^{k_{n-1}}$개의 잎 노드(즉, 레벨이 p_{n-1}인 중간 복원 노드)들을 가지고 있다. 따라서 레벨이 p_{n-2}인 중간 복원 노드들의 갯수를 x_{n-2}라고 하면 x_{n-2}는

$$2^{k_{n-1}} x_{n-2} = x_{n-1} \le \lceil \frac{S}{2} \rceil$$

를 만족하고 따라서

$$x_{n-2} \le \lceil \lceil \frac{S}{2} \rceil \cdot \frac{1}{2^{k_{n-1}}} \rceil$$

이다.

마찬가지 방법으로 레벨이 $p_{n-3}=\sum_{i=1}^{n-3} k_i+1$인 중간 복원 노드들의 갯수 x_{n-3}은

$$x_{n-3} = \lceil x_{n-2} \cdot \frac{1}{2^{k_{n-2}}} \rceil \le \lceil \lceil \lceil \frac{S}{2} \rceil \cdot \frac{1}{2^{k_{n-1}}} \cdot \frac{1}{2^{k_{n-2}}} \rceil$$

임을 쉽게 알 수 있고 일반적으로 레벨이 $p_m=\sum_{i=1}^{m} k_i+1$인$(1 \le m < n)$ 중간 복원 노드들의 갯수 x_m은

$$x_m \le \lceil \underbrace{\lceil \cdots \lceil}_{n-m} \frac{S}{2} \rceil \cdot \frac{1}{2^{k_{n-1}}} \rceil \cdot \frac{1}{2^{k_{n-2}}} \rceil \cdots \frac{1}{2^{k_{m+1}}} \rceil$$

이다.

이러한 과정에 의하여 계산한 각 해당 레벨에서의 중간 복원 노드들의 갯수 $x_1, x_2, \ldots, x_{n-1}$을 모두 더하면 이 값은 많아야

$$\lceil \frac{S}{2} \rceil + \sum_{m=1}^{n-2} \lceil \underbrace{\lceil \cdots \lceil}_{n-m} \frac{S}{2} \rceil \cdot \frac{1}{2^{k_{n-1}}} \rceil \cdot \frac{1}{2^{k_{n-2}}} \rceil \cdots \frac{1}{2^{k_{m+1}}} \rceil$$

이고 따라서 이 보조정리는 성립한다. □

마지막으로 다음의 정리에 제안한 복원방법을 위하여 필요한 허프만 복원 테이블의 크기에 대한 상한 값을 보인다.

정리. S개의 심볼에 대하여 높이가 $K+1$인 오른쪽으로 치우친 허프만 트리 T와 $\sum_{i=1}^{n} k_i = K$인 튜플 $B=(k_1, k_2, \ldots, k_n)$가 주어졌다면 이에 대한 허프만 복원 테이블은 많아야

$$\lceil \frac{3S}{2} \rceil + \sum_{m=1}^{n-2} \lceil \underbrace{\lceil \cdots \lceil}_{n-m} \frac{S}{2} \rceil \cdot \frac{1}{2^{k_{n-1}}} \rceil \cdot \frac{1}{2^{k_{n-2}}} \rceil \cdots \frac{1}{2^{k_{m+1}}} \rceil + \sum_{i=1}^{n} (2^{k_i} - k_i - 1)$$

개의 항을 갖는다.

증명. 허프만 복원 테이블의 항들의 갯수는 심볼의 갯수와 중간 복원 노드의 갯수 그리고 중복 항들의 갯수를 합한 것과 같다. 따라서 이 정리는 앞의 세 개의 보조정리에 의하여 당연히 성립한다. □

Ⅳ. 평 가

본 장에서는 앞의 3 장에서 제안한 허프만 부호 복원방법을 평가한다. 즉, 우리의 복원방법을 JPEG 표준에 적용하였을 때 필요한 허프만 복원 테이블의 크기를 여러가지 가능한 튜플 B에 대하여 각각 보이고 그 실시간 동작 가능성을 분석한다. 그런데 이를 위하여 JPEG 표준에 대한 설명이 필요한데 이는 지면관계상 생략한다[3].

1. 허프만 복원 테이블의 크기

JPEG 표준에서 허프만 부호화 하여야 할 심볼들은 이산 여현 변환 및 양자화 과정을 거친 AC 및 DC 계수에 대한 정보이다. 이러한 심볼들은 그 크기가 최대 $K=16$ 비트인 허프만 부호로 부호화 되며, 가능한 심볼의 최대 갯수는 DC의 경우에는 12이고 AC의 경우에는 162이다. 그런데 JPEG 표준에서는 영

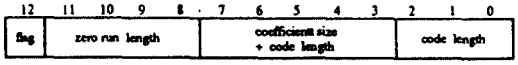
상의 luminance AC와 DC 그리고 chrominance AC와 DC 자료 각각에 서로 다른 허프만 부호를 사용하므로 네 개의 허프만 복원 테이블이 필요하다.

우리의 허프만 부호 복원방법을 JPEG 표준에 적용하기 위해서는 먼저 튜플 $B=(k_1, k_2, \ldots, k_n)$를 결정하여야 한다. 여기서 $K=16$이므로 $\sum_{i=1}^{n} k_i=16$이다. 튜플 B의 후보로서 한 개의 허프만 부호 복원에 필요한 최대 클럭수에 따라 (4, 4, 4, 4), (6, 6, 4) 그리고 (8, 8) 등을 들 수 있다. 물론, 이 외에도 많은 다른 가능한 튜플이 존재하나 허프만 부호 복원기를 일관성 있게 구현하기 위해서는 튜플 B의 원소중 처음 $n-1$개의 원소는 서로 같은 값을 갖게 하고 마지막 원소보다는 큰 값을 갖도록 설정하는 것이 좋다. 이러한 조건을 만족하면 입력 비트열에서 심볼이 발견될 때까지 각 클럭에서 같은 갯수의 비트를 쉬프트하여 필요한 비트열을 얻을 수 있으므로 회로를 설계하기가 용이하다[16].

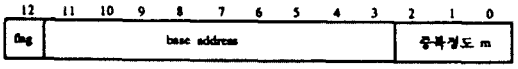
표 1에 이러한 튜플에 대하여 허프만 복원 테이블 구성에 필요한 항의 갯수의 상한값을 3.3 절의 정리에 의하여 계산한 결과를 보인다. 그런데, DC의 경우에는 심볼의 갯수가 많아야 12개 이므로 이들에 대한 허프만 부호의 최대 길이는 12 비트이다. 따라서, 튜플 (4, 4, 4, 4)의 경우 이러한 DC 계수들은 3 개의 클럭이내에 복원될 수 있기 때문에 실제로 튜플 (4, 4, 4)를 사용하는 것과 같다. 마찬가지로 튜플 (6, 6, 4)

표 1. JPEG용 허프만 복원 테이블 항의 갯수의 상한값.
Table 1. Upperbounds on the number of entries of the Huffman decoding tables for JPEG application.

튜플	상한값		
	AC	DC	합계(즉, $2\times AC+DC$)
(4, 4, 4, 4)	294	57	702
(6, 6, 4)	370	133	1,006
(8, 8)	737	512	2,498



(a) 심볼 정보가 기록된 경우.
(a) The case when symbol information is stored.



(b) 다음 테이블 참조를 위한 기본주소가 기록된 경우.
(b) The case when the base address for the next table reference is stored.

그림 7. JPEG에의 적용을 위한 허프만 복원 테이블의 형태.
Fig. 7. The structure of the Huffman decoding table for JPEG application.

표 2. 실제 영상에 대한 허프만 복원 테이블 구성에 필요한 항의 갯수.
Table 2. The number of necessary entries of the Huffman decoding tables for real images.

images	tuples								
	(4, 4, 4, 4)			(6, 6, 4)			(8, 8)		
	Lum. AC, DC	Chrm. AC, DC	total	Lum. AC, DC	Chrm. AC, DC	total	Lum. AC, DC	Chrm. AC, DC	total
standard	198,34	202,40	474	268,72	270,96	706	528,258	530,264	1580
a-6_75	92,20	64,24	200	160,64	76,64	364	530,256	282,256	1324
abwh_75	88,18	70,20	196	160,64	144,64	432	518,256	329,256	1359
acid_75	122,20	74,20	236	180,64	146,64	454	534,256	390,256	1436
acmeteam_75	92,20	62,20	194	164,64	136,64	428	522,256	328,256	1362
adpcop_75	84,20	46,18	168	152,64	98,64	378	334,256	268,256	1114
africa_75	84,18	16,15	133	154,64	63,66	374	400,256	255,256	1167
afterbrrner_75	104,20	68,16	208	160,64	138,64	426	350,256	278,256	1140
agatest_75	92,24	60,18	194	168,66	136,64	134	522,256	326,256	1360
agong_75	86,20	56,16	178	162,64	132,64	422	522,256	290,256	1324
airbrush_75	84,18	74,20	196	158,64	146,64	432	518,256	322,256	1352
total(max)	474			706			1580		

의 경우에는 실제로 튜플 (6, 6)을 사용하는 것과 같다. 표 1에 보인 상한값은 이러한 사실을 고려하여 계산한 값이다.

표 1에 보인 상한값의 타당성을 조사하기 위하여 다수의 영상을 JPEG 표준에 의하여 압축한 후 이를 복원하는데 필요한 항의 갯수를 계산하였다. 표 2에 이러한 결과를 보인다. 표에서 보인 바와 같이 실제로 필요한 항의 갯수는 표 1의 상한값보다 상당히 작다.

그림 7에 튜플 (6, 6, 4)를 사용하였을 때 허프만 복원 테이블의 형태를 보인다[16]. 그림에서 보는 바와 같이 테이블의 각 항을 13 비트로 구성할 수 있기 때문에 허프만 복원 테이블을 구성하는데 1,006×13 =13,078 비트면 충분하며 이는 Luthi 등[11]이 제안한 복원방법에서 사용한 테이블 크기의 40%에 불과하다. 표 3에 이러한 사실을 보인다.

표 3. Luthi 등[11]의 방법과의 비교.
Table 3. Comparison with the results by Luthi[11].

복원방법	테이블 크기	비트수
우리의 방법	1,006×13	13,078 비트
Luthi의 방법	2,720×12	32,640 비트

2. 실시간처리 가능성

JPEG 복호기에서 허프만 부호 복원기의 출력에 연결되어 있는 기능모듈은 역 양자화기, 역 이산 여현 변환기 등을 들 수 있는데 복호기의 실시간 동작을 위해서는 허프만 부호 복원기가 매 클럭당 한 개의 계수를 연속적으로 공급하여 이들의 입력을 만들 수 있어야 한다. 그러나 제안한 복원방법에서는 하나의 허프만 부호를 복원하는데 두 개 이상의 클럭이 필요한 경우도 있으므로 이러한 조건을 만족하지 못한다. 그런데 JPEG 표준에서 허프만 부호화된 심볼에는 연속된 제로계수의 갯수(zero run length)에 관한 정보가 포함되어 있고 이는 허프만 부호 복원기가 한 개의 심볼을 복원하여 한개 이상의 계수를 출력할 수 있음을 의미한다. 따라서 허프만 부호 복원기의 출력에 적당한 크기의 버퍼를 설치함으로써 제안한 복원방법을 실시간 처리에 사용할 수 있다.

튜플 (4, 4, 4, 4), (6, 6, 4) 그리고 (8, 8) 등을 채택한 허프만 부호 복원기의 출력에 각각 4, 8 그리고 16개의 복원된 심볼을 저장할 수 있는 버퍼를 설치하였을 때, 다수의 실제 영상에 대하여 그 실시간 처리 가능성을 조사하였다. 이를 위하여 본 논문에서 제안한 방법에 의한 허프만 부호 복원기와 출력 버퍼 그리고 기타 입출력 장치 등을 시뮬레이션하기 위한 프로그램을 C 언어로 작성하였다. 작성한 프로그램은 JPEG 표준으로 압축된 영상에 대해 허프만 부호 복원기가 이의 출력 버퍼를 통하여 복원기의 다음 모듈에 지속적으로 자료를 (즉, 매 클럭당 한 개의 계수를) 공급하는지의 여부를 판단한다. 여기서 다음단으로의 자료공급은 복원기의 출력 버퍼에 반 정도의 복원된 심볼이 채워져 있을때 시작하며 그 후 부호의 복원 도중에 출력 버퍼의 언더플로가 발생하면 실시간 처리가 가능하지 않다고 판단한다. 그리고 만약 출력 버퍼가 가득차서 더이상 심볼을 저장할 공간이 존재하지 않을 경우에는 빈 공간이 생길 때 까지 허프만 부호 복원기의 동작을 잠시 중단한다.

표 4에 이러한 실험결과를 보인다. 표에서 보인 바와 같이 출력 버퍼의 크기를 32 정도로 하였을 경우 우리의 모든 튜플에 대하여 실시간 처리가 가능하다. 그러나 출력 버퍼의 크기를 4로 한 경우에는 튜플 (4, 4, 4, 4)로는 실시간 처리를 할 수 없으며 튜플 (6, 6, 4) 역시 약간 불안하다. 끝으로 출력 버퍼의 크기를 8 이상으로 하면 튜플 (6, 6, 4)를 실시간 처리에 사용할 수 있다.

3. 허프만 부호 복원기의 구조

본 절에서는 제안한 허프만 부호 복원방법의 구현 가능성을 보이기 위하여 JPEG 표준에 의하여 압축된 영상 자료를 복원하는데 사용할 수 있는 튜플 (6, 6, 4)를 사용한 허프만 부호 복원기의 구조를 보이고 이의 동작원리를 설명한다[16].

그림 8에 이러한 허프만 부호 복원기의 주요 자료흐름도를 보인다. 그림에서 복원부분에 필요한 자료를 공급하는 입력 인터페이스 부분은 IBR, 배럴 쉬프트 BSi, 5 비트 가산기, 그리고 BSi의 현재 제어신호를 저장하기 위한 레지스터 ACC 등으로 구성되어 있다. 여기서 IBR은 필요시 16 비트를 입력받아 32 비트의 자료를 배럴 쉬프터에 지속적으로 공급하는 기능을 수행하며 단순한 레지스터가 아닌 몇 개의 레지스터와 조합회로로 구성된 하나의 기능모듈이다. 또한 BSi의 제어신호 BSiCtr은 5 비트로 구성되어 있으며 BSi가 쉬프트 하여야할 비트수를 나타낸다.

복원부분은 입력 인터페이스에서 제공된 비트열을 이용해서 실제 복원을 수행하는 부분이다. 그림 8에서 복원부분은 매스커(masker), 테이블 기본주소 저

표 4. 여러 튜플에 대한 실시간 처리 가능성.
Table 4. Possibility of real time processing for various tuples.

	tuples											
	(4, 4, 4, 4)				(6, 6, 4)				(8, 8)			
	buffers				buffers				buffers			
images	4	8	16	32	4	8	16	32	4	8	16	32
a-6_75	u	o	o	o	o	o	o	o	o	o	o	o
abwh_75	u	o	o	o	o	o	o	o	o	o	o	o
acid_75	u	o	o	o	o	o	o	o	o	o	o	o
acmeteam_75	u	o	o	o	o	o	o	o	o	o	o	o
adpcop_75	u	o	o	o	o	o	o	o	o	o	o	o
africa_75	u	o	o	o	o	o	o	o	o	o	o	o
afterbrrner_75	u	u	o	o	o	o	o	o	o	o	o	o
agatest_75	u	o	o	o	o	o	o	o	o	o	o	o
agong_75	u	o	o	o	u	o	o	o	o	o	o	o
airbrush_75	u	u	o	o	o	o	o	o	o	o	o	o
o : Normal operation, u : Underflow												

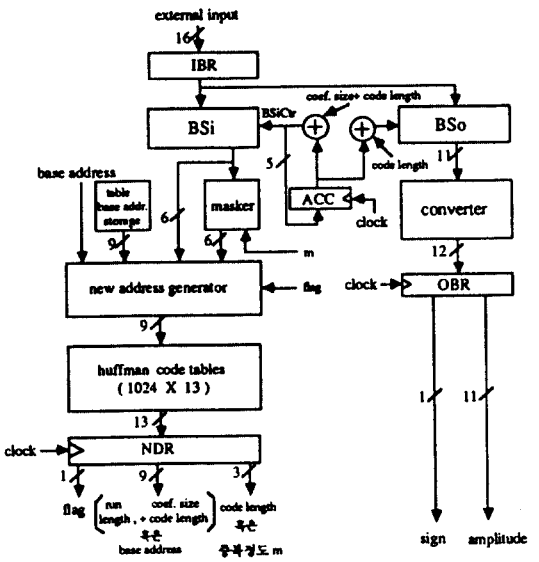


그림 8. JPEG용 허프만 부호 복원기의 주요 자료 흐름도.
Fig. 8. The basic data flow diagram of the Huffman code decoder for JPEG application.

장기(table base address storage), 새 주소 발생기(new address generator), 허프만 복원 테이블(Huffman code table) 그리고 13 비트 레지스터 NDR 등으로 구성되어 있다.

매스커는 현재 참조한 허프만 복원 테이블의 항이 다음 테이블 참조를 위한 기본주소를 포함하고 있을 때 이와 함께 포함된 중복 정도 m의 값을 이용하여 BSi 출력 중 필요한 비트만 취하여 다음 테이블 참조를 위한 오프셋을 구하는 기능을 한다. 테이블 기본주소 저장기는 그 크기가 4×9 비트인 레지스터로서 영상의 luminance 자료에 대한 AC와 DC 계수 그리고 chrominance 자료에 대한 AC와 DC 계수 등을 각각 복원하기 위하여 처음 참조하여야 할 허프만 복원 테이블의 기본주소 4 개를 포함하고 있다. 새 주소 발생기는 기본주소와 오프셋을 더하여 다음 테이블 참조를 위한 주소를 만들어내는 기능을 수행한다. 그리고 허프만 복원 테이블은 앞에서 설명한 바와 같으며 NDR은 새 주소 발생기에 의하여 얻은 주소로 허프만 복원 테이블을 읽은 항이 저장된다.

마지막으로 출력 인터페이스 부분은 배럴 쉬프트 BSo, 4 비트 가산기, 자료 변환기(converter) 그리고 출력 레지스터 OBR 등으로 구성되어 있으며 현재 복원된 부호 다음에 이어지는 계수값에 해당하는 비트열을 12 비트의 2진 보수로 변환한다. 이렇게 변환된 값은 출력 레지스터 OBR에 저장되어 출력된다.

지금까지 기술한 허프만 부호 복원기는 클럭의 한 주기 동안 두 가지 일을 반복적으로 수행한다. 클럭의 falling edge부터 시작하여 반 주기 동안 허프만 복원 테이블을 참조하기 위한 주소를 계산하고 다음 반 주기 동안에 허프만 복원 테이블을 읽어서 그 값을 클럭의 falling edge에서 NDR에 저장한다. 만약

이때 플래그(NDR의 비트 12) 값이 0이라면 NDR의 내용을 이용하여 BSo을 제어함으로써 IBR로부터 현재 복원된 부호 다음에 연속되는 0이 아닌 계수값에 대한 비트열을 취하고 자료 변환기를 거쳐 반 클럭 후에 이 계수값과 NDR에 저장되어 있는 zero run length 값을 출력한다.

이와 동시에 BSi의 제어신호 BSiCtr은 다음 허프만 부호의 복원을 시작할 수 있도록 조정되어야 한다. 이를 위하여 현재의 BSiCtr 값에 NDR에 저장되어 있는 coefficient size와 code length를 더하여 새로운 BSiCtr 값으로 한다. 이후 IBR의 (BSiCtr+1)번째 비트로부터 6비트를 BSi를 통하여 구하고 이 값과 테이블 기본 주소 저장기의 기본주소를 더하여 허프만 복원 테이블을 참조하기 위한 새로운 주소를 생성한다.

한편, 플래그 값이 1인 경우에는 IBR로부터 다음 6 비트를 가져와 복원을 계속하여야 한다. 따라서 BSiCtr의 값은 현재의 값에 6을 더하여 새로운 값으로 한다. 새로운 BSiCtr 값으로 BSi를 제어하여 IBR로부터 가져온 6 비트의 자료는 NDR에 저장되어 있는 중복정도 m 값에 의하여 매스킹된 후 역시 NDR에 저장되어 있는 기본주소에 더하여 다음 허프만 복원 테이블을 참조하기 위한 주소를 만든다. 이러한 과정 중에서 만약 새로운 BSiCtr 값이 16 보다 크게 되면 다음 클럭에서 BSiCtr로부터 16을 빼고 IBR을 16 비트만큼 왼쪽으로 쉬프트함과 동시에 IBR의 오른쪽 16 비트에 새로운 비트열을 입력 받아 복원을 계속 수행할 수 있도록 한다.

지금까지 설명한 허프만 부호 복원기의 동작을 확인하기 위하여 그림 9에 보인 바와 같은 VHDL 모델을 구현하였다. 그림에서 중앙에 보인 모듈은 제안한 허프만 부호 복원기를 VHDL의 structural model로 묘사한 것이며 이는 NAND 게이트를 기준으로 약 5,000 개의 게이트와 13,078 비트의 RAM으로 구성되어 있다. 한편, Front_end는 복원기에 필요한 제어신호와 입력 비트열을 보내고 복원을 시작하기 전에 허프만 복원 테이블을 구성하는 등의 일을 수행하기 위한 모듈이며, Back_end는 버퍼의 상태를 감시하고 복원기로부터 전달된 복원된 심볼들을 저장하는 기능을 수행하는 부분이다. 그리고 복원기와 Back_end 간의 속도문제를 완화하기 위해 적당한 크기의 버퍼를 그 사이에 설치하였다. 이러한 Front_end와 Back_end 그리고 버퍼는 VHDL의 behavioral model로 구현하였다. 이렇게 구성된 VHDL 모델을 JPEG 표준으로 압축한 다수의 영상을 입력으로한 시뮬레이션을 수행하여 그 동작을 확인하였다[16].

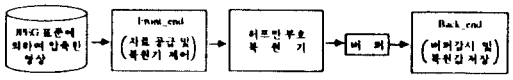


그림 9. 제안한 방법에 의한 허프만 부호 복원기를 시험하기 위한 VHDL 모델.

Fig. 9. VHDL model for testing of our Huffman code decoder.

V. 결 론

본 논문에서는 비교적 적은 량의 하드웨어를 사용하면서 실시간 처리에 사용할 수 있는 새로운 허프만 부호 복원방법을 제안하였다. 새로 제안한 허프만 부호 복원방법에서는 먼저 정수로 구성된 튜플 B를 정의하고 입력된 비트열로부터 B에서 정의된 수 만큼의 연속된 비트를 반복적으로 취하여 이를 토대로 허프만 복원 테이블을 참조하여 복원을 수행한다. 이 방법은 하나의 부호를 복원하는데 한 개이상의 클럭이 필요하지만 MPEG 또는 JPEG 표준에서 정의된 각 심볼은 연속된 제로 계수의 갯수를 포함하고 있으므로 복원기의 출력단에 약간의 버퍼를 장치함으로써 실시간 처리가 가능하다. 또한 이 방법은 주어진 허프만 트리가 한쪽으로 기운 형태를 갖는 경우 허프만 복원 테이블의 크기를 작게 할 수 있다. 실제로 본 논문에서는 이러한 허프만 트리에 대하여 필요한 허프만 복원 테이블 크기의 상한값을 보였다. 제안한 허프만 부호 복원방법을 JPEG 표준에 의하여 압축된 영상을 복원하는데 사용할 수 있도록 회로를 설계하는 것은 약 5,000 개의 게이트와 1024×13 비트의 RAM으로 가능하며 이 중 RAM의 크기는 Luthi 등[11]이 제안한 허프만 부호 복원기에서 사용하는 RAM의 크기에 비하여 약 40%에 불과하다.

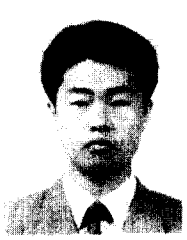
참 고 문 헌

1. M. Bolton, R. Boulton, J. Marting, S. Ng and S. Turner, "A Complete Single-Chip Implementation of JPEG Image Compression Standard," Proc. CICC, 1991, pp. 12.2.1-12.2.4.
2. S.-F. Chang and D. G. Mersserschmitt, "VLSI

Designs for High-Speed Huffman Decoder," Proc. ICCD, 1991, pp. 500-503.

3. *Digital Compression and Coding of Continuous-tone Still Images*, ISO Committee Draft 10918-1, Jan. 1991.
4. *Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to about 1.5 Mbit/s*, ISO Committee Draft 11172-2, Nov. 1991.
5. E. Horowitz and S. Sahni, *Fundamentals of Data Structures in Pascal*, 2nd edition, Computer Science Press, 1987.
6. A. K. Jain, *Fundamentals of Digital Image Processing*, Prentice-Hall, 1989.
7. S.-M. Lei and M.-T. Sun, "An Entropy Coding System for Digital HDTV Application," IEEE Trans. on Circuits and Systems for Video Technology, Vol. 1, No. 1, March 1991. pp. 147-155.
8. H.-D. Lin and D. G. Messerschmitt, "Improving the Iteration Bound of Finite State Machines," Proc. ISCAS-89, 1989, pp. 1328-1331.
9. H.-D. Lin and D. G. Messerschmitt, "Finite State Machine has Unlimited Concurrency," IEEE Trans. on Circuits and Systems, Vol. 38, No. 5, May 1991, pp. 465-475.
10. H.-D. Lin and D. G. Messerschmitt, "Designing a High-Throughput VLC Decoder Part II-Parallel Decoding Methods," IEEE Trans. on Circuits and Systems for Video Technology, Vol. 2, No. 2, June 1992, pp. 197-296.
11. D. A. Luthi, P. Tong and P. A. Ruetz, "A Video-Rate JPEG Chip Set," Proc. CICC, 1992, pp. 26.2.1-26.2.4.
12. A. Mukherjee, N. Ramganathan and M. Bassiouni, "Efficient VLSI Designs for Data Transformation of Tree-Based Codes," IEEE Trans. on Circuits and Systems, Vol. 38, No. 3, March 1991, pp. 306-313.
13. K. K. Parhi, "High-Speed VLSI Architectures for Huffman and Viterbi Decoders," IEEE Trans. on Circuits and Systems_II : Analog and Digital Signal Processing, Vol. 39, No. 6, June 1992, pp. 385-391.
14. M.-T. Sun, "VLSI Architecture and Implementation of A High-Speed Entropy Decoder," Proc. ISCAS-91, 1991, pp. 200-203.
15. M.-T. Sun, K.-M. Yang and K.-H. Tzou, "High-Speed Programmable ICs for Decoding of Variable-Length Codes," Proc. of SPIE, Vol. 1153, Applications of Digital Image Processing XII, 1989, pp. 28-39.
16. 임종석 외 3인, 영상 압축/복원용 허프만 부호화 구조에 관한 연구, 최종보고서, 한국전자 통신연구소, 1993.

본 논문은 1993년도 한국전자통신연구소의 지원을 받아 수행한 연구임.



金 炳 漢(Byoung Han Kim) 正會員
1991년 2월 : 부산대학교 전산통계학과(공학사)
1993년 8월 : 서강대학교 전자계산학과(공학 석사)
1993년 8월 ~ 현재 : 삼성전관 정보제품 연구소 연구원
※주관심분야 : 디지탈 영상처리, 디지탈 신호처리 및 응용.

林 鍾 錫(Chong Suck Rim) 正會員
1981년 2월 : 서강대학교 전자공학과(공학사)
1983년 2월 : 한국과학기술원 전기 및 전자공학과(공학석사)
1989년 5월 : Univ. Maryland 전기공학과(공학 박사)
1983년 3월 ~ 1990년 8월 : 한국전자통신 연구소 연구원
1990년 9월 ~ 현재 : 서강대학교 전자계산학과 조교수
※주관심분야 : VLSI CAD, VLSI 설계