An efficient search of binary tree for huffman decoding based on numeric interpretation of codewords

Byeong-Il Kim[1], Tae-Gyu Chang[1] and Jong-Hoon Jeong[2]
[1] School of Electric and Electrical Engineering, Chung-Ang University
#221 Heukseuk-dong Dongjak-gu Seoul 156-756, Korea,
Tel. : +822-820-5318, Fax : +822-812-1293
e-mail : tgchang@cau.ac.kr
[2]Samsung Electronics Co. Ltd.,
#416 Maetan-3Dong, Paldal-Gu, Suwon City, Kyngki-Do,442-742, Korea

Abstract: *This paper presents a new method of Huffman decoding which gives a significant improvement of processing efficiency based on the reconstruction of an efficient one-dimensional array data structure incorporating the numeric interpretation of the accrued codewords in the binary tree. In the proposed search method, the branching address is directly obtained by the arithematic operation with the incoming digit value eliminating the compare instruction needed in the binary tree search.*

The proposed search method gives 30% of improved processing efficiency and the memory space of the reconstructed Huffman table is reduced to one third compared to the ordinary 'compare and jump' based binary tree. The experimental result with the six MPEG-2 AAC test files also shows about 198% of performance improvement compared to those of the widely used conventional sequential search method

1. Introduction

Binary tree search can be regarded as a promising means for Huffman decoding owing to its inherent advantages of searching efficiency and narrowed variations of time required for the search of an individual codeword [1][2]. However, the binary tree search suffers from the complexities involved with the construction and handling of the linked-list based binary tree [3][4]. Especially, the comparison and the jump, which are the key operational units constituting the binary tree search, play as the major causes of decreased processing efficiency by disturbing the flow of pipelining and by adding the coding complexity in general.

It is well known that the search efficiency can be significantly improved if the search can be aided or guided by some numeric operation, as exemplified by the data hashing and the content addressable memory techniques [5][6]. Recently published several variants of the binary tree search can be considered as an effort to improve the processing efficiency by partially adopting the numeric aid in the search [5][6].

This paper presents a means to completely eliminate the use of conditional statement in the branching operation of the Huffman decoding, consequently giving the significant improvement of processing efficiency compared to the ordinary 'compare and jump' based binary tree search method. The branching address can be obtained by the direct arithmetic operation with the incoming binary digit, importantly, without invoking any condition checking statement. The proposed search technique results from the efficient one-dimensional array data structure devised by adopting the numeric interpretation of the accrued codewords in the tree. The non-skewed characteristics of the Huffman tree are the major feature that enables the realization of the proposed search technique. The non-skewed characteristics refers to that of the tree in which a node branches always to a paired child nodes having the values of zero(0) and one(1), respectively.

2. Huffman Decoding Based on Numeric Interpretation of Codewords

Huffman decoding is based on the reconstruction of the efficient one-dimensional array data structure. Figure 1 shows the construction of the binary tree data structure in a form of a static one-dimensional array, where the array index assigned for each node follows the order of layer in the tree first then the accrued code values within the same layer, i.e., from left to right in the tree. Therefore, the array index follows the increasing order of the accrued code values. For each leaf node, the return value replaces the binary digit associated with the node.

Based on the reconstructed array, the branching address can be obtained by the direct arithmetic operation with the incoming binary digit as shown in equation (1).

$$addr := addr + DATA(addr) + new\_digit() \quad (1)$$

Where *addr* indicates an array index of the Huffman decoding table, *DATA(addr)* is the value of the array element referenced by the *addr*, and *new_digit()* refers to the function which returns the value of the newly read digit from the bitstream to decode.

The Huffman codeword table, which is illustrated in table 1, is constructed based on the array depicted in figure 1. The constructed table consists of one dimensional array where each element contains either the corresponding offset value to jump for the next search or the return value in case for the leaf node. The two columns with the headings of 'left-0' and 'right-1', which are not part of the table but are included only for illustration, show the absolute branch address values for the next search associated with the value of codeword's new binary input digit, i.e., zero and one, respectively.

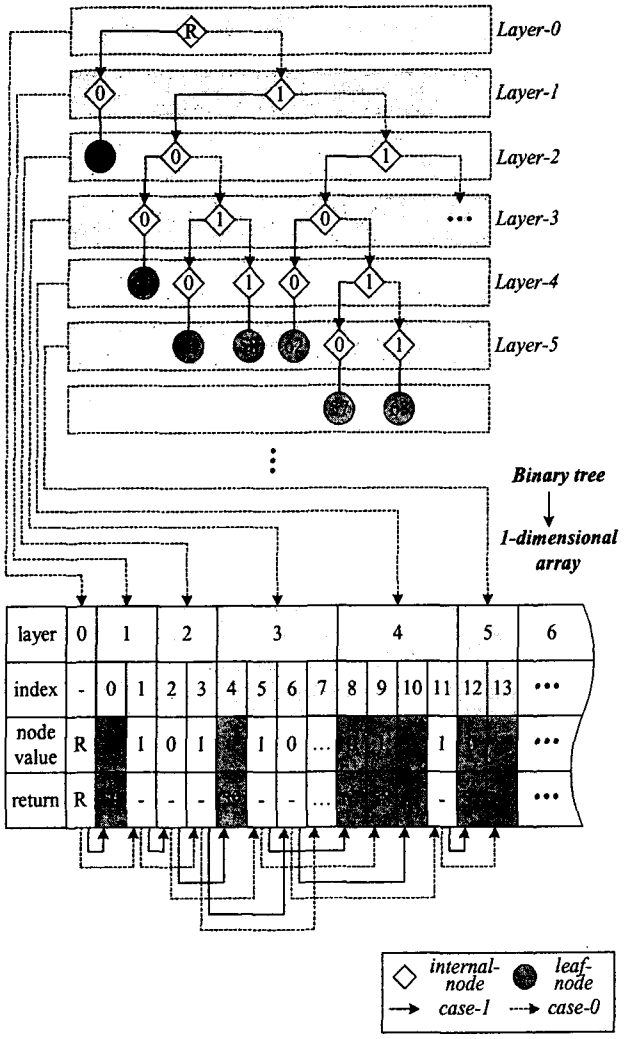

Figure 1. Reconstruction of the efficient one-dimensional array data structure from the Huffman tree.

The required memory space is reduced to one third compared to the ordinary binary tree, where, for each node, three memory locations are required to hold the address for left branch, the address for right branch, and the data for the node, respectively. On the other hand, the proposed array requires only one memory location for each node, since only the offset value to jump to the left branch node or the return value for the leaf node is stored as shown by the box marked with bold lines.

The Huffman decoding procedure based on the proposed numeric computation of the branch offset address using the equation (1) is described by the flow diagram shown in figure 2. The Huffman decoding procedure based on the ordinary 'compare and jump ' based binary tree search is described by the flow diagram shown in figure 3.

As compared in the two flow diagrams, the compare statement needed for the branch in the conventional search can be eliminated in the proposed search method, yielding about 33% of improvement in processing efficiency as expressed in the following.

Table 1. One-dimensional array structure reconstructed from the Huffman tree.

index	left-0	right-1	offset () : right-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
...	...	...	...	...

Percentage improvement of the processing efficient

$$\triangleq\left(1-\frac{\#\text{of instructions per search of the proposed method}}{\#\text{of instructions per search of the conventional method}}\right)\times 100$$

$$=\left(1-\frac{2}{3}\right)\times 100\approx 33[\%]$$

The actual improvement of processing efficiency becomes much higher than 33% if the pipelining stall effect of the compare and the jump instructions is considered.

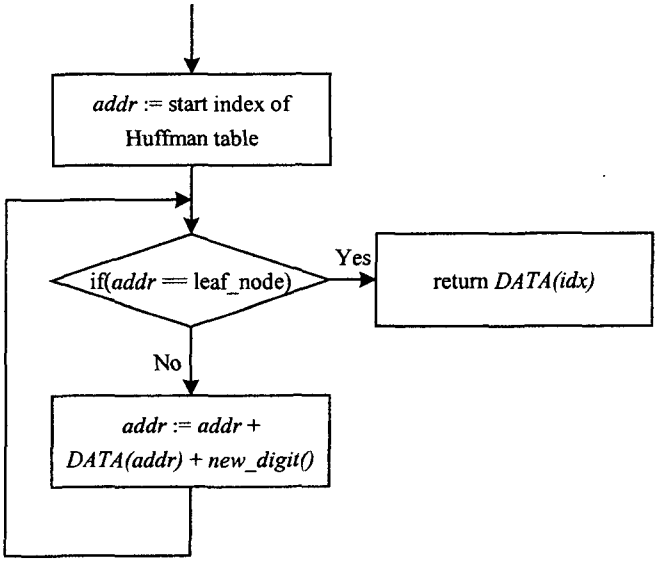

Figure 2. Flow diagram of the Huffman decoding based on the numerical computation of the branch address.

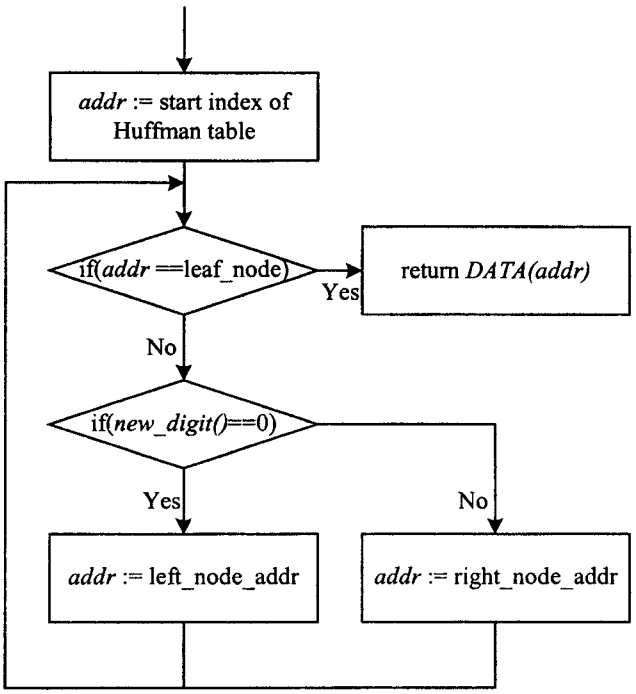

Figure 3. Flow diagram of the Huffman decoding based on the 'compare and jump' instruction.

3. Experimental Results : MPEG-2 AAC

The proposed Huffman decoding method is realized for the MPEG-2 AAC decoder, and the average required number of searches is measured for the performance evaluation of the Huffman decoder. Six MPEG-2 AAC encoded test files provided in the MPEG website are used in the performance evaluation.

To provide a practically realistic figure of the performance improvement, the performance results of the suggested Huffman decoding method are compared with those of the sequential search based Huffman decoding, which is in wide use for its implementation simplicity.

Twelve different Huffman codebooks are used in the MPEG-2 AAC decoder to achieve the optimized compression reflecting the human pshychoacoustics according to the signal characteristics in different frequency bands.

The codebook-wise results of the average number of searches are compared for the two Huffman decoding methods in table 2. It is shown, for this specific MPEG-2 AAC test file, that the performance improvement ranges from 30% to 777% depending on the codebook. In average, the proposed search method achieves 75% of performance improvement over the conventional sequential search method for this specific MPEG-2 AAC test file.

The same performance evaluation is carried for the six different MPEG-2 AAC test files and the results are summarized in table 3. It is shown that the improvement of the processing efficiency of the proposed search method ranges from 24% to 776% depending on the files. Such

Table 2. Results of the average number of searches for each Huffman table

codebook number	search depth/ table size	total number of searches	average number of searches		percentage improvement (%)
			proposed search method operation	sequential search method	
scalefactor	19/121	14022	3.91	5.09	30.01
1	16/81	5624	4.51	12.50	177.98
2	16/81	7857	5.39	18.61	245.11
3	16/81	16170	3.77	6.46	71.11
4	16/81	8680	4.77	11.42	139.32
5	16/81	28794	3.08	4.51	46.35
6	16/81	1086	5.17	14.04	172.58
7	16/64	2958	2.64	3.29	24.80
8	16/64	582	4.81	11.13	131.28
9	16/169	234	2.94	5.09	73.26
10	16/169	2594	6.51	33.60	415.63
11	16/289	7640	7.41	65.02	776.43

Table 3. Performance comparison results for the six MPEG-2 AAC test files.

MPEG-2 AAC test files	total number of searches		percentage improvement (%)
	proposed search method	sequential search method	
L1_44100.aac	407,591	1,168,714	187
L2_44100.aac	404,608	1,238,404	206
M1_32k.aac	360,727	1,236,965	243
M1_44k.aac	371,423	941,376	153
Sin2_32000.aac	174,997	511,710	192
Sin2_44100.aac	239,362	740,700	209
average	**326,451**	**972,978**	**198**

performance improvement of the Huffman decoding can be considered as a significant result considering that the Huffman decoding is one of the major units occupying the most of the resources required for the MPEG-2 AAC decoder.

4. Conclusions

This paper presents a new method of Huffman decoding which gives a significant improvement of processing efficiency compared to the ordinary 'compare and jump' based binary tree search method as well as to the sequential search method. In the proposed search method, the branching address is directly obtained by the arithmetic operation with the incoming digit value eliminating the

compare instruction needed in the binary tree search. The realization of the proposed search is based on the reconstruction of an efficient one-dimensional array data structure incorporating the numeric interpretation of the accrued codewords. The improved processing efficiency is about 33% and the required memory space of the reconstructed Huffman table is reduced to one third compared to the ordinary 'compare and jump' based binary tree.

The proposed Huffman decoding method is applied together with the sequential Huffman decoding method to the implementation of the MPEG-2 AAC decoder for the purpose of performance evaluation. Through the experimental results with the six MPEG-2 AAC test files, it is shown that, in average, 198% of performance improvement is achieved compared to those of the sequential method. The performance improvement of the proposed Huffman decoding can be considered as a significant result considering that the Huffman decoding is one of the major units occupying the most of the resources required by the MPEG-2 AAC decoder.

References

[1] S.HO and P.LAW, "Efficient hardware decoding method for modified Huffman code", Electronics letters, 1991, Vol.27, pp. 855-856

[2] K. L. Chung and J. G. Wu, "Level-compressed Huffman decoding", IEEE, 1999, vol. 47, pp. 1455-1457

[3] W.Hamming, "Coding and information theory", prentice-hall, 1986, pp. 51-78

[4] Khalid sayood, "Introduction to data compression", morgan kaufmann, 1996, pp 39-73

[5] H.D.Lin and David G, "High throughput reconstruction of Huffman-coded images", IEEE, 1989, pp. 172-175

[6] Keshab K. Parhi, "High speed Huffman decoder architectures", IEEE, 1991, pp. 64-68

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

[8] M.bosi and K.Brandenburg, ISO/IEC MPEG-2 Advanced Audio Coding, J. Audio Eng. Soc, 1997, Vol.45, pp 789-814