International Journal of Fuzzy Logic and Intelligent Systems, vol. 4, no. 1, June 2004 pp. 34-39

Efficient Multi-way Tree Search Algorithm for Huffman Decoder

Hyungtai Cha and Kwanghee Woo

School of electronic engineering, Soongsil university, Seoul, Korea

Abstract

Huffman coding which has been used in many data compression algorithms is a popular data compression technique used to reduce statistical redundancy of a signal. It has been proposed that the Huffman algorithm can decode efficiently using characteristics of the Huffman tables and patterns of the Huffman codeword. We propose a new Huffman decoding algorithm which used a multi way tree search and present an efficient hardware implementation method. This algorithm has a small logic area and memory space and is optimized for high speed decoding. The proposed Huffman decoding algorithm can be applied for many multimedia systems such as MPEG audio decoder.

Key-word : Huffman decoder, Multi-way search, Huffman Hardware Design.

1. Introduction

The Huffman coding algorithm is mostly used in data compression to reduce statistical redundancy. Most of its compression applications use only one codebook, while MPEG audio system uses many codebooks to process an audio signal for a whole frequency range[1]-[4]. The fast Huffman coding algorithm should be applied for a real time process of a decoder and an algorithm implemented by a small logic area and small memory is required to minimize hardware.

This paper is organized as follows: Section 2 introduces the conventional Huffman decoding algorithm. Section 3 presents the Huffman decoder using multi way tree search algorithm. We properly optimize to implement the proposed decoding algorithm into hardware and apply it to the MPEG audio decoder in Section 4. We verify the proposed algorithm through the experiment and result in Section 5. Finally, a conclusion is drawn in Section 6.

2. Conventional Huffman Decoder

Many algorithms has been proposed for efficient Huffman decoding. Also, this algorithm has been used in real system very often because of the effectiveness of the coding structure. These are mostly separated into two methods, the tree search algorithm [5,7] and the bit-paralleled algorithm [8-11].

2.1 Tree search algorithm

The tree search algorithm is generally used in the Huffman decoder. In this method, a codeword of a tree structure is searched after reading one bit from a packet bit stream. It could be implemented by the finite state machine (FSM) [5]. The method compares the state from the input bit with a codeword of variable length. Therefore its decoding period is not constant, and it requires too much time to decode a codeword of long length.

2.2 Bit-paralleled algorithm

The bit-paralleled algorithm is the fastest method among methods of hardware implementation. It can decode one codeword per clock cycle by comparing the input codeword with its hardware logic. After a common bit pattern in the input bit stream is matched, it is used as the first parameter for searching memory. The remaining codeword except for the common bit pattern in the input bit stream, is used as the second parameter for searching memory. The bit-paralleled algorithm has advantages due to a constant period that is irrespective of the input bit stream.

However, the size of the hardware logic become larger as the size of the codebook is increased. This is due to the fact that the bit-paralleled algorithm should match the input bit stream and its logic. A MEPG audio system uses too many codebooks for the bit-paralleled algorithm to be an efficient decoding method. The Huffman coder of the MPEG-2 audio AAC has eleven codebooks of spectrum data, and one codebook of scale factor data. The total number of the symbols are 1,362.

3. Huffman Decoder Using Multi-way Tree Search

We propose a multi way tree search algorithm for the Huffman decoder. This algorithm results in the smaller sized hardware and faster speed. We use the FSM to search the tree and the Huffman table which is saved on memory in advance by representing each states with FSM. Because the input bit stream is used as the memory address, it can be compared with the memory immediately. To increase the speed of the

Manuscrip received Mar. 29, 2004; revised May. 21, 2004. This work was supported by the Soongsil University Research Fund.

decoder, we make the decoder process several bits in one clock cycle.

3.1 The binary tree search without the memory search

Table 1 shows a 16 length codeword per codebook from the Huffman codebook 1~3 of the MPEG-2 audio ACC by length [1].

Table 1. Huffman table for MPEG-2 audio AAC

Codebook #1			Codebook #2			Codebook #3		
Index	Codeword	Len.	Index	Codeword	Len.	Index	Codeword	Len.
28	0	1	28	000	3	00	0	1
43	10000	5	43	0010	4	27	1000	4
0D	10001	5	0D	00110	5	01	1001	4
27	10010	5	29	00111	5	09	1010	4
31	10011	5	25	01000	5	03	1011	4
29	10100	5	27	01001	5	24	11000	5
25	10101	5	1F	01010	5	04	11001	5
2B	10110	5	2B	01011	5	0C	110100	6
1F	10111	5	31	01100	5	0A	110101	6
3A	1100000	7	22	011010	6	1E	110110	6
16	1100001	7	16	011011	6	0D	110111	6
20	1100010	7	2E	011100	6	1C	111000	6
2E	1100011	7	2A	011101	6	2A	111001	6
22	1100100	7	30	011110	6	28	1110100	7
2A	1100110	7	26	011111	6	1F	1110101	7
4C	1100110	7	0C	100000	6	25	1110110	7

Figure 1 shows the codebook as binary tree.

The tree in Figure.1 is constructed from the binary tree with branches of child nodes of one bit from the encoded bit stream. Each node of the tree is constructed by memory and the search method is implemented by FSM [5]. Each child node of two pairs consist of the even or odd address of the memory. The data format of the memory in Figure 1 is represented by either a leaf node or a non-leaf node.

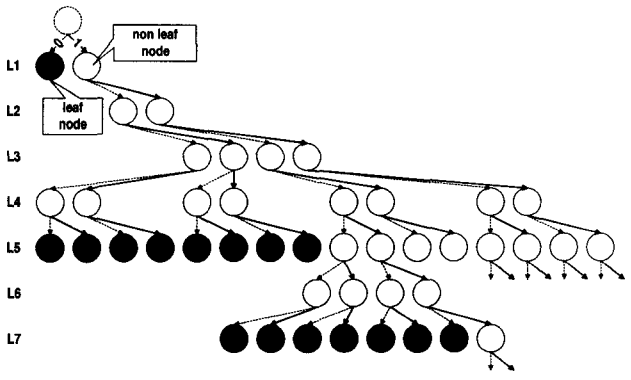

Fig. 1. Binary tree for codebook #1

Leaf node: 0, index
Non-leaf node: 1, [segment]

In this notation, segment is used as the address of a child node, and either '0' or '1' from the bit stream is used as the offset address. To determine the address of the memory to be used for the searching tree, we add the value of the offset to the value of the segment address. The result of the addition is the next address.

Next address = Segment + Offset

In table 2, we show the heap of the memory for searching the tree of Figure 1.

Table 2. Memory heap of codebook #1

Addr.	Value	Addr.	Value	Addr.	Value	Addr.	Value
00	0, 28	01	1, [02]	02	1, [04]	03	1, [12]
04	1, [06]	05	1, [0C]	06	1, [08]	07	1, [0A]
08	0, 43	09	0, 0D	0A	0, 27	0B	0, 31
0C	1, [0E]	0D	1, [10]	0E	0, 29	0F	0, 25
10	0, 2B	11	0, 1F	12	1, [14]	13	1, [26]
14	1, [16]	15	1, [24]	16	1, [18]	17	1, [1E]
18	1, [1A]	19	1, [1C]	1A	0, 3A	1B	0, 16
1C	0, 26	1D	0, 2E	1E	1, [20]	1F	1, [22]
20	0, 22	21	0, 2A	22	0, 4C	23	1, [..]
24	1, [..]	25	1, [..]	26	1, [..]	27	1, [..]
28	1, [..]	29	1, [..]	2A	1, [..]	2B	1, [..]

In Table 3, we show the process for searching the index 29h at the level 5 of Figure 1.

Table 3. tree search of index 29

Clock	Input	Seg.	Offset	Addr.	Value
1	1	00	1	01	1, 02
2	0	02	0	02	1, 04
3	1	04	1	04	1, 0C
4	0	0C	0	0C	1, 0E
5	0	0E	0	0E	0, 29

1. If input 1 is occurred on clock 1, the value of the memory address is Segment ('00'h) + Offset ('1'h) = '01'h.
2. Because the value of the '01'h address of the memory is "1, 02" and it is not matched with the value of the codebook, it is used for the segment of the next address.
3. If in clock 2 the input is '0', the value of the memory address is Segment ('02'h) + Offset ('0'h) = '02'h.
4. Because the value of the memory address '02'h is "1, 04", and it is not matched with the value of the codebook, it is used for the segment of the next address.
5. By the repetition of the same manner, we get the "0, 29" at the memory address '0E'h and the index '29'h of the

codeword.

As stated above, the method of binary tree search with a 1 bit input per one clock consumes 7 clocks for decoding the symbol of level 7. In the same manner, 19 clocks are enough for the Huffman table of the MPEG-2 audio AAC with the maximum 19 bits. Also, leaf nodes and non-leaf nodes of the tree are constructed on the memory, but the memory is wasted by the unnecessary construction of non-leaf nodes over the real symbols.

3.2 Multi way tree search algorithm

We improve the binary tree of Figure 1 by increasing the number of bits from the bit stream to 2 or 3 bit, and so decrease the level of the tree and the number of non-leaf nodes. Figure 2 shows the improved structure of tree.

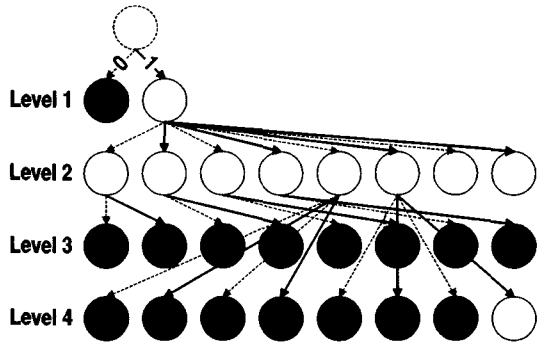

Fig. 2 tree search of multi way tree search of order 3

In Figure 2, the root of the tree start from 1 bit, and each node with the variable input becomes up to 3 bits according to the distribution of the codeword. An input value up to 3 bits is used for the offset address. Figure 4 shows the heap of memory used for the variable length search.

Table 4. Memory heap for FSM

Addr.	Value	Addr.	Value	Addr.	Value	Addr.	Value
00	0, 28	01	[3, 08]	02	[x, xx]	03	[x, xx]
04	[x, xx]	05	[x, xx]	06	[x, xx]	07	[x, xx]
08	[1, 10]	09	[1, 10]	0A	[1, 10]	0B	[1, 10]
0C	[2, 18]	0D	[2, 18]	0E	[., ..]	0F	[., ..]
10	0, 43	11	0, 0D	12	0, 27	13	0, 31
14	0, 29	15	0, 25	16	0, 2B	17	0, 1F
18	0, 3A	19	0, 16	1A	0, 26	1B	0, 2E
1C	0, 22	1D	0, 2A	1E	0, 4C	1F	[., ..]

Figure 2 and Table 4 show the more efficient usage of the memory over the normal binary tree due to the reduction of the number of non-leaf nodes.

In this manner, Codebook #2 and #3 of Table 1 are constructed on the memory heap in sequence, and with the starting address of the segment according to the number of the codebook, it is possible to search the value of the codebook at any given address.

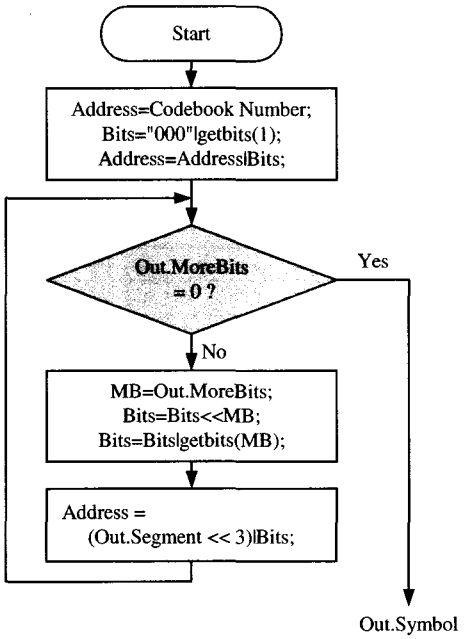

Fig. 3 Flowchart for Huffman decoding

The algorithm of the multi way tree searchis represented as a flow chart in Figure 3. The offset address is initialized as '000', and the length of the first input is only one bit. According to the number of the codebook, the memory address is determined by adding the predefined address of the segment to the offset address constructed by the input bits. If the value at the given memory is matched with the value of the codebook, the index value is the output. But if not, more bits as input are needed to search the next node. Like this, the repetition of searching the memory with the address obtained by the addition of the segment address to offset is performed. At this time, if the number of input bits as input is less than 3, the new offset address is obtained by shifting the value of the previous offset address toward the left by number of the previous input bits.

In Table 5, we show the process for searching the index 29h at level 5 of Figure 2.

Table 5. search tree of index 29

Clock	Input	Seg.	Offset	Addr.	Value
1	1	00	1	01	3, 08
2	010	08	2	0A	1, 10
3	0	10	4	14	0, 29

1. If input 1 is occurred on clock 1, the value of the memory address is Segment ('00'h) + Offset ('1'h) = '01'h.

2. Because the value of the '01'h memory address is "3, 08", 3 bits are needed and the '08'h is used as the segment of

the next address.

3. If in clock 2 the input is '010', the value of the memory address is Segment ('08'h) + Offset ('2'h) = '0A'h.

4. Because the value of the memory address '02'h is "1, 10", 1 bit is needed and the '10'h is used as the segment of the next address.

5. In clock 3, because the input is '0', we can get the value of '14'h by shifting the value of the previous offset toward the left by 1 bit and the adding offset ('4'h) to segment ('10'h).

6. Because the value at the memory of the address '14'h is "0, 29", the index '29'h of the codeword become the output.

As mentioned above, the tree search method with up to 3 input bits per clock consumes up to 8 clocks for decoding the Huffman table of MPEG-2 audio AAC with the maximum 19 bits. Also, due to the reduction of the number of non-leaf nodes, memory is saved.

In general, the maximum number of bits to be used as input is determined by the characteristics of the codebook. But the faster the speed of decoding, the more the memory is wasted due to the increase of don't care states. Considering the trade off between the speed of decoding and the efficiency of the memory usage, the maximum number of bits must be determined properly.

4 Hardware Implementation

In this section, we designed the Huffman Decoder used in the MPEG audio Layer-3 and the MPEG-2 audio AAC. We used VHDL (Very High Speed Integrated Circuit Hardware Description Language) and sysnopsys as development tools. VHDL is suitable for implementing algorithm with hardware and sysnopsys is used to synthesize the result from the VHDL code output.

The Huffman decoder of the MPEG audio decodes the bit stream which is decoded by analyzing the bit stream and choosing the Huffman table according to each band. Once one Huffman codebook is selected, the segment has start address of the codebook in memory heap space.

Until the input bits are matched by codeword, it reads the bit stream up to 3 bits. If the number of input bits to read is less than or equal to 2, shifting the bit stream makes a 3 bit offset address. In case of a mismatch with the codeword, the value of the memory consists of 2 bits witch represents the number of the input bit and 8 bits for the segment address in the next cycle.

The Figure 4 shows the hardware block diagram of the Huffman decoder implementing the multi way tree search of order 3. Also, We construct the Huffman table in ROM, so this hardware has flexibility. If the ROM is changed, this algorithm can be applied to the MPEG audio Layer-3 and the MPEG-2 audio AAC decoder.

Figure 5 shows the data format of the ROM address of the codebook data. In figure 5 a), we designed it so that the address format can represent a 2k word address space by the 8 bit segment pair, and the 3 bit offset part. Figure 5 b) shows the data format of a leaf node of the tree. '00'represents the upper 2 bits and the lower 9 bits are constructed by the decoded index value. According to the MPEG-2 audio AAC, the Huffman codebook can have up to 289 index entries in one table, so we use 9 bits to contain this value. Figure 5 c) shows a non-leaf node of the tree. The upper 2 bits are for the number of input bits in next clock cycle, and the lower 8 bits is for the segment address of the next clock cycle.

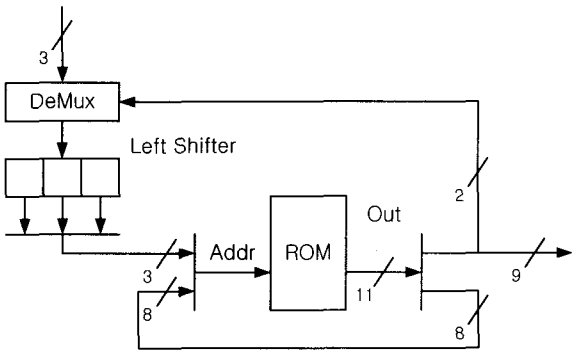

Fig. 4. Block diagram of Huffman decoder

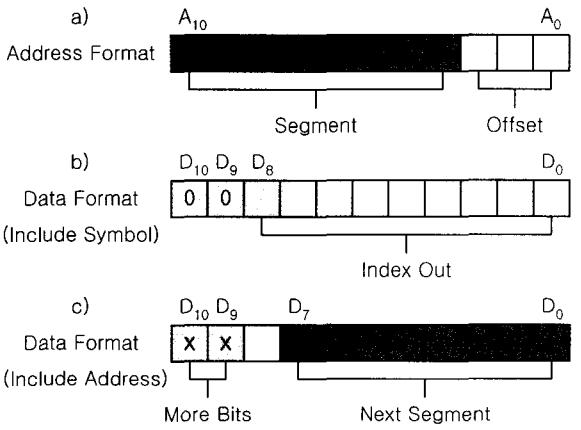

Fig. 5. Address and data format of table

5 Experiments and Result

In this section, we explain the validation of our proposal about the MPEG audio decoder with the multi way tree search algorithm, specially when the tree has a octal nodes. We choose the bit streams, which are longer than 1 minute and are compressed by the MPEG-2 AAC LC profile provided by the MPEG group.

In Figure 6, we show the amount of clock consumption of the sequential search method and the multi way tree search algorithm. The sequential search method is the method from the VM (Verification Model) provided by the MPEG group, starting from the shortest codeword and comparing memory with the same length codeword.

According to the result of our experiments, the average number of clock needed to read the bit stream to decode the one sample and search the memory is 17 for the sequential

search method, but only 2.48 for multi way tree search of order 3. In particular, we can reduce the maximum 300-clock cycle to a 6-clock cycle while easily controlling the clock due to the rapid decrease of differences between the decoding cycles of the samples.

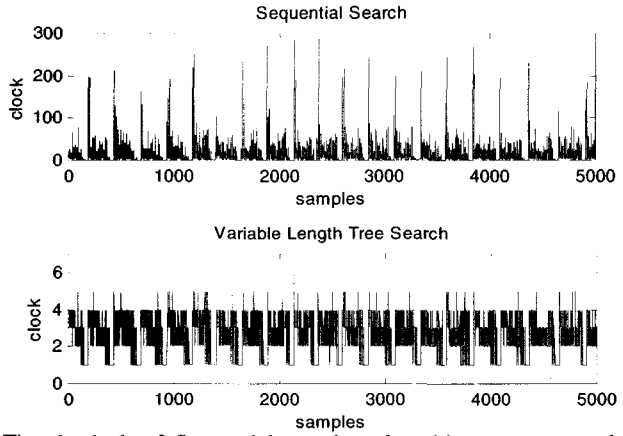

Fig. 6. clock of Sequential search and multi way tree search

Table 6. Implemented Huffman Decoder

	Layer 3	AAC
Number of codebooks	15	12
Number of symbols	1,394	1,362
Maximum length of codeword	19 bits	19 bits
Decoding speed per symbol	8 clock (Max)	8 clock (Max)
Memory Space	2.4k words	2.0k words
Area of logic gate	400 gates	

The Table 6 shows the memory size and the decoding speed of the Huffman decoder used by the Layer 3 and the ACC.

6. Conclusion

We proposed the Huffman Decoding Algorithm for the system which has many Huffman tables. The Variable Length Algorithm is used to quicken the search. We used a lower logic gate and memory for our method, rather than the traditional method and proved the algorithm by application of the Huffman decoder of the MPEG audio decoder.

Because the input bit stream is used as the memory address , the memory search is done at once. Also, Since there is no need to save the codeword, the number of bits representing the memory address used for saving the codebook is reduced to half.

References

[1] ISO/IEC 13818-7, "Generic Coding of Moving Pictures and Associated Audio Information - Part 7: Advanced Audio Coding", 1997

[2] ISO/IEC 14496-3, "Information Technology - Coding of Audiovisual Objects - Part 3: Audio, Subpart 4: T/F Coding", 1998

[3] ISO/IEC 11172-3, "Information Technology - Generic Coding of Moving Pictures and Associated Audio: Part 3: Audio", 1992

[4] ISO/IEC JTC1/SC29/WG11 N2005, "Revised Report on Complexity of MPEG-2 AAC Tools", Feb. 1998

[5] Vikram Iyengar, Krishnendu Chakrabarty "An efficient finite-state machine implementation of Huffman decoders", *Information Processing Letters*, V.64 N.6, 271-275, Dec. 1997

[6] Hong-Chung Chen, Yue-Li Wang, Yu-Feng Lan, "A memory-efficient and fast Huffman decoding algorithm", *Information Processing Letters*, V.69, pp 119-122, 1999

[7] K. Woo, G. Kim, H. Hahn, H. Cha "Efficient Huffman decoder using Octal tree search Algorithm", *Journal of the Korean Institute of Communication Sciences*, vol. 25, No. 12B, pp2033-2038, Dec. 2000

[8] Seung Bae Choi, Moon Ho Lee "A Fast Huffman Decoder via Pattern Matching", *ISPACS*, pp 134-138, 1994

[9] Reza Hashemian, "Memory Efficient and High-Speed Search Huffman Coding", *IEEE Transactions on Communications*, V.43 N.10, Oct. 1995

[10] Reza Hashemian, "Efficient variable - length coding under an assigned maximum code-length constraint", Proceedings of the *IEEE International Symposium on Circuits and Systems* - Volume 2, May. 1996

[11] Park S, Cho H, Cha JJ, "High speed search and an area efficient Huffman decoder", *IEICE Transactions on Fundamentals of Electronics Communications & Computer Sciences*, V.E82-A N.6, Communications, V.43 N.10, Oct. 1995

Hyungtai Cha

He received the M.S. and Ph.D. degree in dept. of Electrical Engineering from the University of Pittsburgh in 1988 and 1993 respectively. He is currently an Associate Professor in the School of Electronic Engineering, Soongsil University. His recent research interests include Multimedia Systems and Applications, Audio and Video Signal Processing, ASIC and DSP Implementation of Digital System, and Communication System.

Phone : +82-2-820-0711
E-mail : hcha@ssu.ac.kr

Kwanghee Woo

Kwanghee Woo received the B.S. degree in Electronic Engineering from Soongsil University in 1998.

He is studying for his M.S. degree in Soongsil University. His recent research interests include Audio Signal Processing and Coding, MPEG Audio coding, ASIC Implementation of Digital System.

E-mail : com3com4@mms.ssu.ac.kr