ISCAS 2000 - IEEE International Symposium on Circuits and Systems, May 28-31, 2000, Geneva, Switzerland

A New Approach of Group-Based VLC Codec System*

Bai-Jue Shieh *Terng-Yin Hsu* *Chen-Yi Lee*

Dept. of Electronics Engineering, National Chiao Tung University

1001, Ta Hsueh Road, Hsinchu, 300, Taiwan, R.O.C.

Abstract

In this paper, the algorithm of a VLC codec system with new group-based approach is presented. Based on the proposed codeword grouping and symbol memory mapping, the group-searching scheme and codec processes are completed by applying numerical properties and arithmetic operations to codewords and symbol addresses. The memory requirement of encoder is reduced by a novel symbol-converting scheme. Therefore, the programmable coding table and symbol representation can be achieved. Based on MPEG-like systems, an architecture design that performs concurrent VLC codec processes with constant symbol rate is presented. Simulation results show 100Msps with 100MHz-clock for both encoding/decoding procedures can be achieved. As a result, it is suitable for those applications that require codec processes simultaneously, such as videoconferencing, and high throughput systems, such as HDTV.

1. Introduction

Recently, progressive applications, such as HDTV, videoconferencing and user-defined coding table system, make challenges to VLC codec technology. The compressed bit stream of HDTV system is more than 100Mbps since the sampling rate is about 52Mpixel/sec and the color profile is 4:2:2. Subsequently, the throughput requirement of VLC codec operation is increased several order of magnitudes than earlier applications. In contrast, the bit rate is much lower for videoconferencing which is established on the limited network bandwidth. However, the 2-way communication needs real time concurrent encoding/decoding procedures. To meet diverse applications and different data types, user-defined coding tables that are generated by precise symbol probability are essential to further increase compression ratio. Because the table information has to be loaded before the codec processes, the VLC codec systems require the programmability to change coding table without redesign.

Both tree-searching and group-based algorithms for VLC codec system have been discussed. By representing coding table as tree structure [1][2], the codeword is encoded/decoded several bits at a time. Therefore, they are not quite suitable for real time processing since the time period is long for a sequence of long codewords. Besides, the IO condition is complex and needs buffer design as well. In contrast, taking the advantage of codeword properties, leading character [3~5] and prefix concatenating with suffix [6], group-based VLC codec designs perform the constant processing rate to reduce the control complexity and increase operation throughput. However, most of them present decoding schemes and lack of encoding algorithm with codeword grouping properties. In addition, when group-searching is completed by leading detector, the architecture is complex to adapt for those tables that are non-monotonic and consist of leading-1 and leading-0 codewords, such as MPEG2 DCT coefficient tables. On the other hand, several architectures, such as PLA-, ROM-, CAM- and RAM-based, for VLC codec systems have been proposed. By searching all possible codewords in parallel, PLA-, ROM- [7~8] and CAM-based [6] designs are popular for the applications with standard-defined tables. However, the disadvantage of PLA and ROM is lacking of table programmability and the cost of CAM-based design is high to preserve enough memory space for all possible patterns. With efficient memory mapping scheme, RAM-based architectures, [1~5][9], not only design cost is reduced by saving memory space but also table programming is achieved by loading table information into memories. Consequently, they can meet the requirement of various applications and obtain the flexibility.

To satisfy the mentioned application, the motivation behind our research is to develop a programmable coding table, low memory requirement, and high throughput VLC codec system. A new group-based VLC codec algorithm which take the advantage of numerical property of codeword and symbol address are presented. The operation throughput is improved significantly. With memory-based architecture, coding tables and symbol representations can be programmed. The organization of this paper is as follows. In section 2, the group-based VLC encoding/decoding algorithm is described. The symbol-converting scheme is presented, too. In section 3, the VLC codec architecture and performance estimation is discussed. Finally, concluding remarks are made in section 4.

2. The Group-Based VLC Encoding / Decoding Algorithm

By observing Huffman procedures, two characteristics are discovered as follows: 1) the symbols being combined will receive the same codeword-prefix; 2) the symbols with the same probability will have the same codelength. Consequently, the symbols that have the same probability and are combined to perform Huffman procedures will receive the same codeword-prefix and codelength. Therefore, the codewords of these symbols are defined as a codeword group, as shown in Fig. 1, and have the following properties:

1. In the group, the codewords can be treated as codelength-bit binary number, VLC_codenum.
2. The codeword that has the smallest VLC_codenum in the group is denoted as VLC_mincode.

* Work supported by the National Science Council of Taiwan, ROC, under Grant NSC88-2218-E-009-022

3. The offset between VLC_mincode and VLC_codenum is called VLC_codeoffset.

Group	symbol	Prob.	prefix	suffix	VLC_ codenum	VLC_ Codeoffset	
G1	X0	0.500	0		1 1-bit	0	mincode
G2	X1	0.200	10	0	4 3-bit	0	mincode
	X2	0.200	10	1	5 3-bit	1	
G0	X3	0.033	11	00	12 4-bit	0	mincode
	X4	0.033	11	01	13 4-bit	1	
	X5	0.033	11	10	24 4-bit	2	

Fig 1: An example of codeword grouping.

In order to generate the table information for both encoder and decoder, the strategy of intra-group symbol memory mapping is that the distance between symbol_address and base_address equal to the VLC_codeoffset, where the base_address is defined as the symbol memory address of VLC_mincode. According to the intra-group symbol memory mapping shown in Fig 2, the encoding/decoding procedures in the group are described as follows:

Decoding procedures: Assume the received bit stream is $(00100010.....)_2$:

① VLC_codeoffset = bit_stream - VLC_mincode
$= (00100010)_2 - (00100000)_2 = 00000010_2 = 2$;

② symbol_address = base_address+ VLC_codeoffset
$= (100) + (2) = 102$;

Finally, the symbol, x2, is accessed with the decoded symbol memory address, 102.

Encoding procedures: Assume the encoded symbol address is 103:

① VLC_codeoffset = symbol_address - base_address
$= (103) - (100) = 3$;

② VLC_codenum = VLC_mincode + VLC_codeoffset
$= (32) + (3) = 35$;

Finally, the encoded 8-bit codeword is $00100011_2 = 35$.

According to the procedures described above, the required group-information is VLC codelength, VLC_mincode and base_address. In addition, the memory space of concurrent VLC codec processes can be saved since the group-information is the same for both encoder and decoder.

symbol	prefix	sufix	VLC _codenum	VLC _codeoffset	sym_ addr
X0	00100	000	32	0	100
X1		001	33	1	101
X2		010	34	2	102
X3		011	35	3	103

group-information: { *codelength = 8; base_address = 100;*
VLC_mincode = 00100000_2;

Fig 2: Example of intra-group symbol memory mapping.

It becomes failed to apply numerical property to whole Huffman coding table while the lengths of codewords are variable. According to Huffman property that codeword can not be the prefix of any other codewords, the codeword is still unique in the table even though arbitrary additional characters are added behind them. Thus, the Pseudo-Constant-Length-Code (PCLC) which is generated by concatenating VLC codeword with redundant characters, *00...0*, is proposed to equalize the codelength. As a result, PCLC codewords can be treated as the binary numbers with max VLC codelength bits and the PCLC_codenums are distinct since the PCLC codewords are unique. Furthermore, the PCLC coding table is established as ascending PCLC_codenums, i.e. $codenum_0 < codenum_1 < ... < codenum_n$, for arithmetic group-searching scheme. Consequently, the PCLC_mincodes are ascending, too, i.e. $mincode_0 < mincode_1 < ... < mincode_n$. To generate encoding/decoding group-information, the inter-group symbol memory mapping scheme is that base_addresses have the same numerical relationship with PCLC_mincodes, i.e. $base\_addr_0 < base\_addr_1 < ... < base\_addr_n$. An example of PCLC coding table and intra-/inter-group symbol memory mapping are shown in Fig 3. The group-information of the table shown in Fig 3 is given in Fig 4.

group	symbol	PCLC_ codeword	PCLC_ codenum	sym_ addr	VLC_ codeoffset
G0	S00	**0 0 1 0 0 1 0 0**	36	0	0
	S01	**0 0 1 0 0 1 0 1**	37	1	1
	S02	**0 0 1 0 0 1 1 0**	38	2	2
	S03	**0 0 1 0 0 1 1 1**	39	3	3
G1	S10	**0 0 1 1 0 0** *0 0*	48	4	0
	S11	**0 0 1 1 1 1** *0 0*	56	7	3
G2	S20	**0 1 0** *0 0 0 0 0*	64	8	0
G3	S30	**0 1 1 0** *0 0 0 0*	96	9	0
	S31	**0 1 1 1** *0 0 0 0*	112	10	1
G4	S40	**1 0** *0 0 0 0 0 0*	128	11	0
G5	S50	**1 1 0** *0 0 0 0 0*	192	12	0
G6	S70	**1 1 1 1 0 0 0** *0*	240	13	0
	S71	**1 1 1 1 0 0 1** *0*	242	14	1
	S72	**1 1 1 1 0 1 0** *0*	244	15	2
	S73	**1 1 1 1 1 0 0** *0*	248	17	4

max VLC codelength: 8-bit; symbol address bitlength: 5-bit.

Fig 3: PCLC coding table and complete intra-/inter-group symbol memory mapping.

group	codelength	PCLC_mincode(8b)	base_addr(5b)
0	8	0 0 1 0 0 1 0 0	0 (00000_2)
1	6	0 0 1 1 0 0 0 0	4 (00100_2)
2	3	0 1 0 0 0 0 0 0	8 (01000_2)
3	4	0 1 1 0 0 0 0 0	9 (01001_2)
4	2	1 0 0 0 0 0 0 0	11 (01011_2)
5	3	1 1 0 0 0 0 0 0	12 (01100_2)
6	7	1 1 1 1 0 0 0 0	13 (01101_2)

Fig 4: The group-information of the table shown in Fig. 3.

Similar to PCLC, the codeword bitstream which is ($codeword_i$ + $codeword_j$ ...) with max VLC codelength bits can be viewed as binary number, bitstream_num. It is found that the decoding codeword belongs to group G_x if the hit condition, i.e. $PCLC\_mincode_x \le bitstream\_num < PCLC\_mincode_{x+1}$, is detected. For encoding process, it satisfies $base\_addr_y \le enc\_symaddr < base\_addr_{y+1}$ when the encoding symbol is located in group G_y. According to the proposed algorithm discussed above, the detail descriptions of VLC codec processes and the corresponding examples based on the coding table in Fig. 3 are given in Fig. 5.

Decoding processes, assume decoded bitstream is 001111100110..... :

1) **group-searching:**
   $PCLC\_mincode_1 \le bitstream\_num < PCLC\_mincode_2$;
   (8'b00110000) ≤ (8'b00111110)<(8'b01000000)
   matching group: G_1;
2) **send group-information:**
   codelength=6-bit, PCLC_mincode=(8'b00110000),
   base_addr(5-bit) = (5'b00100);
3) **the valid VLC_codeoffset is the most significant codelength bits of the result of subtracting PCLC_mincode from bitstream_num:**
   VLC_codeoffset= bitstream_num - PCLC_mincode
   = (8'b00111110) - (8'b00110000) =(8'b00001110);
   valid VLC_codeoffset = (6'b000011) = 3;
4) **VLC_codeoffset operand which the wordlength is equal to symbol address:**
   VLC_codeoffset = (5'b00011);
5) **calculating decoded symbol_address:**
   symbol_address= base_address + VLC_codeoffset;
   = (5'b00100) + (5'b00011) = (5'b00111) = 7;
6) **fetch symobl:**
   sym_mem[7] = S11;

Encoding processes, assume encoded symbol address is 17(5'b10001):

1) **group-searching:**
   $base\_addr_6 \le symbol\_address$;
   (5'b01101) ≤ (5'b10001)
   matching group: G_6;
2) **send group-information:**
   codelength = 7-bit, PCLC_mincode = 8'b11110000,
   base_addr(5-bit) = (5'b01101);
3) **valid VLC_mincode is the most significant codelength bits of PCLC_mincode and the wordlength of its operand is equal to max codelength bits:**
   valid VLC_mincode=7'b1111000;
   VLC_mincode = 7'b1111000 = 120 = 8'b01111000;
4) **VLC_codeoffset is the result of subtracting base_address from symbol_address:**
   VLC_codeoffset = symbol_address – base_address
   =(5'b10001) - (5'b01101) = 5'b00100;
5) **calculating encoded VLC_codenum:**
   VLC_codenum = VLC_mincode + VLC_codeoffset
   = (8'b01111000) + (5'b00100) = 8'b01111100
6) **valid encoded codeword is less significant codelength bits of VLC_codenum:**
   codeword = 7'b1111100

Fig 5. The detail description of VLC coding processes and corresponding examples.

It has to use symbol to find the corresponding symbol address in encoding procedure. For MPEG-like system, it is essential to convert Run-Level symbols into short wordlength format to save the memory space. The 8-bit converted symbols of the proposed scheme are the sum of encoding Level and Converting-Based-Symbol (CBS), which accumulates the max Level from Run_0 to Run_{n-1} for each Run_n, as shown in Fig 6. Therefore, the memory requirement of symbol information of encoder is reduced by the proposed symbol-converting scheme.

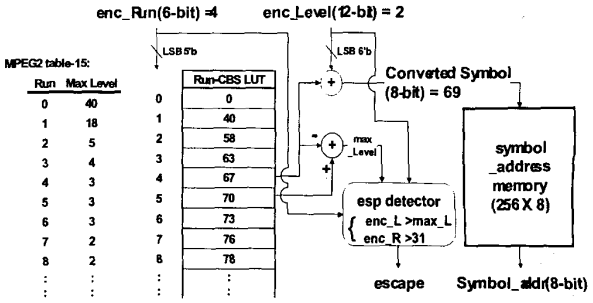

Fig. 6: Symbol converting scheme and Run-CBS LUT based on MPEG2 table-15.

3. The Group_Based VLC Codec System Architecture

The proposed VLC codec system is designed for MPEG-like applications with coding tables up to 256-entry 12-bit symbols and 16-bit codewords. It performs concurrent encoding/decoding procedures with the same group-information and achieves table programmability by loading table-information into on-chip memories. Besides, the processes for sign bit and escape Run/Level following the VLC codewords are included to obtain the complete coding results. Converting Run-Level pair to 8-bit converted symbol reduces the requirement of symbol_address memory. Therefore, the total memory space is $2^5 \times 8$, $2^8 \times 8$, $2^8 \times 12$ and 32×29 bits for Run-CBS LUT, symbol_address memory, symbol memory and 32-entry group-information respectively. Block diagram of the proposed VLC codec system for MPEG-like applications is shown in Fig. 7. It mainly consists of the following components: 1) group-based VLC encoder/decoder: it is composed of group-detectors and combonational logic to realize group-searching and encoding/decoding procedures described in previous section; 2) input FIFO and dec_bitstream selector: the input bitstream is buffered by the input FIFO. According to previous decoding results, the selector sends codeword bitstream to VLC decoder, where sign bit and escape R/L following the codeword are detected; 3) enc_bitstream concatenater and output FIFO: the concatenater adds the sign bit or escape R/L behind the codeword to complete the codeword bitstream. Then, the encoded bitstream is pushed into output FIFO; 4) special code detector: to increase the decoding throughput, the special codes, such as escape and EOB, are detected by comparing the decoded symbol_address rather than symbol. As a result, the bitstream selector does not have to wait symbol fetching and directly selects the next dec_bitstream right after the codeword is decoded; 5) enc_en/dec_en ctrl: according to the condition of input data and FIFO status, enable controllers determine the operations of encoder/decoder; 6) symbol_address/symbol MEM: the on-chip memories store symbol-information for VLC codec processes; 7) symbol converter/recoverer: converter transforms Run-Level pairs to 8-bit converted symbols and detects the cases of escape and EOB. According to decoded results, recoverer finds the correct Run and restores the Level.

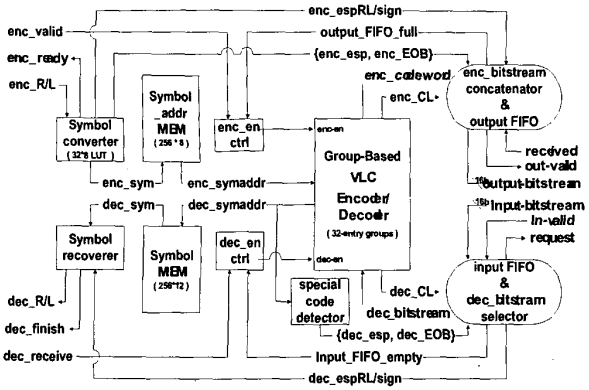

Fig. 7: Block diagram of the proposed VLC codec system for MPEG applications.

Because special code detector determines the additional wordlength in decoding process, it needs not enter stall-state for waiting the symbol access before the selection of next codeword bitstream. As a result, the performance of the proposed VLC codec system is up to 1 symbol/cycle both for encoding/decoding procedures. Based on the HDTV system, several simulation results are given in Table 1. It shows that the operation throughput of the proposed design achieve 100Msymbol/s at 100MHz-clock rate with 0.6-μm CMOS process and 5V supply voltage. Because the bitstream is aligned to 16 bits, some overhead of operation cycles is induced by the FIFOs used to perform the transformation of data format from 32-bit to 16-bit. In addition, the comparison of available VLC codec designs is shown in Table 2.

Table 1: Simulation results based on HDTV system(1-frame)

(4:2:2) @1920x1080			
# of bits	3439392	1912640	2114464
# of symbols	590302	252817	289129
Encode cycle	590348	252863	289173
Decode cycle	590337	252862	289170

Table 2: Comparison of existing VLC codec designs.

	[9]	[3]	[1]	Proposed
Enc/Dec mode	switched			concurrent
Throughput (MSPS)	40.5	30	Enc: 11.9 Dec: 7.6	100
Clk rate	81	30	83.3	100
Process	0.4-μm	1.0-μm	2.0-μm	0.6-μm
Symbol length	12-bit	8-bit	8-bit	12-bit
Memory space for 256-entry coding table	4*16*256 =16k-bit	256x12 +16*16 +128*10*12 =18k-bit	512X12 =6k-bit	32*8+256*8 +256*12+ 32*29 =6.3k-bit
Table limitation	no	Leading 1	no	no

4. Conclusion

In this paper, the VLC codec algorithm with new group-based approach has been presented. Based on the codeword grouping and symbol memory mapping, the encoding/ decoding procedures are completed by applying numerical properties to codeword and symbol address. By transforming VLC table into the proposed PCLC (Pseudo-Constant-Length-Code) format, the group-searching scheme is performed by arithmetic operations rather than pattern matching. Furthermore, the memory requirement of encoding process is reduced by the proposed symbol-converting scheme. Based on MPEG-like system, an architecture design that performs concurrent codec processes with constant symbol rate has also been presented. Simulation results show that this design achieves 100Msymbol/s with 100MHz-clock rate. As a result, it is suitable for those applications that require encoding/decoding procedures simultaneously, such as videoconferencing, and high throughput systems, such as HDTV.

5. REFERENCES

[1] Amar Mukherjee, N. Ranganathan, Jeffrey W. Flieder, and Tinku Acharya, " MARVLE : A VLSI Chip for Data Compression Using Tree-Based Codes," *IEEE Trans. on Very Large Scale Integration (VLSI) System*, Vol.1, No.2, pp.203-213, June.1993.

[2] Heonchul Park and Viktor K. Prasanna, " Area Efficient VLSI Architectures for Huffman Coding," *IEEE Trans. on Circuits and System*, Vol.40, No.9, pp.568-575, Sept.1993.

[3] Peter A. Ruetz, Po Tong, Daniel Luthi and Peng H. Ang, "A Video-Rate JPEG Chip Set," *Journal of VLSI Signal Processing*, vol. 5, pp.141-150, 1993.

[4] Seung Bae Choi and Moon Ho Lee, "High Speed Pattern Matchinig for a Fast Huffman Decoder," *IEEE Trans. on Consumer Electronics*, vol. 41, No.1 , pp.97-103, Feb. 1995.

[5] Belle W. Y. Wei and Teresa H. Meng, "A Parallel Decoder of Programmable Huffman Codes," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 5, No. 2, pp.175-178, April 1995.

[6] Cheng-The Hsieh and Seung P. Kim, "A Concurrent Memory-Efficient VLC Decoder for MPEG Applications," *IEEE Trans. on Consumer Electronics*, vol. 42, No. 3, pp.439-446, August 1996.

[7] Shaw-Min Lei and Ming-Ting Sum, "An Entropy Coding System for Digital HDTV Applications," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 1, No. 1, pp.147-155, march 1991.

[8] Shih-Fu Chang and David G. Messerschmitt, " Designing a High-Throughput VLC Decoder Part I – Concurrent VLSI Architectures," *IEEE Trans. on Circuits and Systems for Video Technology*, vol.2, No.2, pp.187-196, June.1992.

[9] Y. Fukuzawa , K. Hasegawa , H. Hanaki, E. Iwata and T. Yamazaki, "A Programmable VLC Core Architecture for Video Compression DSP," in *IEEE SiPS 97 Design and Implementation formerly VLSI Signal Processing*, pp.469-478, Nov. 1997.