

A New Approach of Group-Based VLC Codec System with Full Table Programmability

Bai-Jue Shieh, Yew-San Lee, and Chen-Yi Lee

Abstract—In this paper, the algorithm and architecture of a variable-length-code (VLC) codec system using a new group-based approach and achieving full table programmability are presented. According to the proposed codeword grouping and symbol memory mapping, both group searching and encoding/decoding procedures are completed by applying numerical properties and arithmetic operations to codewords and symbol addresses. By a novel symbol conversion, the memory requirement of the encoding process is reduced and the programmability of codewords and symbols is achieved. For MPEG applications, a 0.6- μm CMOS design that performs concurrent VLC codec processes is shown. This VLSI implementation occupies an area of $5.0 \times 4.5 \text{ mm}^2$ with 110 k transistors and satisfies a coding table up to 256-entry 12-bit symbols and 16-bit codewords. In addition, both encoding and decoding throughputs of this design achieve 100 Msymbols/s at a 100-MHz clock rate. Therefore, the proposed VLC codec system is suitable for applications which require high operation throughput, such as HDTV, and simultaneous compression and decompression, such as videoconferencing.

Index Terms—Group-based, HDTV, Huffman coding, VLC codec, VLC/VLD.

I. INTRODUCTION

WITH THE ADVANCES of technologies in multimedia and communication, pictures, photographs, and video-films are used in many applications. Meanwhile, the supported images and motion pictures are asked to enhance qualities and resolutions. This request results in higher data rates and more complex data types. Efficient data compression techniques that satisfy the requirements of various applications and save the costs of transmission and storage are demanded. The Huffman code [1], also called the variable-length code (VLC), is the most popular lossless data compression technique which is recommended by many image and video standards, such as JPEG, MPEG, and H.263. The Huffman coding reduces data redundancy based on assigning shorter codewords to more frequent symbols, and vice versa. Hence, the compression result is very close to the entropy of source messages.

Recently, progressive applications such as HDTV, videoconferencing, and user-defined table systems are design challenges for VLC codec technologies. To achieve high-quality and high-

resolution video services, the compressed data rate of a HDTV system is more than 100 Mbits/s, since the sampling rate is about 52 Mpixel/s and the color profile (Y:U:V) is 4:2:2. Subsequently, the VLC codec throughput of HDTV systems is increased several order of magnitudes than that of earlier applications, such as MPEG2 MP@ML. In contrast, a videoconferencing system which is established on limited network bandwidth is a low bit rate application. However, the two-way communication needs real-time compression and decompression. The cost is high to implement an encoder and a decoder, while the control complexity and buffer size are increased to switch encoding and decoding processes with a single VLSI design. Therefore, a concurrent VLC codec system with shared function units is the optimal solution in this case. To meet diverse applications and data types, user-defined tables which are generated by related source data are essential for further increasing compression ratios. Before systems begin to deal with input data, user-defined tables have to be loaded into memories. Consequently, a VLC codec design needs the programmability to change coding tables without redesigning original architecture.

VLC codec algorithms and architectures have been discussed in the literature. In terms of characteristics, the algorithms can be divided into two classes: tree-based and group-based. According to Huffman tree structures, tree-based VLC codec schemes encode/decode a codeword from leaf/root node to root/leaf node several bits at a time [4], [8]–[11]. These schemes are not quite suitable for high-performance real-time applications because the time period is long for a sequence of long codewords. Besides, their I/O conditions and buffer designs are complex since the operation clock cycles are variable for every codeword. In contrast, using codeword properties, such as leading characters in [7], [12], [13] and prefixes concatenating with suffixes in [14], group-based VLC codec algorithms perform constant operation rates to enhance performance and reduce control complexities. However, most of them deal with decoding methods and without encoding approaches. In addition, when algorithms use the leading character property, monotonic codewords and regular leading characters are essential for reducing design complexities and programmability costs. Hence, these algorithms are difficult to be modified for the codewords which are nonmonotonic and have both leading-1 and leading-0 prefixes, such as MPEG2 DCT coefficient table one. Several categories of VLC codec architectures, such as PLA-, ROM-, CAM-, and RAM-based, have been proposed. Completing the codec processes by matching all possible patterns in parallel, PLA, ROM, and CAM-based VLC codec designs are popular for

Manuscript received April 27, 1999; revised June 12, 2000. This work was supported by the National Science Council of Taiwan, R.O.C., under Grant NSC88-2218-E-009-022. This paper was recommended by Associate Editor N. Ranganathan.

The authors are with the Department of Electronics Engineering, National Chiao Tung University, Hsinchu 300, Taiwan, R.O.C. (e-mail: titany@royals.ee.nctu.edu.tw).

Publisher Item Identifier S 1051-8215(01)01245-9.

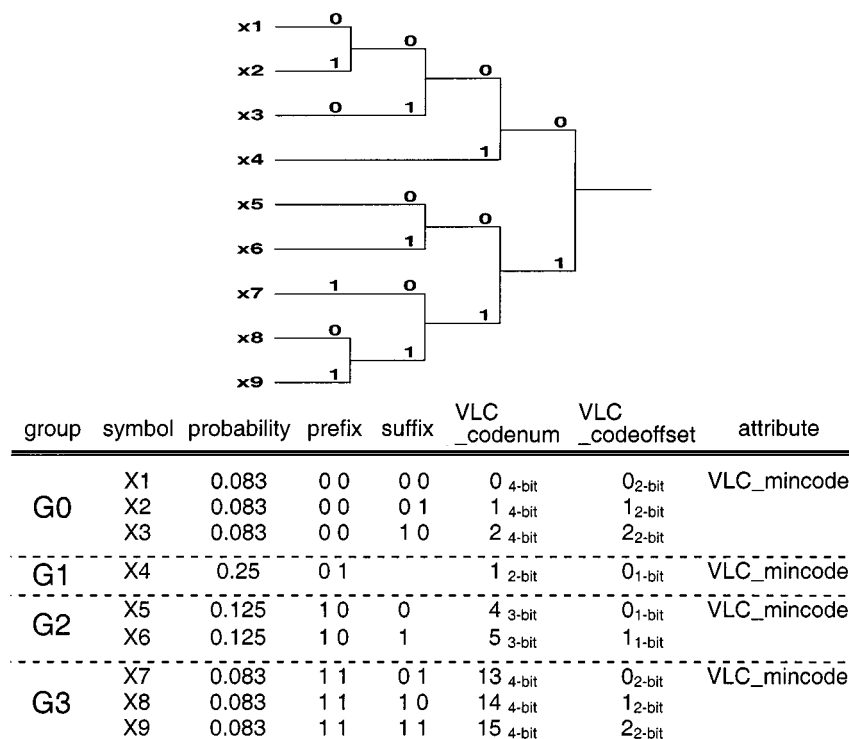


Fig. 1. Huffman code and codeword grouping.

standard-defined table applications [2]–[4], [6], [14]. Nevertheless, PLA and ROM-based systems lack programmability, and CAM-based designs require high costs to store all possible patterns. With efficient memory-mapping schemes, RAM-based VLC codec architectures in [7]–[13], [15] reduce design costs by saving memory space and obtain table programmability by changing memory contents. Consequently, these architectures can meet the requirements of various applications.

In this paper, we present the algorithm and architecture of a VLC codec system with a new group-based approach. Based on the proposed codeword grouping and memory mapping, numerical properties can be applied to codewords, symbol addresses, and bit streams. Therefore, the encoding/decoding procedures as well as the group searching scheme are accomplished by arithmetic operations instead of by pattern matching. Additionally, with a novel symbol conversion technique, the VLC codec system can reduce the memory requirement of the encoding process and achieve the programmability of codewords and symbols. For MPEG applications, we show a 0.6- μm CMOS design of the VLC codec system. This design performs concurrent encoding and decoding processes and satisfies a programmable table up to 256-entry 12-bit symbols and 16-bit codewords. Moreover, both compression and decompression rates of this design are 100 Msymbols/s at a 100-MHz clock rate.

The organization of this paper is as follows. In Section II, a group-based VLC codec algorithm is described. Several techniques that save memory space for storing symbol information are discussed, too. In Section III, the architecture of a VLC codec system for MPEG applications is presented. After that, chip implementation and performance estimation are shown. Finally, concluding remarks are made in Section IV.

II. GROUP-BASED VLC CODEC ALGORITHM

A. Definition of Codeword Groups

An example of the Huffman code and codeword grouping is illustrated in Fig. 1. The Huffman procedure assigns characters “0” and “1” to the combined source symbols with the lowest probability, respectively. The result of the combination is viewed as a composite symbol having the probability equal to the sum of the probabilities of the combined symbols. This procedure is applied as much as possible until all symbols are combined together. Based on the result of this procedure, the proposed codeword group is a set of codewords whose source symbols are combined to perform the Huffman procedure and receive the same codeword length. According to this definition, the codeword groups have the following properties.

- 1) In a group, the codeword can be treated as a codeword length-bit binary number, called VLC_codenum, since the codeword length is the same.
- 2) The codeword that has the smallest VLC_codenum in a group is denoted VLC_mincode.
- 3) A VLC_codeoffset is the offset value between the VLC_mincode and the VLC_codenum. Because codewords in the same group have the same prefix, the bit length of VLC_codeoffsets is the word length of suffixes.

In Fig. 1, the symbols x7, x8, and x9 belong to the codeword group G3. In this group, the codewords have the same codeword length, 4-bit, and prefix, 2'b11. The word length of the suffixes is 2-bit. Therefore, the 4-bit VLC_codenums are 13, 14, and 15, the VLC_mincode is 4'b1101, and the 2-bit VLC_codeoffsets are 0, 1, and 2. Although codeword lengths are identical source symbols which are not combined will belong to different groups, such as x1, x2, and x3 in G0 and x7, x8, and x9 in G1. Besides,

symbol	prefix	suffix	VLC _codenum	VLC _codeoffset	symbol address
X0	0 0 1 0 0	0 0 0	32	0	100
X1		0 0 1	33	1	101
X2		0 1 0	34	2	102
X3		0 1 1	35	3	103
X4		1 0 0	36	4	104
X5		1 0 1	37	5	105
X6		1 1 0	38	6	106
X7		1 1 1	39	7	107

group information: *codeword length = 8;*
VLC_mincode = 0010000₂;
base address = 100;

Fig. 2. Example of intra-group symbol memory map and group information.

there is only one symbol in group G1 since symbol X4 completes the Huffman procedure alone.

B. Intra-Group Encoding/Decoding Procedures

In addition to grouping codewords, it is necessary for both encoding and decoding procedures to map symbols onto memories and extract codeword group information. During intra-group symbol memory mapping, the memory address of a symbol in a group is calculated by the VLC_codeoffset of this symbol and the base address which denotes the symbol address of the VLC_mincode of the group. In other words, the symbol address is the sum of the VLC_codeoffset and the base address. After applying this arithmetic relation, VLC_codeoffsets, decoded symbol addresses, and encoded codewords can be found by numerical calculations rather than by pattern matching. Therefore, the group information to be stored is composed of codeword lengths, VLC_mincodes, and base addresses. Furthermore, the memory space of concurrent VLC codec systems can be saved since their encoders and decoders share the group information.

Based on the memory map and the group information in Fig. 2, intra-group encoding/decoding procedures can be described as follows.

Decoding procedure—assume the decoded codeword is (00 100 101)₂:

- 1) VLC_codeoffset = VLC_codenum (00 100 101)₂ – VLC_mincode (00 100 000)₂ = 00 000 101₂ = 5;
- 2) symbol_address = VLC_codeoffset (5) + base_address (100) = 105;
- 3) the decoded symbol, x5, is accessed by the symbol_address, 105.

Encoding procedure—assume the encoded symbol address is 103:

- 1) VLC_codeoffset = symbol_address (103) – base_address (100) = 3;
- 2) VLC_codenum = VLC_codeoffset (3) + VLC_mincode (32) = 35;
- 3) the encoded 8-bit codeword is 00 100 011₂ = 35.

C. Group-Searching Scheme

Because the encoding/decoding procedures are performed after the group information is acquired, an efficient group-searching scheme with low complexity and high

group	symbol	PCLC _codeword	PCLC _codenum	symbol address	VLC _codeoffset
	S00	0 0 1 0 0 1 0 0	36	0	0
	S01	0 0 1 0 0 1 0 1	37	1	1
G0	S02	0 0 1 0 0 1 1 0	38	2	2
	S03	0 0 1 0 0 1 1 1	39	3	3
	S10	0 0 1 1 0 0 0 0	48	4	0
				5	
G1				6	
	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
	S30	0 1 1 0 0 0 0 0	96	9	0
G3	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
	S60	1 1 1 0 0 0 0 0	224	13	0
G6	S61	1 1 1 0 1 0 0 0	232	14	1
	S70	1 1 1 1 0 0 0 0	240	15	0
	S71	1 1 1 1 0 0 1 0	242	16	1
G7	S72	1 1 1 1 0 1 0 0	244	17	2
				18	
	S73	1 1 1 1 1 0 0 0	248	19	4
	S80	1 1 1 1 1 0 1 0	250	20	0
	S81	1 1 1 1 1 0 1 1	251	21	1
G8	S82	1 1 1 1 1 1 0 0	252	22	2
	S83	1 1 1 1 1 1 0 1	253	23	3

The length of PCLC codewords is 8-bit and that of symbol addresses is 5-bit.

Fig. 3. PCLC table and intra-/inter-group symbol memory map.

group	valid	codelength	PCLC_mincode(8-bit)	base_addr(5-bit)
0	1	8	0 0 1 0 0 1 0 0	0 (00000 ₂)
1	1	6	0 0 1 1 0 0 0 0	4 (00100 ₂)
2	1	3	0 1 0 0 0 0 0 0	8 (01000 ₂)
3	1	4	0 1 1 0 0 0 0 0	9 (01001 ₂)
4	1	2	1 0 0 0 0 0 0 0	11 (01011 ₂)
5	1	3	1 1 0 0 0 0 0 0	12 (01100 ₂)
6	1	5	1 1 1 0 0 0 0 0	13 (01101 ₂)
7	1	7	1 1 1 1 0 0 0 0	15 (01111 ₂)
8	1	8	1 1 1 1 1 0 1 0	20 (10100 ₂)
9	0	0	0 0 0 0 0 0 0 0	0 (00000 ₂)

Fig. 4. Group information of the coding table shown in Fig. 3.

operation rate determines the performance of a group-based VLC codec system. To realize such a group searching scheme, the following pseudo-constant-length-code (PCLC) and inter-group symbol memory mapping are used. If all codeword lengths are the same, the numerical properties of codewords in a group can be applied to the whole coding table. A PCLC procedure is applied to equalize codeword lengths by adding redundant characters 00...0 behind VLC codewords. Hence, PCLC codewords which have the same length as the longest VLC codeword can be treated as binary numbers, PCLC_codenums. Because the VLC code is a prefix code, PCLC codewords and PCLC_codenums can be distinguished from each other. Accordingly, a PCLC table is established by ascending PCLC_codenums, i.e., $\text{codenum}_0 < \text{codenum}_1 < \dots < \text{codenum}_n$. This results in ascending PCLC_mincodes, i.e., $\text{mincode}_0 < \text{mincode}_1 < \dots < \text{mincode}_n$. Based on the PCLC table, the base addresses have to be assigned in PCLC_mincode order, i.e., $\text{base_addr}_0 < \text{base_addr}_1 < \dots < \text{base_addr}_n$ for inter-group symbol memory mapping. An example of the PCLC table and the intra-/inter-group symbol memory map is shown in Fig. 3. The group information of this PCLC table

Decoding process, assume the decoded bitstream is 001111100110..... :**1) Do group searching:**

$$\text{PCLC_mincode}_1(8'b00110000) \leq \text{bitstream_num}(8'b00111110) < \text{PCLC_mincode}_2(8'b01000000);$$

The matching group: G_1 ;

2) Send group information:

$$\text{codelength} = 6\text{-bit}, \text{PCLC_mincode}(8\text{-bit}) = 8'b00110000, \text{base_addr}(5\text{-bit}) = 5'b00100;$$
3) Find the valid VLC_codeoffset, which is the codelength most significant bits of the result of subtracting the PCLC_mincode from the bitstream_num.

$$\text{bitstream_num}(8'b00111110) - \text{PCLC_mincode}(8'b00110000) = 8'b00001110;$$

The valid $\text{VLC_codeoffset} = 6'b000011 = 3$;

4) Extract the VLC_codeoffset operand, which has the same wordlength as the symbol address.

$$\text{VLC_codeoffset} = 5'b00011;$$
5) Calculate the decoded symbol address.

$$\text{symbol_address} = \text{base_address}(5'b00100) + \text{VLC_codeoffset}(5'b00011) = 5'b00111 = 7$$
6) Fetch the decoded symbol.

$$\text{symbol_memory}[7] = S_{11};$$
Encoding process, assume the encoded symbol address is 19(5'b10011):**1) Do group searching.**

$$\text{base_addr}_7(5'b01111) \leq \text{symbol_address}(5'b10011) < \text{base_addr}_8(5'b10100);$$

The matching group: G_7 ;

2) Send group information.

$$\text{codelength} = 7\text{-bit}, \text{PCLC_mincode}(8\text{-bit}) = 8'b11110000, \text{base_addr}(5\text{-bit}) = 5'b01111;$$
3) Find the valid VLC_mincode, which is the codelength most significant bits of the PCLC_mincode. The wordlength of the VLC_mincode operand is the max codelength bits.

$$\text{The valid VLC_mincode} = 7'b1111000;$$

$$\text{The VLC_mincode operand} = 7'b1111000 = 120 = 8'b01111000;$$
4) Extract the VLC_codeoffset operand, which is the result of subtracting the base_address from the symbol_address.

$$\text{VLC_codeoffset} = \text{symbol_address}(5'b10011) - \text{base_address}(5'b01111) = 5'b00100;$$
5) Calculate the encoded VLC_codenum operand.

$$\text{VLC_codenum} = \text{VLC_mincode}(8'b01111000) + \text{VLC_codeoffset}(5'b00100) = 8'b01111100$$
6) Fetch the valid encoded codeword, which is the codelength less significant bits of the VLC_codenum operand.

$$\text{codeword} = 7'b1111100$$

Fig. 5. Detailed descriptions of the VLC codec processes and corresponding examples.

is given in Fig. 4, where the valid bit indicates whether the group information is used.

According to PCLC tables and symbol memory maps, the proposed group searching scheme is realized by applying numerical properties to bit streams and symbol addresses. Similar to PCLC codewords, a decoded bit stream that has the same length as the PCLC codewords is treated as a binary

number, bitstream_num . Because the bit stream is a sequence of concatenated codewords, such as codeword_i - codeword_j -etc., a relation between the bit stream and the PCLC table can be expressed by $\text{PCLC_codenum}_i \leq \text{bitstream_num} < \text{PCLC_codenum}_{i+1}$. Therefore, the group searching scheme is accomplished by the following numerical comparisons. The decoded codeword belongs to group G_x when the hit condition

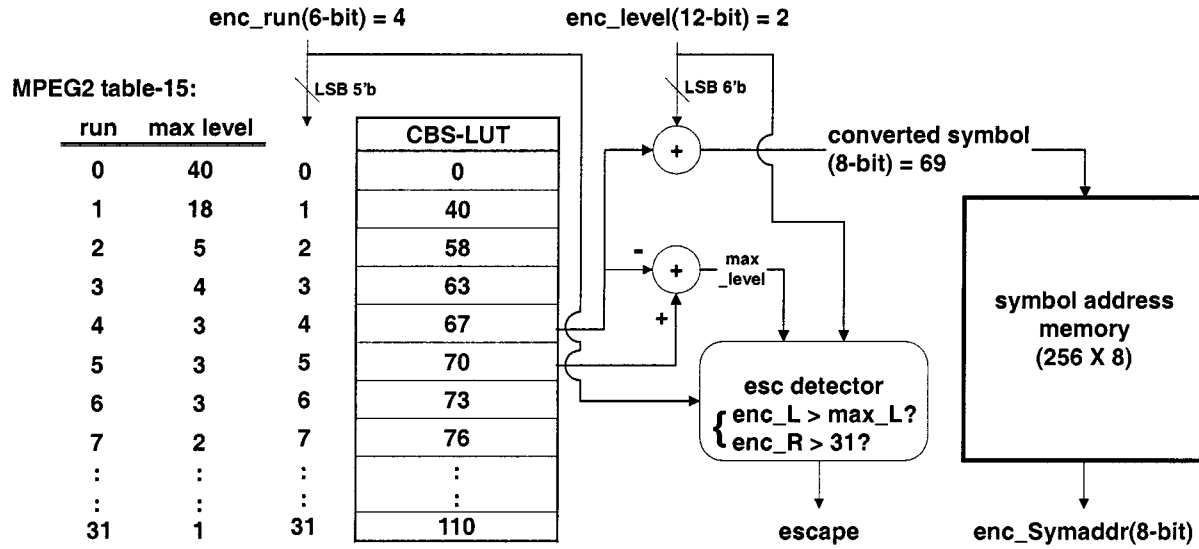


Fig. 6. Symbol conversion and CBS-LUT based on MPEG2 table-15.

group	symbol	PCLC _codeword	PCLC _codenum	symbol address	VLC _codeoffset
G1	S10	0 0 1 1 0 0 0 0	48	4	0
	S11	0 0 1 1 0 1 0 0	52	5	1

(a)

group	symbol	PCLC _codeword	PCLC _codenum	symbol address	VLC _codeoffset
G1	S10	0 0 1 1 0 0 0 0	48	4	0
G2	S11	0 0 1 1 1 1 0 0	52	5	0

(b)

group	symbol	PCLC _codeword	PCLC _codenum	symbol address	VLC _codeoffset
G1	S10	0 0 1 1 0 0 0 0 0	48	4	0
G1	S11	0 0 1 1 1 1 0 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	5	0
G3	S31	0 1 1 1 0 0 0 0	112	6	1

(c)

Fig. 7. Three techniques for saving the symbol memory space.

$PCLC_mincode_x \leq bitstream_num < PCLC_mincode_{x+1}$ is encountered. Besides, the hit condition will be $base_addr_y \leq symbol_address < base_addr_{y+1}$ if the encoded symbol is located in group G_y .

D. Overall Group-Based VLC Codec Processes

Before realizing the codec processes, the word lengths of both VLC_codeoffset and VLC_codenum operands have to be determined, since it is difficult to implement arithmetic units with variable length inputs. To perform memory mapping, the supported symbol memory must satisfy the requirement of coding tables. Consequently, the value of VLC_codeoffsets will not exceed the address space of the symbol memory. For this reason,

it is reasonable that the word length of the VLC_codeoffset operand equals that of the symbol address. On the other hand, because hardware components are designed for all codewords, VLC_codenums have to be extended to the maximum codeword length bits. However, the numerical value of VLC_codenums cannot be changed by this operation. It is necessary for the VLC_codenum operand to do sign-bit extension of an unsigned number before transmitting to arithmetic units.

Based on the word lengths of the operands discussed above, the VLC codec algorithm is completed by the group searching scheme and the intra-group encoding/decoding procedures. Detailed descriptions of the VLC codec processes and corresponding examples based on the coding table from Fig. 3 are presented in Fig. 5.

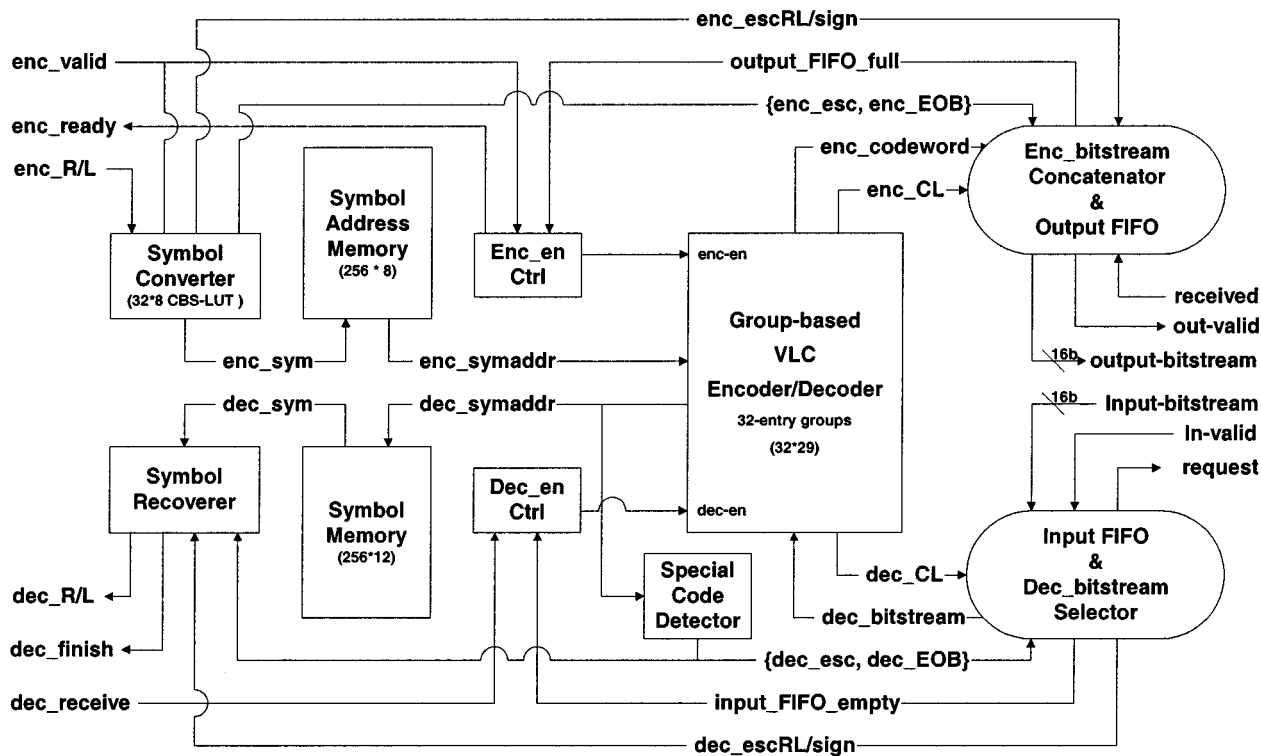


Fig. 8. Block diagram of the proposed VLC codec system for MPEG applications.

TABLE I
ANALYSIS OF SYMBOL MEMORY EFFICIENCY FOR SEVERAL IMAGE
CODING TABLES

Table Item	MPEG-2 TB01	MPEG-2 TB09	MPEG-2 TB15	JPEG Cy	JPEG CbCr
for enc/dec					
# of groups	7	6	22	15	15
# of sym_mem location	48	63	122	164	162
# of non-used location	15	0	9	2	0
decode only					
# of groups	7	6	22	15	15
# of sym_mem location	34	63	113	162	162
# of non-used location	1	0	0	0	0
partition group					
# of groups	7+1	6	22+5	15+2	15
# of sym_mem location	33	63	113	162	162
# of non-used location	0	0	0	0	0

E. Memory Requirement Reduction for Symbol Information

Because memory modules may occupy large area, minimizing memory requirements can reduce the cost of a system. Data to be stored for the proposed VLC codec processes are group information, encoded symbol addresses, and decoded symbols. For a table with 256-entry 12-bit symbols and 16-bit codewords, the size of the symbol address memory is $2^{12} \times 8$ bits for fetching 8-bit symbol addresses by 12-bit encoded symbols. The symbol memory space is $2^8 \times 12$ bits for accessing 12-bit decoded symbols by 8-bit symbol addresses. Besides, it needs $n \times (1 + 4 + 16 + 8)$ bits storage space for n-entry

group information which consists of 1-bit valid, 4-bit codeword length, 16-bit PCLC_mincode, and 8-bit base address.

It is essential to shorten the 12-bit symbols since the memory efficiency is low for storing 256-entry symbol addresses in 2^{12} locations. For MPEG DCT coefficient tables, one technique that converts Run-Level-Pairs (RLP) into 8-bit converted symbols is presented in [15]. However, it is not a proper method for programmable symbols because different RLPs can be transformed into the same converted symbol. A novel symbol conversion is shown in Fig. 6 based on MPEG2 table-15. To generate compact conversion results for arbitrary RLPs, the proposed converted symbols are the sum of the encoded level and the conversion-based symbol (CBS) which accumulates the maximum level from run_0 to run_{n-1} for each run_n . Escaped RLPs are detected by comparing the value of the encoded level with the maximum level of the encoded run. With a memory-based CBS look-up table (CBS-LUT), this symbol conversion technique is suitable for programmable RLPs. Furthermore, the memory requirement for finding encoded symbol addresses is reduced to $(2^5 \times 8 + 2^8 \times 8)$ to obtain 8-bit CBS's by 5-bit encoded runs and fetch 8-bit symbol addresses by 8-bit converted symbols. Therefore, with this symbol conversion, the total memory space of the proposed VLC codec processes is now reduced to $((2^5 \times 8 + 2^8 \times 8) + 2^8 \times 12 + n \times 29)$ bits.

Three techniques for saving the symbol memory space are shown in Fig. 7. According to the proposed intra-group symbol memory mapping, discontinuous VLC_codeoffsets induce unused locations in the symbol memory, such as locations 5, 6, and 18 in Fig. 3. For user-defined coding tables, reassigning the codewords in continuous numerical sequence results in saving memory space as shown in Fig. 7(a). Because the codeword

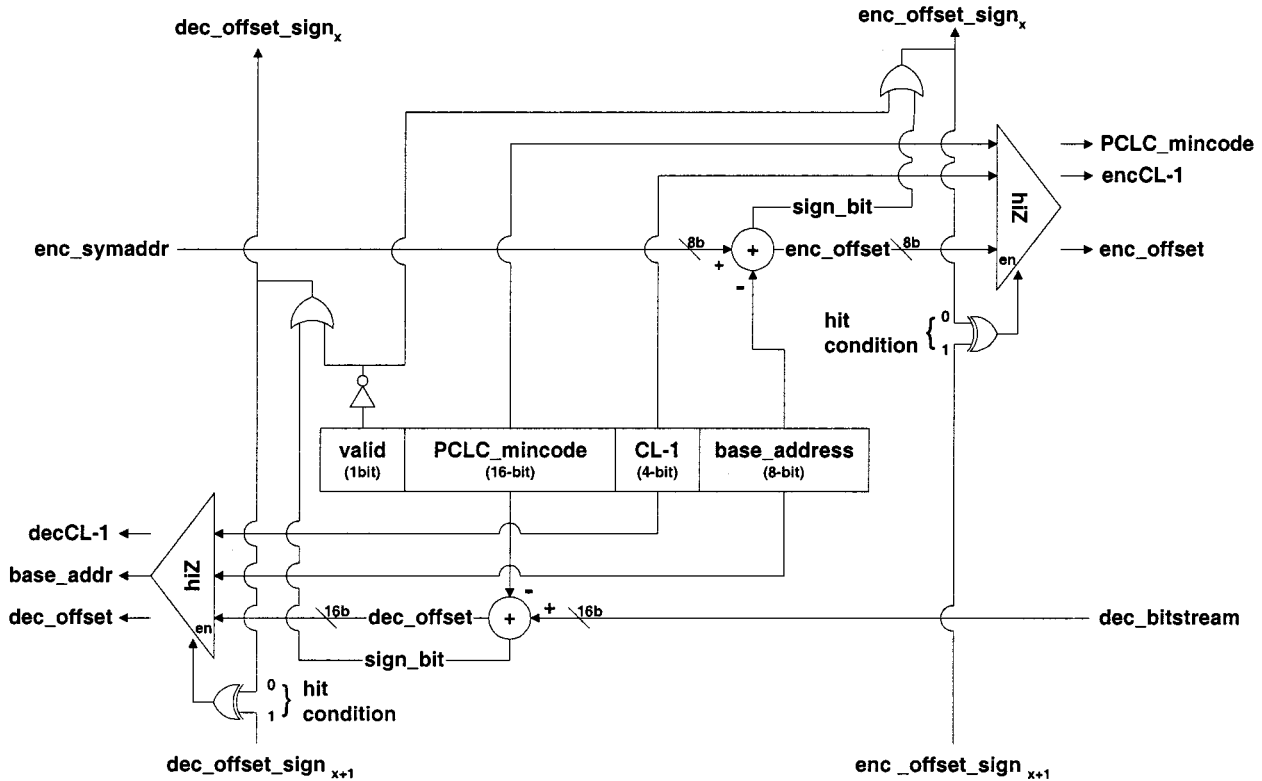


Fig. 9. Detailed schematic of a group detector.

length is identical, the changed codeword do not affect the compression ratio. Nevertheless, this technique cannot be applied to standard-defined tables where the codewords cannot be changed. In this case, partitioning the discontinuous codewords into individual groups is the method to reduce the memory requirement as shown in Fig. 7(b). In addition, it is not necessary for decompression applications to perform inter-group symbol memory mapping. If the intra-group memory mapping is satisfied, the base address can be any location in the memory. Consequently, another group, such as G_3 in Fig. 7(c), is allowed to occupy the unused memory locations for increasing the memory efficiency. An analysis of symbol memory efficiency for several image coding tables is given in Table I.

III. GROUP-BASED VLC CODEC SYSTEM ARCHITECTURE

The proposed VLC codec system is designed for MPEG applications with coding tables up to 256-entry 12-bit symbols and 16-bit codewords. This system performs concurrent encoding and decoding procedures by accessing the same group information and achieves table programmability by loading data into on-chip memories. To complete the VLC codec processes for MPEG videos, this design includes the operations of sign bits and escaped run-levels (escRL) following VLC codewords. By the efficient symbol conversion, the memory requirement is reduced to $(2^5 \times 8 + 2^8 \times 8 + 2^8 \times 12 + 32 \times 29)$ bits for a CBS-LUT, a symbol address memory, a symbol memory, and 32-entry group-information. Block diagram of the proposed VLC codec system is shown in Fig. 8. It mainly consists of the following components.

- 1) The group-based VLC encoder/decoder is composed of group detectors and combinational logic circuits to realize the VLC codec processes.
- 2) The input FIFO stores the input bit stream. According to previous decoded results, the Dec_bitstream selector transmits codeword bit streams to the VLC decoder. Besides, this selector detects sign bits and escRLs when VLC codewords are decoded.
- 3) The Enc_bitstream concatenater adds sign bits or escRL's behind VLC codewords and concatenates encoded results into a single bit stream. Then, every 32 bits of the encoded bit stream in the concatenater is shifted into the Output FIFO.
- 4) The special code detector recognizes special codes, such as escape and EOB, by checking decoded symbol addresses instead of decoded symbols. Without waiting for symbol fetching, this detector can determine the length of the additional bits following a VLC codeword. Hence, the next codeword bit stream can be found by the Dec_bitstream selector immediately and the decoding throughput can be increased.
- 5) The Enc_en and Dec_en Ctrls determine the operations of the VLC encoder and decoder according to the condition of input data and FIFOs.
- 6) Both symbol address and symbol memories are the on-chip memory modules for storing symbol information.
- 7) The symbol converter performs symbol conversion and detects escaped RLP's and EOB symbols. On the other hand, the symbol recoverer finds correct runs and signed levels based on decoded results.

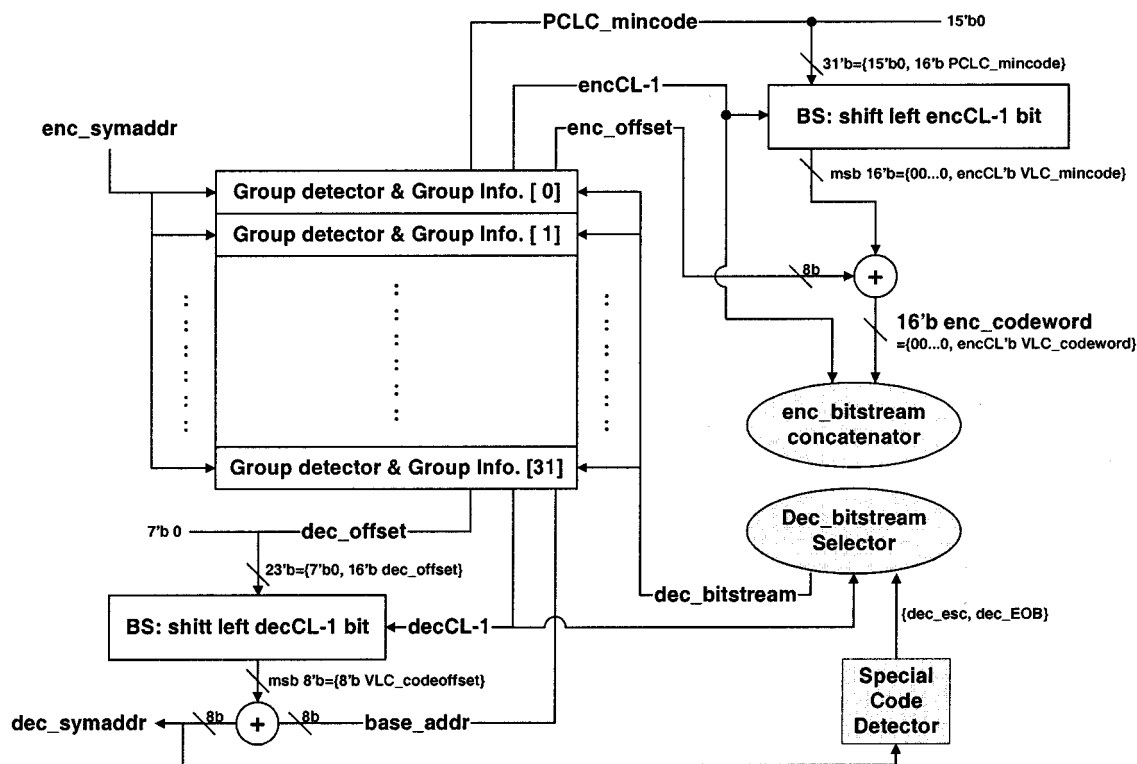


Fig. 10. Architecture of group-based VLC encoder/decoder.

A. Detail Architecture of Main Components

1) *The Group Detector*: A schematic of the group detector is given in Fig. 9. The format of the stored group information is {valid, PCLC_mincode, CL-1, base_address}. The word length of the PCLC_mincode is 16 bits, to satisfy coding tables having 16-bit codewords. Because a codeword is at least 1 bit, the codeword length minus one (CLB1) is stored to reduce memory space. The 8-bit base_address is designed for a 256-entry symbol memory. In addition, two subtractors realize the arithmetic operations, ($8'b \text{ enc_symaddr} - 8'b \text{ base_addr} = 8'b \text{ enc_offset}$) and ($16'b \text{ dec_bitstream} - 16'b \text{ PCLC_mincode} = 16'b \text{ dec_offset}$). The numerical comparison results, sign_bits, are transmitted to the XOR gates. According to the group searching scheme, the hit condition of group G_x can be expressed by ($\text{sign}_x = 0, \text{sign}_{x+1} = 1$). Therefore, the XOR gate of the matching group turns on the tri-state buffers to transmit the group information. For this reason, the sign_bit of unused group detectors must be “1” to guarantee that the result of group searching is correct.

2) *The Group-Based VLC Encoder/Decoder*: An architecture of the Group-based VLC Encoder/Decoder is presented in Fig. 10. Monotonic codewords with leading characters, such as JPEG AC tables, generate 15 groups when the codeword lengths vary from 2 to 16-bit. For this reason, 32 group detectors are sufficient for most of coding tables containing both leading-1 and leading-0 codewords. Nevertheless, the number of group detectors has to be increased for irregular or sparse coding tables, which have a large number of codeword groups. The tri-state buffers of every group detector are connected together to transmit the matching group information, since only one group detector encounters the hit condition. Two barrel shifters (BS)

select the valid VLC_mincode and VLC_codeoffset for the encoding and decoding processes, respectively. Because adding zero bits 15'b0 and 7'b0 to the inputs of two barrel shifters performs the sign-bit extensions of unsigned numbers, the outputs of the barrel shifters are the fixed-length operands with correct numerical values. After the arithmetic operations are completed, the encoded codeword length minus one, encCL-1, and the encoded VLC_codenum, enc_codeword = 16'b {00...0, encCL'b VLC_codeword}, are transmitted to the Enc_bitstream Concatenator. On the other hand, the 8-bit decoded symbol address, dec_symaddr, is sent to both Symbol Memory and special code detector. Besides, the decoded codeword length minus one (decCL-1) is feedback to the Dec_bitstream Selector for finding the next codeword bit streams.

3) *The Dec_bitstream Selector and the Special Code Detector*: A block diagram of the Dec_bitstream Selector that detects 16-bit codeword bit stream, 18-bit escRL, and 1-bit sign is depicted in Fig. 11. The operation of the special code detector is presented here, too. To decode one complete codeword at a time, two 32-bit buffers—MSB32'breg and LSB32'breg—are used for storing the bit stream. The start pointer of the decoded VLC codeword in the buffers is the decCL_acc which accumulates the length of decoded bits. According to this pointer, one barrel shifter selects undecoded 32 bits, dec_bitstream32, from the buffers. Then, the 16 most significant bits of the dec_bitstream32, dec_bitstream are transmitted to the VLC decoder. After receiving the decoded codeword length, the other barrel shifter shifts decCL-1 bits from the 31 less significant bits of the dec_bitstream32 to find escRL's and sign bits. The special code detector determines the lengths of additional bits when decoded symbol addresses

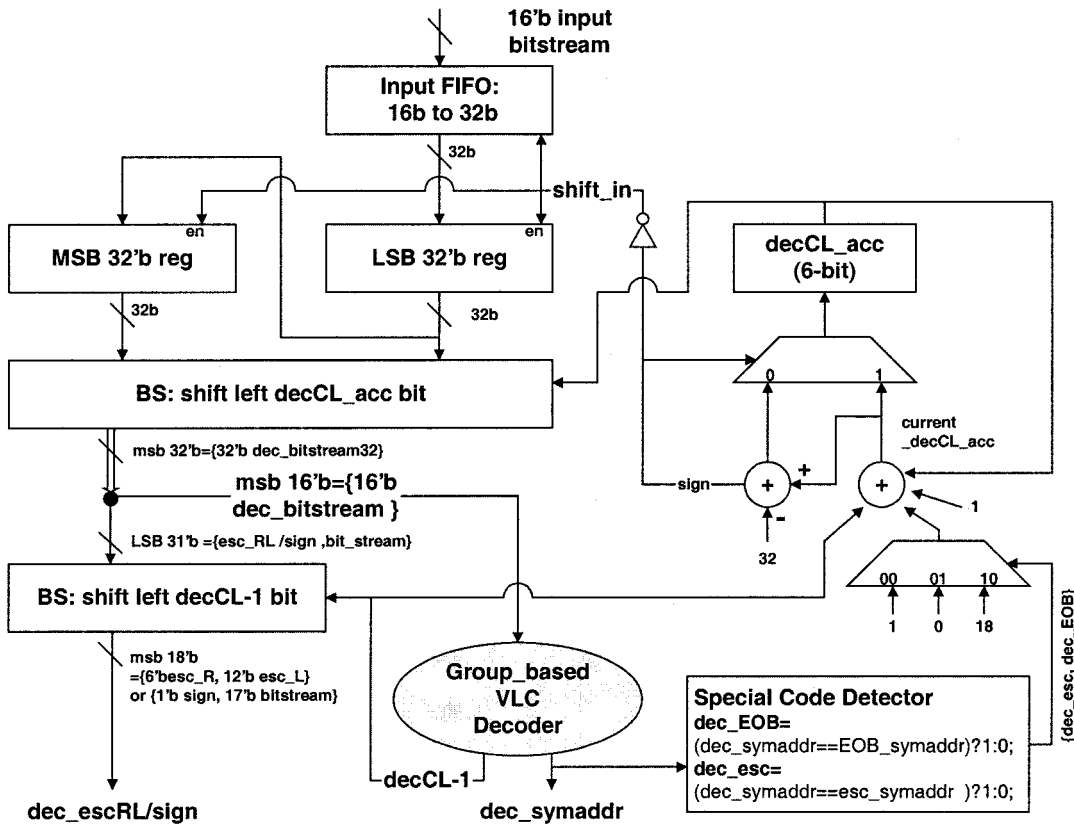


Fig. 11. Block diagram of Dec_bitstream selector and special code detector.

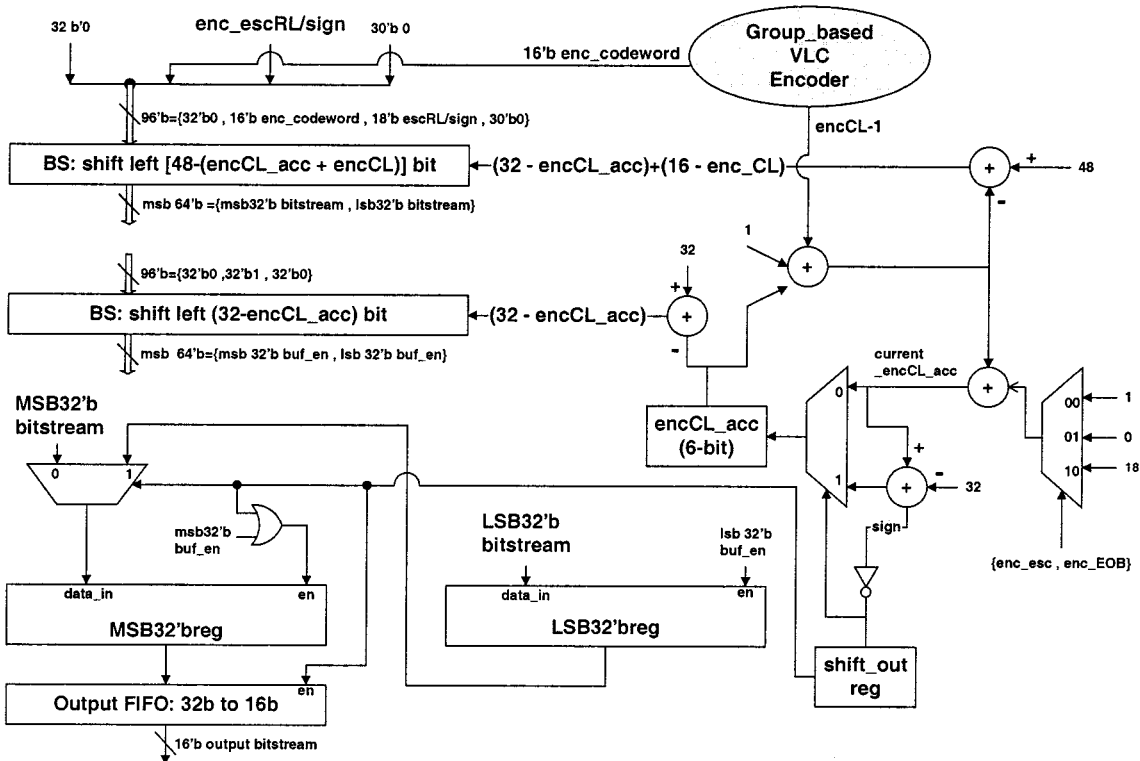


Fig. 12. Block diagram of Enc_bitstream concatenator.

are available. Therefore, current decCL_acc can be calculated immediately. If current decCL_acc exceeds 32, the stored data of the LSB32'breg will replace that of the MSB32'breg and

32 bits input bit stream in the Input FIFO will be shifted into the LSB32'breg. Consequently, the next decCL_acc must be updated from current decCL_acc subtracted by 32.

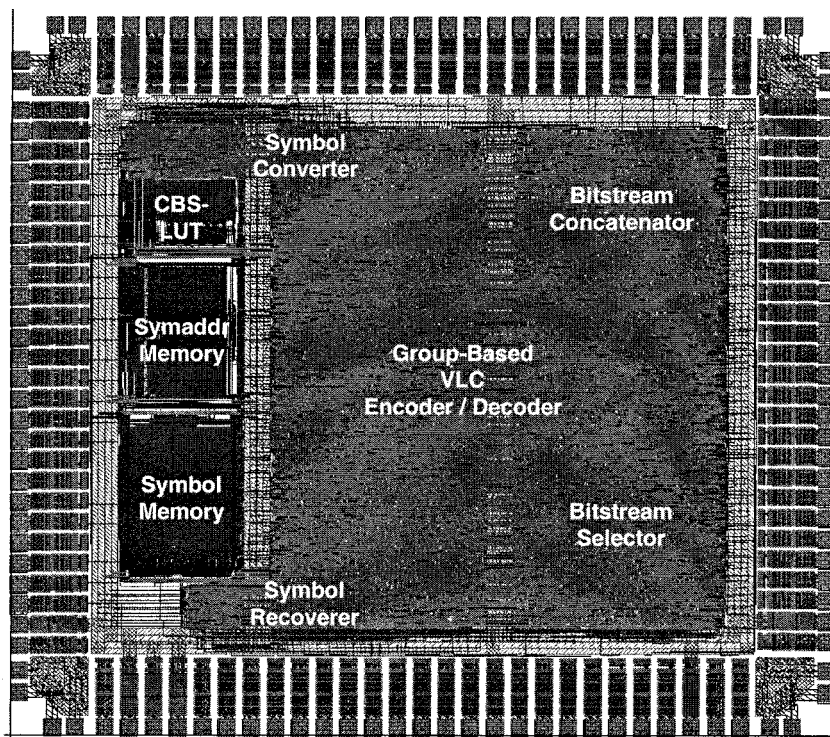





Fig. 13. Chip layout of the proposed VLC codec system.

TABLE II
SIMULATION RESULTS BASED ON HDTV SYSTEMS (I-FRAME)

image: (4:2:2)@ 1920 X 1080 simulation results			
# of bitstream (bit)	3439392	1912640	2114464
# of symbols	590302	252817	289129
Encode cycle	590348	252864	289173
Decode cycle	590337	252862	289170

Average symbol rate: 99.98 Msps @ 100MHz-clk rate. Some overheads are introduced due to the stalls of the bit stream FIFOs.

4) *The Enc_bitstream Concatenator*: A block diagram of the Enc_bitstream Concatenator is illustrated in Fig. 12. To deal with escRLs, two 32-bit buffers, MSB32'breg and LSB32'breg, are applied to perform the concatenation scheme. The encCL_acc, which accumulates the length of encoded results, is the start pointer for storing current encoded bits to the buffers. According to this pointer, the concatenation scheme is using one barrel shifter for shifting the encoded bits to the inputs of correct registers and the other barrel shifter for transmitting the signals, buf_en, to enable these registers. Therefore, the encoded bits can be concatenated without overwriting the previous encoded results. When current encCL_acc exceeds 32, the shift-out signal is activated to transmit the encoded bit stream in the MSB32'breg to the Output FIFO and overwrite the MSB32'breg by the LSB32'breg. Like decCL_acc, the next

encCL_acc is updated from current encCL_acc subtracted by 32.

B. Chip Implementation and Performance Estimation

The proposed VLC codec system was implemented using 0.6- μm CMOS SPTM process. It consists of two major parts: 1) an in-house 5-V standard cell library and 2) memory modules. To satisfy a coding table up to 32 codeword groups and 256-entry 12-bit symbols and 16-bit codewords, the memory modules of this system are 32×8 -bit CBS-LUT, 256×8 -bit symbol address memory, 256×12 -bit symbol memory, and 32×29 -bit group information. In addition, both output and input FIFOs are 64-bit buffers. Nevertheless, their sizes have to be modified to meet application requirements. For simplifying the I/O control, these FIFOs align the output and the input bit streams to 16 bits, i.e., 2 B.

TABLE III
COMPARISON WITH EXISTING VLC CODEC DESIGNS

Design	[15]	[7]	[8]	proposed
Function	VLC enc/dec (with symbol conversion and recovery)	VLC enc/dec (with symbol conversion and recovery)	VLC enc/dec	VLC enc/dec (with symbol conversion and recovery)
Mode	switched	NA	NA	concurrent
Table constrain	no	leading 1	no	no
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	$4 \times 16 \times 256$ = 16K bits	256×12 + 16×16 + $128 \times 10 \times 12$ = 18K bits	512×12 = 6K bits	$32 \times 8 + 256 \times 8$ + 256×12 + 32×29 = 6.3K bits
Transistor count	65k (without memory)	not available	50k (with memory)	110k (with memory)
Chip Area (mm ²)	2.5×1.8 (core area)	not available	6.8×6.9 (with pad)	5.0×4.5 (with pad)
Clock rate	81 MHz	30 MHz	83.3 MHz	100 MHz
System performance (symbol/sec)	40.5 Msps (enc/dec)	30 Msps (enc/dec)	enc: 11.9 Msps dec: 7.6 Msps	100 Msps (enc/dec)

To enhance system performance, this VLSI solution is designed to achieve concurrent codec processes and constant symbol rate, i.e., one symbol per cycle. Because of the special code detector, the Dec_bitstream Selector can determine next codeword bit streams without stalls. Therefore, the pipeline stages for this concurrent VLC codec system is organized as follows:

- stage 1) the Symbol Converter/Recoverer;
- stage 2) the Symbol Address/Symbol Memories;
- stage 3) the Group-based VLC Encoder/Decoder and the bit stream Concatenator/Selector.

Additionally, the pipeline is stalled when the enc_valid/dec_receive signals are disabled or the output_FIFO_full/input_FIFO_empty flags are set. Simulation results based on HDTV systems are given in Table II. These results show that the operation performance of this chip design achieves 100 Msymbols/s at 100-MHz clock rate with 5-V supply voltage. Because the bit streams are aligned to 16 bits, some overheads are induced due to the stalls of the Input and Output FIFOs. Moreover, a comparison with existing VLC codec designs is given in Table III. It shows that the symbol rate of the proposed design is about 2.5 times [15] and 3 times [7].

IV. CONCLUSION

In this paper, the algorithm and architecture of a VLC codec system with a new group-based approach have been presented. Based on the codeword grouping and symbol memory mapping, both encoding and decoding procedures are completed by applying numerical properties to codewords and symbol addresses. Using the proposed PCLC table, the group searching

scheme is accomplished by arithmetic operations. In addition, by a novel symbol conversion, not only memory space reduction for symbol information but also full table programmability can be achieved. A 0.6- μm CMOS chip that performs table programming and concurrent VLC codec processes has been designed for MPEG applications. Simulation results show that this VLSI solution achieves compression/decompression rates up to 100 Msymbol/s at a 100-MHz clock rate. Thus, the proposed solution is suitable for high throughput applications, such as HDTV, and concurrent VLC codec applications, such as video-conferencing.

ACKNOWLEDGMENT

The authors would like to thank their colleagues within the SI2 group of NCTU for many fruitful discussions.

REFERENCES

- [1] D. A. Huffman, "AA method for the construction of minimum-redundancy codes," *Proc. IRE*, vol. 40, pp. 1098–1101, Sept. 1952.
- [2] S.-M. Lei and M.-T. Sum, "AA parallel variable-length-code decoder for advanced television applications," in *Proc. 3rd Int. Workshop on HDTV*, Aug. 1989.
- [3] S.-M. Lei and M.-T. Sum, "An entropy coding system for digital HDTV applications," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 1, pp. 147–155, Mar. 1991.
- [4] A. Mukherjee, N. Ranganathan, and M. Bassiouni, "Efficient VLSI design for data transformations of tree-based codes," *IEEE Trans. Circuits Syst.*, vol. 38, pp. 306–314, Mar. 1991.
- [5] A. Mukherjee, H. Bheda, and T. Acharya, "Multibit decoding/encoding of binary codes using memory-based architectures," in *Proc. Data Compression Conf.*, Snowbird, UT, Apr. 1991, pp. 352–361.
- [6] S.-F. Chang and D. G. Messerschmitt, "Designing a high-throughput VLC decoder Part I-B concurrent VLSI architectures," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 2, pp. 187–196, June 1992.

- [7] P. A. Ruetz, P. Tong, D. Luthi, and P. H. Ang, "A video-rate JPEG chip set," *J. VLSI Signal Processing*, vol. 5, pp. 141–150, 1993.
- [8] A. Mukherjee, N. Ranganathan, J. W. Flieder, and T. Acharya, "MARVLE: A VLSI chip for data compression using tree-based codes," *IEEE Trans. VLSI Syst.*, vol. 1, pp. 203–213, June 1993.
- [9] H. Park and V. K. Prasanna, "Area efficient VLSI architectures for Huffman coding," *IEEE Trans. Circuits Syst.*, vol. 40, pp. 568–575, Sept 1993.
- [10] Y. Ooi, A. Taniguchi, and S. Demura, "A 162Mbit/s variable length decoding circuit using an adaptive tree search technique," in *Proc. IEEE 1994 Custom Integrated Circuits Conf.*, May 1994, pp. 107–110.
- [11] R. Hashemian, "Design and hardware implementation of a memory efficient Huffman decoding," *IEEE Trans. Consumer Electron.*, vol. 40, pp. 345–352, Aug. 1994.
- [12] S. B. Choi and M. H. Lee, "High speed pattern matching for a fast Huffman decoder," *IEEE Trans. Consumer Electron.*, vol. 41, pp. 97–103, Feb. 1995.
- [13] B. W. Y. Wei and T. H. Meng, "A parallel decoder of programmable Huffman codes," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 5, pp. 175–178, Apr. 1995.
- [14] C.-T. Hsieh and S. P. Kim, "A concurrent memory-efficient VLC decoder for MPEG applications," *IEEE Trans. Consumer Electron.*, vol. 42, pp. 439–446, Aug. 1996.
- [15] Y. Fukuzawa, K. Hasegawa, H. Hanaki, E. Iwata, and T. Yamazaki, "A programmable VLC core architecture for video compression DSP," *Proc. IEEE SiPS '97 Design and Implementation (formerly VLSI Signal Processing)*, pp. 469–478, Nov. 1997.



Bai-Jue Shieh was born in Taipei City, Taiwan, R.O.C. in 1974. He received the B.S. and M.S. degrees from National Chiao Tung University, Hsinchu, Taiwan, R.O.C., in 1996 and 1998, respectively, both in electrical engineering. Since September 1998, he has been working toward the Ph.D. degree in the Department of Electronics Engineering, National Chiao Tung University, as part of the SI2 Research Group.

His research interests include IC design flow, cell-based and fully-custom VLSI design, video signal processing, system-on-chip design technology, cell library design, and memory circuit design.



Yew-San Lee was born in Muar City, Johore, Malaysia, in 1971. He received the B.S. and M.S. degrees in June 1995 and 1997, respectively, from the Department of Electronics Engineering, National Chiao Tung University, Hsinchu, Taiwan, R.O.C. Since September 1997, he has been working toward the Ph.D. degree in the Department of Electronics Engineering, National Chiao Tung University, as part of the SI2 Research Group.

His research interests include advanced VLSI design for video signal processing and compression, error detection and correction coding, high-performance cell library and memory circuit design, digital phase-locked loop, mix-mode IC design, and related CAD design.



Chen-Yi Lee received the B.S. degree from National Chiao Tung University, Hsinchu, Taiwan, R.O.C., in 1982, and the M.S. and Ph.D. degrees from Katholieke University Leuven (KUL), Belgium, in 1986 and 1990, respectively, all in electrical engineering.

From 1986 to 1990, he was with IMECNSDM, working in the area of architecture synthesis for digital signal processing (DSP). In February 1991, he joined the faculty of the Electronics Engineering Department, National Chiao Tung University, where he is currently a Professor. His research interests include VLSI algorithms and architectures for high-throughput DSP applications. He is also active in various aspects of high-speed networking, system-on-chip design technology, very low bit-rate coding, and multimedia signal processing.