Codec Design for Variable-length to Fixed-length Data Conversion for H.263

Chua-Chin Wang, *Senior Member, IEEE*, Gang-Neng Sung, and Jia-Hao Li
Dept. of Electrical Engineering
National Sun Yat-Sen University
Kaohsiung, Taiwan 80424
Email: ccwang@ee.nsysu.edu.tw

***Abstract*— A codec (encoder-decoder) design for interfacing variable-length and fixed-length data conversion is proposed in this paper. The poor memory efficiency of the variable-length compression approach can be avoided while its advantages can be preserved. The introduction of the proposed codec converts the variable-length symbols into fixed-length packets which can be hardwaredly and parallelly processing without waiting for any other information, which indicates the decoding process is context-free. The proposed codec encodes extra symbols in the redundant bits of the padding bits of the fixed-length packets. This novel encoding scheme relaxes the intrinsic poor bit rate of the traditional fixed-length data compression.**

Keywords : fixed-length, variable-length, compression, H.263, sparsity

I. Introduction

The effective real-time high-quality video signal of the upcoming DTV (digital TV) era heavily replies on the transmission and storage of the broadcast data, [2]. Both of the transmission and storage of such a huge amount of data demand efficient compression and decompression (comp-decomp) schemes to meet the requirements of the real-time signals, e.g., video signals. Many comp-decomp approaches have been developed, e.g., VQ (vector quantization), DCT [5], run-length, entropy constrain, etc. When it comes to the high-speed data compression, H.263 has been recognized as a more powerful data compresseion standard than the MPEGs, i.e., MPEG2, MPEG4, etc., [8]. The combination of DCT (discrete cosine transform), run-length coding and Huffman coding has brought the success to H.263 and MPEG, [3], [7]. These variable length approaches enjoys a higher compression rate than the fixed-rate or fixed-length methods, e.g., [1], [9], based on the theoretical analysis [6]. However, the variable-length approaches suffers from two intrinsic difficulties : poor memory efficiency (that is, the sparsity problem in the memory), and data dependency which result in that the decoding is not context-free. Though the former problem can be relaxed by using sophisticated algorithms [4], the trade-off is the processing time which might lead to the sacrifice of real-time quality. The latter problem is caused by no guard bits or fields in the continuous bit stream to extract correct words as well as symbols. It makes the parallel processing or pipelining very hard to be implemented. Hence, we proposed to introduce a fixed-length and variable-length conversion interface (codec) design to avoid these two difficulties while preserve the advantage of the variable-length comp-decomp approaches. The introduction of the proposed codec converts the variable-length symbols into fixed-length packets which can be hardwaredly and parallelly processing without any waiting latency. In short, the most important feature of the proposed codec is to encode more symbols in the redundant bits (i.e., the padding bits) of the fixed-length packets.

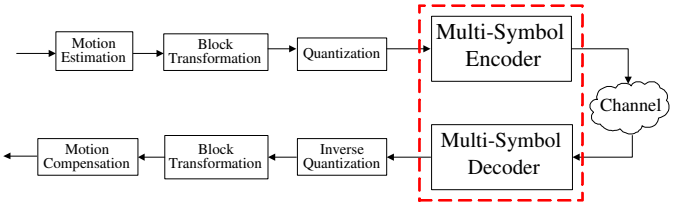

Fig. 1. Application of the proposed codec

II. Codec Interface Design for H.263

It is well known that the fixed-length compression is suffered from poor compression rates. On the contrary, the variable-length approaches are slow due to the data dependency and table lookup operations. Our approach will preserve the advantage of the fixed-length methods and relax the poor compression rate by encoding extra symbols in redundant bits. The proposed codec can be use in fixed-length and variable-length compressions of H.263 as shown in Fig. 1 to attain the advantages of these two methods.

A. Multi-symbol encoding

A source H is defined as an ordered pair $H = (S, P)$, where S represents a set of source symbols $S = \{S_1, S_2, S_3, \ldots, S_n\}$ with probability distribution $P(S_i) = P_i$, and $P_1 \geq P_2 \ldots \geq P_n$. Then, a Huffman tree can be constructed basing upon the balance of the probability of the two sub-trees at each node [4]. For instance, $S_1 = 0, S_2 = 10, S3 = 110, \ldots, S_n = 111\ldots1$, as shown in Fig. 2. The length of the longest symbol is assumed to be n. In order to achieve the fixed-length compression, padding "1"s or "0"s are required to modify the mentioned S set and generate the fixed-length packets. All of the padding bits are called redundant bits (RB) which will deteriorate the memory efficiency as well the compression bit rate even if they are simply assigned to be "0" or "1", since no information is conveyed in this way.

A simple thought to utilize these redundant bits to improve the compression rate is to encode several symbols, called basic symbols (BS), together into a packet. Thus, many redundant bits are combined, which might be more than the number of basic symbols. Those symbols encoded by the redundant bits are named extra symbols (ES). Hence, the X-Y multi-symbol encoding indicates that a packet is composed of X BS's and Y ES's, where the Y ES's are denoted by the padding bits for the X BS's. For instance, a 3-1 multi-symbol encoding is summarized as follows.

Assume the 3 BSs are S_i, S_j, S_k as follows.

$$\begin{aligned} S_i &= (a_1 a_2 \ldots a_i a_{i+1} a_{i+2} \ldots a_n) = (a_1 a_2 \ldots a_i 0 \ldots 0) \\ S_j &= (b_1 b_2 \ldots b_j b_{j+1} b_{j+2} \ldots b_n) = (b_1 b_2 \ldots b_j 0 \ldots 0), \\ S_k &= (c_1 c_2 \ldots c_j c_{k+1} c_{k+2} \ldots c_n) = (c_1 c_2 \ldots c_j 0 \ldots 0), \end{aligned}$$

Proceedings of the 2006 International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP'06)
0-7695-2745-0/06 $20.00 © 2006 IEEE

where $a_1 \dots a_i$, $b_1 \dots b_j$, and $c_1 \dots c_k$ are the original bits of Huffman tree representation of S_i, S_j, and S_k, respectively. The rest of the bits are padding redundant bits. A fourth symbol, $S_l = (d_1 d_2 \dots d_l d_{l+1} d_{l+2} \dots d_n)$, where $d_1 \dots d_l$ are the original bits, is considered to be conveyed by the overall redundant bits in S_i, S_j, and S_k. The pre-requisite is $n \leq (n-i)+(n-j)+(n-k)$. Then, the encoding steps are

(E1). Label the position of the leading redundant bits of S_i, S_j, and S_k, respectively, which are p_i, p_j, and p_k.

(E2). Extend S_i, S_j, and S_k to be n-bit codewords. Then, initialize all of the redundant bits to be "0".

(E3). Start with the first bit of S_l, i.e., d_1, which is XORed with a_i. The outcome is placed at a_{p_i}. Then, the second bit of S_l, which is d_2, is XORed with a_{p_i}. Again, the resulted bit is placed at $a_{p_{i+1}}$. The XOR operation is iteratively executed till the LSB of S_i is replaced.

(E4). Then, the LSB of S_j is replaced by the XORed outcome of the LSB of S_i and the first bit of the rest bits of S_l. The XOR operation is executed from LSB of S_j toward its MSB until all of the bits of S_l is encoded. If all of the remaining bits of S_l are not encoded when all of the RBs in S_j run out, the first bit of the remaining original bits of S_l will be XORed with the b_{p_i}. Then, the outcome is placed in c_{p_k} of S_k. The XOR operation is executed toward the end of S_k till all of the remaining bits of S_l is encoded.

Fig. 3 shows an example of the 2-1 multi-symbol encoding procedure. The proposed approach can be easily applied to a higher order encoding, e.g., 3-2, 4-2, 5-3, etc., as long as the length of all of the redundant bits are larger than the overall bits of the symbols to be encoded.

B. Multi-symbol decoding

The decoding procedure at the receiver terminal is basically a reverse process of the mentioned encoding procedure. An illustration of the 2-1 multi-symbol decoding procedure is given in Fig. 4. It is summarized as follows.

D1). Find the leading "0" in S_i, S_j, and S_k of which positions are labeled as p_i, p_j, and p_k, respectively.

D2). Start from the c_{p_k} toward its MSB which is XORed with its next bit. The generated bit is pushed into a stack.

D3). The XOR operations are executed till the c_{p_k} is hit. The, the XOR operation starts from the LSB of S_j toward its MSB till b_{p_j} is hit.

D4). The top of the stack is then XORed with the LSB of S_i. The outcome is also pushed into the stack. The XOR operations of adjacent bits are then executed toward MSB until a_{p_i} is hit. Every generated bit is pushed into the stack.

D5). Pop the first n bits of the stack which is S_l with padding 0's.

C. Multi-symbol conversion for H.263

Better compression ratio leads to a lower bit rate which is highly welcomed in digital video signal transmissions. It is an interesting problem to know whether there is an optimal solution for the multi-symbol encoding to attain an optimal compression rate. The mentioned multi-symbol encoding is defined as an X-Y encoding, which indicates that there are Y symbols encoded in the redundant bits of a total of X symbols. Referring to Fig. 5 which is the Huffman tree table for H.263. The shortest code is 3 bits, while the longest one is 13 bits including the sign bit. Statistically, it will stand a better chance for 3 BS's to encode one ES after a series of simulations.

Hence, the entire variable-length to fixed-length codec design for H.263 is based on the mentioned algorithms and analysis. A codec using 3-1 multi-symbol encoding scheme is implemented as follows.

Encoder : It is basically a finite state machine (FSM) as illustrated in Fig. 6. The only difference between the flow chart of Fig. 6 and the encoding algorithm (Step E1 to E4) is that we check the remaining redundant bits after each extra symbol is encoded. If there are enough redundant bits for the next symbol, it will be encoded right after the previous extra symbol by the similar XOR operations. Hence, the compression ratio can be further reduced. Notably, at most 2 extra symbols can be encoded theoretically in such an encoding scheme.

Decoder : A single-symbol (SS) decoder is required to decompose the received BS symbol into redundant bits and original symbol bits. The received data are fed into a redundant-bit (RB) counter to find out the number of RB's. The original received data are shifted right and then left by the same number of RB's count to recover the symbol bits. Both of the RB's count and symbol bits are output for the next stage.

A 2-stage decoder is shown in Fig. 7 to derive the ES bits. The RB's count of every basic symbol SS decoder is fed to the ES decoder. So are the received data's. The ES decoder is one FSM which realize the 3-1 decoding scheme addressed in Step D1 to D5. Notably, the ES_active output of the ES decoder indicate whether the ES is validated or not.

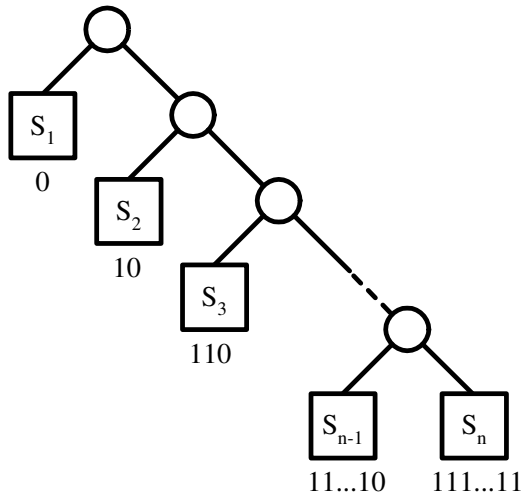

Fig. 2. A Huffman tree example (a skewed tree)

III. Simulation and Implementation

TSMC (Taiwan Semicondutor Manufacturing Company) 0.18 μm 1P6M CMOS technology cell library is adopted to implement the proposed design. Fig. 8 shows the layout of the design. Post-layout simulation results of 3-1 encoding H.263 is revealed in Fig. 9. By contrast, Fig 10 is the decoding results. The operating clock is 166 MHz at all simulation corners. The bit rate is 900 Mbps to 1.5 Gbps given a 100 MHz system clock, which meets the 5 Kbps to 1.0 Gbps requirement of H.263.

IV. Conclusion

A novel multi-symbol encoding method is proposed in this work. It enhances the poor bit rate of prior fixed-length compression approaches by encoding more symbols in the redundant bits. Parallel decoding at the receiver terminal is maintained. On the other hand, the poor memory efficiency and slow speed of prior variable-length compression approaches are avoided.

Proceedings of the 2006 International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP'06)
0-7695-2745-0/06 $20.00 © 2006 IEEE

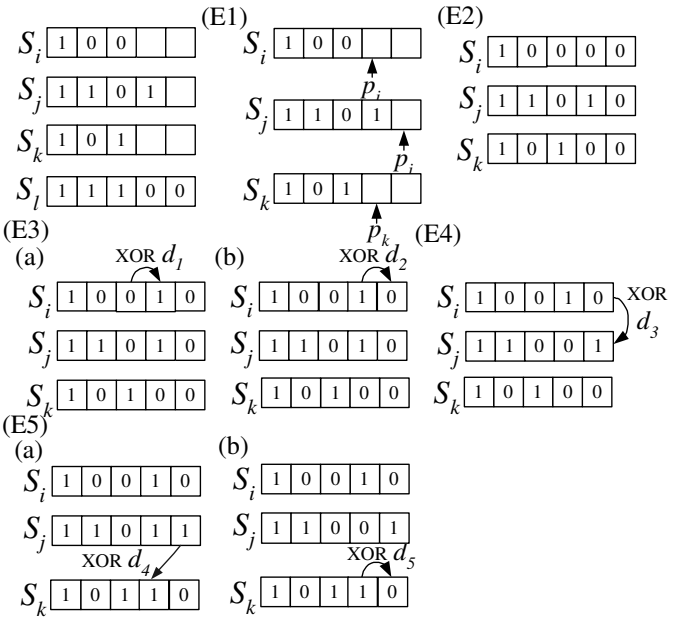

Fig. 3. An example of 3-1 multi-symbol encoding

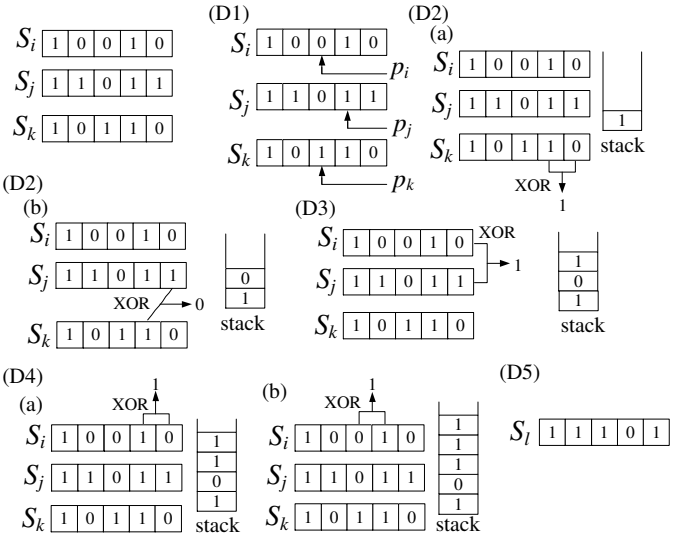

Fig. 4. An example of 3-1 multi-symbol decoding

Last	Run	Level	Bits	Code
0	0	1	3	10s
0	0	2	5	1111s
0	0	3	7	0101 01s
0	0	4	8	0010 111s
0	0	5	9	0001 1111s
0	0	6	10	0001 0010 1s
0	0	7	10	0001 0010 0s
0	0	8	11	0000 1000 01s
0	0	9	11	0000 1000 00s
0	0	10	12	0000 0000 111s
0	0	11	12	0000 0000 110s
0	0	12	12	0000 0100 000s
•	•	•	•	•
•	•	•	•	•
•	•	•	•	•
1	34	1	13	0000 0101 1001s
1	35	1	13	0000 0101 1010s
1	36	1	13	0000 0101 1011s
1	37	1	13	0000 0101 1100s
1	38	1	13	0000 0101 1101s
1	39	1	13	0000 0101 1110s
1	40	1	13	0000 0101 1111s

Fig. 5. H.263 symbol table

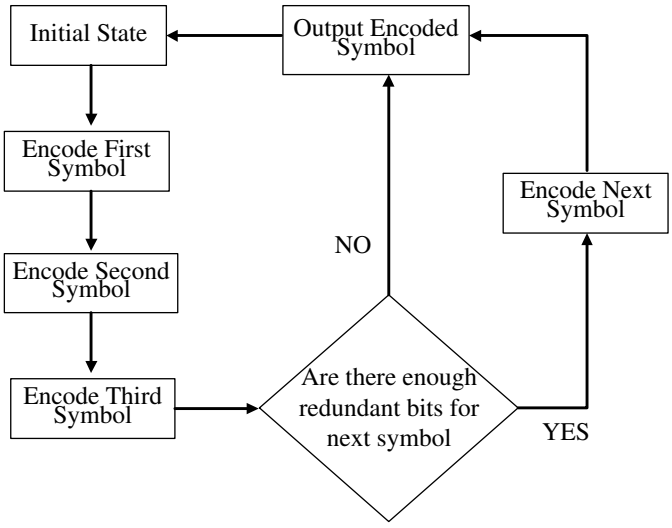

Fig. 6. FSM of the 3-1 encoder design

ACKNOWLEDGMENT

This research was partially supported by National Science Council under grant NSC 94-2213-E-110-022 and NSC 94-2213-E-110-024, and "Aim for the Top University" project of NSYSU and MOE, Taiwan. The authors would like to thank CIC of National Science Council (NSC), Taiwan, for their thoughtful help in the chip fabrication of the proposed work.

REFERENCES

[1] G. Cote, M. Gallant, and F. Kossentini, "Semi-fixed-length motion vector coding for H.263-based low bit rate video compression," *IEEE Trans. on Image Processing,* vol. 8, no.10, pp. 1451-1455, Oct. 1999.

[2] K. Challapali, X. Lebegue, J. S. Lim, W. H. Paik, R. S. Girons, E. Petajan, V. Sathe, P. A. Snopko, and J. Zdepski, "The grand alliance system for US HDTV," *Proceedings of the IEEE,* vol. 83, no. 2, pp. 158- 174, Feb. 1995.

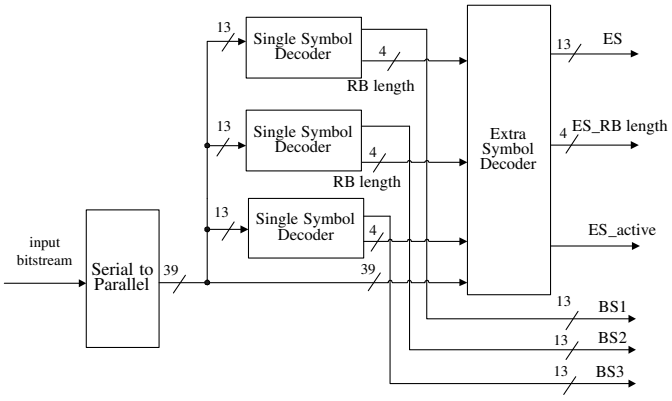

Fig. 7. Schematic of the 3-1 decoder

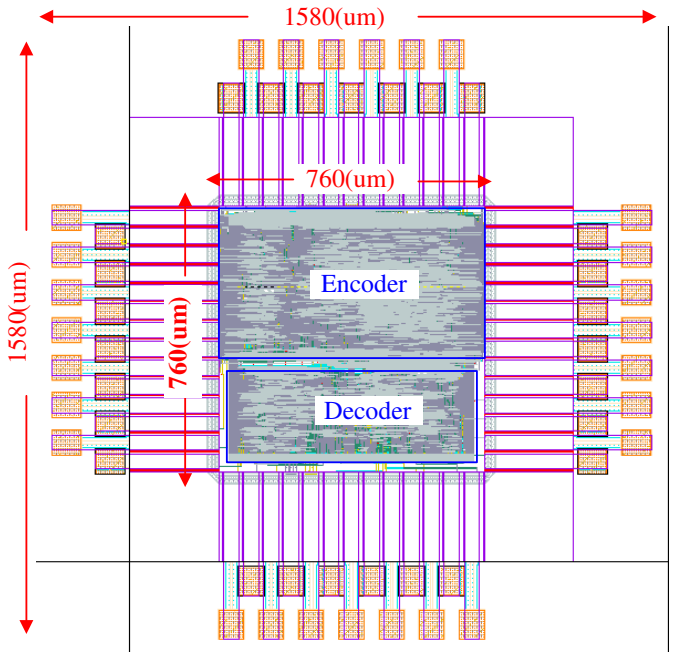

Fig. 8. Layout of the proposed codec

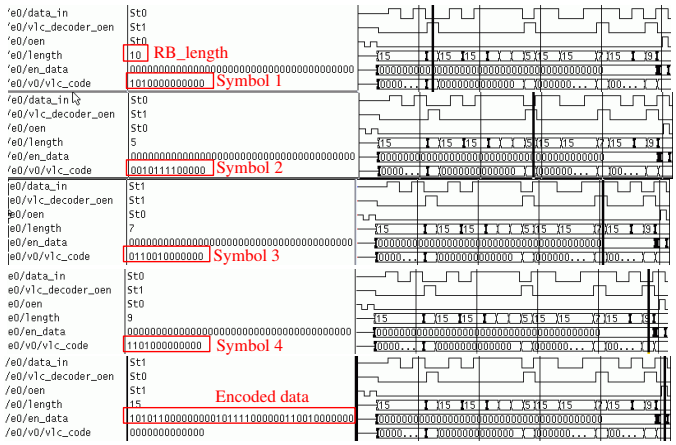

Fig. 9. Post-layout simulation result of the encoder

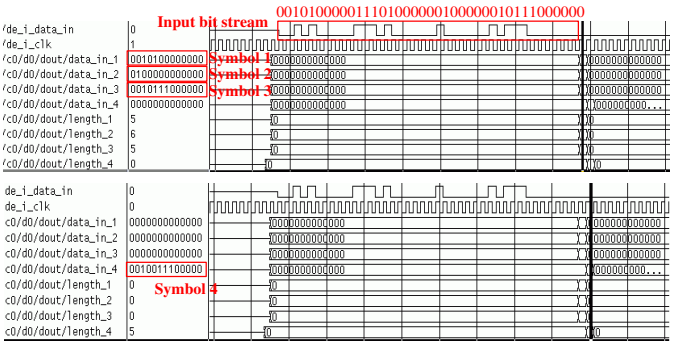

Fig. 10. Post-layout simulation result of the decoder

[3] S. B. Choi, and M. H. Lee, "High speed pattern matching for a fast Huffman decoder," *IEEE Trans. on Consumer Electronics,* vol. 41, no. 1, pp. 97-103, Feb. 1995.

[4] R. Hashemian, "Memory efficient and high-speed search Huffman coding," *IEEE Trans. on Communications,* vol. 43, no. 10, pp. 2576-2581, Oct. 1995.

[5] T. Le, and M. Glesner, "A new flexible architecture for variable length DCT trageting shape-adaptive transform," *1999 IEEE Inter. Conf. on Acoustic, Speech, and Signal Processing (ICASSP'99),* vol. 4, pp. 1949-1952, Mar. 1999.

[6] T. Lynch, "Comparison of time codes for source encoding" *IEEE Trans. on Communications,* vol. COM-22, no. 2, pp. 151-162, Feb. 1974.

[7] A. Mohri, A. Yamada, T. Yoshida, H. Sato, H. Takata, K. Nakakimura, M. Hashizume, and K. Tsuchihashi, "A real-time digital VCR encode/decode and MPEG-2 decode LSI implemented on 1 dual-issue RISC processor," *IEEE J. of Solid-State Circuits,* vol. 34, no. 7, pp. 992-1000, July 1999.

[8] I. E. G. Richardson, "Video codec design : developing image and video compression systems," John Wiley & Sons Inc., Reading : England, 2002.

[9] D. Yu, and M. W. Marcellin, "A fixed-rate quantizer using block-based entropy-constrained quantization and run-length coding," *1997 Data Compression Conference (DCC'97),* pp. 310-316, Mar. 1997.