CODEC DESIGN FOR VARIABLE-LENGTH TO FIXED-LENGTH DATA COMPRESSION BY USING MUTLI-SYMBOL ENCODING[§]

Chua-Chin Wang† , Yih-Long Tseng, and Chun-Chih Chen

Department 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 compression 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 words into a fixed-length packets which can be hardwaredly and parallelly processing without waiting for any other information. The proposed codec is to encode more 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.

Indexing terms : fixed-length, variable-length, compression, DTV, sparsity gate drive

1. INTRODUCTION

The effective real-time high-quality video signal of the up-coming DTV (digital TV) era heavily replies on the transport and storage of the broadcast data. Both of the transport and storage of such a huge amount of data demand efficient compression and decompression (comp-decomp) schemes to meet the requirements of the DTV signals. Many comp-decomp approaches have been developed, e.g., VQ (vector quantization), DCT [4], run-length, entropy constrain, etc. When it comes to the high-speed data compression, the combination of Huffman coding and run-length coding has been deemed as one of the most successful methods, [2], [6]. These variable length approaches enjoys a higher compression rate than the fixed-rate or fixed-length methods, e.g., [1], [7], based on the theoretical analysis [5]. However, the variable-length approaches suffers from two intrinsic difficulties : poor memory efficiency (a.k.a. sparsity problem in the memory), and data dependency. Though the former problem can be relaxed by using sophisticated algorithms [3], 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) 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 words into a fixed-length packets which can be hardwaredly and parallelly processing without waiting for any other information. The most important feature of the proposed codec is to encode more symbols in the redundant bits of the padding bits of the fixed-length packets.

[§] This research was partially supported by National Science Council under grant NSC 91-2218-E-110-001 and 91-2622-E-110-004.

†the contact author

2. CODEC INTERFACE DESIGN

Our approach will preserve the advantage of the fixed-length methods and relax the poor compression rate by encoding extra symbols in redundant bits.

2.1. 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 complete skewed Huffman tree can be constructed [3] : $S_1 = 0, S_2 = 10, S3 = 110, \ldots, S_n = 111\ldots1$, as shown in Fig. 1. The length of the longest symbol is $n-1$. 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 symbols. All of the padding bits are called redundant bits 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 to be able to more than the number of basic symbols. Those symbols encoded by the redundant symbols 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 redundant bits of the X BS's. For instance, a 2-1 multi-symbol encoding is summarized as follows.

Assume the 2 BSs are S_i and S_j, where $i < j$ is assumed without any loss of robustness.

$$S_i = (a_1 a_2 \dots a_i a_{i+1} a_{i+2} \dots a_n) = (a_1 a_2 \dots a_i 0 \dots 0)$$
$$S_j = (b_1 b_2 \dots b_j b_{j+1} b_{j+2} \dots b_n) = (b_1 b_2 \dots b_j 0 \dots 0),$$

where $a_1 \dots a_i$ and $b_1 \dots b_j$ are the original bits of Huffman tree representation of S_i and S_j, respectively. The rest bits are padding redundant bits. A third symbol, $S_k = (c_1 c_2 \dots c_k c_{k+1} c_{k+2} \dots c_n)$, where $c_1 \dots c_k$ are the original bits, is considered to be conveyed by the overall redundant bits in S_i and S_j. The prerequisite is $n \leq (n-i) + (n-j)$. Then, the encoding steps are

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

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

(E3). Start with the first bit of S_k, i.e., c_1, which is XORed with a_i. The outcome is placed at a_{p_i}. Then, the second bit of S_k, which is c_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_k. The XOR operation is executed from LSB of S_j to its MSB until all of the bits of S_k is encoded. It will leave b_{p_j} to $b_{p_{n-j-(n-(n-i))}}$ in S_j as "0".

Fig. 2 shows an example of the 2-1 multi-bit encoding procedure. The proposed approach can be easily applied to 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 to be encoded.

2.2. Multi-symbol decoding

The decoding procedure at the receiver terminal is basically a reverse process of the mentioned encoding procedure. An illustration is given in Fig. 3. It is summarized as follows.

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

D2). Start from the b_{p_j} toward the LSB which is XORed with its next bit. The generated bit is pushed into a stack.

D3). The XOR operations are executed till the end of S_j.

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

D5). Pop the first n bits of the stack to get S_k with padding 0's.

2.3. Compression performance analysis

Better compression ratio leads to a lower bit rate which is highly welcomed by digital video signal transmissions. It is an interesting problem to know whether there is an optimal solution for the multi-symbol encoding to attain a best 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. Given the length, n, of a symbol, the compression ratio of the X-Y multi-symbol encoding provided that the overall redundant bits can accommodate the Y symbols is defined as follows :

$$\text{COMP}_{\text{X-Y}} = \frac{\text{X}}{(\text{X} + \lfloor \frac{n-1}{n} \cdot \text{X} \rfloor)}, \quad \text{Y} = \lfloor \frac{n-1}{n} \cdot X \rfloor \quad (1)$$

A chart to illustrate the compression ratio is given in Fig. 4. Although the lowest ratio is found to be close to be high X and high Y, the compression ratio will approach to a 0.5 bound. If the hardware complexity to carry out the algorithms addressed in encoding (Step E1 to E4) and decoding (Step D1 to D5), 3-2 multi-symbol scheme which provide a 0.6 compression ratio will be a much better option to carry out the codec design.

2.4. 3-2 Multi-Symbol Codec

Hence, the entire variable-length to fixed-length codec design is based on the mentioned algorithms and analysis. A codec using 3-2 multi-symbol encoding scheme is implemented.

Encoder : It is basically a finite state machine (FSM) as illustrated in Fig. 5. The only difference between the flow chart of Fig. 5 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 a 3-2 encoding scheme.

Decoder : A single-symbol (SS) decoder is required to decompose the received BS symbol into redundant

bits and symbol bits. The schematic of the SS decoder is shown in Fig. 6. The raw_data is fed into a redundant-bit (RB) counter to find out the number of RB's. The original raw_data is 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 second 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 raw_data's. The ES decoder is one FSM which realize the 3-2 decoding scheme addressed in Step D1 to D5. Notably, the symbol_i_act output of the ES decoder indicate whether the symbol_i is validated or not.

3. SIMULATION AND IMPLEMENTATION

Avanti 0.35 μm 1P4M CMOS technology cell library is adopted to implement the proposed design. Fig. 8 shows the die photo of the design. Post-layout simulation results of 3-2 encoding 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 MPEG4.

4. 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. It is very suitable to be placed between the fixed-length and variable-length compressions to accommodate the advantages of these two methods.

5. 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] 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.

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

[4] 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.

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

[6] 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.

[7] 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.

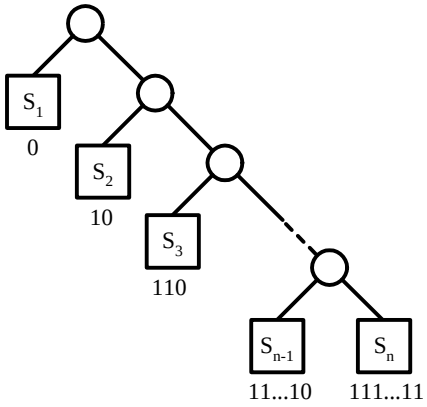

Figure 1: Skewed Huffman tree

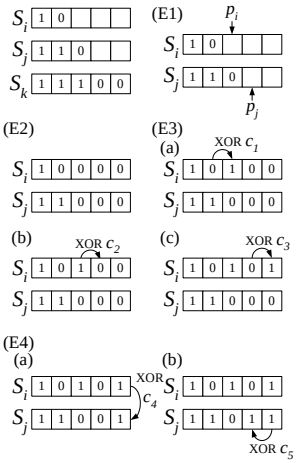

Figure 2: An example of 2-1 multi-symbol encoding

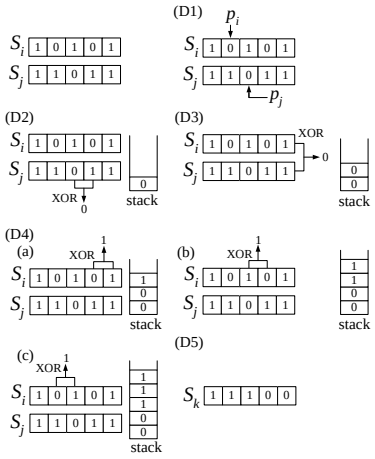

Figure 3: An example of 2-1 multi-symbol decoding

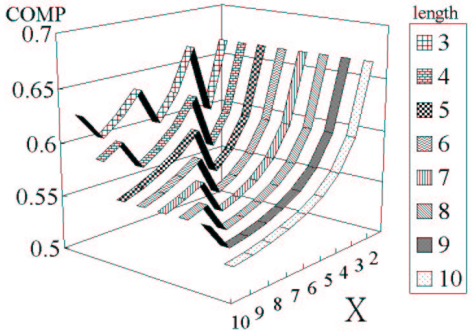

Figure 4: The compression ratio of differnt X-Y multi-symbol encoding

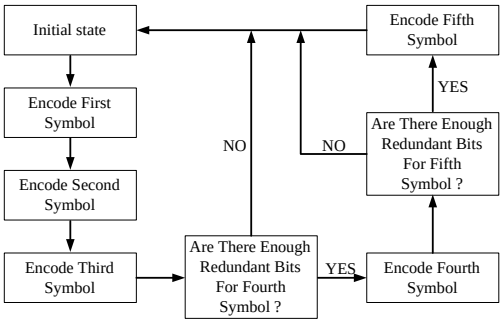

Figure 5: FSM of the 3-2 encoder design

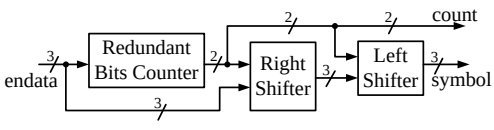

Figure 6: Single-symbol decoder

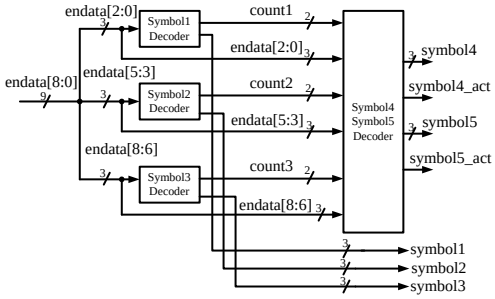

Figure 7: Schematic of the 3-2 decoder

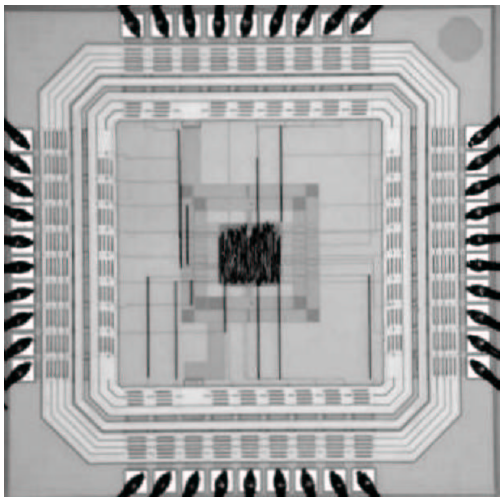

Figure 8: Die photo of the proposed codec

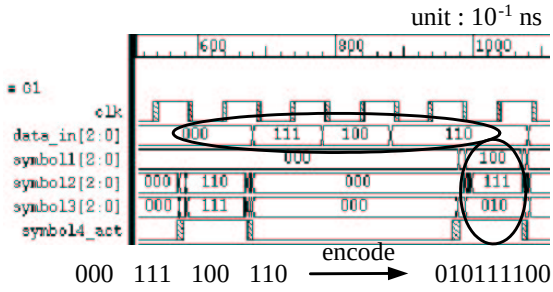

Figure 9: Post-layout simulation result of the encoder

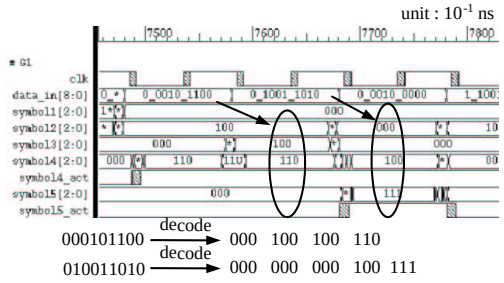

Figure 10: Post-layout simulation result of the decoder