DUAL-BLOCK-PIPELINED VLSI ARCHITECTURE OF ENTROPY CODING FOR H.264/AVC BASELINE PROFILE

Tung-Chien Chen, Yu-Wen Huang, Chuan-Yong Tsai, Bing-Yu Hsieh, and Liang-Gee Chen

DSP/IC Design Lab, Graduate Institute of Electronics Engineering and
Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan
{djchen, yuwen, cytsai, bingyu, lgchen}@video.ee.ntu.edu.tw

ABSTRACT

Direct VLSI implementation of context-based adaptive variable length coding (CAVLC) for residues, as a modification from conventional run-length coding, will lead to low throughput and utilization. In this paper, an efficient CAVLC design is proposed. The main concept is the two-stage block pipelining scheme for parallel processing of two 4×4-blocks. When one block is processed by the scanning engine to collect the required symbols, its previous block is handled by the coding engine to translate symbols into bitstream. Our dual-block-pipelined architecture doubles the throughput and utilization of CAVLC at high bitrates. Moreover, a zero skipping technique is adopted to reduce up to 90% of cycles at low bitrates. Last but not least, exponential-Golomb coding for other general symbols and bitstream encapsulation for network abstraction layer are integrated with CAVLC engine as a complete entropy coder for H.264/AVC baseline profile. Simulation results show that our design is capable of real-time processing for 1920×1088 30fps videos with 23.6K logic gates at 100MHz.

1. INTRODUCTION

Digital video compression technique has played an important role that enables efficient transmission and storage of multimedia data where bandwidth and storage space are limited. The new video coding standard, H.264/AVC [1][2], developed by Joint Video Team (JVT) significantly outperforms previous standards in compression performance[3]. The higher coding efficiency comes from the new features and functionalities including the entropy coding tools of context-based adaptive variable length coding (CAVLC). In this paper, the first published entropy coding engine for H.264/AVC baseline profile encoder is proposed.

By usage of the instruction profile and the symbol-count analysis, the reason of why entropy coding needs to be accelerated by hardware will be described. Afterwards, the architecture of the entropy coding engine is designed based on the hardware/software (HW/SW) partition. Some of the important issues of our prototype entropy coding engine are as follows. The dependency caused by content-adaptability will confine the hardware utilization and decrease the throughput. The dual-buffer architecture with block-pipeline schedule is proposed to improve the hardware utilization. The zero skip technique according to coding block pattern (CBP) in macroblock (MB) header is used to save the redundant computation especially in the low bit-rate situation. To reduce the system load in platform-based system, the network abstraction layer (NAL) encapsulation is implemented in bitstream packer. The analysis of bitstream buffer size is used to favor the burst data transmission from entropy coding engine onto the system bus. The rest of this paper is organized as follows. In Section 2, the background and the profiling analysis are mentioned . In Section 3, the architecture design of the H.264/AVC entropy coding engine is described. Section 4 shows the simulation and VLSI implementation results of our entropy coding design. Finally, the conclusion is in Section 5.

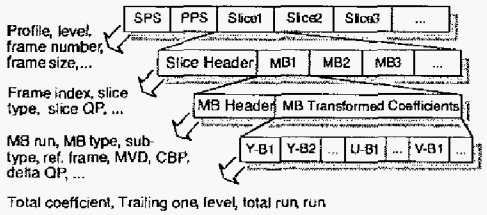


Fig. 1. Basic hierarchy of bitstream in baseline profile.

2. FUNDAMENTALS

There are two entropy coding schemes adopted by H.264/MPEG-4 AVC [4]. One is the VLC-based coding. The other is the binary arithmetic coding. In this paper, only the VLC-based coding will be described. Two VLC-based coding techniques, Exp-Golomb code and CAVLC, cooperate in H.264/AVC. CAVLC encodes the transform residues while Exp-Golomb is responsible for the rest symbols, such as prediction modes, block types, and motion vectors. In this section, the bitstream hierarchy and the analysis of instruction set profile will be mentioned. For more details , please refer to [1][5].

2.1. Bitstream Hierarchy

Figure 1 shows the hierarchical structure of bitstream. The whole sequence can be categorized into four layers, sequence layer, as slice layer, MB layer, and block layer. The first three layers begin with their corresponding headers. The sequence parameter set and picture parameter set (SPS/PPS) define the coding tools and sequence information such as profile, level, frame size, frame number, etc. The slice header represents the slice information such as slice mode and initial quantization parameter, while MB header represents the MB information such as block types, predicted modes, and motion vectors (MV's). After MB header, transformed coefficients of each MB are codded by CAVLC. In CAVLC, each category of symbols has several context-based adaptive VLC

Table 1. Symbol rates of Foreman sequence in CIF 30fps video format under different quantization parameters.

QP	Symbol Rate (Symbol Count / Sec)					
	SPS	PPS	Slice Header	MB Header	Luma coeff.	Chroma coeff.
10	1.8	1.5	270.4	240918	1889623.6	364628.4
15	1.8	1.5	270.4	221623	1014633.6	166223.6
20	1.8	1.5	270.4	182482	458573.6	59964
25	1.8	1.5	270.4	137481	195131.2	25455.6
30	1.8	1.5	270.4	93405.2	75994.8	10519.2
35	1.8	1.5	270.4	61870.8	31268	4050.4
40	1.8	1.5	270.4	39392.4	12354	1833.2
45	1.8	1.5	270.4	21510.4	3783.6	1044.8
Foreman (CIF, 30fps)						

tables, and the selection of these tables depends on the statistic of block's content and previous transmitted symbols to match the most probable statistics.

2.2. Profiling

We use iprof, a software instruction level analyzer, to make profiling at a processor-based platform The profiling condition considers CIF format, 30fps, quantization parameter as 20,and CAVLC for residue coding. Entropy coder requires 115.4 million instructions/s (MIPS) of computation complexity. It will exhaust the system RISC resource in platform-based VLSI system for software implementation. Besides, the entropy coding is a kind of bit-level operation and cannot be efficiently handled by general purpose processors (GPP's). Therefore, dedicated hardware of entropy coding is a must.

Table 1 shows the symbol rate of Foreman sequence in CIF format at 30fps. The symbol rates of the SPS, PPS, and slice header in bitstream (Fig.1) are very low. Besides, these symbols are almost fixed for the specified profile and level. The symbol rates of MB header and coefficients are much higher. There are 396×30 macroblocks per second and up to million symbols/s for CIF format. As for HW/SW partition, the SPS, PPS, and slice header will be generated by system processor. The remaining part including scanning, coding, and bitstream packing will be mapped into hardware as an MB engine.

3. ARCHITECTURE DESIGN

In this section, we will introduce the architecture design for entropy coding engine in H.264 baseline profile. Figure 2 shows the block diagram. The bitstream of SPS, PPS, and slice header are generated by system processor because their symbol rates of them is very low. The information about MB header and transform quantized residues are assumed to be inputted from prediction and reconstruction engine of the encoder. The final bitstream will be outputted to the system buffer in network abstraction layer (NAL) format. This entropy coding engine is divided into three stages. At symbol level, the Exp-Golomb code unit and CAVLC unit take MB information and transformed coefficients in proper order, and translate them into codewords by table look-up. At codeword level, the bitstream packer concatenates the generated codewords. This compressed result is then stored in bitstream buffer, and then outputted via bus interface. The architecture design of each stage will be described in the following subsections.

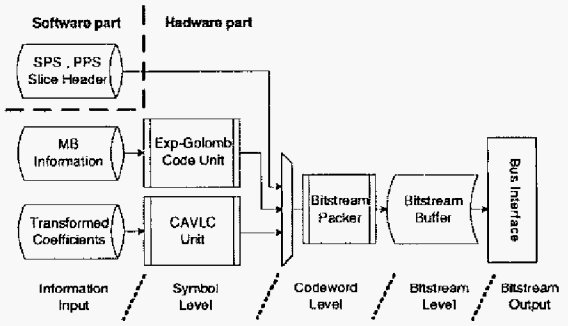


Fig. 2. Block diagram of entropy coding engine in H.264 encoder.

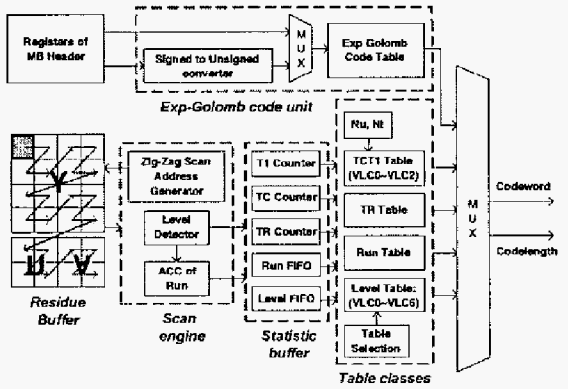


Fig. 3. Basic architecture of Exp-Golomb Code / CAVLC Unit.

3.1. Exp-Golomb Code / CAVLC Unit

3.1.1. Basic Architecture

Figure 3 shows the basic architecture that has one degree of parallelism in terms of reading one coefficient from residue buffer or coding one symbol. When one MB starts to be processed, the MB information is translated into codeword by using Exp-Golomb code table. Afterwards, the transformed coefficients are coded by using CAVLC. The macroblock (MB) is be divided into several 4×4 blocks, and those 4×4 blocks are processed one after another in double zigzag order. Each 4×4 block is processed through two phases, scan phase and coding phase. In scan phase, the transformed coefficients are read from residue buffer in reverse zig-zag order. Then, the run-level symbols and required statistics are extracted by level detector and stored in statistic buffer. In coding phase, the symbols are translated into codewords by usage of the corresponding class of tables. The selection of VLC tables within a class is according to the related statistic and the previous coded symbol.

3.1.2. Dual-Buffer Architecture with Block Pipeline Scheme

Compared with the traditional VLC tables that use fixed static probability distribution model, CAVLC utilizes the inter-symbol correlation to further reduce the statistics redundancy. Not until the scanning of a 4×4-block is finished can we know the statis-

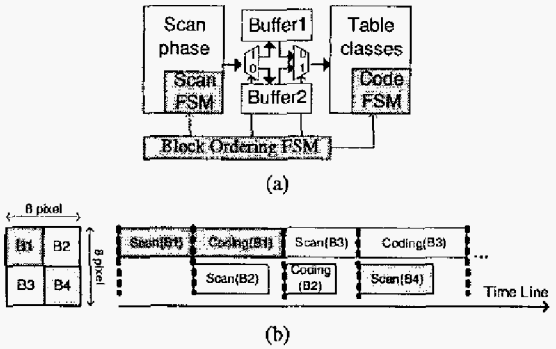


Fig. 4. (a) Dual-buffer architecture of bitstream generator. (b) Block pipeline flow of proposed dual-buffer architecture.

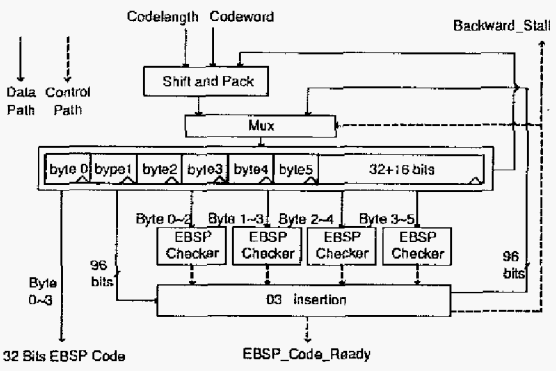


Fig. 5. Architecture of 96-bits packer that supports translation from RBSP to EBSP.

tic of total coefficients, trailing ones, and total zeros that are the adaptive factors of most tables. Thus, the scan and coding phase of each block must be processed in sequential order. Though this structure of basic architecture is similar to those used for JPEG and MPEG-1/2/4, its utilization and throughput are only half.

To deal with this problem, we propose an advanced dual-buffer architecture and the corresponding block pipeline scheme as shown in Fig. 4. There is a pair of pingpong mode statistic buffers (Fig. 4(a)). After the scan phase of the first 4×4-block, the run-level symbols and statistics are stored in the first buffer, and the coding phase is processed. At the same time the scan phase of the second 4×4-block is processed in parallel by usage of the second buffer. As shown in Fig. 4(b), by switching the pingpong mode buffers, scanning and coding of the 4×4-blocks within a macroblock can be processed simultaneously with the interleaved matter. In this way, both the throughput and utilization are doubled .

3.1.3. Zero Skipping by CBP Look-Ahead

The symbol count of transformed coefficients decreases with the increasing of quantization parameter because of the larger percentage of zero coefficients. In this situation, the throughput of dual buffer architecture will be confined by the scan phase. To further improve our design, a zero skipping technique is applied. When the coefficients within an 8×8-block are all zero, the 4×4-blocks inside are unnecessary to be coded in this situation. We can save the operation cycles and power by skipping the redundant scan process including the memory access toward residual buffer. In this method, the coded block pattern (CBP) in macroblock header is used for the skipping decision. This scheme is useful for the well-predicted MB or in low bitrate situation.

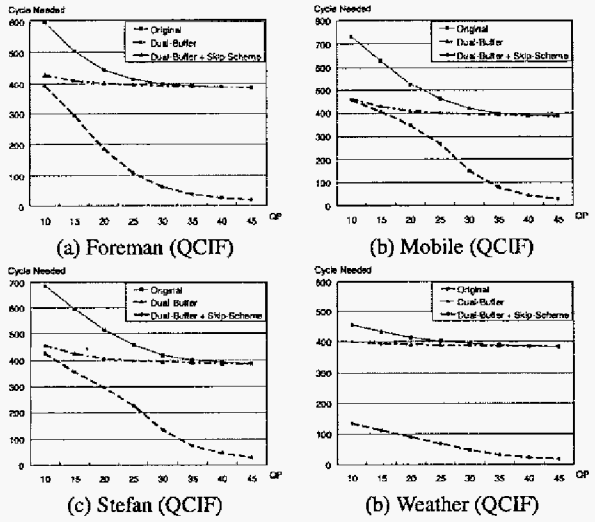
3.2. Bitstream Packer

H.264/AVC defines a byte-stream format to transmit a sequence as the ordered stream of bytes in network abstraction layer (NAL). In this system, the usage of emulation prevention bytes guarantees that start code prefix can be uniquely identified. The emulation prevention on byte basis requires the format translation from raw byte sequence payloads (RBSPs) to encapsulates byte sequence payload (EBSP). When successive three bytes of "0x000000", "0x000001", "0x000002", or "0x000003" is found in original bitstream, the byte, "03", is inserted. Large amount of memory access and the byte-level computation are required and will consume significant system resources. This overhead can be simply alleviated by integrating the RBSP-EBSP conversion into the bitstream packer.

Unlike using traditional 32-bits bitstream packer[6], a 96-bits bitstream packer as Fig. 5 is designed. This 96-bits bitstream packer can concatenate variable-length codewords together and segmenting them into 32-bits word like before. It can also checks EBSP-compatible format in parallel and inserts dummy bytes in serial if needed. The shift-and-pack concatenates the inputted codeword and the bitstream left in register array. When data in register array are large than 48 bits, four EBSP checkers operate simultaneously. Signal of EBSP_Code_Ready is high if the checkers are all passed. At the same time, the first 32-bits word is compatible with EBSP format and will be outputted to the bitstream buffer. Otherwise, the dummy byte insertion is performed in serial, and the circuit of coding core must be paused via feedback stall signal. By simulation of variable sequences, the occurrence probability of dummy byte insertion is very small (less than 0.001%). Therefore, the backward stall seldom occurs, and the throughput of the coding core is still very high.

3.3. Analysis of Bitstream Buffer

The entropy coding engine acts as an output interface of the encoder. After bitstream packer, the concatenated bitstream in EBSP-format will be stored temporarily and then transmit to system buffer via system bus. A bitstream buffer is used to favor the burst transmission from core engine onto the system bus. If the bitstream buffer is full while the coding procedure of one MB is not finished, the entropy coding engine must be halted immediately, and the system bus is requested to handle this exceptional condition. If the size of bitstream buffer is too small, such exceptional condition occurs too frequently, which will decrease the utilization of both the system bus and coding engine. To decide the minimum size of the bitstream buffer and to guarantee that buffer fullness seldom takes place, we collect statistics of variable sequences and analyze the bits usage of each MB. Almost all MB's (over 99.9%), either in I frame or P Frame, has less than 2K-bits size of bitstream. Therefore, the bitstream buffer with 2K-bits storage capacity is enough.



(a) Foreman (QCIF) (b) Mobile (QCIF)

(c) Stefan (QCIF) (b) Weather (QCIF)

Fig. 6. The statistics for symbol count per frame of four difference sequences in QCIF format : Foreman stand for general sequence with media motions. Mobile has complex textures. Weather is the static sequence while Stefan has large motions.

4. SIMULATION AND IMPLEMENTATION

4.1. Simulation Result

Figure 6 shows the cycle count requirements of three entropy coding engines : basic architecture, dual-buffer architecture with block pipeline scheme, and the former with zero skip technique. Compared with basic architecture, dual-buffer architecture with block pipeline scheme can process the scan phase and coding phase of two neighboring 4×4-blocks in parallel and thus enhance the hardware utilization. It can almost half the processing cycles when the quantized residue energy is large at high bit-rate situation. However, when prediction is fine or at low bit-rate situation, most residuals are zero, and the scan phase dominates the processing cycles. The zero skipping technique according to CBP can further improve the design by passing over the redundant scan process in this situation.

4.2. Implementation Result

The proposed entropy coding engine with dual-block-pipelined architecture, zero skipping technique, NAL encapsulation bitstream packer, and 2K-bits bitstream buffer is implemented by using cell-based design flow and 0.18um UMC/Artisan cell library. Table 2 shows the gate count profile. To achieve full hardware utilization by dual buffer architecture, two block statistic buffers are required. Additional 6000 gates are needed. Table 3 shows the local memory requirement. Three types of memories are required. The coefficient memory and bitstream memory are used as input and output buffer for system consideration. The upper 4×4-block total coefficient memory is used to story the 4×4-block total coefficients required by following blocks. The entropy coding engine requires about 500 cycles for high-quality application (QP=10-20) and about 200 cycles for low-bitrate application (QP=30-45).

Table 2. Gate count profile of entropy coding engine.

Item		Gate Count
VLC Tables	Exp-Golomb Code	973
	Total Coefficient and Trailing ones	864
	Level	1012
	Total Run	646
	Run After Level	263
Content Statistic Buffer		12283
96-bits bitstream Packer		4976
Control		2567
Total		23584

Table 3. On-chip memory requirement of entropy coding engine.

Memory Specification			
Address x Word	Type	Number	content
192x32	dual	1	Coefficients
160x20	single	1	Upper block4x4 total coefficient
64x32	single	1	Bitstream in EBSP format

5. CONCLUSION

In this paper, an entropy coding design for H.264/AVC baseline profile encoder is described. We consider the feature of context adaptation and propose the dual-buffer architecture. The scan and coding procedure are interleavingly processed in block pipeline method, which enhances the hardware utilization and processing throughput. The zero skipping technique can further skip the redundant computations in low bit-rate situation. Besides, for system consideration, the RBSP-EBSP conversion is integrated in bitstream packer, and the bitstream buffer is also used to reduce the interaction with the system. It can encode 1920×1088 30fps videos in realtime with 23.6K logic gates at 100MHz.

6. REFERENCES

[1] Joint Video Team, *Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification*, ITU-T Rec. H.264 and ISO/IEC 14496-10 AVC, May 2003.

[2] *Joint Video Team Reference Software JM8.5*, http://bs.hhi.de/ suehring/tml/download/, 2004.

[3] T. Wiegand, H. Schwarz, A. Joch, F. Kossentini, and G. J. Sullivan, "Rate-constrained coder control and comparison of video coding standards," *IEEE Transactions on CSVT*, vol. 13, no. 7, pp. 688–703, July 2003.

[4] D. Marpe, H. Schwarz, and T. Wiegand, "Context-based adaptive binary arithmetic coding in the H.264/AVC video compression standard," *IEEE Transactions on CSVT*, vol. 13, no. 7, pp. 620–644, July 2003.

[5] Iain E G Richardson, "H.264 / MPEG-4 Part 10 : Variable Length Coding," Mar. 1999.

[6] S. M. Lei and M. T. Sun, "An entropy coding system for digital HDTV applications," *IEEE Transactions on CSVT*, vol. 1, no. 1, pp. 147–155, Mar. 1991.