

A HW CABAC Encoder with Efficient Context Access Scheme for H.264/AVC

X.H. Tian, T.M. Le *SM-IEEE*, X. Jiang, Y. Lian *SM-IEEE*

Department of Electrical and Computer Engineering
National University of Singapore
Contact: elelmt@nus.edu.sg

Abstract—In this paper, we propose a hardware Context-based Binary Arithmetic Coder (CABAC) targeting the main profile of H.264/AVC standard. The encoder fully supports different coding modes including RDO coding. An efficient memory access scheme is proposed and shown to reduce context memory access rate, context memory size, and RDO context state backup and restore operation delay. Coding throughput of 1 bin/cycle is achieved by pipelined structure. The encoder is physical implemented with 362MHz clock frequency and power reduction technique is also utilized.

I. INTRODUCTION

THE H.264/Advance Video Coding (AVC) video compression standard [1], [2] has been introduced to improve coding efficiency and network friendliness. The standard suggests normative implementations of 2 types of entropy coding techniques: Context-based Adaptive Variable Length Coding (CAVLC) [3] for the baseline and extended profiles, and Context-based Adaptive Binary Arithmetic Coding (CABAC) [4] for the main and higher profiles. CABAC achieves higher coding efficiency than CAVLC with 9%-14% bit rate saving [4] but at the expense of increased complexity in the entropy coder. Rate-Distortion Optimization (RDO) is another technique adopted in H.264/AVC that selects the best coding mode for each macroblock (MB) with minimum RD cost. The combination of RDO and CABAC achieves bit rate saving of around 20%, but the computation cost is significantly increased, occupying around 1/3 of encoder instructions [5]. Therefore, it is reasonable to design a HW IP block to accelerate CABAC and related RDO operations.

Several HW CABAC encoder designs have been proposed. Pipelined structures have been used in [6] and [7], but because data dependence of context access stage and arithmetic coding stage is not removed, the designs cannot achieve coding throughput of 1 bin/cycle. Cache buffer and low power techniques are adopted in [8] to reduce memory access rate and dynamic power. RDO coding mode is not supported in [6], [7], and [8]. Osorio *et al.* [9] allocates 2 context memory blocks to reduce context state restore operation in RDO coding, but the scheme cannot support P8x8 inter picture RDO coding modes. Three large FIFO buffers are allocated in Nunez-Yanez *et al.*'s design [5] to support different RDO coding modes, however, the context access scheme is not efficient, and both context memory size and context state backup and restore (B&R) delay are high. In this paper, we propose a high throughput HW CABAC encoder that fully supports different coding modes including RDO. A

context access scheme is proposed to reduce memory size, context RAM access rate, and context state B&R operation delay. In the following sections, HW structure of the CABAC encoder is introduced in Section II, and context access scheme is discussed in Section III. Circuit implementation results and conclusion are given in Sections IV and V.

II. PROPOSED HW ENCODER ARCHITECTURE

CABAC encoder specified in the H.264/AVC standard includes 3 function blocks: binarizer, context modeler, and binary arithmetic coder (BAC). Input syntax element (SE) is binarized into bin string in the binarizer. A context modeler selects the context (probability) model of each regular bin in the string. The BAC encodes regular bins and bypass bins in two different coding engines and outputs encoded bit stream.

A. Top-level HW CABAC encoder architecture

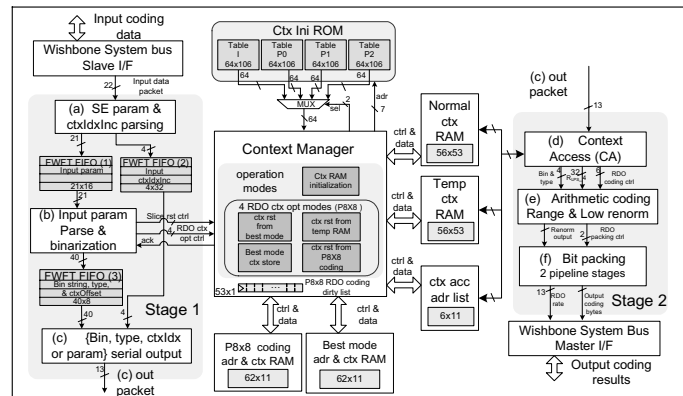


Figure 1. Block diagram of the HW CABAC encoder.

Top-level architecture of the proposed HW CABAC encoder is shown in Fig. 1. The encoder consists of two coding stages and a Context Manager (CM) block. Binarization is implemented in units (a), (b), and (c) of Stage 1. Input data received from the slave interface of the Wishbone system bus [10] is separated into context index increment (ctxIdxInc) and SE parameters in unit (a); SEs are binarized into bin strings and controls operation of context manager in unit (b); context index (ctxIdx) of regular bin is calculated and data pairs of {bin, ctxIdx} of each bin string are output in series in unit (c). Units (a), (b) and (c) work simultaneously with outputs buffered in 3 FIFOs (0.78 Kbits). FIFOs are of FWFT-typed (first word fall through) buffers which can remove 1 wait cycle of FIFO read operation. SE

binarization is fully implemented in HW compared to other designs [5-9] to reduce data transfer rate of bin string on system bus. Stage 2 implements context modeler in unit (d) and BAC in units (e) and (f). Unit (d), (e), and (f) can fetch one context model, encoder one bin, and pack output coded bits, respectively, in one cycle. The 3 units constitute a 3-stage pipeline, and the coding throughput at Stage 2 is 1 bin/cycle. The encoded results of Stage 2 are output through the master interface of the Wishbone system bus. The design of master and slave interfaces of the Wishbone bus enhances the portability and reusability of the CABAC encoder. Constant address burst transfer is supported in the slave interface with input speed of 1 packet per cycle. Compared to [5-9], the context model initialization at the beginning of each slice is first implemented in HW to reduce SW computation cost and data transfer delay. As shown in Fig. 1, CM fetches context initialization parameters from ROM tables and concurrently calculates 4 context models per cycle through a 5-stage pipeline to accelerate initialization procedure. More details of coding procedure including binarization of SE and HW oriented fast algorithm of BAC are provided in our previous work [11]. In comparison to [11], this paper focuses on a more efficient context access scheme, physical layout of the encoder design, and gate level power reduction scheme.

B. Binary Arithmetic Coder (BAC)

In Fig. 1, unit (e) encodes each bin by updating Range and Low values of the coding interval. Three processing routes are designed based on the bin types. Regular bin coding route selects one of the 4 Range_{LPS} values received from unit (d) and processes MPS and LPS bin in two paths. For MPS bin, only 0 or 1 bit renormalization is needed after updating Range and Low; while for LPS bin, 5-bit leading zero detection (LZD) circuit decides renormalization length. Separation of MPS and LPS processing simplifies coding procedure and reduces critical path length. Bypass bin and End of Slice (EOS) flag are processed in the other 2 routes. Range and Low are updated and renormalized in the same cycle.

Unit (f) is the bit packing and output stage of CABAC encoder. It is designed in 2 pipeline stages to cut the critical path. Least significant zero (LSZ) checking circuit is utilized to parse output bin strings and outstanding (OS) bin strings. Output string and confirmed OS bin string are packed in the output buffer. Full output bytes from the buffer and the length of confirmed stuffing bytes are output. This design can tolerate maximum length of 63 OS bits before the OS bit value is confirmed. Zero bit stuffing circuit is used in EOS packing, which can output 3 bytes excluding stuffing bytes. During RDO coding, bit rate of each mode is accumulated and output.

III. EFFICIENT CONTEXT ACCESS SCHEME

CABAC coding performance depends highly on the accuracy of context models. The context model is fetched from RAM, updated, and written back during the coding of each regular bin. High frequency of context RAM access significantly increases encoder's power consumption [8]. During RDO coding, large number of context models are required to be backed up or restored, which causes long operational delay, and requires large backup memory

resources. To solve the problems above, an efficient context access scheme is proposed in this design.

A. Context line access and buffering

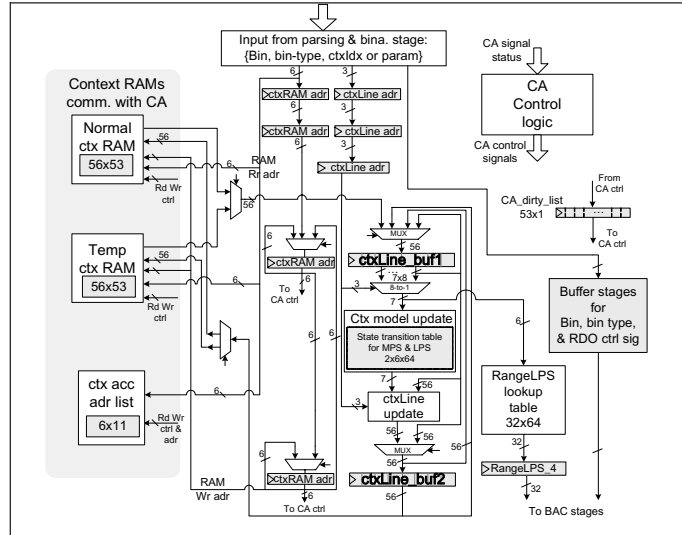


Figure 2. Block diagram of the context line access and buffering of unit (d).

Fig. 2 shows context access scheme in unit (d) of Stage 2. Context RAM is accessed by a context line instead of single context model. One context line of RAM contains 8 context models. Input ctxIdx of regular bin is partitioned into 6-bit context RAM address and 3-bit context line address. Context RAM address is buffered in 4 stages and used to locate one context line from RAM. Context line address is buffered in 3 stages to select 1 of the 8 context models in each context line for regular bin. 2 context line buffers are allocated in unit (d) and 16 different context models are buffered during coding. Selected context model from ctxLine_buf1 is used to fetch 4 possible Range_{LPS} values, and it is updated by a lookup table. Because context models of same SE type are located closely in context RAM, RAM access rate reduction is significant with context line buffering, especially for residual SE coding.

B. Context Memory Reallocation

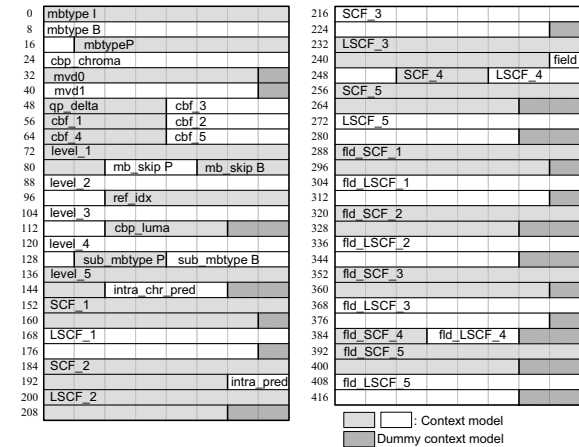


Figure 3. Context model reallocation map in context RAM

The original context allocation specified in the H.264/AVC standard is not optimized for the context line access and buffering scheme. In order to further reduce memory access rate, the context models in the context RAM are reallocated, as shown in Fig. 3. The change in color in the context line indicates changes of SE type of context models. The principle is to allocate the context models of one SE type in the same context line to avoid additional RAM read. For the SE type with over 8 context models, such as SCF, LSCF, and residual coefficient level, 2 context lines are allocated. 53 lines of 424 context models are allocated in the context RAM with only 6.8% dummy context models inserted.

C. RDO Context Operation Support

During RDO coding, large operation delay is introduced by context model backup and restore during RDO coding mode change. In our design, a Temp context RAM (56x53), 3 additional small RAM blocks (context address access list (6x11), best mode address & context RAM (62x11), P8x8 coding address & context RAM (62x11)), and 2 dirty lists (Fig. 1) are allocated, except the Normal context RAM (56x53), to support all RDO coding modes. During non-P8x8 RDO coding, Normal RAM and Temp RAM store the original and updated context models respectively, and Normal RAM is unchanged. In P8x8 RDO coding, the 3 small RAM blocks store the intermediate context state including (1) addresses of the accessed context lines in one RDO mode, (2) addresses and context lines of best P8x8 coding mode for updating the Normal RAM, (3) addresses and context lines of original context state for recovering the Normal RAM after P8x8 RDO coding. 4 pipelines are designed in CM to support 4 types of context state B&R modes during P8x8 coding with context copy speed of 1 context line per cycle. Details of P8x8 context state B&R operations of the 4 pipelines are shown in Fig. 4. Buffering stages of context line and RAM address are inserted to implement the pipeline operations.

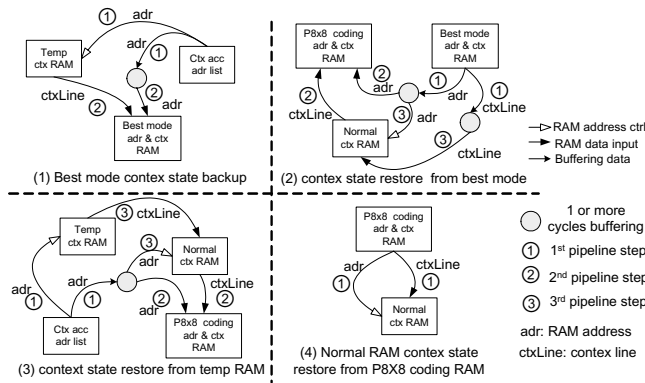


Figure 4. Four pipelined P8x8 context state backup and restore modes

D. Comparisons of Context Access Performances

Context access performance is obtained from coding of QCIF IPP Foreman test sequence for both RDO and non-RDO coding. With the context line access and buffering scheme, both context RAM read and write frequencies are reduced comparing to [5], which is the only design that fully support RDO coding modes. Fig. 5 (a) and (b) show that, with video

PSNR in 33–49dB, before context RAM reallocation, the average RAM read and write frequency ratio (this design to [5]) is 30.4% (both) in non-RDO coding, and 26.3% (read), 19.6% (write) in RDO coding; and after context RAM reallocation, the ratio reduces to 24.8% (both) in non-RDO coding, and 21.4% (read), 14.7% (write) in RDO coding. Context RAM access reduction is more significant in low QP coding. Context line buffers work similar to small cache, with no delay cost of data fetch of cache miss, compared to [8-9]. Simulation results verify the significant benefits of context line access and buffering scheme and context memory reallocation to the context RAM access frequency reduction.

Compared to [5], the operations of context model restore are removed in non-P8x8 RDO coding. In P8x8 RDO coding, B&R operations are faster than [5] because of the context line access scheme and pipelined context copy operation in CM. The average cycles of P8x8 context state B&R are reduced to 15.6% and 14.6% of [5] before and after context memory reallocation, and benefit is more significant in high PSNR range (Fig. 5 (c)). In [5], 46Kbits backup context memory is allocated to support RDO coding. In comparison, our design only allocates 7.37 Kbits context RAM to fully support RDO coding, which reduces to 16.0 % of [5].

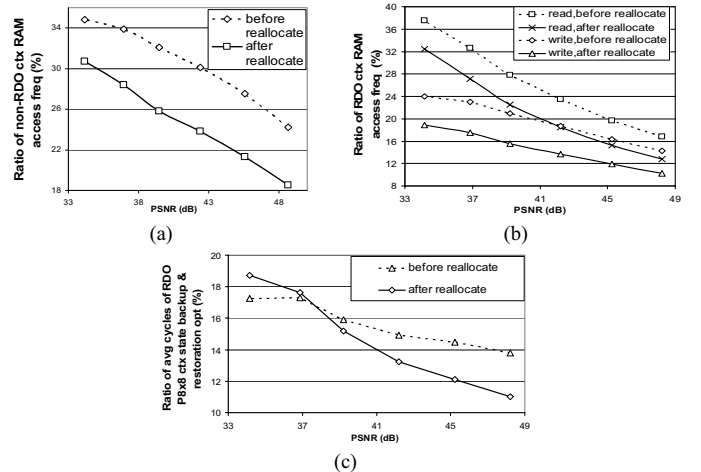


Figure 5. Ratio (this design to [5]) of context RAM access frequency and RDO P8x8 context state B&R operation delay

IV. CIRCUIT SYNTHESIS AND LAYOUT RESULTS

The CABAC encoder is fully verified at RTL level and gate level using the test vectors and reference coding results generated from the H.264/AVC reference software JM 9.3. More details of RTL level and gate level simulation and verification scheme adopted here are described in our previous work [12]. The encoder is synthesized and implemented with main stream 0.13μm process technology. The layout design was completed and the chip diagram is shown in Fig. 6. The core size is 0.77mm² and 99 I/O pins are assigned. Post-layout clock frequency is 362MHz. As far as we are aware of, this design is the only HW CABAC encoder with post-layout results. Both synthesis and post-layout results of our design are given, which are more reliable than the synthesis-only result. The detailed performance metrics including clock frequency and circuit area are listed in Table I.

Table I illustrates that this design achieves higher clock frequency of CABAC encoder compared to the FPGA and CMOS synthesis results of reference designs. Considering that coding throughputs of [6] and [7] are lower than 1 bin/cycle, the coding speed of this design is faster than most other designs except [9], which has throughput of 1.9~2.3 bin/cycle. However, comparing to all the other designs, our design fully supports RDO coding modes with efficient context access scheme that reduce context RAM size and operation delay. The circuit area of different designs is also listed in Table I. The area is not comparable, because most of the designs only implemented context access and BAC, which is a part of Stage 2 of our design. Input parameter buffering and SE binarization of Stage 1, context model initialization (utilize 27.1Kbit ROM table) and pipelined P8x8 RDO context state backup and restore operation (utilize 8.15Kbit RAM blocks) of context manager are only completely implemented in this paper with larger area occupation.

Considering the power consumption of this design, RTL level power reduction technique is also adopted to reduce dynamic power consumption of memory blocks. Clock gating is inserted to the 4 large context RAMs and 3 FIFO buffers to reduce internal power. In order to compare power consumption with available reference data, gate level SAIF (Switching Activity Interchange Format) file is generated from QCIF Foreman sequence simulation in both non-RDO and RDO coding modes at 200MHz clock frequency. SAIF file is imported to state-of-the-art power analysis tool.. Average power consumption calculated in the PSNR range of 33~49 dB is listed in Table II. Considering 25% power reduction with process technology improvement (using [13] as a guideline), power data of [6] and [8] in 0.13 μ m process are scaled to 75% of original results. Power consumption of this design is still relatively lower, as shown in Table II.

TABLE I. PERFORMANCE OF CABAC ENCODING CIRCUITS

CABAC design	Process technology	Clock frequency	Circuit Area	
[5]	XilinxVirtex4 FPGA	130MHz	Slice 1158	
[6]	0.18 μ m synthesis	263MHz	0.423 mm ²	
[7]	0.35 μ m synthesis	150MHz	4.57K gate 4.1Kb RAM	
[8]	0.18 μ m synthesis	200MHz	0.31mm ²	
[9]	0.35 μ m synthesis	186MHz	19.4K gate excluding memory resource	
This design	0.13 μ m synthesis	568MHz	0.58 mm ²	Area including: 8.15Kb RAM 27.1Kb ROM
	0.13 μ m post-layout	362MHz	0.77 mm ²	
	XilinxVirtex4 FPGA	145MHz	Slice 2559	

TABLE II. GATE-LEVEL CABAC POWER CONSUMPTION COMPARISON

Design	Process (um)	Total power	Power of Stage 2 and Normal context RAM access	
			Actual power	Power scaled to 0.13 μ m process
[6]	0.18	N/A	48mW	36mW
[8]	0.18	N/A	20.7mW	15.5mW
This Design	Non-RDO	26.5mW	12.2 mW	12.2mW
	RDO	22.2mW	9.4mW	9.4mW

V. CONCLUSION

In this paper, a HW CABAC encoder targeting the main profile of the H.264/AVC standard is proposed. Compared to the other designs [5-9], this design fully supports RDO coding, context model initialization, and binarization in HW. HW support of RDO coding is necessary to significantly reduce power consumption and context access delay of host processor. An efficient context access scheme is proposed including techniques of context line access and buffering, context memory reallocation, and pipelined context model backup and restore operation in P8x8 RDO coding. Context memory size is reduced to 16.0% of [5]. In non-RDO coding, context RAM access frequency is reduced to 24.8% of [5]. In RDO coding, average context RAM read, write frequencies, and operation delay of P8x8 context state backup and restore have been reduced to 21.4%, 14.7%, and 14.6% of [5].

Our design is verified, synthesized, and implemented at the GDS-II stage. The coding speed is higher than most other designs, and post-layout speed is suitable for the video applications of real time CIF coding in full RDO mode and HDTV coding in non-RDO mode. Power reduction technique is also applied to the memory blocks, and the power consumption of our design is lower than [6] and [8] with consideration of process technology difference.

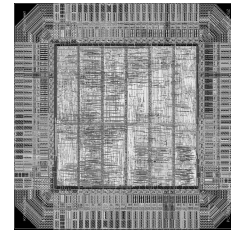


Figure 6. Chip layout of CABAC

REFERENCES

- [1] "Draft ITU-T Recommendation H.264 and Draft ISO/IEC 14496-10 AVC," in Joint Video Team of ISO/IEC JTC1/SC29/WG11 & ITU-T SG16/Q.6 Doc. JVT-G050, T. Wieg, Ed., Pattaya, Thailand, Mar. 2003.
- [2] T. Wiegand, G.J. Sullivan, *et al.*, "Overview of the H.264/AVC Video Coding Standard," *IEEE Trans. CSVT*, vol. 13, no. 7, pp. 560-576, Jul. 2003.
- [3] Gisle Bjontegaard and Karl Lillfeldt, "Context-adaptive VLC (CAVLC) coding of coefficients", JVT-C028, Fairfax, Virginia, USA, 6-10 May 2002.
- [4] D. Marpe, H. Schwarz, and T. Wiegand, "Context-based adaptive binary arithmetic coding in the H.264/AVC video compression standard," *IEEE Trans. CSVT*, vol. 13, no. 7, pp. 620-636, Jul. 2003.
- [5] Y.L. Nunez-Yanez, V.A. Chouliaras, *et al.*, "Hardware-assisted rate distortion optimization with embedded CABAC accelerator for the H.264 advanced video codec", *IEEE Trans. CE*, vol. 52, no. 2, pp. 590-597, May 2006.
- [6] H. Shojania and S. Sudharsanan, "A high performance CABAC encoder", *The 3rd International IEEE-NEWCAS Conference*, pp. 315-318, Jun. 2005.
- [7] L. Li, Y. Song, T. Ikenaga, and S. Goto, "A CABAC encoding core with dynamic pipeline for H.264/AVC main profile", *APCCAS'2006*, pp 761-764, Dec.2006.
- [8] C.C. Kuo and S.F. Lei, "Design of a low power architecture for CABAC encoder in H.264", *APCCAS'2006*, pp 243-246, Dec.2006.
- [9] R.R. Osorio and J.D. Bruguera, "High-throughput architecture for H.264/AVC CABAC compression system", *IEEE Trans. CSVT*, vol. 16, no. 11, pp. 1376-1384, Nov. 2006.
- [10] WISHBONE System-on-a-Chip Interconnection Architecture for Portable IP Cores, Revision: B.3, Sep. 2002.
- [11] X.H. Tian, T.M. Le, *et al.*, "A CABAC Encoder Design of H.264/AVC with RDO Support," *IEEE RSP'2007*, pp. 167-173, 2007.
- [12] T.M. Le, X.H. Tian, *et al.*, "System-on-Chip Design Methodology for a Statistical Coder," *IEEE RSP'2006*, pp.82-88, 2006.
- [13] G.R. Cho and T. Chen, "On the impact of technology scaling on mixed PTL/static circuits", *IEEE ICCD'2002*, pp. 322-326, Sept. 2002