A High-speed H.264/AVC CABAC Decoder for 4K Video Utilizing Residual Data Accelerator

Kenji Watanabe
Synthesis Corporation
watanabe@synthesis.co.jp

Gen Fujita
Dept. Engineering Informatics,
Osaka Electro-Communication University

Toru Homemoto, Ryoji Hashimoto
Graduate School of Information
Science and Technology, Osaka University

Abstract— The implementation of a parallel decoder for CABAC (Context-based Adaptive Binary Arithmetic Coding), which is adopted in the H.264/AVC video coding standard, is extremely difficult due to inherent data dependency. Therefore, the CABAC decoder constitutes a bottleneck when decoding 1080 HD (1,920×1,080) or higher video sequences in real time. In this paper, we propose a VLSI (Very Large Scale Integration) architecture for the CABAC decoder that adopts a multi-bin decoding architecture in conjunction with techniques that improve the maximum clock frequency. The implementation results show that the proposed architecture achieves an average throughput of 1.48 bins per clock and a maximum clock frequency of 394 MHz, demonstrating that our architecture is capable of decoding 4K (4,096×2,048 @ 30 fps) video in real time.

I. Introduction

H.264/AVC [1] is a video coding standard jointly developed by ITU-T and ISO/IEC. H.264/AVC achieves higher coding efficiency compared with preceding video coding methods, such as MPEG-4 [2], due to a variety of enhanced coding techniques, such as variable block-size, quarter-pel accurate motion compensation, and context-adaptive entropy coding.

H.264/AVC adopts two entropy coding methods: CAVLC (Context-based Adaptive Variable Length Coding) and CABAC (Context-based Adaptive Binary Arithmetic Coding). CAVLC is a refinement of VLC, which has been used in numerous still image and video coding standards. CABAC is based on arithmetic coding, which can achieve superior coding efficiency compared with conventional Huffman coding. H.264/AVC high profile, adopted by the Blu-ray Disc Movie (BDMV) specification, requires decoding support for both entropy coding methods. Both of the entropy coding methods use neighbor block information, which results in an increase in the computational cost. Furthermore, the data dependency inherent to the algorithm, especially in CABAC, limits the possibility of parallel decoding, thus demanding sequential processing of the bitstream. CABAC decoding is performed by first converting the bitstream into a 'bin' string consisting of intermediate expressions represented by the binary digits '0' and '1.' Then, a de-binarization process is performed, in which the 'bin' string is interpreted by decoding the string into syntax elements. The throughput required for 'bin' decoding increases proportionally with the bit-rate of the bitstream. Hence, CABAC decoding can emerge as a bottleneck when decoding a high-bit-rate H.264 bitstream. Under this context, a high-speed CABAC decoder architecture is required for decoding 1080 HD (1,920×1,080 @ 30 fps) or higher resolution video in real time. Furthermore, high bin throughput per clock cycle would be desirable, especially for low-power products such as mobile devices.

A number of H.264/AVC CABAC decoder architectures have been proposed so far [3–8]. However, most of these architectures [3–5] require multiple cycles to generate each bin, thus demanding high clock frequencies to decode high-bit-rate video in real time.

According to [9], the maximum bit-rate of a 1080 HD H.264/AVC high profile level 4.1 bitstream is 62.5 Mbps, and under these conditions, the average number of bins in one picture is 2.69×10^6. However, depending on the picture type, the number of bins per picture fluctuates, even in the same bitstream. The number of bins in an I-Picture is 5.54×10^6, which is approximately two times larger than the average number of bins per frame. Hence, the architecture described in [3–5] does not satisfy this performance requirement. Furthermore, [6] is designed for H.264/AVC main profile level 4.0, and thus does not achieve sufficient performance to decode 4K (4,096×2,048 @ 30fps) video.

Ref. [7] uses a multi-bin architecture that is capable of generating 2 bins per clock cycle in certain conditions in conjunction with techniques to shorten the critical path. This architecture not only achieves real time decoding of 1080 HD video but also reaches a bin-decoding throughput that corresponds to 1.2 times that of H.264/AVC high profile level 5.1 intended for 4K video. However, the number of bins in each picture can fluctuate greatly, as described above, and hence, 1.2 times the average number of bins is not enough for decoding I-Pictures. Thus, either a higher efficiency or a higher clock frequency would be needed for decoding 4K video in real time. Similarly, the architecture described in [8], which supports H.264/AVC

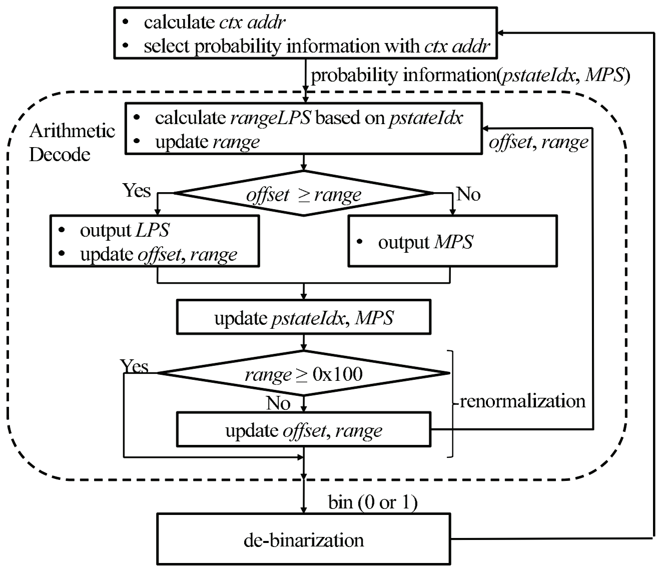

Fig. 1. Flowchart of CABAC Decoding

high profile level 5.1, is subject to the same problems as [8].

To solve this problem, we propose a VLSI (Very Large Scale Integration) architecture for the multi-bin CABAC decoder by improving our past CABAC decoder architecture [10], which takes advantage of statistical tendencies among syntax element types in general H.264/AVC bitstreams.

The proposed architecture achieves a 49% reduction in critical path. Moreover, our new architecture adopts a bin string buffer with the purpose of alleviating the peak throughput required when decoding I-Pictures. With the introduction of these techniques, our new architecture provides sufficient performance to decode 4K video in real time.

The remainder of this paper is organized as follows. In Section II, we present an overview of the CABAC decoding process. Then, Section III describes our proposed architecture in detail. After that, the implementation results and performance comparisons are shown in Section IV. Finally, we conclude our work in Section V.

II. Overview of CABAC Decoding

H.264/AVC specifies two entropy coding methods: CABAC and CAVLC. CABAC is known to be superior in compression efficiency and, hence, is widely use for encoding H.264/AVC video sequences, especially when higher image quality is required. However, parallel decoding of CABAC is greatly limited by data dependency. Hence, real-time decoding of high-bit-rate or high-resolution CABAC bitstreams presents a challenging task.

H.264/AVC bitstreams are composed of ‘syntax elements,’ each of which can be classified into two data types:

- header data
- residual data

The header data contains prediction data such as macroblock (MB) type and motion vector. The residual data contains prediction errors such as the differential of the coefficient. In typical high-bit-rate bitstreams, the majority of syntax elements consist of residual data. Therefore, it is important to decode the residual data efficiently.

Fig. 1 shows an overview of the CABAC decoder. It contains three steps, as follows:

1. **Probability Information Selection**: Probability information such as *pstateIdx* and *MPS* (Most Probable Symbol) is decided in this step. In CABAC, 460 tuples of probability information are used. When decoding a bin, one piece of probability information is selected from 460 tuples depending on *context*, which changes based on the syntax element type, bit position, neighbor block information, and so on. In the probability information selection, a variable *ctx addr* is used, which corresponds one-to-one to the probability information.

2. **Arithmetic Decoding**: This step generates a bin based on the probability information.

3. **De-binarization**: This step decodes a bin string and generates the value of a syntax element.

The arithmetic decoding process is shown inside the dashed line in Fig. 1. Using the probability information selected by *ctx addr* and two variables (*offset*, *range*), 1 bit of bin is output in this process. Then, the values of the selected probability information, *offset*, and *range* are updated. In addition, if the new *range* value is below 0x100, *offset* and *range* are updated again in the renormalization process.

There are data dependencies between these variables and the probability information, and the initialization of these variables in conjunction with the probability information is only executed at the beginning of each picture. Therefore, performing parallel decoding of CABAC within the same picture is extremely difficult. Moreover, the result of de-binarization affects the calculation of the next *ctx addr*, hence requiring architecture-level enhancements to achieve an efficient computation of this step.

III. Proposed Architecture

A block diagram of the proposed entropy decoder is shown in Fig. 2. The bitstream is input to the CAVLC controller or the CABAC arithmetic decoder via the variable-length shift register. The arithmetic decoder receives neighbor MB information from the MB memory controller and then calculates *ctx addr*. After that, the arithmetic decoder generates 1 bit of bin and updates the

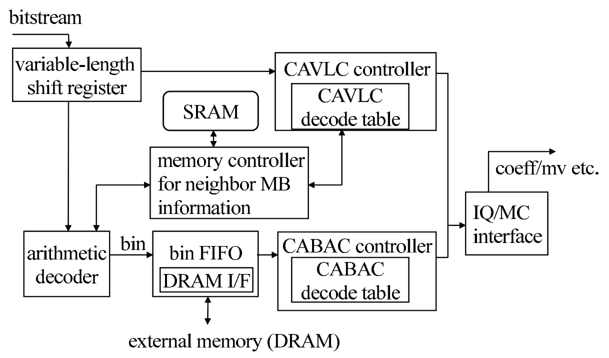

Fig. 2. Block Diagram of Entropy Decoder

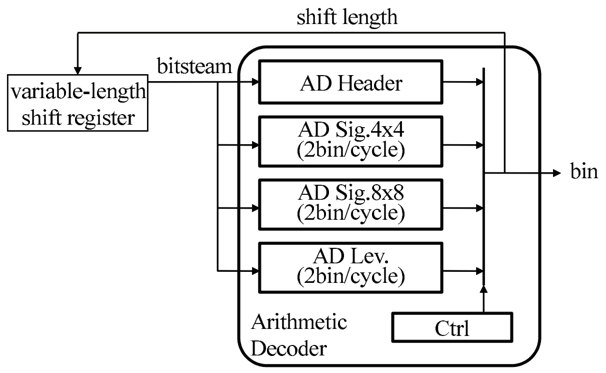

Fig. 3. Block Diagram of Proposed Arithmetic Decoder

neighbor MB information. The generated bin is buffered into external memory through a bin FIFO, which alleviates the peak number of bins required to be processed within a frame. The bin string is interpreted into syntax elements such as the motion vector or IT (Integer Transform) coefficients and then outputs to the IQ (Inverse Quantization) or MC (Motion Compensation) processing block.

In the following subsections, the arithmetic decoder and the bin FIFO are described in detail together with the main features of our proposed architecture.

A. *Arithmetic Decoder*

A block diagram of our arithmetic decoder is shown in Fig. 3. In a high-bit-rate bitstream, most of the syntax elements are residual data; hence, our new architecture makes use of the architecture described in [10], which can efficiently decode residual data using three accelerators (AD Sig.4x4, AD Sig.8x8, AD Lev.). The 'AD Header' decodes header data such as MB type. To achieve a throughput of over 1 bin per clock cycle, the bitstream buffer must be shifted in the same clock cycle as the arithmetic decoding. Consequently, the critical path includes a variable-length shift register. The proposed architecture reduces the critical path of the variable-length shift register in addition to those of the AD accelerators.

Apparently, [8] is capable of decoding all syntax elements in 2 bins/cycle. However, the complexity of syntax element transition conditions in the decoding header data is much higher than that of the residual data. Since the critical path increases according to the complexity of the transition conditions, we adopt an architecture in which the header data is decoded in 1 bin/cycle, in order to achieve higher operating frequency.

A block diagram of the AD Header used in [10] is shown in Fig. 4(a). Probability information is taken out from the register file with *ctx addr*. Then, the module generates bin and updates *offset* and *range*. After that, renormalization is performed if needed. This sequence is executed by the module that we call the 'arithmetic calculation module.' The number of words in the register file in the AD Header is comparatively large (85 words). Therefore, the probability information selection has a long latency.

The AD Header of the proposed architecture is shown in Fig. 5. The probability information is selected and stored at the previous clock time after the bin generation and *ctx addr* calculation, whereas it is selected at the beginning of the clock time in the conventional architecture. This module arrangement enables parallel operation of register file selection and variable-length shift, which results in a reduction in latency.

A block diagram of the AD Lev. in the conventional architecture [10] is shown in Fig. 4(b). The other accelerators (AD Sig.4x4 and AD Sig.8x8) use a similar architecture to AD Lev. In contrast to the AD Header, the number of words is relatively small, which contributes to latency reduction in the selector. However, in this architecture, two arithmetic calculation modules are sequentially connected, and therefore, these modules constitute the critical path and significantly contribute to the increase in latency.

Fig. 6 shows a block diagram of the AD Lev. used in the proposed architecture. The proposed architecture solves the problem above using two techniques, which can be described as follows:

1) **ctx addr Prediction**
   In contrast to the AD Header, 2 bins are generated in one clock cycle in the AD Lev. The probability information used to generate the second bin depends on the result of the first bin decoding. Therefore, the *ctx addr* used in the second stage is not determined at the previous clock time. Then, the *ctx addr* predictor selects the three pieces of probability information that can possibility be used at the next clock time and stores them into the registers. Hence, probability information can be selected with negligible latency in the next clock cycle.

2) **Parallel Decoding of Second Stage of Arithmetic Decoder**

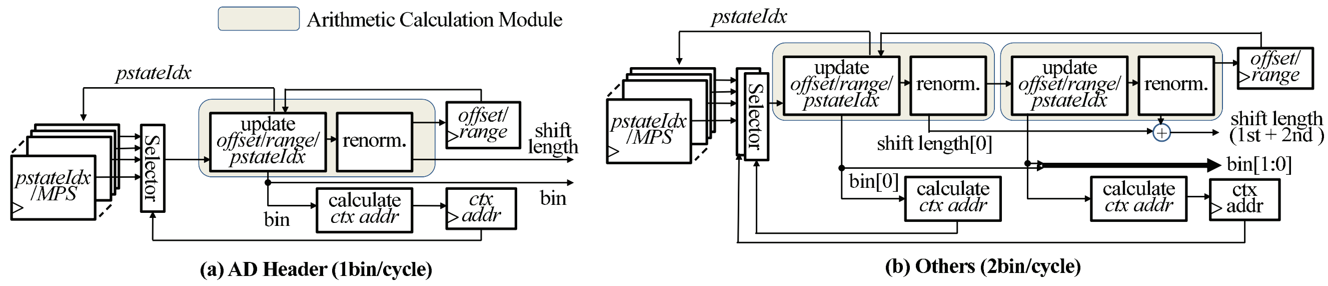

Fig. 4. Conventional Architecture of Arithmetic Decoder [10]

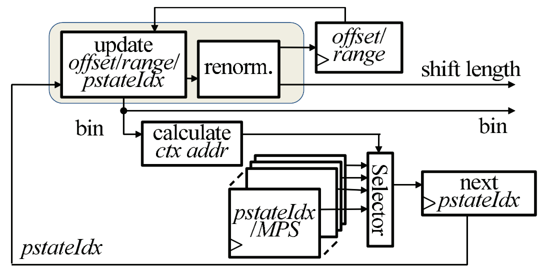

Fig. 5. Proposed Architecture (AD Header)

In the proposed architecture, the second stage of the arithmetic calculation module is doubled. Each module speculatively generates the second bin assuming the first bin value equals '0' and '1.' This technique enables the second stage modules to start calculating before finishing the first stage bin decoding.

Technique 2) is also used in [7, 8]. In the proposed architecture, by combining Techniques 1) and 2), the second stage can be started just after the *pstateIdx* decision, which further reduces the latency.

The bit shift amount is determined in the renormalization of the first and second stages of the arithmetic calculation modules. In our architecture, the variable-length shift register can operate independently between the first stage and the second stage, hence decreasing the total latency.

Fig. 7 compares the latencies of [10] and the proposed architecture. The new architecture operates the variable-length shift and *pstateIdx* selection in parallel. Also, the probability information selection of the second stage and the variable-length shift can be executed earlier. Consequently, the proposed architecture achieves an approximately 49% reduction in latency.

B. Bin FIFO

As described in Section I., the amount of calculations executed within the CABAC is proportional to the amount of bin strings. The amount of bin strings fluctuates depending on the picture type. Therefore, in conventional architectures, an extremely high clock frequency would be required to cover the theoretical peak performance. In our architecture, the generated bin string is buffered into external memory through the bin FIFO module, consequently reducing the required peak performance.

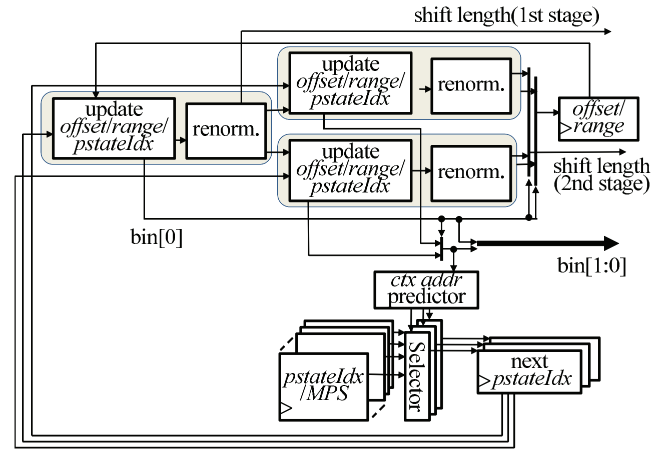

Fig. 6. Proposed Architecture (AD Sig.4x4/AD Sig.8x8/AD Lev.)

TABLE I
Average Number of Clock Cycles per MB

video sequence	without buffering	with buffering	reduction ratio [%]
crowd_run	728.8	282.3	61.3
ducks_take_off	749.0	320.9	57.2
in_to_tree	706.4	298.6	57.7
old_town_cross	737.3	315.0	57.3
park_joy	782.5	296.1	62.2

To verify the effect of the bin FIFO, we evaluated the worst case for the average number of cycle counts per MB in all pictures. The bit-rate of the 4K video sequences used for evaluation is set to 300 Mbps, which is the maximum bit-rate defined in the H.264/AVC high profile. We assumed that the bin string corresponding to 30 pictures is buffered. For the case "with buffering," the maximum

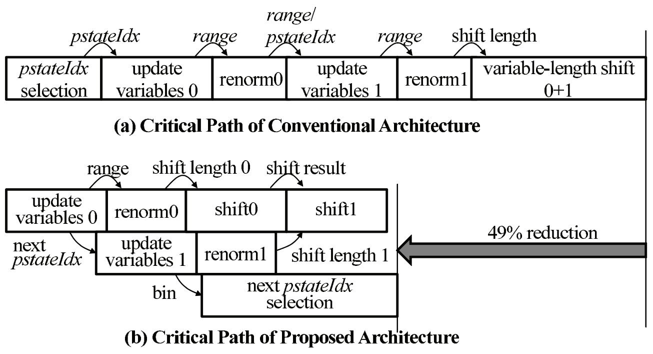

Fig. 7. Reduction of Critical Path

TABLE II
Implementation Results

Area (whole of the entropy decoder)	286,933 [gates]
Area (only the arithmetic decoder)	144,370 [gates]
Maximum Operation Frequency	394 [MHz]

cycle count of the moving average between consecutive 30 pictures is calculated. Table I shows the evaluation results, which indicate that the required performance is decreased by 57.2–62.2% using the bin FIFO.

IV. Implementation Results

We implemented the proposed architecture using Verilog HDL (Hardware Description Language) and applied logic synthesis with the *SYNOPSYS Design Compiler* using a 40 nm standard cell library. The results are shown in Table II.

According to Table I, assuming that the average clock cycle count per MB equals 320.9, the required clock frequency would be 316 MHz for decoding a 4K video that contains 983,040 MB/s. Therefore, the proposed architecture achieves real-time decoding of 4K video.

Table III compares our proposed architecture with conventional architectures. Compared with conventional architectures, the proposed architecture has a higher throughput of 1.48 bins per cycle. Hence, our architecture can decode video sequences using a lower clock frequency.

TABLE III
Comparison with Conventional Architectures

	[4]	[5]	[7]	[8]	Ours
Technology [nm]	180	180	130	90	40
Area [gates]	81,162	76,333	47,081	NA	144,370
cycles/MB	NA	396	326	NA	321
bin/cycle	0.254	0.86	1.08	1.95	1.48
Max. Freq. [MHz]	225	140	333	215	394

V. Conclusion

In this paper, we proposed a high-speed architecture for the H.264/AVC CABAC decoder that achieves real-time decoding of 4K video and a maximum clock frequency of 394 MHz. By using a *ctx addr* predictor and doubling the second stage of the bin decoder, the critical path of the multi-bin decoder has been reduced significantly, and real-time decoding of 4K video has been achieved.

The proposed architecture is able to decode a video sequence at a lower clock frequency than conventional architectures. Hence, our architecture is also suitable for mobile devices.

Acknowledgements

We would like to thank Roberto Y. Omaki for productive discussions on the topics of this paper. We would also like to thank our colleagues for implementing the H.264/AVC CODEC.

References

[1] ITU-T Rec. H.264/ISO/IEC 11496-10: "Advanced video coding," International Standard, Oct. 2004.

[2] ISO/IEC 14496-2: "Coding of audio-visual objects : visual," International Standard, Dec. 1999.

[3] C. H. Kim and I. C. Park, "High speed decoding of context-based adaptive binary arithmetic codes using most probable symbol prediction," *IEEE Symp. Circuits and Systems (ISCAS) 2006*, pp.1709-1710, May 2006.

[4] Y. Yi and I. C. Park, "High-speed H.264/AVC CABAC decoding," *IEEE Trans. Circuits and Systems for Video Technology*, vol.17, no. 4, pp.490-494, Apr. 2007.

[5] Y. C. Yang and J. I. Guo, "A high throughput H.264/AVC high profile CABAC decoder for HDTV applications," *IEEE Trans. Circuits and Systems for Video Technology*, vol.19, no. 9, pp.1395-1399, Sept. 2009.

[6] P. Zhang, D. Xie, and W. Gao "Variable-bin-rate CABAC engine for H.264/AVC high definition real-time decoding," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol.17, no. 3, pp.417-426, Mar. 2009.

[7] Y. Hong, P. Liu, H. Zhang, Z. You, D. Zhou, and S. Goto, "A 360Mbin/s CABAC decoder for H.264/AVC level 5.1 applications," *Proc. SoC Design Conference 2009*, pp.71-74, Nov. 2009.

[8] T.-D. Chuang, P.-K. Tsung, P.-C. Lin, L.-M. Chang, T.-C. Ma, Y.-H. Chen, and L.-G. Chen, "A 59.5mW scalable/multi-view video decoder chip for Quad/3D Full HDTV and video streaming applications," *2010 IEEE Int. Solid-State Circuits Conf. Dig. Tech. Papers*, pp.330-331, Feb. 2010.

[9] K. Kato, R. Hashimoto, G. Fujita, and T. Onoye, "VLSI architecture of multi-symbol CABAC decoder for H.264/AVC high profile," *IEICE Technical Report*, SIP2007-121, ICD2007-110, IE2007-80, pp.65-70, Oct. 2007 (in Japanese).

[10] R. Hashimoto, K. Kato, M. Saitsuji, T. Tanaka, H. Uetsu, G. Fujita, and T. Onoye, "VLSI architecture of H.264 multi-symbol entropy decoder for 1080HD," *22nd Signal Processing Symposium*, P2-2, Nov. 2007 (in Japanese).