LETTER

Parallel Decoding of Context-based Adaptive Binary Arithmetic Codes Based on Most Probable Symbol Prediction

Chung-Hyo Kim,[†] *Nonmember* and In-Cheol Park[††], *member*

Summary
Context-based adaptive binary arithmetic coding (CABAC) is the major entropy-coding algorithm employed in H.264/AVC. Although the performance gain of H.264/AVC is mainly due to CABAC, it is difficult to achieve a fast decoder because the decoding algorithm is basically sequential and computationally intensive. In this letter, a prediction scheme is proposed that enhances overall decoding performance by decoding two binary symbols at a time. A CABAC decoder based on the proposed prediction scheme improves the decoding performance by 24% compared to conventional decoders.

Key words:
video coding, arithmetic codes, prediction theory, entropy codes

1. Introduction

The newest international video coding standard H.264/AVC developed by the joint video team of the MPEG and ITU can produce a video stream of perceptually equivalent quality at about half a bit-rate compared to MPEG-2. The performance gain is mainly resulted from context-based adaptive binary arithmetic coding (CABAC) employed in H.264/AVC main profile [1]. The CABAC, a binary arithmetic code [2] associated with the context modeling technique, was reported in [3]; it can reduce the bit rate by 32% compared to other compression methods such as Huffman [4] and Exp-Golomb codes [5] if appropriate context models are provided. Although more than 90% of the H.264 main profile stream is encoded using the CABAC, its decoding algorithm is basically sequential and needs large computation to calculate range, offset and context variables, making it difficult to achieve high decoding performance [6-7]. The CABAC decoding complexity required to process high definition images in real time is about 3 giga-operations per second. Although this computing complexity is still less than block processing complexity, the CABAC decoding becomes a major bottleneck in real time processing due to its serial processing nature. On the other hand, block processing can be easily enhanced by applying parallel and pipeline techniques.

In this letter, we propose a parallel CABAC decoding method that can decode two binary symbols at a time to achieve a high-speed decoder meeting the real-time processing requirement of the H.264/AVC standard. In the proposed decoding method, the first binary symbol is decoded as the conventional scheme, while the second one is decoded with predicting that the first symbol is the most probable one.

2. CABAC encoding and decoding

A sequence of syntax elements to be encoded is first converted to a sequence of codewords each of which is a binary string consisting of binary symbols called bins, as the CABAC deals with only binary symbols. The binary string of a codeword is constructed under specific rules, and a codeword is valid if its binary string satisfies the rules. The binary symbol with the higher probability is called the most probable symbol (MPS) and the other is the least probable symbol (LPS). A specific context model containing the LPS probability and the MPS value is assigned to each bin. By applying binary arithmetic coding, each bin is sequentially encoded using the associated context model to produce a number called an offset. Note that the context model is not fixed but adaptively updated for the next encoding.

The decoding is similar to the encoding. Given an offset, a CABAC decoder repeatedly decodes bins until all the codeword are found. The CABAC decoder has a merging unit to check whether the sequence of decoded bins matches with a valid codeword. Except the merging unit, the encoding and decoding procedures are almost the same. To decode a bin, the binary arithmetic decoder (BAD) needs the corresponding range, offset and context model. A decoding example is shown in Fig. 1, where the initial offset is set to 0x113 by taking the higher 9 bits from the encoded bit-stream, 10001001100. Note that a different context model can be used for each bin decoding.

Starting from the initial range, 0x1FE, it is narrowed after each bin is decoded. The range is divided into two sub-ranges, R_{MPS} and R_{LPS}, where R_{LPS} is calculated by multiplying the range and the LPS probability specified in the context model, and R_{MPS} is computed by subtracting

Manuscript received January 2006.
[†] The author is with Power Generation Laboratory, Korea Electric Power Research Institute, 103-16, Moonji-dong, Yuseong-gu, Daejeon, Korea
[††] The author is with the Department of Electrical Engineering, Korea Advanced Institute of Science and Technology, 373-1, Guseong-dong, Yuseong-gu, Daejeon, Korea

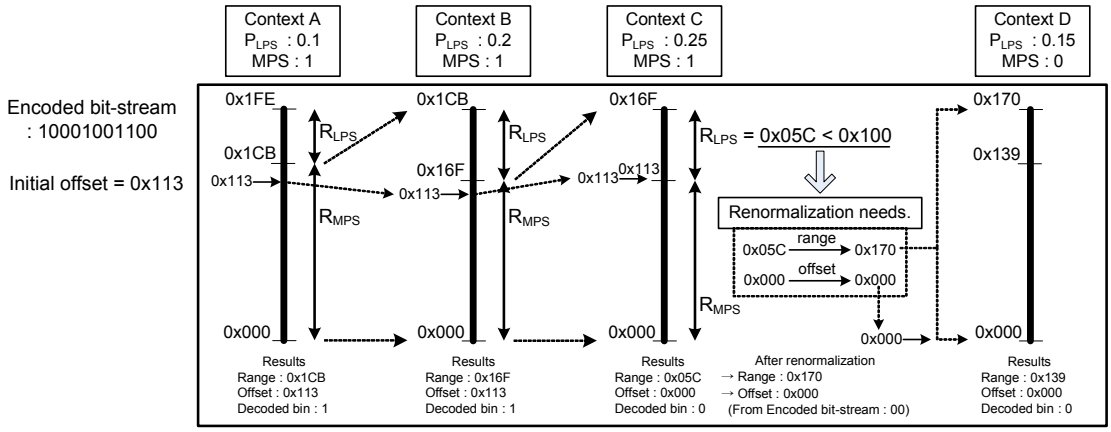

Fig. 1. The decoding procedure

R_{LPS} from the range. If the offset is less than R_{MPS}, the bin is decoded as the MPS and the range to be used for the next decoding is set to R_{MPS}. Otherwise, the bin is regarded as the LPS and the next range is set to R_{LPS}.

Since the range becomes narrow as the decoding progresses, more bits are needed to represent it. To avoid this situation, the range is renormalized to make it equal to or greater than 0x100. If the range is less than 0x100, it is shifted left until the range is in [0x100, 0x1FE], and the offset is updated by appending *a* bits of the encoded bit-stream, where *a* is the shift amount. If the decoded bin is the MPS, the renormalization requires at most one shift, because the MPS probability is always greater than or equal to 0.5. In case of LPS decoding, the renormalization occurs always and requires multiple shifts. For example, the third decoding in Fig. 1 requires 2-bit shift to renormalize the range and offset. In addition, the offset should be adjusted by subtracting R_{MPS} before the renormalization. Table 1 summarizes how variables are updated according to the decoded bin.

Table 1. Variable update after 1 bit decoding

Case		MPS decoding	LPS decoding
No renormalization	Frequency	Frequent	None
	Range	R_{MPS}	-
	Offset	No change	-
Renormalization	Frequency	Rare	Always
	Shift amount *n*	1	Arbitrary
	Range	$R_{MPS} << 1$	$R_{LPS} << a$
	Offset	Offset << 1	(Offset − R_{MPS}) << *a*

3. Proposed prediction scheme

If the decoded bin is the MPS, the next range and offset required for the next bin decoding can be calculated with simple operations even if the renormalization is required. Based on the analysis, we propose a prediction scheme to decode two bins at a time. The first binary symbol is decoded as the conventional scheme, while the second is decoded with predicting that the first symbol is the MPS. There is a problem to be solved for the parallel decoding. To start the second predicted decoding, we should know the context model to be used for the second bin decoding. Since a context model is selected out of 399 ones by referring to the previous bin decoding, it is difficult to determine the context model to be used for the second bin decoding. In addition, a context model used for a bin decoding is updated after the bin is decoded, which means that the updated context model should be used for the second bin decoding if the same context model is applied to the first and second bins. Analyzing the pattern of context models applied to two sequential bins contained in a syntax element, we found that the two context models are different in most cases and the second context model can be selected independent of the first context model. As shown in Table 2 [8], predicting the second context model is hard in case 4, because it is dependent on the first bin. In cases 2 and 3, it is sometimes difficult to predict the second context model because the context model updated by the first bin is required when *k* or f(*x*) is equal to *n*. However, the percentage of those unpredictable cases is much less than that of the predictable cases. In addition, f(*x*) and *k* are usually *n+1* even in cases 2, 3, and 4. Based on the observation, the second bin is decoded with predicting that the first bin is the MPS and the next context model is used for the second decoding.

Table 2. Neighboring context composition

Case	1st context model	2nd context model	Percentage
1	constant *n*	constant *n+1*	44.2%
2	constant *n*	constant *k*	8.8%
3	constant *n*	f(*x*) ; *x* is independent of the 1st decoding	29.4%
4	constant *n*	f(*x*) ; *x* is dependent on the 1st decoded bin	17.6%

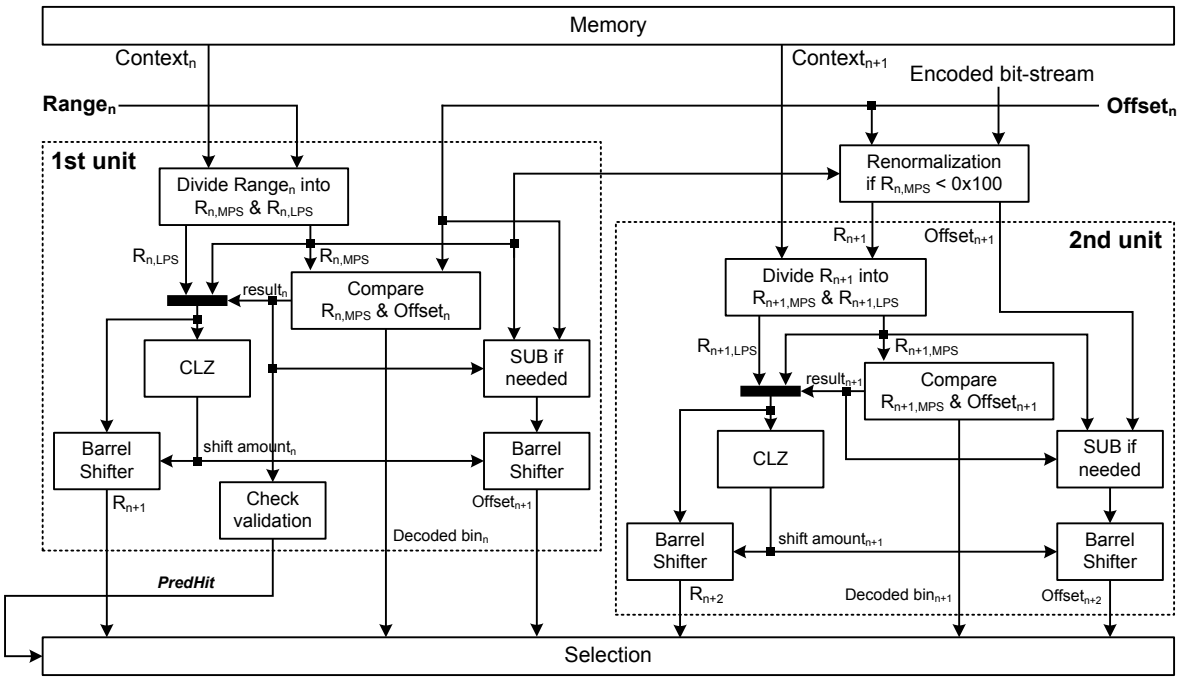

Fig. 2. The proposed CABAC decoder

The proposed CABAC decoder shown in Fig. 2 includes two BAD units and reads two sequential context models at a time. The first BAD unit is the same as the conventional decoder except additional two output signals, $R_{n,MPS}$ and PredHit, where $R_{n,MPS}$ is the range to be transferred to the second BAD unit and PredHit is to indicate whether the decoding result of the first unit is the MPS and the next context model is valid for the second decoding. Before starting the second bin decoding, the second unit performs renormalization if $R_{n,MPS}$ is less than 0x100. If PredHit is asserted, the decoding result of the second unit is valid. Otherwise, it is discarded. Since there are little changes in the range and offset when the MPS is decoded in the first unit, the second unit can start after a little delay taken to calculate $R_{n,MPS}$.

4. Experimental Results

The proposed decoder was described in Verilog HDL and synthesized in 0.18um CMOS technology. The bin decoding is processed in three cycles. Two context models are read in the first cycle, bins are decoded in the second cycle, and the decoded bins are merged in the third cycle.

The critical part of a conventional decoder is the BAD unit that takes 3.3ns. As the second BAD needs some additional delay to calculate $R_{n,MPS}$ and select valid results, its delay increases to 4.5ns in the proposed decoder. To reduce the additional delay in the second cycle, the calculation of $R_{n,MPS}$ and the selection of valid results are retimed to the first cycle and the third cycle, respectively, as shown in Fig. 3. With the retiming, the proposed decoder can work at the same clock frequency as that of the conventional decoder.

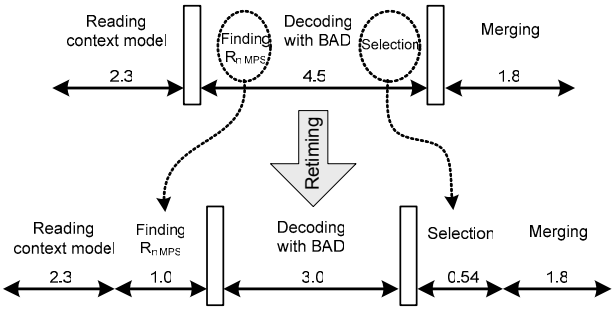

Fig. 3. Retiming to reduce critical path delays (unit : ns)

Table 3 shows the simulation results for two benchmark files each of which consists of 120 progressive frames of 720*480 pixels. The case that has at least two bins remaining to be decoded in a syntax element is called a predictable case, and the other is an unpredictable case. The prediction is hit only when the second predicted decoding is valid, that is, when the first BAD unit decodes the MPS and the next context model is valid for the second decoding. If the prediction is hit, we can decode two bins at a time and thus can save three cycles. The prediction accuracy is 67% on the average if predictable

cases are considered. The conventional method that does not employ the proposed scheme decodes 0.33 bins per cycle, which is the same throughput as the unpredictable case, while the proposed method can decode 0.41 bins per cycle as shown in Table 3. Therefore, with the proposed predicted decoding, we can decode 24% more bins compared to the conventional serial decoder.

Table 3. Simulation results for two benchmark files

File	Case	No. of bins	Prediction hit (%)	bins/cycle
Car.yuv	Predictable	14499682	69.07	0.56
	Unpredictable	13745309	-	0.33
	Total	28244991	-	**0.41**
Cheer.yuv	Predictable	20441764	65.42	0.55
	Unpredictable	20567685	-	0.33
	Total	41009449	-	**0.41**

5. Conclusions

We have presented a prediction-based CABAC decoding scheme that offers improved performance by decoding two bins simultaneously. In the proposed scheme, the first bin is decoded conventionally, while the second bin is decoded with predicting that the first bin is the MPS and the next context model is needed for the second decoding. Experimental results show that the proposed prediction scheme can improve the overall decoding performance by 24% compared to the conventional decoder.

Acknowledgments

This work was supported in part by Institute of Information Technology Assessment through the ITRC and by IC Design Education Center.

References

[1] D. Marpe, H. Schwartz, and T. Wiegand. “Context-Based Adaptive Binary Arithmetic Coding in the H.264/AVC video compression standard,” IEEE Trans. on CSVT, vol. 13, pp. 620-636, July 2003.

[2] I. H. Witten, R. M. Neal, and J. G. Cleary. “Arithmetic coding for data compression,” Communications of the ACM, vol. 30, pp. 520–540, June 1987.

[3] D. Marpe, G. Blattermann, G. Heising, and T. Wiegand. “Video compression using context-based arithmetic coding,” ICIP 2001, vol. 3, pp. 558-561, Oct. 2001.

[4] D. Huffman, “A method for the construction of minimum redundancy codes,” Proc. IRE, vol. 40, pp. 1098-1101, Sept. 1952.

[5] J. Teuhola, “A compression method for clustered bit-vectors,” Inform. Proceedings Lett., vol. 7, pp. 308-311, Oct. 1978.

[6] Aravind AL, Bindu P.Rao, Sudhir S. Kidva, Sreenu Babu, Sumam David, and Ajit V. Rao, "Quality and complexity comparison of H.264 intra mode with JPEG2000 and JPEG”, ICCP 2004, vol. 1, pp. 525-528, Oct. 2004.

[7] S. Saponara, C. Blanch, K. Denolf, and J. Bormans, “The JVT advanced video coding standard: complexity and performance analysis on a tool-by-tool basis,” in Proc. 13th Int. Packetvideo Workshop, pp. 98-109, Apr. 2003.

[8] Thomas Wiegand, “Draft ITU-T Recommendation H.264 and Final Draft International Standard of Joint Video Specification (ITU-T Rec. H.264 | ISO/IEC 14496-10 AVC)", pp.160-187, May 2003.