A High-throughput Low-latency Arithmetic Encoder Design for HDTV

Yuan Li, Shanghang Zhang, Huizhu Jia, Xiaodong Xie, and Wen Gao, *Fellow*, IEEE
National Engineering Laboratory for Video Technology
Peking University
Beijing, China
Email: yuanli@pku.edu.cn

***Abstract* — In this paper, we propose a high-throughput low-latency arithmetic encoder (AE) design suitable for high definition (HD) real-time applications employing advanced video coding standards such as H.264/AVC or AVS and using a macroblock (MB) level pipeline. First, in order to derive the performance requirement on the AE, a buffer model in connected with which it is designed is thoroughly analyzed. Then, using joint algorithm-architecture optimization and multi-bin processing techniques, we introduce a novel binary arithmetic coder (BAC) architecture with throughput of 2~4 bins per cycle sufficient for real-time encoding. Furthermore, a hybrid context memory scheme is presented to meet the throughput requirement on the BAC. Simulation result shows that our design can support 1080p at 60 fps for AVS HDTV real-time coding with a bin rate up to 107K per MB line. Synthesized with the TSMC 0.13μm technology, the AE can run at 200MHz and costs 47.3K gates. By operating at 130MHz, the design is also verified in an AVS HD encoder on a Xilinx Virtex-6 FPGA prototype board for 1080p at 30 fps.**

I. INTRODUCTION

Today's video codec adopts a series of innovative coding techniques to achieve high compression efficiency. Both H.264/AVC [1] and AVS [2] use entropy coding to reduce the redundancy of transformed coefficients after prediction, as well as the motion information etc. Context-based Adaptive Binary Arithmetic Coding (CABAC) [3] is adopted in H.264/AVC Main/High profiles which can save up to 14% bit rate compared to Context-based Adaptive Variable Length Coding (CAVLC). AVS also adopts a similar Context-based Binary Arithmetic Coding (CBAC) [4] in Jiaqiang profile which can save up to 13% bit rate than Context-based 2D Variable Length Coding (C2DVLC). However, the arithmetic coding techniques cause additional computational complexity. In order to implement a real-time HDTV encoder, it becomes necessary to develop a high-throughput AE architecture with strong considerations for the system latency.

Several AE architectures have been proposed in literature to increase the throughput in recent years. Osorio [5] introduced a multi-bin BAC by packing the equally probable (EQ) bins with normal ones. The updates of Range and Low are also separated to achieve a throughput around 2bins/cycle in average. In Liu's [6] article, a 4-stage pipelined BAC architecture is presented, which can process one bin per cycle. Since the delay of accessing neighboring syntax elements (SE) is long, the throughput of this encoder is only 0.67bin/cycle. Tian [7] proposed a RDO-support CABAC encoder with a throughput of 1bin/cycle. By using high operating frequency, the throughput of this encoder is higher than most of others. Reported by Chen [8], a 6-stage pipelined BAC with multi-bin processing capability is presented. By optimizing the renormalization step, the average throughput of 1.42bins/cycle can be achieved for the design.

These state-of-the-art AEs focus on the throughput mostly, while for real-time coding applications, the system latency caused by the large size of the front-buffer of the AE is also important. In this paper, we analyze front-buffer behavior and its requirement on the BAC processing speed and then propose a novel BAC architecture with throughput of 2~4 bins per cycle by using joint algorithm-architecture optimization and multi-bin processing techniques. A hybrid context memory scheme is also presented to meet the throughput requirement on the BAC. Implemented for AVS, the proposed AE can achieve low delay and real-time performance for HDTV (1080p at 60fps).

The rest of this paper is organized as follows. Section II analyzes the buffer model and performance requirement. The AE architecture and optimization strategies are proposed in Section III. Section IV shows the implementation result and comparison with other works. Finally, we conclude this paper in Section V.

II. BUFFER MODELING

In AE, the throughput of BAC is the bottleneck since the binarizer can generate large amount of bins by, for example, processing multi SEs simultaneously. For real-time design, MB-level pipelining is usually adopted to increase the throughput and reduce latency. However, the quantity of input bins (from binarizer) of BAC for a MB could fluctuate tremendously, which makes the MB-level pipelining of AE very difficult. It is obvious that we can use a large buffer to smooth the input fluctuation and in our design we also make such an arrangement. The problem then becomes the

978-1-4673-5762-3/13/$31.00 ©2013 IEEE

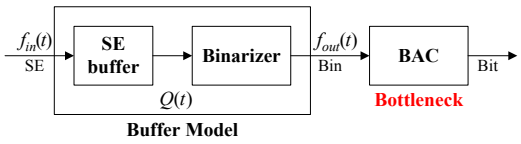

Figure 1. The proposed buffer model.

balancing between the input (front) buffer size and its throughput requirement, which directly translates into chip area and system latency. In the following, we analyze the assumed buffer model (as depicted in Fig. 1) to illustrate the relationship between the buffer size and the throughput requirement.

Let $f_{in}(t)$ be the input to the buffer with variable, time t, let $f_{out}(t)$ be the output of the buffer with a constant read-out speed C and let T be the length of a continuous time period. Let $Q_{in}(t,T)$ and $Q_{out}(t,T)$ be the quantities of data written to the buffer and data read out from the buffer, respectively, during time period $(t, t+T]$. Let $Q(t)$ be the quantity of data held in the buffer at time t. Then we see the following equations hold.

$$f_{out}(t) = \begin{cases} C, & if\ Q(t) \neq 0 \\ 0, & if\ Q(t) = 0 \end{cases}. \quad (1)$$

$$Q_{in}(t,T) = \int_t^{t+T} f_{in}(t)dt. \quad (2)$$

$$Q_{out}(t,T) = \begin{cases} TC, & if\ Q(t) \neq 0\ for\ t \in (t, t+T] \\ \int_t^{t+T} f_{out}(t)dt, & if\ buffer\ is\ empty\ at\ some\ t \end{cases}. \quad (3)$$

$$Q(t) = Q(t-T) + Q_{in}(t-T,T) - Q_{out}(t-T,T). \quad (4)$$

Based on these equations, we will prove the following hypothesis about the buffer model.

Hypothesis 1: *For any starting time t, if there exists an upper bound Q_0 for the total input data to the buffer in a given time period T_0 ($Q_{in}(t,T_0) \leq Q_0$), then we can set the Q_0 as the buffer size and $C = \frac{Q_0}{T_0}$ as the constant output speed of the buffer to guarantee that the buffer will never overflow, which is $Q(t) \leq Q_0$ for any t.*

Proof: We first prove the following lemma.

Lemma 1: *If the buffer is empty at time t_0 and is full or overflowed at t_1, then we can get that the time period from t_0 to t_1 is larger than T_0, that is $t_1 - t_0 > T_0$.*

Proof: Because the total input data to the buffer is monotonically increasing and Q_0 is the upper bound for any given time period T_0, the total input data cannot exceed Q_0 unless the time period is equal or larger than T_0. Moreover, during the time period from t_0 to t_1, there must have data output from the buffer. So the time period from buffer empty to buffer full or overflowed is strictly larger than T_0.

Based on Lemma 1 we can further prove the Hypothesis 1. Suppose at time instance of t_f the buffer overflows at the first time, then from Lemma 1, $t_f > T_0$ must be true. Considering a time instance $t_s = t_f - T_0$, the amount of data in the buffer at time t_s can be obtained by the following equation.

$$Q(t_s) = Q(t_f) - Q_{in}(t_s,T_0) + Q_{out}(t_s,T_0) \quad (5)$$

TABLE I. STATISTICAL RESULTS OF Q_0 AND T_0 (GOP OF IBBP)

Sequence	QP	Bit rate (Mbps)	Q_0 (Bins)	T_0		Throu. (Bins/cy.)
				Cycle	MB	
bluesky	14	114.3	100657	49020	120	2.05
sunflower	12	121.6	96706	49020	120	1.97
mobcal_ter	21	99.4	107160	49020	120	2.19
fireworks	40	106.3	71686	49020	120	1.46
pedestrian_area	20	103.7	77930	49020	120	1.59
BasketballDrive	16	113.5	79688	49020	120	1.63

It is obvious that $Q(t_f) > Q_0$ and $Q_{in}(t_s,T_0) \leq Q_0$. For $Q_{out}(t_s,T_0)$, we have two mutually exclusive situations as follows.

1) *From t_s to t_f, the buffer is empty at time instance t_e.*
2) *From t_s to t_f, the buffer has never been empty.*

For Situation 1), we get that the time period from buffer empty (t_e) to buffer overflowed (t_f) is $t_f - t_e < T_0$, which contradicts with Lemma 1. For Situation 2), since the buffer is never empty, the total output data from the buffer is $Q_{out}(t_s,T_0) = T_0C = Q_0$. Then (5) becomes

$$Q(t_s) = Q(t_f) - Q_{in}(t_s,T_0) + Q_{out}(t_s,T_0) > Q_0 - Q_0 + Q_0 = Q_0.$$

This means that at time t_s the buffer already overflows and this contradicts with the assumption that the time t_f is the very first time that buffer overflows. The Hypothesis 1 is thus proved.

With this buffer model, if we find Q_0 for a given T_0 (this implies that when a window of time T_0 is checked across the entire input bin stream, there are at maximum Q_0 number of bins), the buffer size can be set to Q_0 and the throughput to $\frac{Q_0}{T_0}$, and this can guarantee the buffer will never overflow. In order to obtain these two parameters, we tested a number of HD (1080p) sequences at level 6.0.3 in AVS, which supports resolution of 1920x1152@60fps and bit rate up to 100Mbps. Table I shows the detailed results. ***From the test result, our design is set to aim at using a buffer capable of holding the data of a MB line (120MBs for 1080p) and achieving the throughput of 2.2 bins per cycle when operating at 200MHz.***

III. PROPOSED ARITHMETIC ENCODER

A. Top-level Arithmetic Encoder Architecture

The proposed AE architecture, which consists of three main functional blocks, is shown in Fig. 2. Block Binarizer converts the input SE into bin string. A line buffer holding neighboring SE for context selection purpose is also located in this block. Block CM (context management) contains the proposed hybrid memory scheme and the context updating logic. BAC block can support a throughput of 2~4 bins per cycle. Since these three blocks are data independent (no feedback loop), we adopt a 3-stage pipelined scheme among them to increase the throughput. For the critical path in CM and BAC, we optimize them according to the following strategies.

B. Optimization of BAC Architecture

First, we construct a baseline BAC architecture with a constant throughput of 2bins/cycle. The critical path in BAC is the calculation of Low in the iteration, especially in LPS. Hence the worst case of 2-bin calculation is in processing two

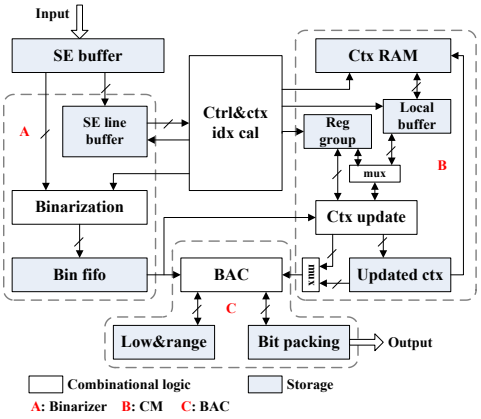

Figure 2. Proposed arithmetic encoder architecture.

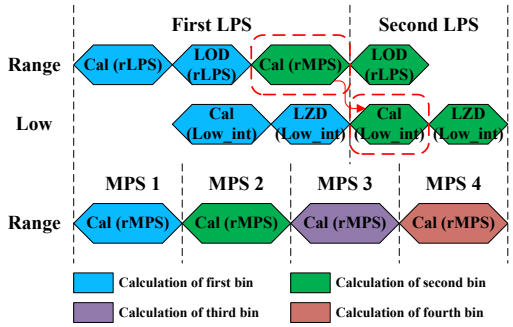

Figure 3. Timing diagram of the worst case for two LPS bins.

consecutive LPS bins. By further observation we found two useful characteristics that can be used for speeding up the worst-case processing. One is that the calculation of Range only depends on itself in the iteration and the operating time is much shorter than Low. In this situation, some operation of Low (calculation of rMPS and rLPS as shown in Fig. 3, rMPS and rLPS indicates the widths of interval for MPS and LPS, seperately), can be pre-calculated in the first iteration after the calculation of Range is done, which significantly reduces the operating time of Low in the second iteration. The other one, e.g. in CBAC, is the initialization of Range when LPS happens. In logarithm domain, the Range is represented by two parameters which are its integral part and its decimal part. The integral part will be initialized when LPS happens, which can make calculation of Low in the second iteration much easier. By applying this strategy (i.e. time borrowing), we can achieve a joint algorithm-architecture optimization that makes it possible a constant throughput of 2bins/cycle and a latency 33% lower than in the case of an architecture with a performance of 1bin/cycle.

Moreover, we adopt the multi-bin processing to further increase the throughput. For coefficient coding, with the quantization parameter (QP) getting smaller, the number of bins increases dramatically when the bit rate rises. Taking AVS as an example, which is also applicable for H.264/AVC, the coefficients are coded by Level (absLevel and Sign) and Run pairs and the type of binarization is unary as shown in Fig. 4. For the context model selection, the bins whose indexes are equal or larger than 2 for absLevel and 1 for value of Run use the same contexts, which are called same-context (SC) bins.

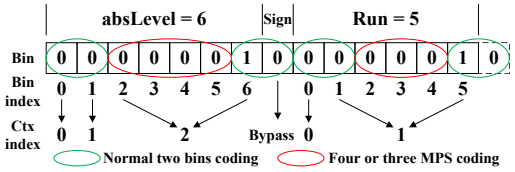

Figure 4. Speed up using multi-bin processing for coefficients.

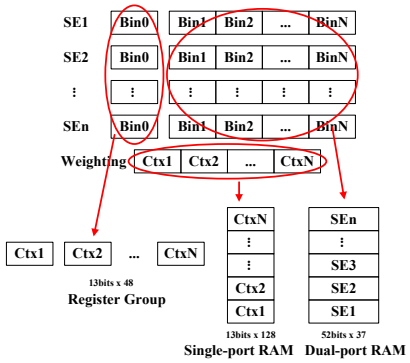

Figure 5. Hybrid memory scheme.

The SC bins consists of several consecutive '0's and a '1' at the end. It is obvious that the MPS of the context for SC bins is '0' in most cases. Since the calculation of MPS is much simpler than LPS, we can process multi consecutive '0's in one cycle. Considering the matching of circuit delay with processing two LPS bins, we adopt 4-bin MPS processing circuit which can encode four or three MPS bins in one cycle (shown in Fig. 3). This multi-bin processing technique further increases the throughput by 15% than the throughput of 2bins/cycle in average.

C. *Hybrid Memory Scheme*

In the AE architecture, we adopt a two-level memory architecture, consists of context RAM and local context buffer, to speed up the memory access [7]. However, in our constant 2bins/cycle scheme, the consecutive two bins may belong to different SEs, which is a bit difficult for context RAM mapping. For example, in AVS, if the first bin is the last bin of SE mb_type, the second bin may be part of the SE mb_part_type or reference index. We call this type of SE the Collision SE (CSE), which must be located in different RAMs. To fulfill all these CSEs, 4～5 dual-port RAMs and the corresponding local buffers are needed at least. After further careful observation, we found that in all bin pairs associated to the CSEs, the bin index of the second bin in bin pair is always 0 (means the first bin of SE). According to this observation we propose a hybrid memory scheme as shown in Fig. 5. All context models used by the first bin of CSEs are stored in register groups for fast access and the other context models are stored in a dual-port RAM for high storage density. For the context weighting technology adopted in AVS [4], two context models may be used for coding one bin (WBin). In our constant 2bins/cycle scheme, the WBins cannot be consecutive and we use a single-port RAM to store the associated context models. Finally, totally 324 context models are stored which is only one more than the 323 context models required to support AVS standard. This memory scheme can also be applied to H.264/AVC due to the same concept.

TABLE II. SIMULATION RESULT WITH 1080P SEQUENCES (GOP IS IBBP)

Sequence	QP	Bit rate (Mbps)	Throu. needed (Bins/cycle)	Throu. actually (Bins/cycle)
bluesky	14	114.3	2.05	2.34
sunflower	12	121.6	1.97	2.36
mobcal_ter	21	99.4	2.19	2.41
fireworks	40	106.3	1.46	2.17
pedestrian_area	20	103.7	1.59	2.39
BasketballDrive	16	113.5	1.63	2.35
Average		109.8	1.82	2.34

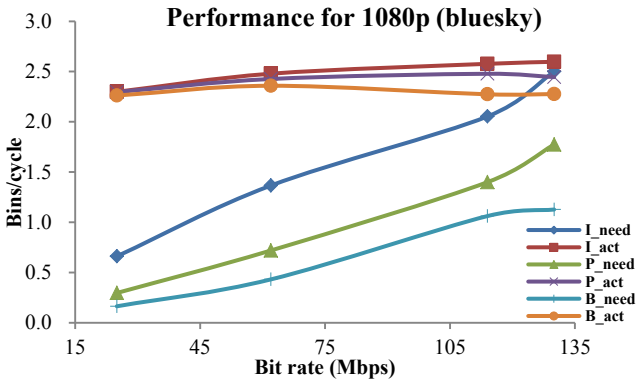

Figure 6. Actual performance of different frame types for 1080p.

IV. IMPLEMENTATION RESULT

Table II shows the performance of our architecture in encoding HD (1080p@60fps, bit rate up to 100Mbps) sequences. The actual throughputs of these sequences, under the given buffer size for storing the data for a MB line, are all higher than the requirement. The simulation results for different bit rates and frame types are shown in Fig. 6. The throughputs needed for I frame per MB line increases from 0.66bins/cycle to 2.50bins/cycle when bit rate is from 20Mbps to 130Mbps, while the actual performance of our AE (from 2.31bins/cycle to 2.60bins/cycle) can follow this trend to meet the throughput requirement. Because the coefficients take the most calculations when bit rate is rising, in this situation, our multi-bin processing for coefficients becomes more and more effective. P and B frames have much less bins than I frame, and the actual throughputs are also lower but still sufficient. Thus the real-time coding for AVS level 6.0.3 can be achieved with low delay.

Our design is synthesized using the TSMC 0.13µm process. The detailed result is shown in Table III. The gate count includes all functional blocks in CBAC and memories for contexts. The comparison with previous works is shown in Table IV. Our design is the best on the throughput per cycle and also has an additional advantage of low system delay, i.e. less than a MB line for 1080p video coding. The gate area of computational logic of CBAC in AVS costs less than the CABAC in H.264/AVC. However, the memory cost of CBAC (13bits x 323) is much more than CABAC (7bits x 400). Hence the total gate count of our design is more than other works targeting H.264/AVC. The functions are also different between these designs as shown is Table IV. In addition, our design has been successfully verified in an AVS HD encoder on a Xilinx Virtex-6 FPGA prototype board operating at 130MHz for 1080p at 30 fps.

TABLE III. SUMMARY OF THE IMPLEMENTED RESULT

Process technology	TSMC 0.13 µm CMOS
Max frequency	200MHz
Memory count	Single-port RAM: 208bytes Dual-port RAM: 241bytes Register group: 78bytes
Total gate count	47.3K
Processing ability	1080p at 60fps with bit rate up to 100Mbps
System latency	Less than a MB line for 1080p

TABLE IV. COMPARISON WITH PREVIOUS WORKS

	Osorio [5]	Tian [7]	Chen [8]	Ours
Standard	H.264/AVC	H.264/AVC	H.264/AVC	AVS
Technology (µm)	0.35	0.13	0.13	0.13
Max frequency	186	578	222	200
Throu. (bins/cycle)	1.9∼2.3	1	1.42	2.34
Gate count	19.4K	44.6K	46.0K	47.3K
Function	BAC + part of BI	Full	Full	Full

V. CONCLUSION

In this paper, a high-throughput AE architecture for HD real-time video coding is proposed, with the minimized latency of a MB line. We establish a buffer model about the relationship between buffer size and throughput requirement. A novel BAC architecture with throughput of 2∼4 bins per cycle and the corresponding hybrid context memory scheme are proposed. Simulation results show that our design is far more superior than the ones in the literature and can support 1080p at 60 fps for AVS HDTV real-time coding, bin rate up to 107K for a MB line. Synthesized with the TSMC 0.13µm technology, the proposed AE can run at 200MHz and costs 47.3K gates. We have successfully implemented this AE in an AVS HD encoder on a Xilinx Virtex-6 FPGA prototype board operating at 130MHz for 1080p at 30 fps.

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, Mar. 2003.

[2] Chinese GB/T20090.2 Information Technology—Advanced Audio Video Coding Standard Part2: Video, 2006.

[3] D. Marpe, H. Schwarz, and T. Wiegand, "Context-based adaptive binary arithmetic coding in the H.264/AVC video compression standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 620-636, July 2003.

[4] L. Zhang, X. Wu, N. Zhang, W. Gao, Q. Wang, and D. Zhao, "Context-based arithmetic coding reexamined for DCT video compression," in *Proc. IEEE ISCAS,* May. 2007, pp. 3147-3150.

[5] R. R. Osorio and J. D. Bruguera, "High-throughput architecture for H.264/AVC CABAC compression system," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 16, no. 11, pp. 1376-1384, Nov. 2006.

[6] P. S. Liu, J. W. Chen, and Y. L. Lin, "A hardwired context-based adaptive binary arithmetic encoder for H.264 advanced video coding," in *Proc. VLSI-DAT*, Apr. 2007, pp. 1-4.

[7] X. H. Tian, T. M. Le, X. Jiang, and Y. Lian, "Full RDO-support power-aware CABAC encoder with efficient context access," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 19, no. 9, pp. 1262-1273, Sept. 2009.

[8] J. W. Chen, L. C. Wu, P. S. Liu, and Y. L. Lin, "A high-throughput fully hardwired CABAC encoder for QFHD H.264/AVC main profile video," *IEEE Trans. Consumer Electron.*, vol. 55, no. 4, pp. 2529-2536, Nov.2010.