International Journal of Control and Automation
Vol. 10, No. 5 (2017), pp.199-208
http//dx.doi.org/10.14257/ijca.2017.10.5.19

High-Throughput Architecture of HEVC CABAC Binary Arithmetic Encoder

Hyungu Jo, Seungyong Park and Kwangki Ryoo

Graduate School of Information and Communication, Hanbat National University, Korea
iookie@nate.com, srrr.kr@gmail.com, kkryoo@hanbat.ac.kr

Abstract

This paper proposes an efficient binary arithmetic encoder hardware architecture for CABAC (Context-based Adaptive Binary Arithmetic Coding) encoding. UHD (Ultra High Definition) images require very high throughput for standard video encoder. CABAC is an entropy coding method that is used in HEVC standard. Entropy coding removes statistical redundancy and supports a high compression ratio of images. However, the binary arithmetic encoder causes a delay in real time processing, making parallel processing difficult, because of the high dependency between data. The function of the proposed CABAC BAE hardware structure is to separate the renormalization and process the conventional iterative algorithm in parallel. In addition, it generates an information bit for outputting a bitstream, and outputs a variable bitstream in one cycle. The new scheme was designed as a four-stage pipeline structure that can reduce critical path optimally. The proposed CABAC BAE hardware architecture was designed with Verilog-HDL and synthesized in 65nm technology. Its gate count is 11.2K and maximum clock frequency is 625MHz. It processes 4 Bins per clock cycle, with a 36% increase in maximum processing speed compared to existing hardware architectures.

Keywords*: HEVC, CABAC, Binary Arithmetic Encoder, Entropy Coding*

1. Introduction

Recent times have seen rapid developments in the network industry (image processing technology and communication technology for broadcasting and communication), coupled with the popularization of high definition television (HDTV) and support for high resolution service of multimedia devices. This has sparked the interest of users in the ultra-high resolution video service and led to the development of Ultra-High Definition UHD) class images with a resolution over 4 times higher than full high definition (FHD) which has resolution of 1920×1080 [1]. Due to the increased interest of these users in ultra-high resolution image services, image compression standards with higher performance than the existing image compression standard H.264/AVC have become necessary. HEVC supports various image compression from low-resolution 176×144 images to ultra-high resolution 4K and 8K images. Compared with H.264/AVC, the previous image compression standard, it has an improved compression efficiency of about 50% or more [2]. However, there is an increase in complexity and computation time, making it difficult to process in real time. The entropy coding method of H.264/AVC standard uses both context-based adaptive binary arithmetic coding (CABAC) and context-based adaptive variable length coding (CAVLC), but HEVC adopts only CABAC with higher compression efficiency [3]. CABAC eliminates statistical redundancy and provides high compressibility of video. After encoding one Bin, the probability model is updated, and the next Bin is encoded with the updated probability model. This method improves the compression efficiency, but after the encoding of the current Bin is finished, the probability model of the next Bin can be updated and the encoding of the next Bin can

ISSN: 2005-4297 IJCA

be performed. [4]. In this paper, we propose a hardware design of a CABAC binary arithmetic encoder with high throughput by applying a four-stage pipeline structure that can operate optimally by separating delayed sections in the computation process - this paper is an elaborate version of our previous work [5]. In order to optimize critical path and output the variable bitstream in one cycle, the information bit is generated and output simply through the LUT. Contents of this paper are as follows. Chapter 2 describes a CABAC Encode, Chapter 3 describes the proposed binary arithmetic encoder (BAE) and Chapter 4 describes Hardware Implementation. Finally, Chapter 5 describes the conclusion of this paper.

2. CABAC Encode

2.1. CABAC Encoder

The CABAC encoder performs adaptive binary arithmetic coding through a context-based modeling method that selects a probability model for each context element. CABAC Encoder consists of Binarizer, Context Modeler, and BAE as shown in Figure 1.

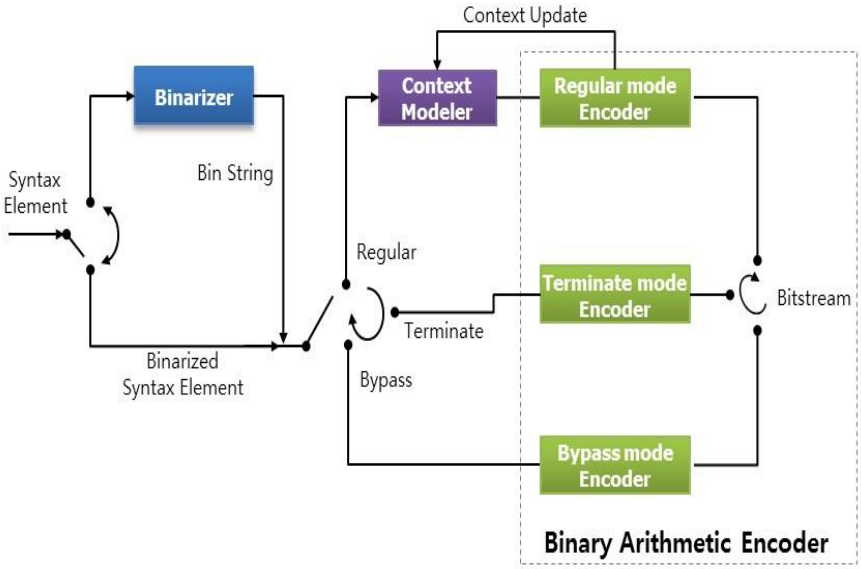

Figure 1. Block Diagram of CABAC Encoder

The Binarizer outputs a syntax element (not a binary value) as a Bin string of a binary sequence. If the syntax element has a binary value, it passes the binarization process. The context modeler estimates the context model probability using the context, which is the surrounding information value of the encoding block [6]. The binary arithmetic encoder performs encoding while setting a range using Bin (1-bit binary value) and the probability value of the context modeler. The binary arithmetic encoder performs encoding while setting a range using Bin (1-bit binary value) and the probability value of the context modeler. When the range becomes smaller than the predetermined size 256, renormalization for updating the range is performed, and a bitstream is generated in the renormalization process. Bin performs binary arithmetic encoding in three modes, Regular mode, Bypass mode, and Terminate mode. The Regular mode encodes with the probability values (pStateIdx, ValMPS) generated in the context modeler. The bypass mode does not use the context modeler and encodes Bin with equal probability. The Terminate mode encodes a Bin of a syntax element (end_of_slice_flag) that determines whether a slice is terminated.

2.2. BAE Process

Binary arithmetic encoder takes a probability value from the context modeler and performs binary arithmetic encoding on the current Bin. Figure 2 shows a flowchart of binary arithmetic encoding. qRangeIdx represents the upper 3 bits of ivlCurrRange and is used to generate rLPS (LPS range). rMPS (MPS range) is calculated as ivlCurrRange - rLPS. The rMPS (MPS range) is calculated as ivlCurrRange - rLPS. If the current Bin is an MPS, Update pStateIdx, and if the current Bin is an LPS, update ivlLow and pStateIdx.

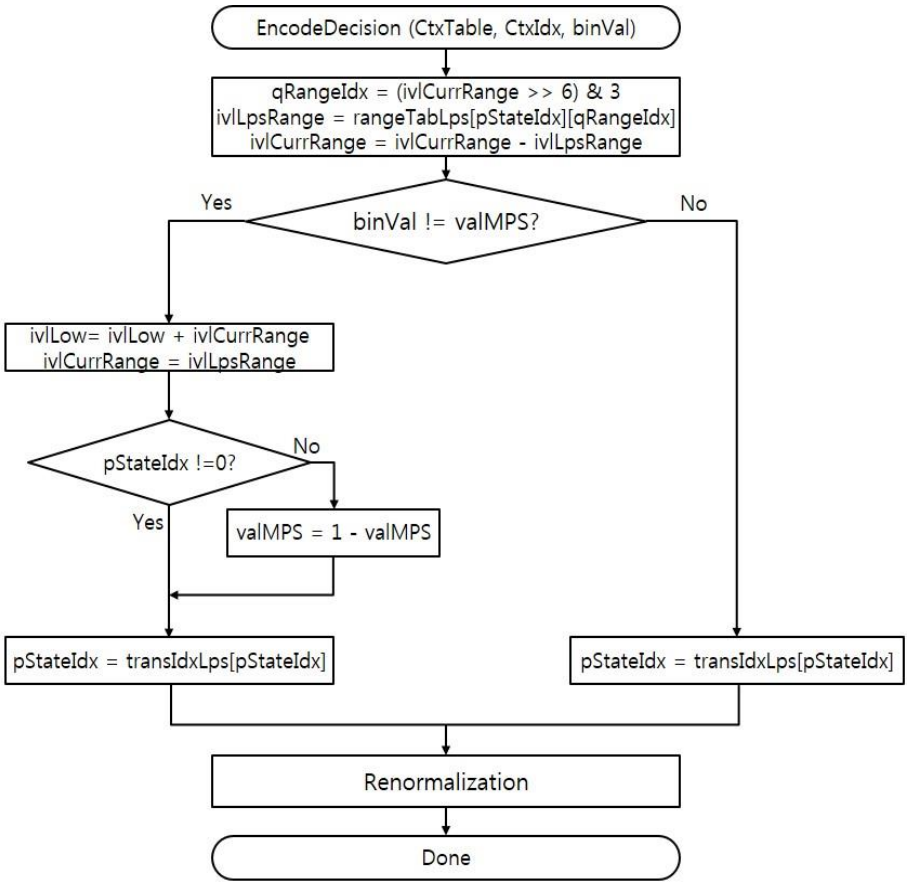

Figure 2. Flowchart of BAE

Figure 3 shows an example of a binary arithmetic encoding process. The binary arithmetic encoding process in the case where the input Bin value (binVal) is '1, 0, 0, 0' and the initial context model value has pStateIdx = 0 and valMPS = 0 is as follows [7]. Since the first input Bin is '1', we compare valMPS with binVal to determine MPS (Most Probable Symbol) and LPS (Least Probable Symbol). Because the current Bin (binVal) and MPS values (ValMPS) are different, the LPS is selected and the LPS range (240) is less than 256, so renormalization is performed. Also, through Update CM (Context Model), valMPS and pStateIdx are updated to the probability model values for the next Bin. CABAC outputs a bitstream whenever it performs renormalization, or updates the ready bit (BitOutStanding) for bitstream output. Binary arithmetic encoding can calculate Low after calculating Range, and can encode next Bin after Range and Low are updated. In addition, the renormalization process can be repeated up to 6 in regular mode, so that the time and output per binary arithmetic encoding process are not constant. Therefore, CABAC binary arithmetic encoder has high data dependency between the current Bin and the previous Bin, which makes parallel processing and high-speed operation difficult in hardware implementation.

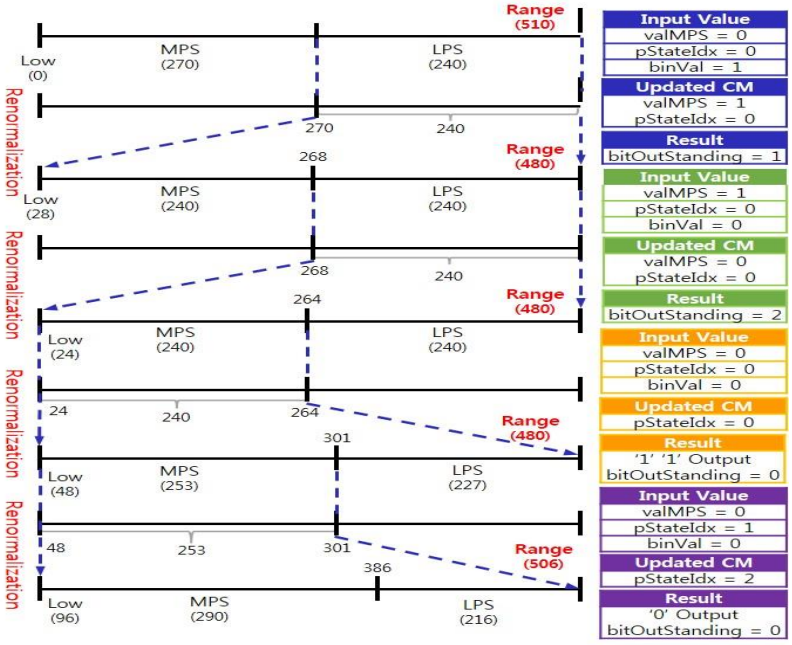

Figure 3. Process of Binary Arithmetic Encode

2.3. Bottleneck in CABAC BAE

The main bottleneck of CABAC is binary arithmetic encoding. Figure 4 and Figure 5 show the pipeline strategy of Single-Bin BAE and Multi-Bin BAE [8-9]. Features of this structure are as follow:

1) Status updates (pStateIdx) and MPS indicators (ValMPS) are pre-executed with context modeling.
2) Updates of range and low, and output of bitstream are processed in separate stages of the pipeline.
3) The output of rLPS (range LPS) is divided into two stages. The first half produces four candidates of rLPS. The second half uses the simpler 4-1 MUX to make the final decision according to bits 7 and 6.

The critical path of CABAC BAE occurs during the renormalization process. The existing algorithm is repeatedly executed until the value of the range becomes 256 or more to occurs a critical path. In this paper, we propose a parallel processing architecture that can reduce the critical path generated in the renormalization process and output the variable bitstream in one clock cycle.

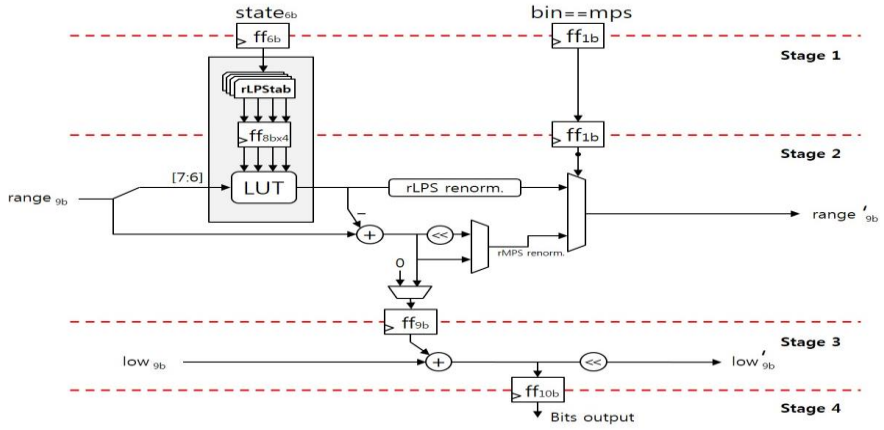

Figure 4. Simplified Architecture of Pipelined Single-Bin BAE

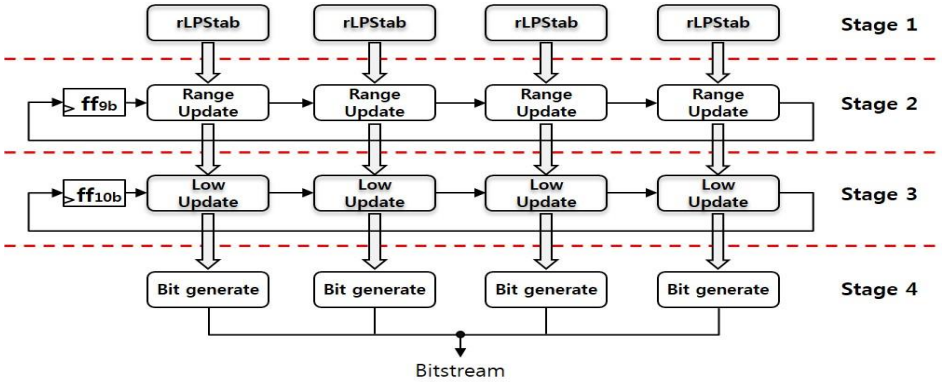

Figure 5. Pipelined Multi-Bin BAE Architecture

3. Proposed CABAC BAE

The function of the proposed CABAC BAE hardware structure is to separate the renormalization and process the conventional iterative algorithm in parallel. The new scheme was designed as a four-stage pipeline structure that can reduce critical path optimally. The existing structure outputs the bitstream through the memory. This structure outputs the number of valid bitstream and bitstream, thereby reducing the hardware area by not using the memory. Figure 6 shows architecture of the proposed BAE.

The proposed BAE generates the information bits necessary to the bitstream output while performing the renormalization. Using the information bits, the bitstream generator can simply output the bitstream. However, the generation of information bits causes a critical path by up to 7 iterative comparisons according to the number of variable used for renormalization. To reduce the critical path, a dedicated LUT can be used to reduce the operational time involved.

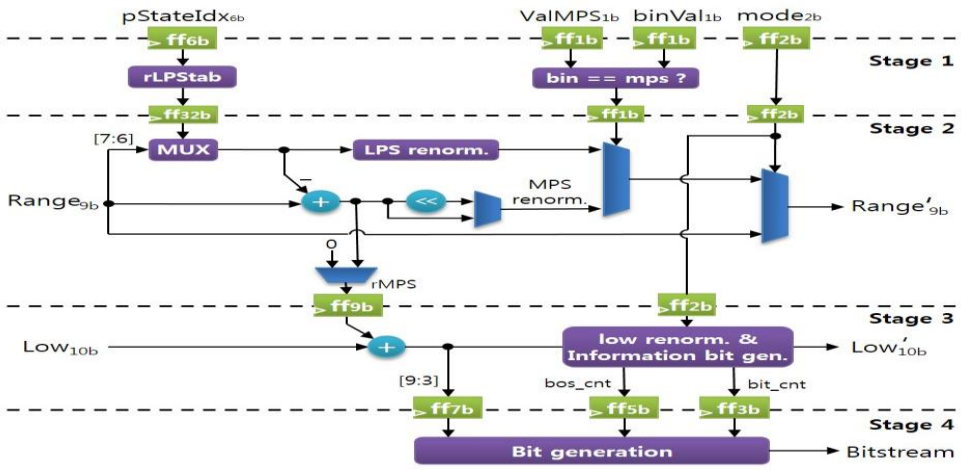

Figure 6. Propose Block Diagram of BAE

3.1. Probable Symbol and rLPS Decide

In Stage 1, MPS and LPS for current Bin are determined as shown in equation (1). Also, as shown in Table 1, four LPS range candidates are output based on the probability value (pStateIdx) from the context modeler.

$$bin = MPS.....if\ (ValMPS = bin)$$
$$bin = LPS.....if\ (ValMPS \neq bin) \quad (1)$$

Table 1. rLPS Table

pStateIdx	qRangeIdx			
	0	1	2	3
0	128	176	208	240
1	128	167	197	227
2	128	158	187	216
…				
61	6	7	9	10
62	6	7	8	9
63	2	2	2	2

3.2. Range Update

Stage 2, performs renormalization when the range of binary arithmetic coding becomes smaller than a certain range, and outputs the number of renormalization (Cnt_RenormE) and the range of MPS (rMPS) required for calculating the low value. If the current encoding Bin is LPS or MPS, then rMPS is equal to (2).

$$\begin{aligned} rMPS &= ivlCurrRange - ivlLPSRange \text{.....} if\ (bin = LPS) \\ rMPS &= 0 \text{.....} if\ (bin = MPS) \end{aligned} \tag{2}$$

The four LPS candidates determine ivlLPSRange with the 7th and 8th bits of the input range (qRangeIdx). If the current encoding Bin is LPS or MPS, then ivlCurrRange is equal to (3).

$$\begin{aligned} ivlCurrRange &= ivlLPSRange \text{.....} if\ (bin = LPS) \\ ivlCurrRange &= ivlCurrRange - ivlLPSRange \text{....} if\ (bin = MPS) \end{aligned} \tag{3}$$

In the regular mode of binary arithmetic coding, existing algorithms generate a critical path by repeatedly performing a maximum of 6 left shift operations until ivlCurrRange becomes 256 or more. In the proposed scheme, to solve the variable operation of renormalization, the renormalization number is calculated by finding the first '1' position from the MSB (Most Significant Bit) of ivlCurrRange, and left renormalization is performed by left shift by the number of iterations. Figure 7 shows the renormalization flowchart of the Range.

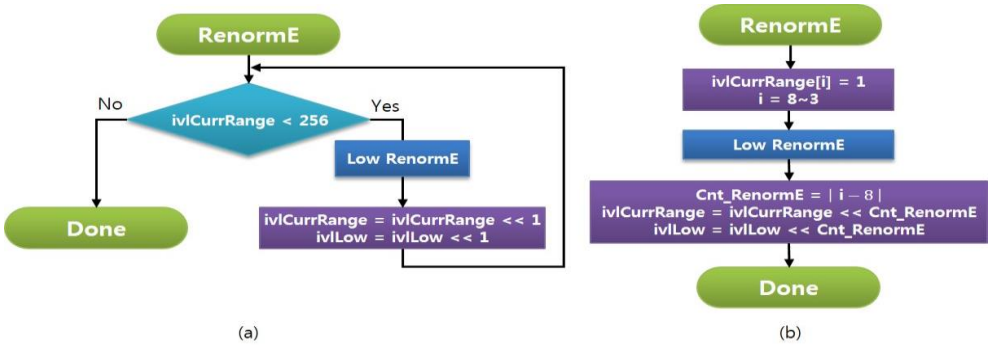

Figure 7. Range Renormalization flow chart, (a) Conventional Renormalization Algorithm (b) Propose Renormalization Algorithm

3.3. Low Update

Stage 3 adds the value of rMPS and low to generate the value of ivlLow, and outputs information bits necessary for bitstream generation with ivlLow and the number of renormalization. If the current encoding Bin is LPS or MPS, then ivlLow is equal to (4).

$$\begin{aligned} ivlLow &= ivlLow + rMPS \text{.....} if\ (bin = LPS) \\ ivlLow &= ivlLow \text{.....} if\ (bin = MPS) \end{aligned} \tag{4}$$

The CABAC encoder generates a bitstream according to the number of renormalization. In the proposed structure, renormalization(B) is performed by left-shifting ivlLow by the number of renormalization as shown in Table 2, and the MSB is set to 0, and renormalization(A) is performed while maintaining the MSB without changing the MSB.

Table 2. Low Renormalization Table According to Number of Renormalization

ivlLow Index	Number of Renormalization (Cnt_RenormE)						
	0	1	2	3	4	5	6
[9]=1 or [0]=0	ivlLow	A	B	B	B	B	B
[9:7]=111 or [7]=0	ivlLow	B	A	B	B	B	B
[9:6]=1111 or [6]=0	ivlLow	B	B	A	B	B	B
[9:5]=11111 or [5]=0	ivlLow	B	B	B	A	B	B
[9:4]=111111 or [4]=0	ivlLow	B	B	B	B	A	B
[9:3]=1111111 or [3]=0	ivlLow	B	B	B	B	B	A
A : ivlLow << Cnt_RenormE B : (ivlLow << Cnt << RenormE) [MSB] <= 0							

The generation of information bits for bitstream output determines bit_cnt (Number of bitstreams to be output) and bos_cnt (Number of bits whose bitstream value has not been determined) according to the number of renormalization and ivlLow. Table 3 shows the output of information bits according to the number of renormalization.

Table 3. Bitstream Information bit Table According to Number of Renormalization

ivlLow [9 : 9-i]			0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Cnt_RenormE(i)	1	bit	1	0	1	1												
		bos	0	1	0	0												
	2	bit	2	1	0	1	2	1	2	2								
		bos	0	1	2	1	0	1	0	0								
	3	bit	3	2	3	1	3	2	3	0	3	2	3	1	3	2	3	3
		bos	0	1	0	2	0	1	0	3	0	1	0	2	0	1	0	0
	4	bit	4	3	4	2	4	3	4	1	4	3	4	2	4	3	4	0
		bos	0	1	0	2	0	1	0	3	0	1	0	2	0	1	0	4
ivlLow [9 : 9-i]			16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
i	4	bit	4	3	4	2	4	3	4	1	4	1	4	2	4	3	4	4
		bos	0	1	0	2	0	1	0	3	0	3	0	2	0	1	0	0
	5	...																
	6	...																

3.4. Bitstream Generation

Stage 4 outputs a bitstream through an information bit for generating a bitstream for the current Bin. The bit generator receives Low_data (upper 7 bits of ivlLow), bos_cnt (Number of bits whose bitstream value has not been determined) and bit_cnt (Number of bitstreams to be output) and outputs a bitstream. The number of bitstream output according to the Bin to be encoded is not constant. The proposed architecture reduces the hardware area by generating an output signal (valid_bit_cnt) indicating the number of variable bitstreams and outputting the bitstream without using the memory. Figure 8 shows the structure of Bitstream Generator and Table 4 shows the table for bitstream output.

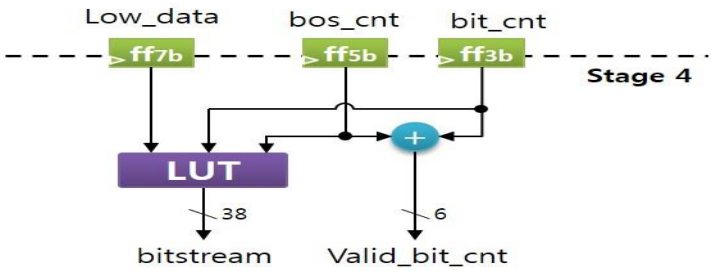

Figure 8. Architecture of Bitstream Generation

Table 4. Table for Bitstream Output

bos_cnt	Bitstream
0	{Low_data[6:0], 0}
1	{Low_data[6], ((1)~Low_data[6], Low_data[5:0], ((30)0)}
2	{Low_data[6], ((2)~Low_data[6], Low_data[5:0], ((29)0)}
3	{Low_data[6], ((3)~Low_data[6], Low_data[5:0], ((28)0)}
...	...
28	{Low_data[6], ((28)~Low_data[6], Low_data[5:0], ((3)0)}
29	{Low_data[6], ((29)~Low_data[6], Low_data[5:0], ((2)0)}
30	{Low_data[6], ((30)~Low_data[6], Low_data[5:0], ((1)0)}
31	{Low_data[6], ((31)~Low_data[6], Low_data[5:0]}

4. Hardware Implementation

4.1. Comparison of Multi-Bin Architecture

Applying the structure for multi-Bin processing to the proposed BAE increases the maximum Bin processing to 2,499Mbin/s. Table 5 shows the gate count, maximum clock frequency, and maximum Bin processing according to Bin per Clock Cycle (BPCC). Figure 9 shows graphs of maximum clock frequency, and maximum Bin processing according to BPCC.

Table 5. Gate Count, Maximum Bin Processing and Maximum Clock Frequency According to BPCC, Synthesized in 65nm

BPCC	1	2	3	4	5	6
Gate Count (NAND gate)	3.17K	5.7K	8.07K	11.2K	13.3K	16.4K
Max. Clock Freq. (Mhz)	1,530	1,110	769	625	500	434
Max. Processing (Mbin/s)	1,530	2,220	2,307	2,499	2,499	2,603

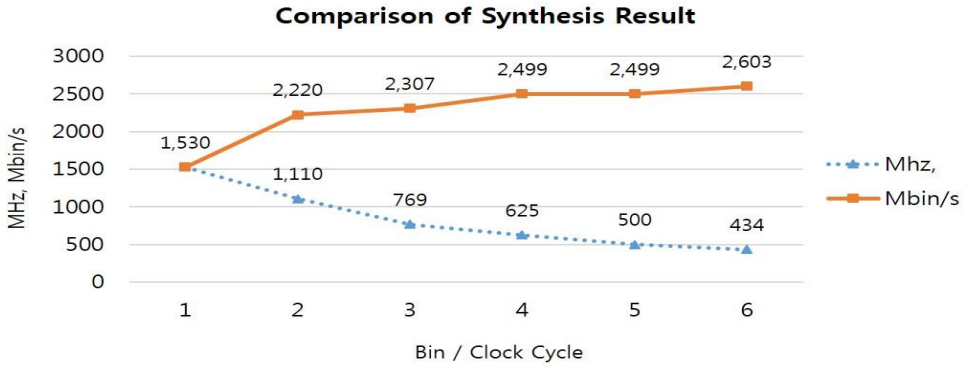

Figure 9. Graphs of Maximum Clock Frequency and Maximum Bin Processing According to BPCC, synthesized in 65nm

4.2. Implementation Results and Comparison with Prior Architecture

In the proposed BAE designed with Verilog HDL. A test vector was created using the HEVC standard model HM16.9. RTL simulation was performed on 1920x1080 and 2560x1600 test sequences for HEVC. It was synthesized in 65nm technology with support from IDEC providing CAD tools. Its gate count is 11.2K and it processes 4 Bins per clock cycle. Maximum clock frequency is 625MHz and the maximum processing time is 2,499Mbin/s. The structure of Zhou [9] performs 4.37 Bins per clock cycle with a maximum clock frequency of 420MHz and 1,836Mbin/s maximum processing time. Maximum processing speed therefore increased by 36% compared to the best performance of existing hardware structure, Zhou [9]. Table 6 compares the hardware implementation results with other structures.

Table 6. Hardware Comparison

	Chen[8]	Zhou[9]	Fei[10]	Peng[11]	Proposed
Throughput (Bin/Clock Cycle)	1.42	4.37	4	1.18	3.99
Max. Clock Freq. (Mhz)	222	420	279	357	625
Technology (nm)	130	90	90	130	65
Max. Processing (Mbin/s)	315	1,836	1,116	440	2,499
Gate Count (NAND gate)	14.7K	-	8.22	24.9K	11.2K

5. Conclusion

The proposed CABAC binary arithmetic encoder hardware structure has a four-stage pipeline structure that can reduce the critical path optimally by separating the renormalization process in order to reduce the computation time and computation amount of the standard binary arithmetic encoding technique. In CABAC, the number of output bitstreams varies depending on the Bin to be encoded. An information bit is generated to output the variable bitstream in one cycle, and the number of valid bitstreams and the bitstream are output using the LUT. The proposed CABAC binary arithmetic encoder hardware structure was verified by comparison with HEVC reference software HM16.9 and verified by Xilinx ISE simulator (ISIM). Designed with Verilog HDL and synthesized in 65nm technology, it has been implemented with about 11,210 gates and has a maximum operating frequency of 625MHz and processes four Bins per clock cycle. Compared with the existing Zhou CABAC encoder hardware structure, the maximum Processing increased by 36%. CABAC is the only entropy coding method adopted in the HEVC standard. This has high compression performance, but it has high computational complexity and high dependency between data, which makes parallel processing difficult. The proposed CABAC binary arithmetic encoder hardware architecture works faster by reducing computational complexity. This improves the performance of the HEVC CABAC Encoder and is applicable to multimedia devices and system for real-time image processing.

Acknowledgments

This paper is a revised and expanded version of a paper entitled [Hardware Architecture of CABAC Binary Arithmetic Encoder for HEVC Encoder] presented at [The Third International Mega-Conference on Green and Smart Technology (GST 2016), Jeju Island, Korea and December 22, 2016]. And, this research was supported by the MSIP (Ministry of Science, ICT and Future Planning), Korea, under the Global IT Talent support program(IITP-2016-R0134-16-1019) and Human Resource Development Project for Brain scouting program(IITP-2016-R2418-16-0007) supervised by the IITP (Institute for Information and Communication Technology Promotion.

References

[1] J.-W. Kim, Y.-H. Kim, B.-H. Choi and J.-H. Park, "Application View for High Efficiency Video Coding," Journal of Broadcast Engineering, vol. 15, no. 4, **(2010)**, pp. 135-145.
[2] G. J. Sullivan, J.-R. Ohm, W.-J. Han and T. Wiegand, "Overview of the High Efficiency Video Coding (HEVC) Standard," IEEE Trans. Circuits Syst. Video Technol., vol. 22, no. 12, **(2012)**, pp. 1649-1668.
[3] V. Sze and M. Budagavi, "High Throughput CABAC Entropy Coding in HEVC," IEEE Transactions on Circuits and System for Video Technology, vol. 22, no. 12, **(2012)**, pp. 1755-1764.
[4] Yongseok Choi and Jongbum Choi, "High-throughput CABAC codec architecture for HEVC", Electronics Letters, vol. 49, no. 18, **(2013)**, pp. 1145-1147.
[5] Hyungu Jo, Gookyi Dennis A.N and Kwangki Ryoo, "Hardware Architecture of CABAC Binary Arithmetic Encoder for HEVC Encoder", Proceedings of the Third International Mega-Conference on Green and Smart Technology (GST 2016), Jeju Island, Korea, **(2016)** December 21-23.
[6] J. Stankowski, D. Karwowski, K. Wegner, O. Stankiewicz, K. Klimaszewski and T. Grajek, "Analysis of CABAC Context Models Efficiency in the HEVC", Proceedings of the 57th International Symposium ELMAR, Zadar, Croatia, **(2015)** September 28-30.
[7] Jeon-Hak Moon, Yoon-Sup Kim and Seong-Soo Lee, "Design of an Efficient Binary Arithmetic Encoder for H.264/AVC," Journal of the Institute of Electronics Engineers of Korea-SD, vol.46, no.12, **(2009)**, pp. 66-72.
[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 Transactions on Consumer Electronics. vol. 56, no. 4, **(2010)**, pp. 2529-2536.
[9] D. Zhou, J. Zhou, W. Fei and S. Goto, "Ultra-High Throughput VLSI Architecture of H.265/HEVC CABAC Encoder for UHDTV Applications," IEEE Transactions on Circuits and System for Video Technology, vol. 25, no. 3, **(2015)**, pp. 497-507.
[10] W. Fei, D. Zhou and S. Goto, "A 1 Gbin/s CABAC encoder for H.264/AVC," in European Signal Processing Conference (EUSIPCO), **(2011)**, pp. 1524-1528.
[11] B. Peng, D. Ding, X. Zhu and L. Yu, "A hardware CABAC encoder for HEVC," in Processing IEEE International Symposium on Circuits and Systems (ISCAS), **(2013)**, pp. 1372-1375.

Authors

Hyungu Jo, received a BS Degree in Semiconductor Engineering from Chongju University, South Korea, in 2014. He is currently pursuing a MENG Degree in Information and Communication Engineering at Hanbat National University, South Korea. His research interests include SoC Design and Verification Platforms, Image Signal Processing and Multimedia Codec Design.

Seungyong Park, received a BS Degree and MENG Degree in Information and Communication Engineering from Hanbat National University, South Korea, in 2010 and 2012 respectively. He is currently pursuing a PhD Degree in Information and Communication Engineering at Hanbat National University, South Korea. His research interests include SoC Design and Verification Platforms, Image Signal Processing and Multimedia Codec Design.

Kwangki Ryoo, received BS, MS and PhD Degrees in Electronic Engineering from Hanyang University, South Korea in 1986, 1988 and 2000 respectively. From 1991 to 1994, he was an Assistant Professor at the Military Academy in South Korea. From 2000 to 2002, he worked as a Senior Researcher at ETRI, South Korea. From 2010 to 2011, he was a Visiting Professor at University of Texas at Dallas. Since 2003, he has been a Professor at Hanbat National University, South Korea. His research interests include Engineering Education, SoC Design and Verification Platforms, Image Signal Processing and Multimedia Codec Design.

