ISSN (Online) 2394-6849

International Journal of Engineering Research in Electronics and Communication Engineering (IJERECE)
Vol 4, Issue 6, June 2017

VLSI implementation of High Throughput HEVC/H.265 CABAC Encoder for UHDTV Applications

[1] Yalagoud A. Patil, [2] Vijaya Prakash A.M.
[1] PG Student, BIT Bengaluru , [2] Professor Dept.of Electronics and Communication
Bangalore Institute of Technology,Bengaluru

Abstract: **Context Adaptive Binary Arithmetic Coding (CABAC) is a technique for entropy coding initially presented in H.264/AVC what's more, now utilized as a part of the most up to date standard High Efficiency Video Coding (HEVC). While it gives high coding productivity, the information conditions in H.264/AVC CABAC make it trying to parallelize and along these lines, restrict its throughput. As needs be, amid the standardization of entropy coding for HEVC, both coding performance and throughput were considered. This paper highlights the key strategies that were utilized to empower HEVC to conceivably accomplish higher throughput while conveying coding increases relative to H.264/AVC. The proposed enhancements including pre-normalization, hybrid way coverage and, lookahead rLPS to diminish the basic way deferral of double Binary Arithmetic Coding (BAE) by abusing the inadequacy of information conditions in rLPS refreshing. The context modeling and binarization segments are likewise upgraded. Therefore, CABAC encoder conveys a normal of 4.37 bins per clock cycle.**

***Index Terms*—Advanced Video Coding (AVC), Context Adaptive Binary Arithmetic Coding (CABAC), Entropy coding, H.264, H.265, High Efficiency Video Coding (HEVC), Super Hi-Vision, Ultra High Definition, Ultra High Definition Television (UHDTV), Very- Large-Scale Integration (VLSI), Video Encoder.**

I. INTRODUCTION

High Efficiency Video Coding (HEVC) is at presently being produced by the Joint Collaborative Team for Video Coding (JCT- VC). It is relied upon to convey up to a half higher coding productivity contrasted with its forerunner H.264/AVC. HEVC utilizes a few new instruments for moving forward coding productivity, including bigger block and transform sizes, extra loop filters, and profoundly adaptive entropy coding. While high coding productivity is vital for lessening the transmission and capacity cost of video, handling speed and range cost likewise should be considered in the improvement of cutting edge video coding to deal with the interest for higher resolution and frame rates.

CABAC is the entropy coding device connected in the most recent video compression standards: High Efficiency Video Coding (H.265/HEVC) and, Advanced Video Coding (H.264/AVC). Contrasted and past entropy coders, CABAC accomplishes strikingly better proficiency yet includes higher intricacy in the in the interval. Particularly, the CABAC calculation forces information conditions on its finest grain level: the receptacle (image) level. This basically confines the level of pipelining and parallelism achievable in hardware implementation, which makes CABAC a known bottleneck of throughput situated video encoders. In the HEVC standard, greatest frame rate is characterized as 800 Mbit/s at level 6.2 high level [3] for 8K UHDTV, which compares to a bin rate (symbol rate) of roughly 1 Gbin/s. In hone, the quantity of bins vacillates intensely between edges also, between coding tree units, in this way a constant CABAC encoder may need to convey a pinnacle bin rate that is considerably higher.

Albeit abnormal state parallelism between various CABAC encoders (e.g., on frame, slice, or wavefront levels) is a decision to acknowledge high throughput, such an answer for the most part experiences memory data transfer capacity overhead, long framework idleness, and expansive equipment cost. Integrating numerous slower CABAC encoders additionally brings down the framework's effortlessness and its adaptability to designs. In this manner, the throughput contributed by a single CABAC encoder is as yet basic.

Very Large-scale integration (VLSI) engineering of CABAC encoder has pulled in a considerable measure of studies in the previous quite a long while. Reference [4] was among the early endeavors to abuse multibin BAE for an execution pick up. [5] actualized a four-arrange BAE pipeline with decreased basic way delay. [6] Enhanced the execution of renormalization in BAE. [7] Outlined a HDTV-arranged CABAC encoder with low equipment cost. [8], [9] proposed

ISSN (Online) 2394-6849

International Journal of Engineering Research in Electronics and Communication Engineering (IJERECE)
Vol 4, Issue 6, June 2017

productive designs of setting displaying and, binarization to guarantee the high usage of BAE. [10] examined approaches that expansion get to proficiency of the setting memory. [11] parallelized two CABAC encoders on the slice level to meet the throughput prerequisite of a 4K UHDTV video encoder chip [12] upgraded the association of setting tables to acknowledge proficient setting displaying in view of Static Random Access Memory (SRAM).[13] proposed an adaptable generator that can produce CABAC encoder outlines for different throughput and cost targets. While all the above advance depends on H.264/AVC [14] and [15] created augmentations that offered support to H.265/HEVC.

II. ANALYSIS AND ENTROPY CODING OF CABAC

The CABAC encoding algorithm comprises of three steps: binarization, context modeling and arithmetic encoding. Fig.1 demonstrates the CABAC encoder block diagram. H.264/AVC characterizes any important data to be encoded by CABAC as a syntax element. In the start of CABAC encoding, the binarization deciphers a non-binary syntax element to a series of bins. At that point, the context modeler peruses in the bin string and creates context a value as per neighboring data of top and left macro-blocks. The context value is an index to the context table worked toward the start of preparing another slice. The context table has 399 entries. The number arithmetic encoder processes bin from the binarization and context a value from the context modeler and yeilds encoded bits.

2.1 Binarization

Character reduction in CABAC is performed by the utilization of a binarization plan to each non-binary syntax element bringing about a one of a kind halfway parallel codeword for a given syntax element, called a bin string.

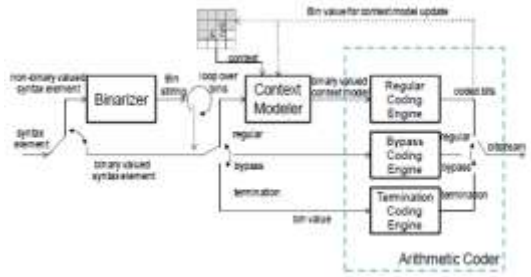

Fig.1: The CABAC Encoder Block diagram

The upsides of this approach are both regarding modeling and implementation.

There are four such fundamental sorts: the unary code, the truncated unary code, the k-th arrange Exp-Golomb code, furthermore, the fixed-length code. Also, there are binarization in view of a connection these basic sorts. As an exemption of these organized sorts, there are five particular, generally unstructured double trees that have been physically decided for the coding of macro-block types furthermore, submacro-block sorts.

2.2 Context Modeling

A standout amongst the most essential properties of arithmetic coding is the probability to use a clean interface between modeling and coding with the end goal that in the demonstrating stage, a model probability distribution is doled out to the given symbols, which then, in the ensuing coding stage, drives the real coding engine to create a sequence of bits as a coded portrayal of the symbols as per the model distribution.Four essential plan sorts of setting models can be recognized in CABAC. The principal sort includes a setting layout with up to two neighboring syntax element in the past of the current syntax element to encode, where the particular meaning of the sort of neighborhood relies on upon the syntax element. The second kind of context models is characterized for the syntax element of mb_type and sub_mb_type. Both the third and fourth kind of setting models is connected to remaining information as it were. The third sort does not depend on past coded information, in any case, on the position in the scanning path. For the fourth sort, displaying capacities are indicated that include the assessment of the aggregated number of encoded levels with a particular incentive preceding the present level bin to encode.

2.3 Arithmetic Encoding

The arithmetic encoder module compromises of three coding engine: the regular coding engine, the bypass coding engine and the termination coding engine.

The principle of binary arithmetic coding is to recursively partition the probability interval. As it gets each new symbol, the present probability interval will be parceled into two sub-intervals. The tag will be refreshed and indicated the lower bound of new subinterval as indicated by the encoding symbol. Once the tag is unquestionably situated in either top or base portion of the interval, a bit will be moved out to bit-stream. In the event that the tag lies in the upper half portion of the interval, a “1” will be created; generally a “0” will be yield. Thusly, the base of the new interval can be followed

ISSN (Online) 2394-6849

International Journal of Engineering Research in Electronics and Communication Engineering (IJERECE)
Vol 4, Issue 6, June 2017

without sitting tight for all symbols coded, in this manner an incremental encoding can be obtained.

2.3.1 Regular Mode

The flowchart of CABAC normal arithmetic coding plot appears in Fig.2. The regular coding engine performs recursive interval subdivision. It takes the context value, current encoding bin, variable Range and variable Low as its information. It gets MPS, pStateIdx from the setting table and gets RangeLPS by turning upward the rangeTabLPS table. The new Range and the new Low are figured in view of whether the encoding binesteem is identical to MPS. Furthermore, the new pStateIdx is refreshed by turning upward the transIdxLPS table and the transIdxMPS table. At long last, the customary coder renormalizes the Range on the off chance that it is too little and yields the encoded bits simultaneously. procedure, one bit at most in the lower bound of the interval that is utilized to represent the condition of the arithmetic encoder is taken off and embedded into the bit-stream. To begin with, the algorithm does not know how often the renormalization loop will run. Second, as this procedure is iterative, it can prompt an inaccessible parallelism. In hardware implements, the renormalization has a tendency to devour different cycles; hence the throughput is altogether diminished. To assault this issue, the renormalization is changed in order to be prepared in a solitary stride. The flowchart of renormalization appears as shown in Fig.3.

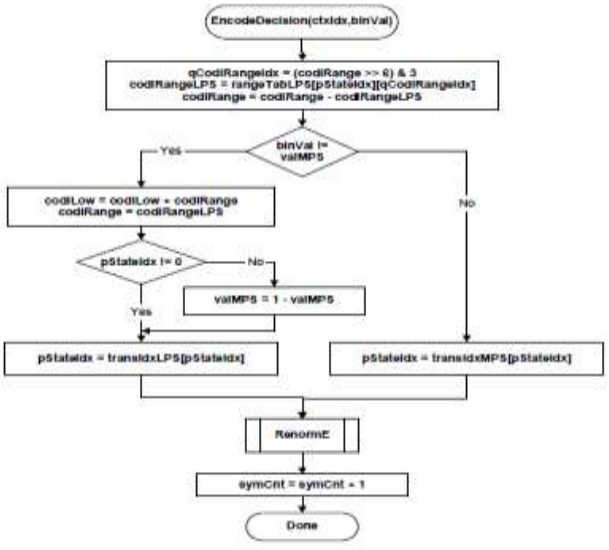

Fig.2: Flowchart for encoding a decision

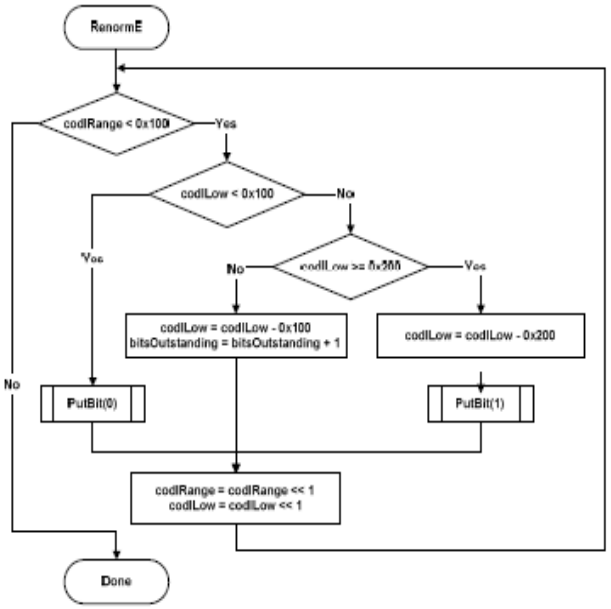

Fig.3: Flowchart of renormalization in the encoder

The renormalization after interval subdivision is required at whatever point the new interval range at no time in the future remains inside its legitimate range [28, 29]. The procedure of renormalization is iterative. Amid each time of cycle of this

The Fig.4 is flow chart when there is a yield bit happened by renormalization. The data dependencies are high since it is connected with renormalization, and this loop works. At the point when there is a yield bit happened by renormalization with

'0' or '1', the value is quite just output and if there comes to be bitsOutstanding which has been gathered up until this point, it yields the inverse estimation of the output first bit. This loop is rehashed till bitsOutstanding moves toward becoming '0', and it returns in a renormalization circle again if this stage is over.

2.3.2 Bypass Mode

The symbol of same appearance recurrence is encoded by bypass mode. The Binary arithmetic engine is utilized to accelerate the encoding procedure if there should arise an occurrence of Range -

RangeLPS ≒ RangeLPS ≒ Range/2. Since

bypass mode encode it without utilizing

ISSN (Online) 2394-6849

International Journal of Engineering Research in Electronics and Communication Engineering (IJERECE)
Vol 4, Issue 6, June 2017

probablity, it is most certainly not packed, however effectiveness of the operation makes strides. Fig. 5 demonstrates flowchart of encoding bypass.

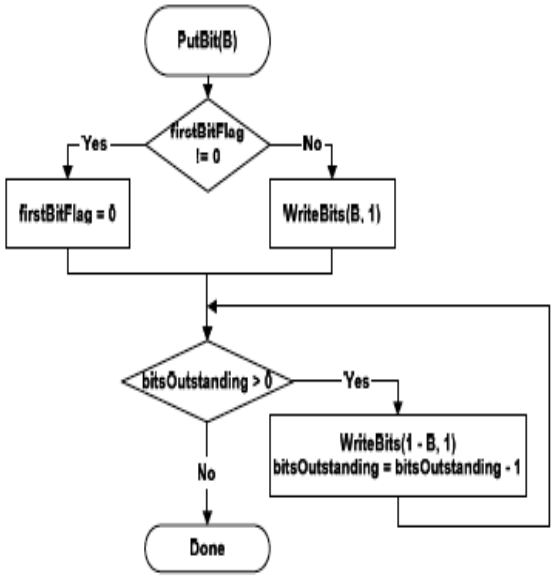

Fig 4: Flowchart of Put Bit

2.3.3 Termination Mode

CABAC encodes it to slice unit. One slice has syntax element of numerous sorts and when syntax element of the last of the slice is encoded, it is encoded by termination mode. Since it implies that it is not the finish of the slice if a symbol of last syntax element is '0', it is equivalent, and it is processed with standard mode and in light of the fact that it is the end of the slice if an symbol is '1', it is dealt with by flush mode. Flush mode yields encoded last Low and stop bit. Fig. 6 demonstrates flowchart of encoding end.

III. THE PROPOSED ARITHMETIC ENCODER

The subject of worry for execution upgrades of CABAC enhances sporadic renormalization and beat output of binary arithmetic encoder. This paper proposes the Binary arithmetic encoder which enhanced the existing complex renormalization and bit output. Despite the fact that there must be a Low and a Range to encode a next input symbol as indicated by the algorithm to propose in a standard report, this value can comprehend that a standardization procedure is over.

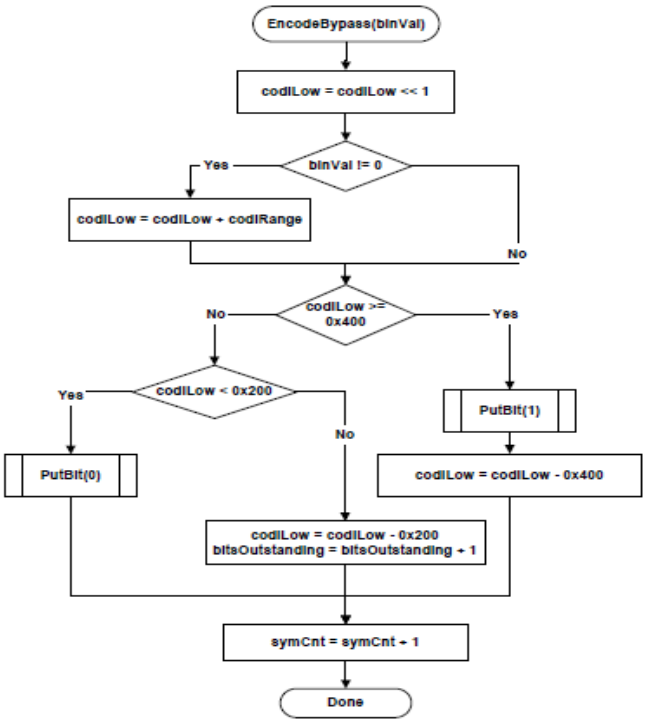

Fig 5: Flowchart of encoding bypass

Notwithstanding, the iteration number of the standardization procedure is not fixed. In this manner, there is the issue that the clock cycles which is important to encode a symbol comes to be distinctive, and the yield is not general.

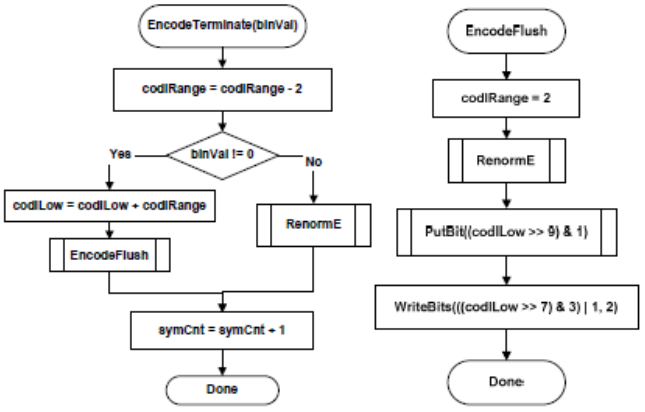

Fig 6: Flowchart of encoding a decision before termination

ISSN (Online) 2394-6849

International Journal of Engineering Research in Electronics and Communication Engineering (IJERECE)
Vol 4, Issue 6, June 2017

The proposed design let bit generator and renormalization module end up noticeably free and worked by pipeline design. Therefore, at each clock cycle, the input symbol is encoded paying little mind to the emphasis of the renormalization procedure. Since proposed design took information reliance between each stage and expelled it, and an aggregate engine worked by a pipeline, the handling of information turned out to be early. Fig. 7 shows architecture of proposed arithmetic coder. The arithmetic encoder comprises of three coding modes: regular, bypass and termination mode. It was intended to hardware without processor. Besides, termination mode produces a stop bit and a stuffing bit in bit generator after it was worked. The bit generator saves data of arithmetic encoding in FIFO and the saved data create a last yield stream by FSM as Fig.8.

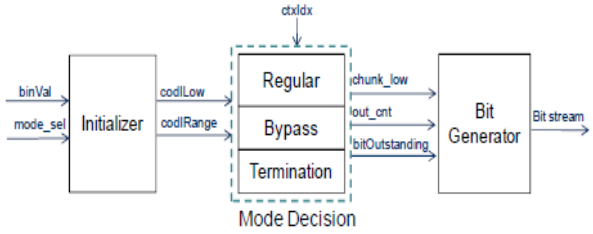

Fig7. Architecture of proposed arithmetic coder

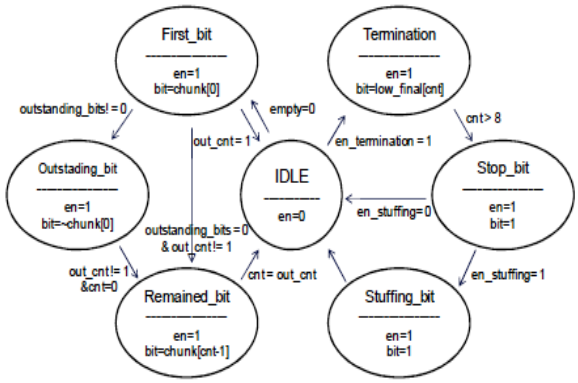

Fig 8. FSM state diagram of bit generator

IV. IMPLEMENTATION RESULTS

Our simulation based on the H.264 reference software JM 9.8 [3]. The proposed arithmetic encoder for CABAC was designed and implemented in VHDL and simulated using ISE simulator in Xilinx Spartan 6E. Table 1. shows comparison of reference software output with HDL output for the throughput in various cases. To verify for the proposed arithmetic encoder, we extracted test vector of input and output with reference software. And we confirmed a correct operation in comparison with the extracted output file from HDL. Furthermore, the renormalization processes the input binary symbol for every clock. And output bit was generated by FSM of bit generator. The same bit stream is checked using HEVC analyser.

TABLE 1: PROPORTION OF BINS BY TYPE (LOW DELAY AND RANDOM ACCESS)

Sequence	P_{MPS}	P_{LPS}	P_{BYPASS}
Anchor	53.7%	17.8%	28.0%
Jellyfish	55.8%	21.2%	23.5%
BQMall	50.7%	19.8%	29.8%
RandAccess	52.6%	18.5%	29.4%
Total	53.2%	19.3%	27.7%

Table 2 demonstrates that the rate of setting coded containers under basic conditions is lower for HEVC than H.264/AVC. Table 3 likewise demonstrates that in the most pessimistic scenario conditions, there are 8× less setting coded containers in HEVC than H.264/AVC. The lessening in setting coded bins is principally credited to the adjustments to coefficient level and movement vector distinction.

TABLE 2: DISTRIBUTION OF CONTEXT CODED, BYPASS AND TERMINATION BINS

	Common condition configuration	**Context**	**Bypass**	**Term**
H.264	HierB	79.5	13.2	5.4
	HierP	79.6	12.3	6.3
H.265	AI_MAIN	68.0	31.8	0.1
	LP_MAIN	77.6	21.0	1.0
	LB_MAIN	78.1	21.1	1.0
	RA_MAIN	72.9	25.8	0.8
	AI_HE10	68.2	32.4	0.1
	LP_HE10	78.5	21.5	1.0
	RA_HE10	73.9	26.5	0.7

ISSN (Online) 2394-6849

International Journal of Engineering Research in Electronics and Communication Engineering (IJERECE)
Vol 4, Issue 6, June 2017

TABLE 3: MEMORY REDUCTION AND WORST CASE BIN IN HEVC OVER H.264

Metric	H.264	H.265	Reduction
Max context coded bins	7804	940	8x
Max bypass bins	13057	13426	1x
Max total bins	20862	14365	1.5x
Number of contexts	446	153	3x
Line buffer for 4k x k	30721	1537	20x
Coefficient storage	8x8x9-b	4x4x4-b	9x
Initialization table	64033	3537	18x

V. CONCLUSION

Entropy coding was a highly active area of development throughout the HEVC standardization process with proposals for both coding efficiency and throughput improvement. The tradeoff between the two requirements was carefully evaluated in multiple core experiments and ad hoc groups. Many techniques were used to improve throughput, including reducing context coded bins, grouping bypass bins together, grouping bins that use the same contexts together, reducing context selection dependencies, and reducing the total number of signaled bins. CABAC memory requirements were also significantly reduced. A summary of the throughput techniques and related contributions are found in Table 3. The final design shows that accounting for implementation cost when designing video coding algorithms results in a design that can maximize processing speed and minimize area cost, while delivering high coding efficiency in the next generation video coding standard.

REFERENCES

[1] Parameter Values for Ultra-High Definition Television Systems forProduction and International Programme Exchange, document ITU-RBT.2020, 2012.

[2] V. Sze and M. Budagavi, "High throughput CABAC entropy codingin HEVC," IEEE Trans. Circuits Syst. Video Technology., vol. 22, no. 12,pp. 1778–1791, Dec. 2012.

[3] High Efficiency Video Coding. document ITU-T H.265/ISO/IEC 23008- 2HEVC, 2013.

[4] Yoonsup Kim and Jeonhak Moon et al.,"Design of High Performance Arithmetic Encoder for CABAC in H.264/AVC", Proceedings of the 8th WSEAS International Conference on Microelectronics, Nanoelectronics, Optoelectronics.

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

[6] Y.-H. Chen et al., "An H.264/AVC scalable extension and high profile HDTV 1080p encoder chip," in Proc. IEEE Symp. VLSI Circuits (VLSIC), Jun. 2008, pp. 104– 105.

[7] Z. Liu and D. Wang, "One-round renormalization based 2-bin/cycle H.264/AVC CABAC encoder," in Proc. 18th IEEE Int. Conf. Image Process. (ICIP), Sep. 2011, pp. 369–372.

[8] J.-L. Chen, Y.-K. Lin, and T.-S. Chang, "A low cost context adaptive arithmetic coder for H.264/MPEG-4 AVC video coding," in Proc. IEEE Int. Conf. Acoust. Speech Signal Process., Apr. 2007, pp. 105–
108.

[9] L.-C. Wu and Y.-L. Lin, "A high throughput CABAC encoder for ultrahigh resolution video," in Proc. IEEE Int. Symp. Circuits Syst. (ISCAS),May 2009, pp. 1048–
1051.

[10] 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. Consum. Electron., vol. 56, no. 4, pp. 2529–2536,Nov. 2010.

[11] X. 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, Sep.
2009.

[12] L.-F. Ding et al., "A 212 MPixels/s 4096 × 2160p multiview video encoder chip for 3D/quad full HDTV

applications ," IEEE J. Solid-StateCircuits, vol. 45, no. 1, pp. 46–58, Jan. 2010.

[13] W. Fei, D. Zhou, and S. Goto, "A 1 Gbin/s CABAC encoder forH.264/AVC," in Proc. Eur. Signal Process. Conf. (EUSIPCO), Sep. 2011,pp. 1524–1528.

[14] D. Marpe, H. Schwarz, and T. Wiegand,Context-Based Adaptive Binary Arithmetic Coding in theH.264/AVC Video Compression Standard, IEEETransaction on Circuits and Systems for VideoTechnology, Vol.13, No.7, pp.620-636

[15] H.264/AVC Reference Software, http://iphome.hhi.de/suehring/tml, ver. JM9.8