A 1 Gbin/s CABAC Encoder for H.264/AVC

Wei Fei, Dajiang Zhou, and Satoshi Goto

Graduate School of Information, Production and System LSI, Waseda University
2-7 Hibikino, Wakamatsu-ku, Kitakyushu, Fukuoka, 808-0135, Japan
phone: +81-080-4272-2462, e-mail: feichongwei@moegi.waseda.jp

ABSTRACT

In this paper, we propose a 1 Gbin/s context-based adaptive binary arithmetic coding (CABAC) encoder architecture for beyond-HDTV applications. CABAC is a crucial part in H.264/AVC main and high profiles that provides a great compression ratio at the expense of high computational complexity. And it is also considered as a very efficient coding method in the developing high-efficiency video coding (HEVC) standard. We try to accelerate the CABAC encoder to provide a high throughput to meet the requirement of beyond-HDTV video. Our design includes the binarization, context modeling and binary arithmetic encoding (BAE) parts and achieves a throughput of 4 bins per cycle. The synthesis result using SMIC 90nm shows that the logic gate count is 36.2K in all and the encoder engine can work at a maximum frequency of 279MHz. Thus the overall throughput can reach over 1 Gbin/s. In our design, the 460 contexts are assigned to 6 SRAMs to attain efficient context modeling. And the binarization part is also optimized to enhance the throughput with low hardware cost.

1. INTRODUCTION

Context-based adaptive binary arithmetic coding (CABAC) is introduced in H.264/AVC video coding standard as an efficient entropy coding tool. It is remarkable for providing a better compression ratio than most other encoding algorithms. For example, it achieves an average more 9%-14% of bit rate saving than the baseline entropy coding method in H.264, CAVLC. And CABAC is still the efficient entropy coding method considered in the developing High-Efficiency Video Coding (HEVC) standard, which is targeted at next-generation HDTV and beyond-HDTV displays.

On the other hand, CABAC requires a significant amount of logic and increases the hardware implementation cost on a large scale. Due to the syntax-element-level dependency as well as bit-level dependency, it is also difficult to parallelize, thus the throughput of a single CABAC encoding engine is limited. With the emergence of QFHD (2160p) and Super Hi-Vision (4320p) video, the throughput requirement is much higher and the real-time CABAC encoder engine is much harder to design. The bin rate for HDTV (1080p) is 40 Mbps at most. As to the QFHD and Super Hi-vision data, it can reach a very high bin rate of above 1 Gbin/s. This makes the design of CABAC encoder very challenging and crucial in the whole video encoding system.

In recent years, many papers have presented CABAC encoder architectures for throughput accelerating and power saving. Some designs are for 1 bin/cycle, such as [3-5] and [8] while some adopt multi-bin architecture, like [6], [7], [9] and [10].

It can be seen from the trend that the designs are transferring from 1 bin/cycle to multi-bin/cycle to provide a higher throughput. The works of [3] and [8] try to enlarge the working frequency as much as possible by designing the encoder with maximum pipelining. And the synthesis results using 0.13 CMOS technique indicate a max frequency of around 600Mbin/s at around 600 MHz. While this is still not enough for the throughput requirement, and the high clock frequency also brings difficulties in applying these architectures in consumer electronics.

But many problems come out with multi-bin architecture. Firstly, the critical path becomes longer. The BAE logic is duplicated and cascaded sequentially to process multiple bins in one cycle. Even though efficient pipeline is employed, the critical path increases and the maximal frequency decreases substantially. For example, the maximal frequency of 1 bin/cycle BAE is 625 MHz in [2], while 344 MHz and 192 MHz respectively for 2 and 4 bin/cycle BAE. Secondly, SRAM based context modeling has very low efficiency to provide multi-bin/cycle. In [7] and [10], CABAC is designed as 2.37 bin/cycle, while the actual throughput is only 1.3 and 1.42 due much to the inefficient context table access. At last, to achieve good throughput, several CABACs may be used sometimes. CABAC in [6] uses two BCMs (binarization and context modeling) and one 4 bin/cycle BAE to achieve an overall throughput of 4bin/cycle. And CABAC in [9] applies four BCMs and two 2 bin/cycle BAE to achieve 3.88 bin/cycle and better working frequency. This will surely increase the hardware cost on a large scale. Furthermore, the multi-engine solution either requires a frame to be partitioned into slices (slice parallelism) which degrades coding efficiency, or results in a longer encoding delay by employing frame parallelism [11].

In order to meet the throughput requirement and save hardware at the same time, a 1Gbin/s CABAC encoder is proposed. In this work, 4-level pipeline is utilized in a 4 bin/cycle BAE to enlarge the throughput of this part. The 460 contexts are assigned to 6 banks of SRAMs and the SRAM reading and updating control logic are developed to

This research was supported by CREST, Japan Science and Technology Agency.

achieve high context modeling efficiency as much as 4.5 bin/cycle on average. The binarization part is also optimized to enhance the throughput.
The rest of the paper is organized as follows. Section II provides an overview of CABAC encoding mechanism and previous design works. Section III introduces the proposed CABAC encoder architecture. Then, the synthesis result and comparison are given in Section IV. And finally the conclusion is drawn in Section V.

2. OVERVIEW OF CABAC ENCODER

CABAC in H.264/AVC can be described as three major steps: 1) Binarization, 2) Context Modeling (CM) and 3) Binary Arithmetic Encoding (BAE). Fig. 1 shows the three stages in H.264 standard.

2.1 Binarization

In the binarization stage, the non-binary valued syntax elements (SE) are mapped to a corresponding binary string. There are five kinds of binarization schemes defined in H.264: Unary (U), Truncated Unary (TU), Exp-Golomb (EGk), Fixed-length (FL) and Unary Exp-Golumb (UEGk). Some kinds of SEs have their special way of binarization. The output of the binarization process is the mapped bins of the SEs and the context indexes for the bins which are used in the context modeling stage to fetch the right probability mode.
Since binarization has no problem in providing a throughput of 2 bin/cycle on average, the previous works focus not much on this part. And sometimes several binarization units are applied to generate good throughput, such as [6] and [9].

2.2 Context Modeling

There are three modes in the binary arithmetic encoding stage: regular mode, bypass mode and final mode. In the regular encoding mode, a context model referred to by context index is used to assign a proper probability mode for each bin. The probability mode is used to encode the bin in the BAE stage and the context model for this bin should also be updated according to the bin's value. When bypass mode is selected, a predefined equal probability model is used to process the bin. And in the final mode, the bin is predicted to be 0 with the highest probability.
CABAC in H.264 adopts 460 context models to attain accurate probability estimation, and each context model is made up of one {MPS, pStateIdx} pair. Due to the large memory area to store these context models, SRAM other than register is usually chosen to implement the context table storing the context models, such as [9] and [10]. The synthesized area of context modeling and BAE implemented by SRAM is 29%-35% compared with register based one according to [2].
Data dependency exits in this part because the context modeling of the current bin may use the updated context of the previous bins. So logic should be developed to take care of the hazard cases.

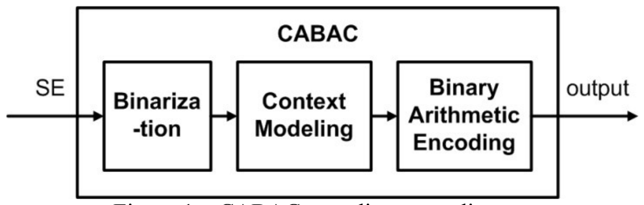

Figure 1 – CABAC encoding steps diagram

Table 1 – Updating of the interval

Prediction correctness	Update of Range	Update of Low
Miss(Bin!=MPS)	rLPS	Low+Range- rLPS
Hit(Bin==MPS)	Range- rLPS	Low

In one bin/cycle architectures, a two-port SRAM (one read and one write ports) is used to perform context reading and updating at the same time, and full pipeline design can be easily employed, such as [5]. And simple control scheme like forwarding can be applied in the pipeline to remove the data hazard caused by the data dependency. Thus steady throughput of one bin/cycle in the pipeline is easy to achieve except for the SRAM initialization process.
While for multi-bin/cycle designs, context access becomes more than 1 bin/cycle. As a result, one two-port SRAM can't meet the reading and updating speed requirement. It is simulated in [2] that the context table can be subdivided into several groups and stored in different SRAMs to add the SRAM ports. And better performance can be obtained if more SRAMs are used. In [10], the context table is divided into 2 banks and 1.42 bin/cycle on average is achieved with 2 bin/cycle BAE.

2.3 Binary Arithmetic Encoding

BAE is used to encode the bins based on their probabilities. The principle is a recursive selection of sub-division of an interval. The interval is set as [0, 1] originally, then divided into two according to the probability mode, and updated to be one of the two intervals based on the correctness of the prediction. Thus the encoded stream can be decoded using this interval information.
The interval is defined by the low bound (Low) and length (Range). The updating of the interval (Low and Range) is defined in Table 1, where the value of rLPS is indexed by pStateIdx read from context modeling. As the encoding process goes on, the value Low become more and more precise, and range is always kept within [0xl00, 0xlFF] by shifting the new Range value. The shift operation is called renormalization. The bits of Low are also shifted out little by little and packed as the final bit stream.
The update process of the value Range and Low and the bit packing process can be pipeline without stall, most designs adopt 4-stage pipeline in this part. And as Fig. 2 shows, several units are cascaded to form multi-bin/cycle architecture. The timing performance is related to the cascade levels. The relationship between the throughput and the cascade levels is shown in Fig. 3, where the synthesis utilizes SMIC 90 nm.

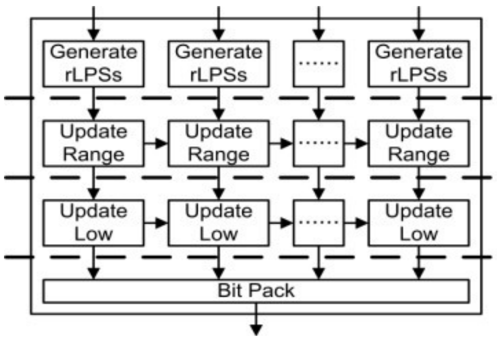

Figure 2 – Pipeline architecture of multi-bin BAE

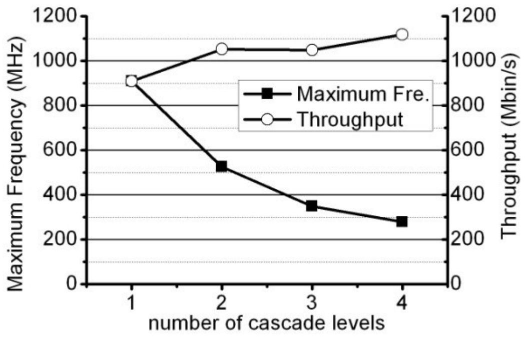

Figure 3 – The relationship between throughput and cascade levels

3. PROPOSED CABAC ENCODER ARCHITECTURE

Fig. 4 depicts the block diagram of our proposed CABAC encoder architecture. It is composed of a binarization unit, a 6-SRAM-based context modeling unit, a 4-stage pipelined BAE and two parallel-in-serial-out (PISO) modules.

3.1 4 bin/cycle binary arithmetic encoder (BAE)

The pipeline architecture describe in Fig. 2 is used in our design of BAE to enlarge the throughput as much as possible. Four Update Range and four Update Low logic units are cascaded to achieve a throughput of 4 bin/cycle in BAE part.

The first pipeline stage in BAE is to generate the four rLPS values for each bin according to the value of pStateIdx from the context modeling module. One of these four values is selected out in Update Range stage and will be used as the rLPS value in Table 1 for interval update. The Update Range stage will update the value of Range four times for the four bins in a cycle and generate the information needed for the update of Low. Then Low is updated in the next stage and the Bit Pack stage will pack the output bit of Update Low into the output stream.

3.2 SRAM based context modeling (CM)

To enhance the throughput of CABAC, the design of CM should also provide a throughput of over 4 bin/cycle on average. 6 two-port SRAMs are used in our design to store the 460 contexts to make efficient context access possible. The 460 contexts are divided into 6 groups according to the feature of context access order.

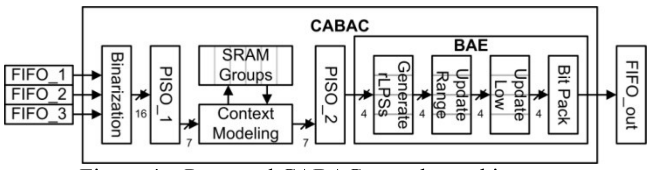

Figure 4 – Proposed CABAC encoder architecture

Table 2 – Examples of group of contexts

SRAM groups	1	2	3	4	5	6
mb_type (SI)	3-5	6	7	8	9	0-2
transform_size		399				
pre_mode			64-66	67	68	69
coded_block_flag (16x16 DC)					85-88	
significant_ flag (16 x16 DC)		106…		107…		105…
last _flag (16 x16 DC)	166…		167…		168…	
abs_level_minus1 (16 x16 DC)(first bin)	229	230	231	227		228
abs_level_minus1 (16 x16 DC)(other bins)	233	234	235		236	232
ref_idx	56	57	58	59	54	55
Mvd	40	41	42	43	44	45-46

The throughput of CM depends much on the grouping of the contexts. In bad grouping, if two adjacent different contexts are store in the same SRAM, only the context for the first bin can be read out, because SRAM has only one read port. Good grouping will try to avoid this kind of hazard. CABAC in [2] groups the contexts based only on the context index (CtxIdx). If they group the contexts into b banks, they use "CtxIdx % b" to determine the group. The design of [5] divides the contexts into 2 banks. The CtxIdxs of sig_coeff_flag and last_sig_coeff_flag types are separately assigned into two banks. And for other CtxIdxs, they assign them to "CtxIdx % 2"-th bank.

In our design, the contexts are grouped by taking the access feature of each context index into consideration. As Tabel 2 shows, 0,1and 2 are assigned to group 6, because only one of these three will be used for one MB and they will never be required in the same clock cycle. Contexts indicated by 6, 7, 8, 9 are assigned respectively to group 2 to group 5 because they may be accessed one following another.

As for the SE ref_idx and mvd, their contexts are almost divided equally among the 6 groups. And the contexts of sig_coeff_flag for one coding block (for example, 16x16 DC) are divided into 3 groups, while those of last_sig_coeff_flag for the same coding block are assigned to the other 3 groups.

Usually the contexts of the same SE type have different probabilities to be visited. The assignment of the more frequently visited contexts should better be given prior consideration. For example, context 227 and 236 are more frequently used than the other contexts of the same SE type, so they are staggered in the context table. The grouping of the contexts is almost balanced.

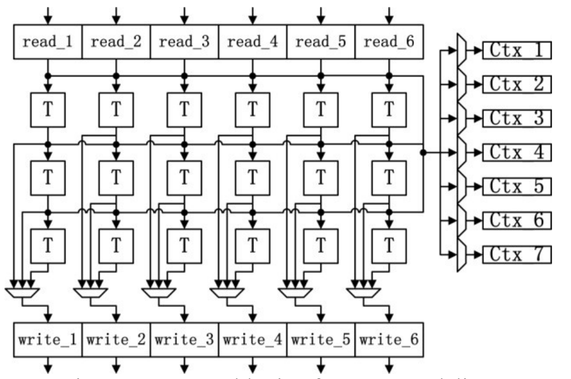

Figure 5 – Control logic of context modeling

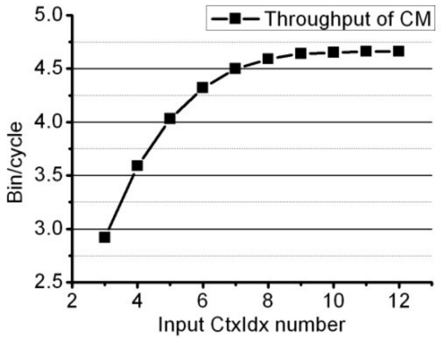

Figure 6 – Different throughput with different input numbers

Table 3 – Examples of binarazation combination

SE combination	Average bin/cycle
2 pre_mode_luma	5.15
pre_mode_chrome & cbp_luma	5.96
Regular coeff & ± 1 coeff	5.12

Every cycle, six contexts can be read out from SRAMs at most. But in the case the same context is accessed again, we take it that the context is legal after transformation. As Fig. 5 shows, read_1 to read_6 are the read out contexts, and each context is transformed for three times. The 18 contexts made up of the read out contexts, the contexts after first and the second transformation are used as the candidates for the context of the corresponding bins. And the 3 contexts after transformation of each read out context are used as the candidates for the updated context to be written back to SRAM. A same forwarding technique as [9] is utilized for each SRAM to update the contexts and remove the data hazard.
The average throughput of CM is affected by the defined maximum input bins. Fig. 6 shows the relationship between the throughput of CM and the input CtxIdx numbers. And in our design, the maximum input number is set to be 7.

3.3 Binarization unit

As Fig. 6 shows, if the throughput of binarization is 5 bin/cycle on average, the throughput of CM can only reach a little higher than 4 bin/cycle. Due to the imbalance of the output of binarization, the throughput of CM is always worse than the expectation from Fig. 6. As a result, binarization should be properly accelerated.

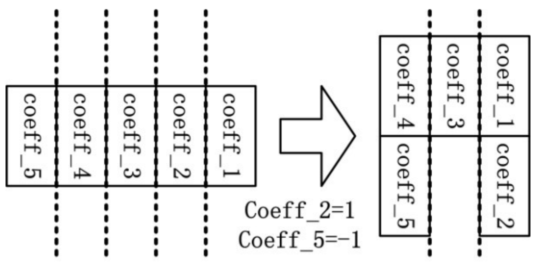

Figure 7 – Examples of saving clock cycles by combining coefficients

We accelerate the bin generation speed by combining some SE types together for binarization. Firstly, we simulate the average bins of each type of SE. Then we combine the process of some SE types together and use FSM to do the control. Table 3 shows some examples of combination.
Since coefficient with absolute value equal to 1 appears frequently, and the binarization result contains only 2 bins, the binarization of these coefficients is combined with a regular coefficient in binarization. As Fig. 7 shows, if there are 5 coefficients to be processed and the absolute values of coefficient 2 and 5 are both 1, it will take 5 clock cycles to do the binarization originally, while in our design, it takes only 3 clock cycles. Thus the throughput can be increased by saving clock cycles for coefficients.
As Figure 4 shows, the input of the binarization is defined as three FIFOs. The first is used to transfer MB information such as MB type, coded block pattern and delta_qp. The second is for the prediction mode, reference index and significant coefficient flags. And the third is for the residual coefficients. The FIFO input architecture will make the whole video encoder easier to pipeline.

3.4 Design of two PISOs

As shown in Fig. 4, two PISOs are used to connect the output of the predecessor stage and provide relatively stable output.
PISO_1 is used to store {bin, CtxIdx} pairs. CtxIdx is made up of 9 bits, 3 to indicate the SRAM group, and 6 for the position in the SRAM. Thus a {bin, CtxIdx} pair contain 10 bits. The input port is designed as 16 bins from binarization stage. The output port is set as 7 bins, because the control logic will become the critical path when the output port is larger than 7. The depth is defined as 20, which is appropriate according to the simulation. The PISO_1 contains logic to decide how much bins will be sent to CM part in the cycle, tell the relationship of the bins to be sent out, and generate the control information for the CM part.
PISO_2 is used to store {bypass, bin, MPS, pStateIdx} groups. Each pair contains 9 bits. The input port is 8. The output port is 4. And the depth is chosen as 10 by simulation.

4. SYNTHESIS RESULT AND COMPARISON

Our design is synthesized using SMIC 90nm library. The synthesis result is shown in Table 4.

Table 4 – The Synthesis Result

Module Name	Gate Count	Critical Path	Throughput
binarization	9.18K	<3.58ns	5.6 bin/cycle
PISO_1	7.49K	<3.58ns	7 bin/cycle
CM	11.07K	<3.58ns	4.5 bin/cycle
PISO_2	0.28K	<3.58ns	4 bin/cycle
BAE	8.22K	3.58ns	4 bin/cycle

Table 5 – Comparison with Other Works

	Synthesis Library	Bins /cycle	Max. Fre. (MHz)	Logic Gate Count	Through-put (Mbin/s)
[3]	0.13μm	1	620	27.5K	620
[4]	0.13μm (TSMC)	0.67	200	34.3K	134
[5]	0.18μm (SMIC)	1	312 CM&AE	14K CM&AE	312
[7]	0.13μm (TSMC)	1.3	222	45K	288.6
[8]	0.13μm (TSMC)	1	578	44.6K	578
[10]	0.13μm (TSMC)	1.42	222	45.9K	315
This	90 nm (SMIC)	4	279	36.2K	1116

Our designed BAE part is targeted at 4 bin/cycle, and the binarization, CM and PISO parts can provide a higher throughput to enhance the throughput of the CABAC encoder. And the design of these parts has no effect on the timing performance because their critical paths are all shorter than BAE. Our designed PISO_1 part and CM part occupy about a half of the circuit area. This is because complex control logic is adopted here to maintain efficient context access.

In Table 5, our design is compared with others in terms of gate count, throughput, etc. Our design can achieve 4 bin/cycle. With the working frequency of 279 MHz, 1 Gbin/s can be provided with only one CABAC engine.

There are several reasons for the throughput improvement in comparison with the other works besides the better synthesis library. Firstly, we enlarge the maximum throughput by utilizing multi-bin per cycle architecture. As it is shown in Fig. 2, the throughput of 4 bin/cycle BAE can provide a 22.9% larger throughput than 1 bin/cycle. [3] and [8] use 1 bin/cycle, so the throughput of the whole CABAC is restricted by BAE and can only reach a throughput of 620Mbin/s though efficient pipeline is applied. Secondly, our design implements the context table using 6 SRAMs to provide efficient context modeling. We make full use of the context access features to group the contexts. Most the grouping work now only divides the contexts into two groups. This is not efficient, because when conflict occurs, only 1 context can be read out and 50% of the throughput is lost. And finally, our proposed work develops high speed binarization and efficient control logic to maintain full use of the throughput of CM and BAE. [7] and [10] are targeted at 2.37 bin/cycle while the actual performance is only 1.3 and 1.42 bin/cycle. And in [6] and [9], double BCM cores and frame parallelism are used in the architecture to enlarge the throughput. While in our design, we make use of efficient control logic to resolve the inefficiency problem and try to avoid the usage of multi-engine or frame parallelism.

5. CONCLUSIOIN

A 1 Gbin/s CABAC encoder with full hardware design is proposed in this paper. Multi-bin architecture is made full use of to increase the throughput of BAE. 6-SRAM-based context modeling is applied and efficient control is developed to obtain highly efficient context modeling. And binarization part is designed for high bin rate processing speed. The simulation using QFHD sequences shows that the 4 bin/cycle throughput of BAE can be fully maintained while the design of the other parts doesn't affect the timing performance.

REFERENCE

[1] Joint Video Team of ISO/IDC MPEG and ITU-T VCEG. "Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification (ITU-T Rec. H.264/ISO/IEC 14 496-10 AVC)".

[2] Y.J. Chen, C.H. Tsai, Liang-Gee Chen, "Architecture design of area-efficient SRAM-basedmulti-symbol arithmetic encoder in H.264/AVC", ISCAS, 2006, pp. 2621-2624.

[3] X. H. Tian, T. M. Le, B. L. Ho, Y. Lian, "A CABAC encoder design of H.264/AVC with RDO support", IEEE/IFIP International Workshop on RSP, 2007, pp. 167-173.

[4] P.S. Liu, J.W. Chen, Y.L. Lin, "A Hardwired Context-Based Adaptive Binary Arithmetic Encoder for H.264 Advanced Video Coding", VLSI-DAT, 2007, pp. 1-4.

[5] W. Zheng, D.X. Li, B. Shi, H.S. Le, M. Zhang, "Efficient pipelined CABAC encoding architecture", IEEE Transactions on Consumer Electronics, 2008, pp. 681-686.

[6] Y.H. Chen, T.D. Chuang, Y.J. Chen, C.T. Li, C.J. Hsu, S.Y. Chien, L.G. Chen, "An H.264/AVC scalable extension and high profile HDTV 1080p encoder chip", IEEE Symposium on VLSI Circuits, 2008, pp. 104-105.

[7] L.C. Wu, Y.L. Lin, "A high throughput CABAC encoder for ultra high resolution video", ISCAS, 2009, pp. 1048-1051

[8] X.H. Tian, T.M. Le, X. Jiang, Y. Lian, "Full RDO-Support Power-Aware CABAC Encoder With Efficient Context Access", IEEE Transactions on Circuits and Systems for Video Technology, 2009, pp. 1262-1273

[9] L.F. Ding, W.Y. Chen, P.K. Tsung, T.D. Chuang, P.H. Hsiao, Y.H. Chen, H.K. Chiu, S.Y. Chien, L.G. Chen, "A 212 MPixels/s 4096×2160p Multiview Video Encoder Chip for 3D/Quad Full HDTV Applications", IEEE Journal of Solid-State Circuit, 2010, pp. 46-58

[10] J.W. Chen, L.C. Wu, P.S. Liu, Y.L. Lin, "A High-throughput Fully Hardwired CABAC Encoder for QFHD H.264/AVC Main Profile Video", IEEE Transactions on Consumer Electronics, 2010, pp. 2529-2536

[11] D. Zhou, J. Zhou, X. He, J. Kong, J. Zhu, P. Liu, S. Goto, "A 530Mpixels/s 4096x2160@60fps H.264/AVC high profile video decoder chip", IEEE Journal of Solid-State Circuits, 2011, in press.