

Context-based Adaptive Binary Arithmetic Coding (CABAC): Design and Implementation Using a Multi-Symbol Binary Arithmetic Coder

B. Keerthi Priya¹, B. Kalyan Chakravarthy²

¹Department of ECE, Gayatri Vidya Parishad College of Engineering(A), Visakhapatnam, Andhra Pradesh, India

²Department of ECE, Gayatri Vidya Parishad College of Engineering(A), Visakhapatnam, Andhra Pradesh, India

Abstract:

The most recent standard for encoding video, known as H.265/MPEG-HEVC, achieves a rise in coding efficiency that is fifty percent higher than that of its predecessor, known as H.264/MPEG-AVC. High-resolution video coding (such as 4K, 8K, etc.) has attracted considerable interest since it was anticipated that H.265/MPEG-HEVC will end up replacing practically all current H.264/MPEG-AVC codecs. Video compression is a crucial method, especially in this day and age where greater quality and quicker frame-rate video are in high demand. There are other video compression algorithms available today, but the H.265 standard is one of the most well-known and widely utilized. The standard's last stage is the construction of a binary arithmetic coding algorithm that employs arithmetic coding theory principles to accomplish efficient encoding of the produced information. Binary arithmetic coding (BAC) is an important component of modern video compression techniques. A novel BAC method is examined in this work to demonstrate the feasibility of breaking the dependency of bin-bin in the refreshing of range and low registers. It is suggested to create new dataflow. The design implementation is performed using the Modelsim tool.

Keywords: CABAC; Video Encoding

1. Introduction

CABAC is used in H.265/HEVC and H.264/AVC. CABAC exceeds its predecessors in coding efficiency, but its computation requires certain knowledge. This causes issues in using parallelism and pipelining in equipment execution[1]. CABAC decoding and encoding is a vital bottleneck in the video decoder and encoder on account of data conditions and the need to deal with very high piece rates continuously. Fig. 1 depicts the important aspects of CABAC decoding, including paired arithmetic decoding, context memory, context choice, and de-binarization (DB). Outside and inside AD are two information flow rings. CABAC translation is harder than with a single pivotal circle[2].

In CABAC decoding, the first loop contains dependencies of data between successive syntax components (SE). Many earlier kinds of research have been undertaken to mitigate its impact, using strategies such as speculative/predictive pipeline execution[3], thorough branch coverage[4], and minimizing the frequency of syntactic element flipping[5]. Also, in HEVC, bins in context coding were lowered in order to reduce the SE loop at the algorithm level[6].

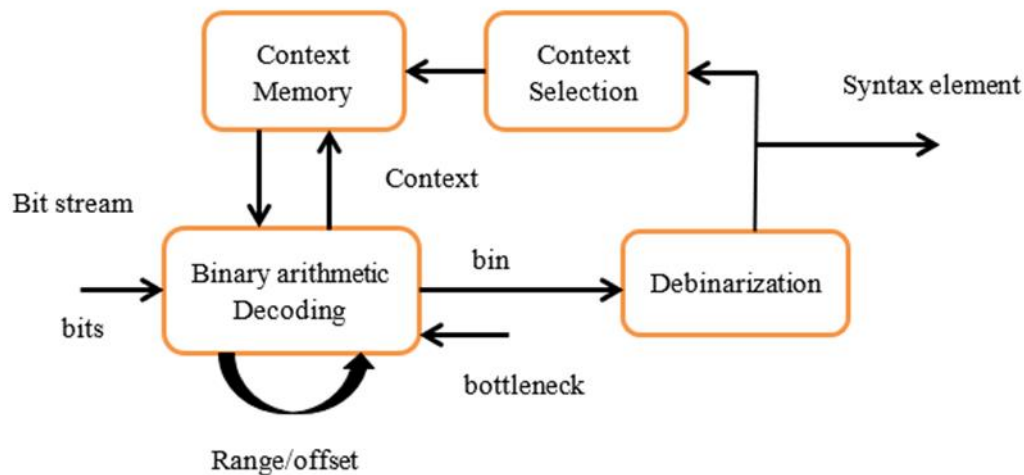


Figure 1. Block diagram of CABAC decoding

Within the AD loop, data dependencies at the bin level are much more important. A multi-symbol AD architecture to process more than one Bin for every Clock Cycle (BFCC) to address this loop is been considered in [7-9]. However, as parallelism was raised in these multi-BFCC architectures, critical path time rose. Clock cycle loop and highest clock frequency restrict performance increase. A multi-BFCC CABAC decoder may handle numerous SEs every clock cycle, increasing execution difficulty. Ref [10] showed a pipelined CABAC decoder. AD remains the CABAC decoder's bottleneck despite subinterval reordering¹¹. SE's plan accomplishes more noteworthy than 3Gbin/s speed utilizing a similar AD engineering as [10] and information parallelism at the syntactic component level, even though it isn't consistent with the last HEVC standard. The recommended AD approach can be utilized with punctuation component level parallelism for speedier CABAC translating in future video coding structures.

H.265/MPEG-HEVC is compatible with most H.264/MPEG-AVC applications, with an emphasis on standard video coding. H.265/enhanced MPEGHEVC's interactivity was decided by recent coherence and mechanical advances in video coding, resulting in huge bit rate reductions for typically similar visual quality [12-13].

2. Content-Based Adaptive Binary Arithmetic Coding

The CABAC technique is a form of entropy coding. It was first used in the H.264/AVC standard, and it is currently utilized in the HEVC standard, which is the most recent standard¹⁴. It is utilized toward the finish of the video encoding interaction to encode the results of the former stages, for example, quantized change coefficients, expectation modes, movement vectors, and intra-forecast course, which are alluded to as language Syntax components (SCs). SE explains how the video is recreated at the decoder.

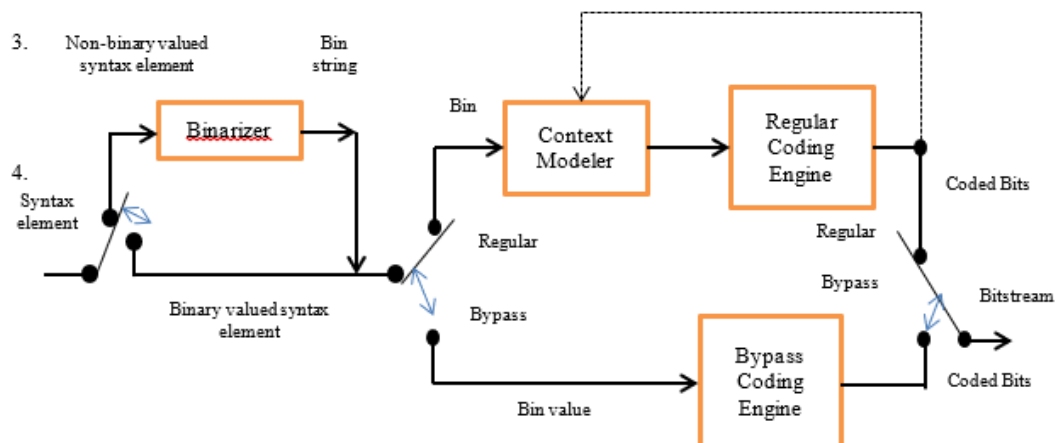


Figure 2. CABAC Encoding

As shown in Fig 2, three major steps are involved in the CABAC encoding methodology in which the first step is binarization, second step is modeling of context and finally binary form of arithmetic coding. The syntactic elements are mapped to binary symbols in the first stage (bins). The estimated likelihood of bins is provided by the context modeler. Finally, the binary arithmetic encoder uses the specified probability to compress bins to bits.

BAC is an arithmetic coding extension that is used for binary data. There is no requirement for a statistical framework for the data because the raw data simply comprises two symbols. Following the coding of a symbol, the frequency of occurrence is noted. The most likely symbol (MSP) has a probability of at least 0.5 for the symbol considered, whereas the symbol with the lowest probability is known as the least probable symbol (LPS). BAE's input is divided into three types: standard bin, bypass bin and terminate bin. Each has a unique coding procedure. While the encoding procedure for normal bins is quite typical, there is no need for a probability model while coding bypass bins, and the context model in the case of terminate bins is non-adaptive.

2.1. Binary Arithmetic Coding

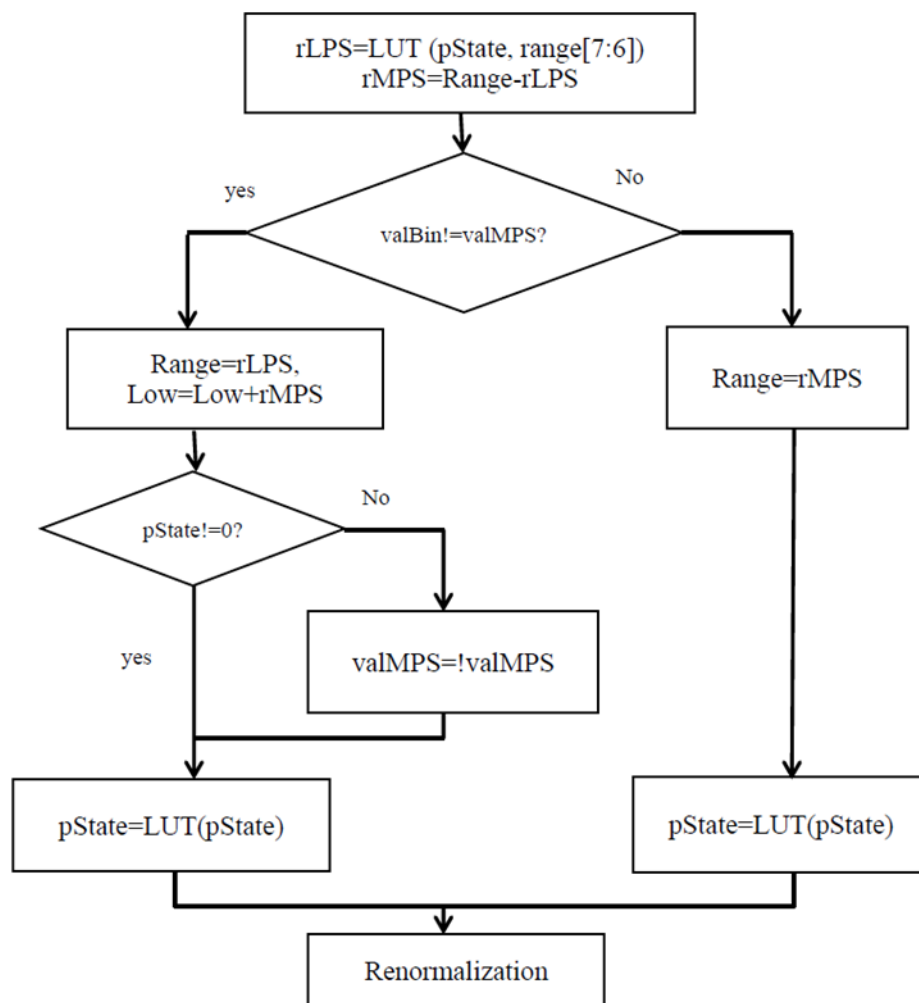


Figure3: Flowchart of encoding a regular bin [14]

BAE's major activity is regular bin coding. As a result, for hardware implementation, we rearranged the bypass bin and terminate bin encoding processes to fit within the main process. Two parameters express the internal state of arithmetic coding: Range (the range of present interval) and Low (the lower bound of this range). The context information supplied comprises the state index probability is termed as $pState$ and the value of MPS is termed $valMPS$.

The method involves four major phases, which are depicted in fig3. The current interval is partitioned in the first step based on the specified probability estimation. As the range of LPS is picked from a lookup table, the interval subdivision is conducted multiplier-free. MPS range is obtained by subtracting the LPS range from Range. Then, depending on the kind of symbol, MPS or LPS, Low and Range are adjusted. MPS refers to the lower interval range part of the interval range, whereas LPS corresponds to the upper interval range part of the interval range. In the third stage, the probability state is updated. The final step is to renormalize Range and Low.

Because there are only a few bits to represent Range and Low, they must be scaled up to ensure precision. During the renormalization process, the most significant bits (MSBs) of Low will be output. When Range falls below the 256-point barrier, renormalization occurs. A bit can be created or accumulated after each round. When a bit is created the next time, any pending bits will be resolved. The loop will be repeated till the Range is more than 256.

In [15], the author utilized a dedicated core for every bin that is utilized and the utilized bin is processed for every cycle. To improve speed, the author introduced two concurrent four-stage pipelined binary arithmetic encoding cores in [16]. They used a LUT to generate bitstream in order to minimize operating time. The author judged [17] to be state-of-the-art architecture for BAE. They built four concurrent pipelined multi-bin BAE cores, each of which can encode up to four standard bins every cycle. The binary arithmetic encoding's critical path time may be drastically reduced by employing the indicated modifications, which include bypassing bin splitting, pre-normalizing, using hybrid path coverage, and using look ahead rLPS. However, the use of a large number of BAE cores might result in a highly occupied area. The BAC architecture used in this paper is shown in fig4. A low-cost approach that balances performance, power and cost is developed and discussed in section 3.

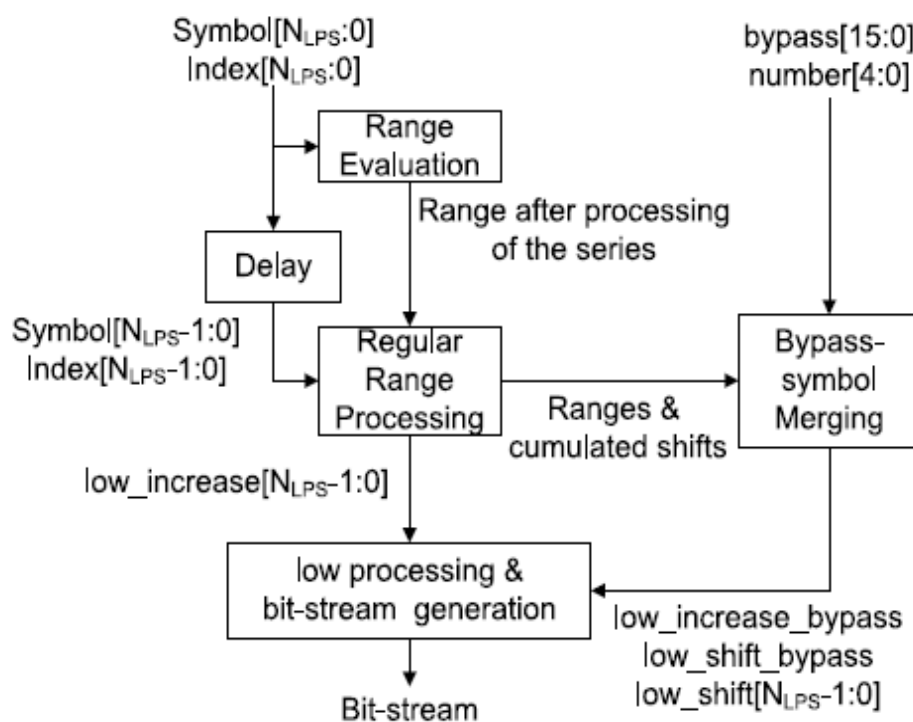


Figure 4. BAC Architecture

3. Proposed Methodology

An investigation into BAC aims to show that altering register range and low registers may be done without the need for b2b dependencies (bin-to-bin). A new process flow of the BAC method is studied in [18]. To start, range esteems are evaluated in four equal courses for an extensive grouping of bins, starting with the most unrealistic images. Second, the bins need to group based on the variable length and the performance is assessed by preceding the update of range registers in pipelined process. The next step is followed by the parallelism of images needed

to be done by accomplishing the segments of images into series of images. Fourth, rMPS factors (MPS range - base subinterval length) and single-clock-cycle images are added to the low register to reduce fundamental paths. The low register for bypass pictures has been expanded to 16 bins to improve throughput. At the HDL level, the most extreme proportion of bins is processed in a single clock cycle and is detected.

3.1 Range assessment using a pipeline-based approach

The logical optimization of BAC procedures appears to have reached its limit. As a result, advancements should be made differently. In the proposed technique, range register functions for subsequent symbols/bins are pipelined in order to maintain a high operating frequency and quickly process additional bins. The addition of consecutive stages to the pipeline allows the design to handle a specified number of bins while limiting just the quantity of used resources.

The standardized likelihood estimate produced from the input file is allocated to the range register in LPS coding. Notwithstanding the way that the gauge contains four prospects picked by the first range esteem (bits 7:6), they would all be able to be used as beginning stages for autonomous assessment courses for a long grouping of images following. The quantity of pipeline stages is dictated by the most extreme length of upheld LPS-driving series that might be characterized at the HDL level.

For MSP, subtract the specified probability estimate from the existing range value to compute the new range. This dependency inhibits effective pipelined scheduling for the symbols that follow. As a result, the MPS-leading series cannot be too long in order to reduce critical pathways.

3.2 Series Division of MPS and LPS

In the simplest situation, the final throughput is the average of the two cases and the probabilities that are weighted. Moreover, it is advantageous to reduce series lengths adaptively in order to improve the likelihood of LPS-leading instances. The suggested technique considers two following series to maximize their combined length as shown below.

$$\max(L_s[i] + L_s[i + 1]) \quad (1)$$

Depending on the ' L_s ' two scenarios can be considered. They are:

$$L_s[i] + L_s[i + 1] = N_{MPS} - dec + L_s[i + 1] \geq 2N_{MPS} \quad (2)$$

$$L_s[i] + L_s[i + 1] = N_{LPS} - dec + L_s[i + 1] \geq N_{LPS} + N_{MPS} \quad (3)$$

Where $L_s[i]$ = Length of i th series

For LPS and MPS, N_{LPS} and N_{MPS} represent the maximum lengths of the preceding series leading sequences.

$L_s[i + 1] = N_{LPS}$, as illustrated in (4) by the simplified equation:

$$dec \leq N_{LPS} - N_{MPS} \quad (4)$$

Equation (4) shows that the LPS-leading series must be at least as long as the MPS-leading example. In the last case, the series must comprise roughly one symbol.

$$dec < N_{MPS} - 1 \quad (5)$$

3.3 Processing of Range

An alternative dataflow process is proposed for the reduction of complexity. This decrease is achieved by decreasing the number of registers in all the paths. The said model is shown in Fig5.

To start, the assessment in the four pathways is restricted to the values of the range. Following the assessment, the standard handling of genuine range esteems is rehased for one of four courses. Third, in this customary processing, a succession of factors expected to refresh the low register is determined. From the last evaluation stage, only a single range value for every succession of images is taken and passed to the primary phase of standard processing. Such range esteems compare to moments later a progression of images has been coded.

A. Join Bit Renormalization

Because the total number of renormalization shifts is possibly large the wide bit representation is required. Because of the adder's lengthy carry chain, it may limit time at the lower updated loop. As a result, the smaller bit width is beneficial for meeting time constraints. Limiting the amount of shifts every clock cycle accomplishes this. If the true value is greater, the addition is performed more than twice.

B. A system using bypass mode

More bypass mode images ought to be taken care of in corresponding with context-coded ones to accomplish quicker throughputs. Utilizing the operation of multiplication, all successive bypass mode images might be connected to work out one addition to the lower register. This methodology, truth be told, is utilized in the HM programming [19] and might be utilized instead of equipment executions. When contrasted with the intricacy of range handling, the expense of a single multiplier is irrelevant. The prior context-coded picture range and bypass mode image link value are multiplied. The following option provides support for variable bit widths because the number of bypass mode images contained within the context-coded images series is subject to change.

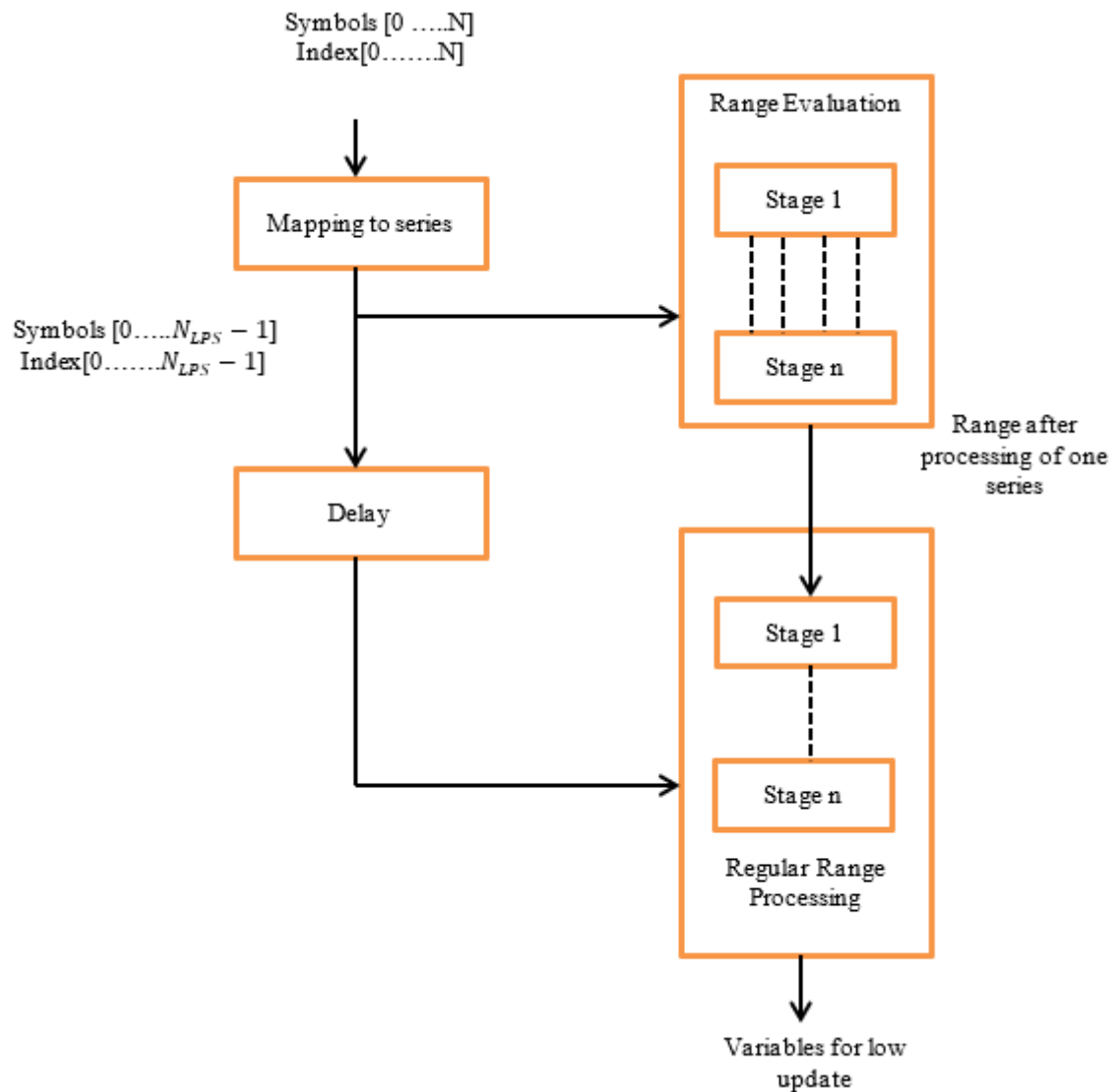
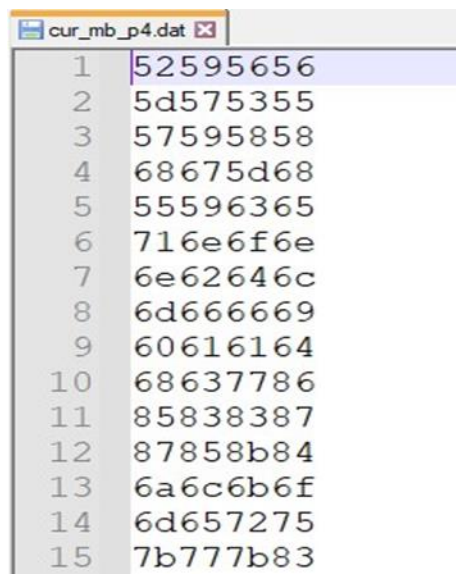


Figure 5. Processing of Range in dataflow



| Line Number | Hexadecimal String |
|-------------|--------------------|
| 1 | 52595656 |
| 2 | 5d575355 |
| 3 | 57595858 |
| 4 | 68675d68 |
| 5 | 55596365 |
| 6 | 716e6f6e |
| 7 | 6e62646c |
| 8 | 6d666669 |
| 9 | 60616164 |
| 10 | 68637786 |
| 11 | 85838387 |
| 12 | 87858b84 |
| 13 | 6a6c6b6f |
| 14 | 6d657275 |
| 15 | 7b777b83 |

Figure 6. Text sequence for testing results

4.Results and Discussions

The CABAC architecture designed in the proposed work allows a different type of hardware configuration. The entire process is coded in Verilog for ease of accessibility. To achieve the implemented results test sequences need to be selected first, to validate the output achieved using the proposed architecture. The input for testing is shown in figure6.

The processing of data range flow is designed in Verilog and some part of the design element is presented in figure7. The BAC approach is investigated in order to illustrate the viability of reducing bin-to-bin connections in range and low register updates.

```

0, Frame Number = 0, mb_x_load = 0, mb_y_load = 0
1165, Frame Number = 0, mb_x_load = 1, mb_y_load = 0
2655, Frame Number = 0, mb_x_load = 2, mb_y_load = 0
5095, Frame Number = 0, mb_x_load = 3, mb_y_load = 0
7535, Frame Number = 0, mb_x_load = 4, mb_y_load = 0
9975, Frame Number = 0, mb_x_load = 5, mb_y_load = 0
12415, Frame Number = 0, mb_x_load = 6, mb_y_load = 0
14855, Frame Number = 0, mb_x_load = 7, mb_y_load = 0
17295, Frame Number = 0, mb_x_load = 8, mb_y_load = 0

```

Figure 7. Verilog code for loading frames

The final output results are synthesized by using different logic gates, LUTs, adders, and multipliers. The simulation result is shown in fig8.

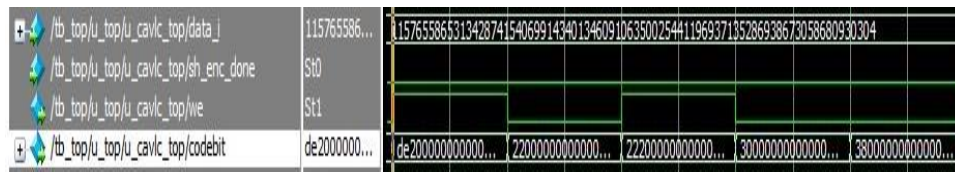


Figure 8. Simulation output

The architecture's capacity to accommodate longer series allows it to process a higher number of symbols for every clock cycle. The stretching of a series of images that are long and short has a comparable effect on the average symbol rate in H.265/HEVC. The advantage is especially significant when it comes to the compression of high-quality H.265/HEVC videos. The proposed design has the capability of encoding a greater number of bypass-mode symbols in a single clock cycle. Table 1 presents the results of a comparison between the performance of the proposed methodology and that of existing CABAC architectures.

Table 1. Comparison with different CABAC Models

| Parameter | T Sai et al., [20] | Zhou et al., [21] | Ramos et al., [22] | Proposed Model |
|-----------------------|--------------------|-------------------|--------------------|----------------|
| Evaluated Standard | AVC | HEVC | HEVC | HEVC |
| Technology [nm] | 130 | 90 | 90 | 90 |
| Symbol rate QP=22 | 5 | 4.38 | 4.94 | 9.87 |
| Max Freq [MHz] | 254 | 420 | 537 | 720 |
| Max throughput [Mb/s] | 1270 | 1839 | 2653 | 6855 |
| Gate count | 10 | 64 | 33 | 122 |

5. Conclusion

The binary arithmetic encoder is one of the very essential components of CABAC due to its processing which is based on serial bits and its internal data dependences. For every clock cycle, more symbols are been processed due to its architecture's capacity that accommodates longer series. Both the expansion of long series and short series have an equivalent impact on the mean symbol rate in H.265/HEVC codec. The dataflow and architectural architecture that is recommended for the BAC allows for significantly greater throughputs as compared to earlier technologies. Low-cost, high-throughput 4-stage pipelined BAC for HEVC CABAC based on a single-core architecture with multiple bypass bin processing. In one clock cycle, the suggested design may handle one standard bin or up to four bypass bins. The suggested architecture reduces gate count while improving processing time.

References

- [1] Tian, Xiaohua, Thanh M. Le, Xi Jiang, and Yong Lian. "Full RDO-support power-aware CABAC encoder with efficient context access." IEEE transactions on circuits and systems for video technology 19, no. 9 (2009): 1262-1273.
- [2] Zhou, Dajiang, Jinjia Zhou, Wei Fei, and Satoshi Goto. "Ultra-high-throughput VLSI architecture of H. 265/HEVC CABAC encoder for UHDTV applications." IEEE Transactions on circuits and systems for video technology 25, no. 3 (2014): 497-507.
- [3] Liao, Yuan-Hsin, Gwo-Long Li, and Tian-Sheuan Chang. "A highly efficient VLSI architecture for H. 264/AVC level 5.1 CABAC decoder." IEEE transactions on circuits and systems for video technology 22, no. 2 (2011): 272-281.

- [4] Lin, Pin-Chih, Tzu-Der Chuang, and Liang-Gee Chen. "A branch selection multi-symbol high throughput CABAC decoder architecture for H. 264/AVC." In 2009 IEEE international symposium on circuits and systems, pp. 365-368. IEEE, 2009.
- [5] Yi, Yongseok, and In-Cheol Park. "High-speed h. 264/avc cabac decoding." IEEE Transactions on Circuits and Systems for Video Technology 17, no. 4 (2007): 490-494.
- [6] Sze, Vivienne, and Madhukar Budagavi. "High throughput CABAC entropy coding in HEVC." IEEE Transactions on Circuits and Systems for Video Technology 22, no. 12 (2012): 1778-1791.
- [7] Yu, Wei, and Yun He. "A high performance CABAC decoding architecture." IEEE Transactions on Consumer Electronics 51, no. 4 (2005): 1352-1359.
- [8] Choi, Yongseok, and Jongbum Choi. "High-throughput CABAC codec architecture for HEVC." Electronics letters 49, no. 18 (2013): 1145-1147.
- [9] Zhao, Yijin, Jinjia Zhou, Dajiang Zhou, and Satoshi Goto. "A 610 Mbin/s CABAC decoder for H. 265/HEVC level 6.1 applications." In 2014 IEEE International Conference on Image Processing (ICIP), pp. 1268-1272. IEEE, 2014.
- [10] Chen, Yu-Hsin, and Vivienne Sze. "A deeply pipelined CABAC decoder for HEVC supporting level 6.2 high-tier applications." IEEE Transactions on Circuits and Systems for Video Technology 25, no. 5 (2014): 856-868.
- [11] Sze, Vivienne, and Anantha P. Chandrakasan. "A highly parallel and scalable CABAC decoder for next generation video coding." IEEE Journal of Solid-State Circuits 47, no. 1 (2011): 8-22.
- [12] Ohm, Jens-Rainer, Gary J. Sullivan, Heiko Schwarz, Thiow Keng Tan, and Thomas Wiegand. "Comparison of the coding efficiency of video coding standards—including high efficiency video coding (HEVC)." IEEE Transactions on circuits and systems for video technology 22, no. 12 (2012): 1669-1684.
- [13] Horowitz, Michael, Faouzi Kossentini, Nader Mahdi, Shilin Xu, Hsan Guermazi, Hassene Tmar, Bin Li, Gary J. Sullivan, and Jizheng Xu. "Informal subjective quality comparison of video compression performance of the HEVC and H. 264/MPEG-4 AVC standards for low-delay applications." In Applications of Digital Image Processing XXXV, vol. 8499, SPIE, 2012. pp. 321-326.
- [14] Coding, High Efficiency Video. "Recommendation itu-t h. 265." International Standard ISO/IEC (2013): 23008-2.
- [15] Pham, Duyen Hai, Jeonhak Moon, Doohwan Kim, and Seongsoo Lee. "Hardware Implementation of HEVC CABAC Binary Arithmetic Encoder." Journal of IKEEE 18, no. 4 (2014): 630-635.
- [16] Jo, Hyungu, and K. Ryoo. "Hardware Architecture of CABAC Binary Arithmetic Encoder for HEVC Encoder." Advanced Science and Technology Letters 141 (2016): 58-63.
- [17] Zhou, Dajiang, Jinjia Zhou, Wei Fei, and Satoshi Goto. "Ultra-high-throughput VLSI architecture of H. 265/HEVC CABAC encoder for UHD TV applications." IEEE Transactions on circuits and systems for video technology 25, no. 3 (2014): 497-507.
- [18] Pastuszak, Grzegorz. "Generative multi-symbol architecture of the binary arithmetic coder for UHD TV video encoders." IEEE Transactions on Circuits and Systems I: Regular Papers 67, no. 3 (2019): 891-902.
- [19] HEVC Software Repository-HM-16.0 Reference Model, (2018) https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/tags/HM-16.0/.
- [20] Tsai, Chen-Han, Chi-Sun Tang, and Liang-Gee Chen. "A flexible fully hardwired CABAC encoder for UHD TV H. 264/AVC high profile video." IEEE transactions on consumer electronics 58, no. 4 (2012): 1329-1337.

- [21] Zhou, Dajiang, Jinjia Zhou, Wei Fei, and Satoshi Goto. "Ultra-high-throughput VLSI architecture of H. 265/HEVC CABAC encoder for UHD TV applications." *IEEE Transactions on circuits and systems for video technology* 25, no. 3 (2014): 497-507.
- [22] Ramos, Fábio Luís Livi, Bruno Zatt, Marcelo Porto, and Sergio Bampi. "High-throughput binary arithmetic encoder using multiple-bypass bins processing for HEVC CABAC." In *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, IEEE, 2018 pp. 1-5