Multiplication free Fast-Adaptive Binary Range Coder using ISW

Sunkara Teena Mrudula[1*], K.E. Srinivasa Murthy[2] and M.N. Giri Prasad[3]

[1]Research Scholar, Dept of ECE, Jawaharlal Nehru Technological University Anantapur, Anantapuramu 515002, A.P, India
[2]Professor, Department of ECE, Ravindra College of Engineering for Women, Kurnool, A.P, India
[3] Professor, Jawaharlal Nehru Technological University Anantapur, Anantapuramu 515002, A.P, India

***Correspondence:** Sunkara Teena Mrudula; sunkaramrudula@gmail.com

ABSTRACT- Data compression is defined as the process of encoding, converting and modifying the bits-structures of data in such a way that reduces less-spaces on the disk. Fast-ABRC, a new context ABRC for compressing the image and video. This paper introduces novel hardware F-ABRC (Fast-adaptive binary range coder) and architecture of VLSI, as it doesn't have requirement of LUTs (Look-up-Tables) and also it is completely multiplication free. To get the result, we will combine the utilization of simple operation to compute the approximation after encoding every single symbol and the PE (probability estimation) on the basis of ISW (Imaginary Sliding Window) with approximation of the multiplication. We have represented our introduced algorithm, which is faster and in comparison, to the existing model it gives superior compression efficiency and the comparison takes place on the basis of two parameters such as power dissipation (Dynamic and Static) and device utilization.

Keywords: F-ABRC, ISW, VLSI.

ARTICLE INFORMATION

Author(s): Sunkara Teena Mrudula , K.E. Srinivasa Murthy and M.N. Giri Prasad;

Received: 15/02/2023; **Accepted**: 27/03/2023; **Published**: 30/03/2023;
e-ISSN: 2347-470X;
Paper Id: IJEER 1502-07;
Citation: 10.37391/IJEER.110131
Webpage-link:
www.ijeer.forexjournal.co.in/archive/volume-11/ijeer-110131.html

Publisher's Note: FOREX Publication stays neutral with regard to Jurisdictional claims in Published maps and institutional affiliations.

1. INTRODUCTION

The term DC (Data compression) contains the relevant encoding information through smaller bits than the given original re-presentation in the signal processing [1]. Moreover, compression can be lossy / lossless. The statistical redundancy is removed for lossless compression minimizing the bits. In the lossless compression, the information is stored as it is. unwanted eminent information is eliminated for minimizing the bits of lossy compression. The data size file which is introduced as the DC is minimized by this process. It is known as source-coding in data transmission context. Before storing and transmitting, the source data has encoded. The channel coding for error detection, correction and line coding (means mapping the data onto the signal) along with source code should not to be confused. The compression [2] is very useful because it minimizes the resources, which needed to transmit and store the data. The computational resources can be consumed in compression process and generally in decomposition process. DC (Data Compression) is subjected to the complexity of space-time. For example, the compression method for video may need expensive hardware for video, which is being decompressed fast as to be noticed, and decompress the video in full before seeing it and may not be convenient or need the additional storage. The DC scheme contains among the various factors, consisting the compression degree, the amount of distortion is represented (when utilizing the lossy DC), and computational resources needed to decompress and compress the data. The AC (arithmetic coding) is very common algorithm, which is utilized in both the lossy and lossless data compression algorithms. It is an entropy encoding method, which is frequently gotten that the symbols can be encoded with smaller bits than the lower seen symbols. It has few benefits over well-known methods like Huffman coding. The term AC covers 2 distinct processes like encoding the messages and decode them.

The term entropy coding is eminent phase in the scheme of video coding to reduce the size of syntax elements into the bitstream without the help of information loss. ABRC which is abbreviation for Adaptive Binary Arithmetic Coding [3] is the entropy tool that can be appeared in AVC/H.264 for 1st time in the standards of video coding. Related with prior CAVLC [4] (context-adaptive of variable length coding), the Adaptive-Binary Athematic-Coding (ABAC) develops the efficiency of compression by 14 and 9% [5]. Because of greater computational complexity, the ABC (Adaptive Binary Coding) can be only sustained in high and main profiles of AVC/H.264. After that, with development in memory cost reduction and throughput, the ABAC can be adopted as the tool of entropy coding for the syntax elements in the HEVC [6]. The ABAC encoding process contains of 3 phases such as context modeling, BAC (Binary Arithmetic Coding) and binarization [7]. If the element of syntax isn't binary, so it can be mapped to the binary sequences that is the term of binarization; otherwise, this phase can be omitted. Then for every single binary, there are 2 coding modes like bypass and regular. In the mode of regular coding, first the binary enters the phase of context modeling where the model of probability can be chosen for binary according to the earlier that encoded the syntax elements (known as context). Then the binary is together and associated with the model of probability that is fed into the BAC (binary arithmetic coder), with the help of subsequent model of updating procedure. In the mode of bypass coding, the binary can be encoded with the help of probability distribution (e.g., no modeling context) that speeds up the process of decoding/encoding. To understand the ABAC, we can distribute the ABAC encoding process into 2 phases such as arithmetic coding and probability estimation. The modelling of banalization and context phases are considering the syntax element of probability distribution, while the BAC phase satisfies the encoding. If the distribution of probability is specified accurately then the arithmetic coding can realize itself the efficiency of optimal coding up to the rounding error. Therefore, the performance tool of arithmetic coding solely based on the probability estimation. In both HEVC and H.264, the tools of ABRC can be optimized with the help of empirical context methods and binarization, which was acquired by experience, intuition and physically performed the statistics. Then, it is very doubtful whether the ABRC tool is an optimal. To minimize the CC (computation complexity), the modification of the Re-norm method will help to minimize the CC. In [8], it is proposed that the faster technique is Re-norm technique, but it is dependent on LUTs. Whereas in [9], the author introduced ABRC, which does not the byte Re-norm. Although, in the interval of ABRC, the multiplication is utilized by the range coder. Henceforth, F-ABRC (Fast-ABRC) implementation is also known as Q-Coder [10], and it follows the M-coder in H.264/AVC and H.264/HEVC, QM-Coder in JPEG and MQ-Coder JPEG2000 make use of the LUTs for multiplication of approximate operation and the probability estimation in interval part of bit-Re-norm (Re-normalization), which produces possible approximation. another replacement to the range, codes and arithmetic codes utilizes the elements of output bit-stream as well as the byte Re-Norm will be achieved at the same time. We make the use of the term ABRC for the universal DC and for the video calling [11]. M-coder in together with ABRC acquires up to 40% less computational complexity is represented [14] for the performance of software. Anyways, from the implementation of hardware, the drawback of ABRC, in the part of interval division, ABRC is to make use of the multiplication. On the other hand, the cost of multiplication can be similar with cost utilizing the look-up-tables in the modern architecture. As byte-Re-Norm is very less complex than the Bit Re-Norm. This paper is mainly dedicated to the F-ABRC for decoding and encoding technique which is the significant component of the CABAC, we make the use of CABAC in image and video compression standards like JPEG, H.265/HEVC, JPEG2000 and H.264/AVC.

This paper introduces F-ABRC that does not utilize the LUTs and multiplication. The introduced algorithm is the modification of F-ABRC, which is basically based on the ISW (Imaginary Sliding Window). Based on the statistical properties of corresponding binary source, it allows to assign precise window-length. Therefore, as compared to the method of state-of-the-art M-Coder, which don't require look-up-tables and very faster. The effectiveness of compression also improvised by it. As rest of this paper is planned in such a way that *section-2* reviews the ABRC, *section-3* reviews the BAC implementation, section 4 dedicated the free look-up-tables and probability estimation of 1's for the binary source and introduces the free look-up-tables and multiplication free of F-ABRC. Then the comparative outcomes for introduced F-ABRC and M-Coder are represented.

2. LITERATURE SURVEY

Several F-ABRC algorithm have been introduced [12]-[14]. To create implementation of AC (arithmetic coding) more practicable and easier, the alphabet size requires to be minimized to the binary so the process of coding can be simplified. The simple and faster implementation of AC-algorithm is utilizing the LUT method [15]. Few architectural develops in the implementation of VLSI of the arithmetic coder has been created by [18] utilizing the well-known method of speculative execution and loop unrolling. Whereas, in [20] introduced the algorithm that utilizes the redundant arithmetic to get further speed up for coder. Anyways, all of the coder is based on the arithmetic coding hardware's, which is defined to compress bi-level data of image and may be very poor for another type of data.

This paper [21] introduces an efficiency of the adaptive- BAC that dependent on a domain of logarithm (LBAC) as well as evaluation of probability dependent on P- LBAC (L-BAC). Combination of the P-LBAC as well as LBAC attain a ratio of the large data compression through less complexity as well as structure of hardware efficiency. The introduces P-LBAC as well as LBAC do not utilize even division and multiplication operation or overview the tables, and only shifting as well as addition operators are needed. The introduced LBAC are designed to approval the coding of various symbols as well as had huge throughput. CABAC [22] is a general part of the existing ISO/IEC/ISO advanced H.264/AVC for the compression of video is described. By CABAC through context modelling, a large degree of the redundancy and adaption reduction is acquired. The framework of the CABAC also consists a novel method of less-complexity for the BAC as well as evaluation of probability, which is well suitable for the efficiency of the hardware as well as implementation of the software.

In [23] introduces an efficiency of hardware-ABRC as well as its VLSI design. To acquire this, they follow a method, according to various requirements, that allows to reduce the capacity of bit in the period partition as well as it allows to explains that how the requirement avoided to make use in case of a loop under renormalization portion of ABRC. In the proposed ABRC, the estimation of probability is based on the lookup table free simulated sliding window. They have introduced an existing size of Adaptive – WSA (window selection algorithm) to gain a performance of the higher compression. they introduce an existing size of adaptive window selection algorithm. In contrast, through an ABRC by a unique window, the introduced scheme gives an adaption of the faster probability at basic decoding/encoding phase, as well as more suitable probability evaluation for the very less entropy of the binary sources. Whereas, in [24] represent a VLSI architecture of the ABAC for lossless compression of data as well as decompression. The important component of it includes of an APEM (adaptive probability estimation modular), an AOU (arithmetic operation unit), and a NU (normalization unit), A new method of bit-suffering, that concurrently resolves combination of source-termination as well as carry-over problems of efficiency, is designed and introduced in an UN. The APEM evaluates the probabilities of conditional of the efficiency of input symbols utilizing a method of table lookup method through 1.28-kbytes memory.

In [25] represents novel methods to parallelize-CABACs (context-based adaptive binary arithmetic coders). Two existing similar methods are defined as PCABACs (or CABACs). Therefore, this type of the coders is designed through transforming general utilized binary multiplication-free ACs (arithmetic coders). Uses of the linear estimates as well as simplifies the H/w (hardware) by predicting that low possible symbol probability is about the similar when presenting the decoding/encoding. There is another codec administrates method of table lookup as well as attains parallelism through a model of parallelized probability (known as QT-coder). QL-coder developed by IBM-coder, as well as QT-coder developed through utilization of CABAC in H.264 standard of compression of video. Whereas in [26] introduces the new scheme of AEC (adaptive entropy-coding scheme) for the compression is represented. It uses an adaptive technique of arithmetic coding to best contrast the entropy of first order of coded indicators as well as to keep rule of the statics of the nonstationary symbol. BAC implementation is discussed in next section.

3. PROPOSED METHODOLOGY

3.1 Implementation of ABAC

In addition to the stationary-discrete-memoryless abbreviation for SDM of the binary-source here we consider $\wp$ = probability of 1's. Encoding codewords for BS (binary-sequence) as $S^L = \{s_1, s_2, \dots, s_L\}, s_t \in \{0,1\}$ is signify as bits-number$\lceil -\log_2 (S^L) + 1 \rceil$ In BA (Binary Arithmetic).

$$Cum(S^L) + 0.5 \times Pro(S^L) \tag{1}$$

Where, $Pro(S^L)$ is represented as the probability and $Cum(S^L)$ is signify as the cumulative probability of the sequence S^L respectively, which is computed by the help of recurrent-operations.

If $s_i = 0$, then

$$\begin{cases} Cum(S^i) \leftarrow Pro(S^{i-1}) \\ Pro(S^i) \leftarrow Pro(S^{i-1})(1-\wp) \end{cases} \tag{2}$$

If $s_i = 1$, then

$$\begin{cases} Cum(I^u) \leftarrow Cum(S^{i-1}) + Pro(S^{i-1})(1-\wp) \\ Pro(S^i) \leftarrow Pro(S^{i-1})\wp \end{cases} \tag{3}$$

There are two registers such as X and Y. As for execution of the integer of an arithmetic encoder depends on these two registers. $Pro(S^L)$ is correlate with register X and $Cum\ (S^L)$ is correlate with Y. To show the register R and L we required more and more accuracy so that it develops with the maximize number of Y. we make use of re- normalization technique for each symbol of an output so that it will avoid registers-underflow as well as it will reduce the coding latency.

3.2 PE (Probability Estimation) based on Imaginary sliding Window (ISW)

In real time applications, probability of 1 is undetermined. In this period, symbol of an input binary A_{t0} for the 1's Probability Estimation (PE) can be solved in place of $\wp$. To get approximate result by examining the content of special – buffer (this buffer encoded as the symbols and keeps B earlier) for the source symbol, based on SW concept i.e. Sliding Window concept the PE algorithm has been developed. Here length of the buffer is BL. Then content of buffer can be moved from one place. After this the last symbol is removed from buffer. To free – cell the novel symbol is rewritten.

The probability of 1's for binary sources are estimated by the help of Krichevsky- Trofimov method:

$$\hat{\wp}_{u+1} = \frac{no_u + \frac{1}{2}}{BL + 1} \tag{4}$$

Before in window, encoding the symbol together with index i no. of one's is no_u.

When we make use of ISW there are many advantages and one of them is the possibility of precision of the statistics change as well as adaption which is fast. However, there is major disadvantage of this algorithm that in encoder and decoder memory the window has to be kept. The ISW technique has been introduced to ignore this disadvantage. The window content doesn't require to store by the ISW technique. In place of this the symbol counts from the source alphabet kept in the window can be approximated,

For BS, we are considering the ISW technique. Index $i, t_i \in \{0,1\}$ in addition to $s_i \in \{0,1\}$ defines (after including s_i, from window it can be removed) as the source input's symbol. Here in place of final one, we consider that the unspecified position's symbol is separated at each time from window. Afterwards, in windows the procedure randomized is re-computed.

PHASE 1: Eliminate the random symbol from window

$$no_{i+1} \leftarrow no_i - t_i \tag{5}$$

Where, t_i is denoted as the random value which can be generated with the help of probabilities.

$$\begin{cases} P(t_i = 1) = \dfrac{no_i}{BL} \\ P(t_i = 0) = 1 - \dfrac{no_i}{BL} \end{cases} \tag{6}$$

PHASE 2: Include novel symbol from source

$$no_{i+1} \leftarrow no_{i+1} + s_i \tag{7}$$

The unknown variable can be created for implementing ISW's algorithm. At corresponding stages of decoder and encoder this kind of unknown variable takes equal value. To ignore unknown variable there is a technique. In the algorithm at 1st stage, we will replace average probabilistic including value t_i . Encoding symbol s_i which can be represented in 2-stages after re-solving the no. of 1's.

PHASE 1: Eliminate the average number of 1's from window.

$$no_{i+1} \leftarrow no_i - \frac{no_i}{BL} \tag{8}$$

PHASE 2: Include the novel symbol from source.

$$no_{i+1} \leftarrow no_{i+1} + s_i \tag{9}$$

By adding the *equation (8) and (9)*, the last rule of re-computing the number of 1's, which can be mentioned as follows:

$$no_{i+1} = \left(1 - \frac{1}{BL}\right).no_i + s_i. \tag{10}$$

On the basis of *eq. (10)*, the PE (Probability Estimation) utilizing the ISW, which is been introduced in [28]. By comparing with M-Coder, it provides improved efficiency because of specific ISW length, which is chosen by the help of statistical properties of the binary source. Nonetheless, it needs multiplications that is insufficient for the implementation of hardware.

3.3 Removal of Multiplication using Reasoning Procedure

To eliminate the multiplication, we utilize the same reasoning, which is described in [29]. After the procedure of re-normalization, the register R fulfills the given inequality.

$$\frac{1}{2}2^{d-1} \leq X < 2^{d-1} \tag{11}$$

From (11), it can be follows the multiplication that can be approximated in following way:

$$\mathcal{T} = X \times \hat{\wp}_i \approx \alpha 2^{d-1} \times \hat{\wp}_i, \tag{12}$$

Let's multiply both sides of (9) by the help of $\alpha 2^{d-1}$, where $\alpha \in \left[\frac{1}{2}, \dots\dots..1\right]$.

$$no'_{i+1} = \left(1 - \frac{1}{BL}\right).no'_i + \alpha 2^{d-1} s_i, \tag{13}$$

Where $no'_i = \alpha 2^{d-1} no_i$, lets define the $BL = 2^{bl}$, where bl is denoted as the integer $+ve$ value. After that the we get the Equ (12) of integer rounding,

$$no'_{i+1} = \begin{cases} no'_i + \left\lfloor \dfrac{2^{d-1}2^{BL} - no'_i + 2^{BL-1}}{2^{BL}} \right\rfloor, if\, s_i = 1 \\ no'_i - \left\lfloor \dfrac{no'_i + 2^{BL-1}}{2^{BL}} \right\rfloor, if\ s_i = 0 \end{cases} \tag{14}$$

And,

$$\mathcal{T} = X \times \hat{\wp}_i \approx \alpha 2^{d-1} \times \hat{\wp}_i = \frac{no'_i}{2^{BL}} \tag{15}$$

To develop the precision approximation, which is mentioned in *eq. (7)*, we quantize the interval such as $\left[\frac{1}{2}2^{d-1};\ 2^{d-1}\right]$ to 4 points:

$$\left\{\frac{3^2}{2^4}2^{d-1}, \frac{3^2+2}{2^4}2^{d-1}, \frac{3^2+4}{2^4}2^{d-1}, \frac{2^4-1}{2^4}2^{d-1}\right\} \tag{16}$$

To execute this, first, we compute the state no'_i on the basis of *eq. (14)* for$\alpha = \frac{3^2}{2^4}$. Then estimate the multiplication in given method below:

$$\mathcal{T} = Y \times \hat{\wp}_i \approx \frac{no'_i + \Delta \times \frac{1}{2^2} no'_i}{2^{BL}}, Where\ \Delta = \frac{X - 2^{d-2}}{2^{d-4}} \tag{17}$$

Considering the estimation of multiplication can be precise for $\hat{\wp}_i < \frac{2}{3}$ [14], we should be work with $\hat{\wp}_i \in \left[0, \dots\dots, \frac{1}{2}\right]$ and utilized the LPS and MPS. In this case, the value of MPS should be varied.

$$\hat{p}_t = \frac{no'_i}{2^{BL}} \frac{1}{\alpha 2^{d-1}} > 0.5\ or\ no'_i > \alpha 2^{d-2} 2^{BL}. \tag{18}$$

Thus, considering the *equation (14), (17) and (18)*, the Fast-ABRC (Adaptive-Binary Arithmetic Coding) is been introduced in algorithm:

Algorithm 1: Encoding Method of Binary symbol s_t

```
S1:   $\Delta \leftarrow (X - 2^{d-2}) \gg (d - 2^2)$
S2:   $\mathcal{T} \leftarrow \left(no + \Delta \times (no \gg 2)\right) \gg bl$
S3:   $\mathcal{T} \leftarrow max(1, \mathcal{T})$
S4:   $X \leftarrow X - \mathcal{T}$
S5:   $if\ s_t \neq MPS$ then
S6:       $Y \leftarrow Y + X$
S7:       $X \leftarrow \mathcal{T}$
S8:       $no \leftarrow no + \left((\alpha 2^{d-1} 2^{bl} - no + 2^{bl-1}) \gg bl\right)$
S9:   $if\ no > \alpha 2^{d-2} 2^{bl}$ then
S10:              $MPS \leftarrow !MPS$:
S11:  $if\ no > \alpha 2^{d-2} 2^{bl}$;
S12:  End if
S13:      else
S14:      $no \leftarrow no - \left((no + 2^{bl-1}) \gg bl\right)$
S15:     End if
S16:  Call the Re-norm Procedure
```

Since, $\Delta \in \{0,1,2,3\}$ the multiplication in line 2 of the Algo-1 that can be implemented on the basis of addition and conditional operations.

3.4 Architecture of Multiple-Encoding Symbols

For increasing the performance of the AC encoding, we introduce the architecture of encoding which is able to coding the multiple symbols per cycle. While preserving the same or greater performance of coding, our architecture gives the flexibility to balance the clock rate by changing the huge number of coding per-cycle.

Figure 1 represents the block diagram of encoding architecture. To encode the multiple symbols per-cycle, we match the 1-symbol of encoding architecture and include the extra hardware in every single pipeline phase. For encoding the no number of symbols per-cycle, we match the 1-symbol of encoding architecture by no time in the phases 0 and 1. As represented in *Figure 1*, the 1-symbol of encoding unit in phases 0 and 1 adds the CU (Context update), RO (Range Operation) and LO (Low-Operation). For more than one symbol encoding, this useful unit are matched. After encode the symbol, these value of low and range are delivered to next useful unit. In stage 0, if multiple encoding symbols will be utilizing the model of context probability after the updating. Hence, the context data requires the multiplexer to select the right one. In phase1, the number of outcomes are produced by the help of LO, which is more variable. For minimizing the complexity and workload oh phase 2, we integrate the all outcomes before passing the information to phase 2.

In phase 2, we add the smaller number of input to help more than one encoding symbol. The BPU (Byte Packing Unit) can processes the 8-bit per cycle. For coding the 1 symbol, the usual number of outcomes from the Phase 1 is less than the 1. As we prolong the design for more than one symbol encoding, the probability for total number of input is being larger than 2^3, which is very small. Such type of exception only arises in some times for every single video frame. Thus, we add the less buffer in front of Phase-2. The input buffer controls the number of input to 2^3-bits. As an outcome, utilizing the input buffer that can maintain the similar structure of byte which packing the unit in Phase 2.

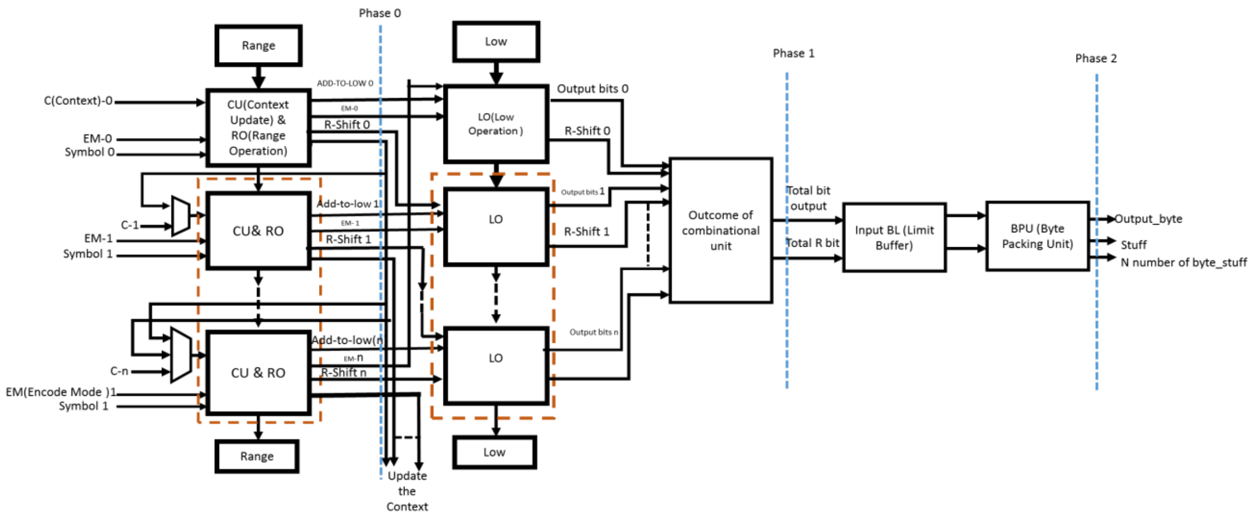

Figure 1: Architecture of Encoding Symbol

Figure 2 represents the details of unit for outcome combination. The outcome combination unit contains of adders and shifters. There are two types of input (I/P) signals in *figure 2*. The signal $O/P\ bits\_i\ (i = 0,1, \ldots\ldots, no)$ means the encoding outcome of 1-symbol and $X\_shift\_i\ (i = 0,1, \ldots\ldots, no\ )$ represents the encoding outcome length. For integrating the outcome, which can be produced by various low-operations, firstly, we shift the prior encoding outcome to right place. Then we utilize the address to integrate the all outcome bits. The outcome combination of unit can output the total number of outcome bits and the sequence of outcome bits.

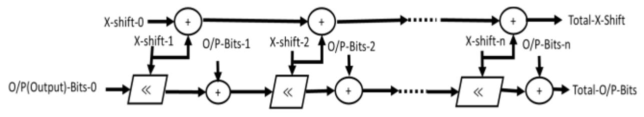

Figure 2: Outcome of Combinational Unit

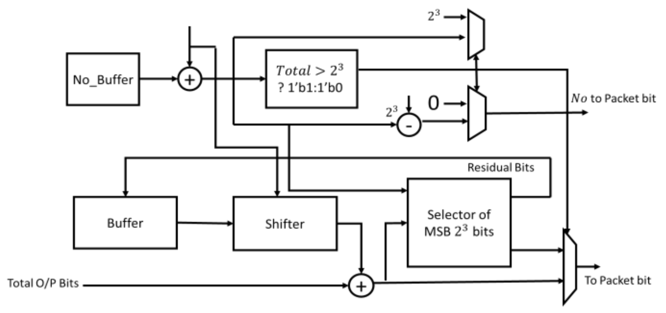

Figure 3: Block diagram of Input -BL (Buffer Limit)

Figure 3 represents the Block diagram of Input -LB (Limit Buffer). There are 2 types of local registers, input signals and 2 output signals. The first register known as "Buffer" is temporarily storing the residual bits if the prior length of packing bits are larger than 2^3 bits. The second register "No_Buffer" is recording the number of bits which is stored in register "Buffer". If the buffer is empty, we integrate bits in input bits and buffer. Then we verify if total number of bits are larger than 2^3. As the total number of bits is larger than 2^3, then we first choose 2^3 MSB bits of the integrated outcome as the output and preserve the bits of residual in buffer. Then, directly we pass the bits to byte packet unit.

4. EXPERIMENTAL RESULTS

This unit represents outcomes of F-ABRC which we have experimented as existing techniques compared with it. The main aim of this paper is not to use multiplications and LUT in ISW. After experimental result VDHL is used to write a code and Xilinx-version of 14.7 is to simulate. We have compared various parameters as well as constraint and some existing techniques which we have to be supposed in every single case, as we introduced F-ARBC is much better than existing techniques. The results are divided into 2-section such as power dissipation and device utilization.

4.1 Device Utilization

Below *table 1* represents 1st column for various architecture and 2nd for technology, 3rd and 4th column represents logic cell (no.) and memory. The number of logic cells is the best parameter that has been utilized for various resources by the help of FPGA technologies. The maximization in any image or video does not affect hardware resources of F-ABRC at fixed number of block size. The outcomes of F-ABRC is represented in *table 1*, when FPGA compared with architectures and implementation takes place. Our result i.e., outcomes doesn't need LUT and gets much better effectiveness of compression. Generally, to say F-ABRC has been developed from the ISW of the binary sources for the non-stationary, which gains trade off accuracy of 1's probability and speed of adaption because the use of different window size. That's why, it is more desirable for the standards of non-standardized codes , image and video coding. As compared to JPEG [34] AND SPIHT [32]. The JPEG is approximately 2.8 more than F-ABRC in logic cell (no.). where F-ABRC requires 10.6 times less than NLS [30]. Like we can see as in MQ-Coder Dyer [39] MQ-Coder-Kai [40] MQ-Coder [37] requires approximate 7.3, 1.6 8 and 16.5 times more logic cell as compared to F-ABRC. More to say, ABRC [41], ABRC [42] and ABRC [43] are 1.6, 1.7 and 1.3 times greater Fast-ABRC. Below the *table 1* represents comparison for different coders.

Table 1: FPGA resources of introduced F-ABRC in comparison with various implementations

Technology used	LC (Logic Cell)	Memory (in bytes Kbit)
Virtex 2-4	10,125	432
Virtex 2000E	83,808	N/A
Virtex 6-LX75T	16,621	0
Virtex 5-LX330	22,996	8.3
Spartan 3-S200	2711	N/A
Spartan 3-S200	2385	N/A
Zynq Z-7020	1017	0
Virtex 4-LX80	15,692	4.17
Altera Stratix	761	2675
Altera Stratix	1596	8192
Virtex 4- XC4VL	6974	4269
Virtex 5- ML507	1544	552960
Virtex 4-LX80	1688	0
Altera Stratix	1296	0
Virtex 4- XC4VF	948	0

4.2 Power Dissipation

Table 2 gives power dissipation for proposed Fast-ABRC in comparison to MQ-Coder, STS, CL-DCT and ABRC. In *table-2*, the architecture whereas second, third and fourth row depicts CL-DCT, MQ-Coder, STS, and the ABRC depicted by 1st row. 2nd row provides maximum operating frequency that is 182.75 MHz for F-ABRC (Fast-ABRC) and only 105.92 MHz for ABRC. The third and fourth column provides the dynamic power and the normalized power. So, it is very clear that the dynamic power and the normalized power value is less than F-ABRC in comparison to MQ-Coder, STS, CL-DCT and the ABRC. The F-ABRC has less normalized power as 182.75, 2077 and 0.114 in comparison to t MQ-Coder, STS, CL-DCT and the ABRC. Lastly, the density of power is represented in 5th column. The F-ABRC has slightly less than the other models. So, our F-ABRC outperforms better than another existing model.

Table 2: Power Dissipation for Proposed F-ABRC in comparison to CL-DCT, MQ-Coder, STS and ABRC

Architecture	MQ Coder	CL-DCT	STS	ABRC	F-ABRC
Frequency (MHz)	48.30	66.4	96	105.92	182.75
Dynamic power (mW)	488.67	96	74	127.05	20.77
Normalized power (mW/MHz)	10.117	1.45	0.77	1.19	0.114
Power density (μW/(MHz × Logic	0.65	0.60	0.75	0.71	0.12

Below *table 3*, In comparison with ABRC and MQ-Coder represents the power consumption. As in both phases, there is need for lossless compression in real-time which is provided by encoding cores. Here, the F-ABRC consumes 166.79, 20.77 and 187.56 m 5W, which is very less than in comparison to MQ-Coder and ABRC; even the consumption of leakage power is taken into the account. It's essential to be noticed that our presented models at the decoder side of F-ABRC are implemented as an analogical.

Table 3: Power consumption for proposed Fast-ABRC in comparison to MQ-Coder and ABRC

Architecture	MQ Coder Kai [40]	ABRC [43]	Fast-ABRC
Static power (mW)	624.68	622.55	166.79
Dynamic power (mW)	69.81	18.15	20.77
Total power (mW)	694.49	640.7	187.56

4.3 Static and Dynamic Power

When there is not an action of the circuit then the power consumed has been defined in place of the static power. It also produces the leakage power as well as standby power. Here, we have taken 2 other models like ABRC and MQ-Coder, whereas the static power of MQ-Coder and ABRC achieves 624.68 and 622.55mW when compared with our F-ABRC as we can see that our models outperform better than other two models. Whenever there is an action of the circuit, the power consumed is defined in place of dynamic power. The dynamic power can be assumed as the main parameter while comparing with other 2 models. Here, we have taken 2 other models as MQ-Coder and ABRC that possesses 69.81 and 18.15 mW when compared with our F-ABRC as we can see that our models outperform better than other two models

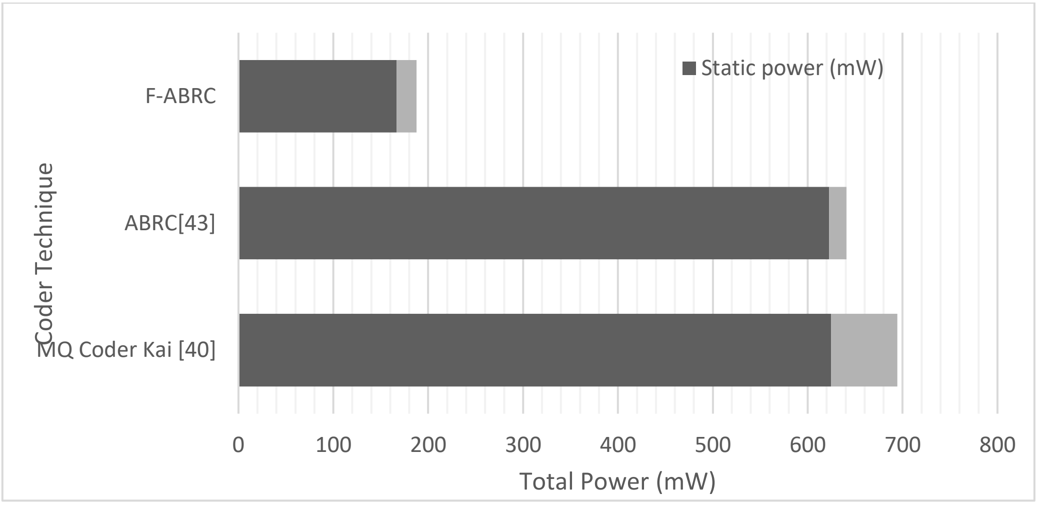

Figure 4: Comparison of Static and Dynamic power

4.4 Total Power

The total power can be defined as the combination of both static and dynamic power. In *figure-5*, we can see that 2 existing model such as MQ-Coder and ABRC achieves 694.49 and 640.7 mW, whereas our model achieves 187.56 mW, which outperforms much better than the other 2 models.

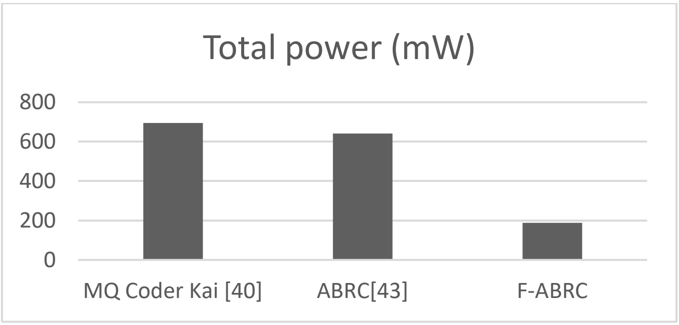

Figure 5: Comparison of Total power

5. CONCLUSION

In this paper, we have represented the multiplication free Fast-ABRC. It performs superior compression-efficiency, very quicker as compared to M-coder and don't need LUT. Basically, our proposed F-ABRC depends on ISW for binary sources of non - stationary, additionally, these methods perform trade-off among precision of PE and adaption speed because of the utilization of various size of window. In device utilization, we can see that FPGA required by F-ABRC is 2.8 times lower than JPEG. As in NLS, which is 10 times more to F-ABRC, similarly several methodologies such as MQ-Coder required 16.5, 1.68 and 7.3 times bigger than F-ABRC in logic cell. Moreover, ABRC [41], ABRC [42] and ABRC [43] are 1.6, 1.7 and 1.3 times greater than Fast-ABRC. Bit-PP [31], BPS [33], 1DSPIHT [32] requires more amount of area in comparison to the F-ABRC.

REFERENCES

[1] K. Holtz and E. Holtz, "Lossless data compression techniques," Proceedings of WESCON '94, Anaheim , CA, USA, 1994, pp. 392-397.

[2] Eswaran, K., &Gastpar, M. (2009). Foundations of Distributed Source Coding. Distributed Source Coding, 3–31.

[3] D. Marpe and T. Wiegand, “A highly efficient multiplication-free binary arithmetic coder and its application in video coding,” in Proc. IEEE Int. Conf. Image Process., 2003, pp. 263–266.

[4] V. Sze and M. Budagavi, “Overview of the high efficiency video coding (HEVC) standard,” IEEE Trans. Circuits Syst. Video Technol., vol. 22, no. 12, pp. 1649–1668, Dec. 2012.

[5] Auli-Llinas, F. (2015). Context-Adaptive Binary Arithmetic Coding With Fixed-Length Codewords. IEEE Transactions on Multimedia, 17(8), 1385–1390.

[6] V. Rosa, L. Max, S. Bampi," High Performance Architectures for the Arithmetic Encoder of the H.264/AVC CABAC Entropy Coder", Electronics, Circuits, and Systems (ICECS), 2010 17th IEEE International Conference.

[7] D. Chevion, E. D. Karnin, and E. Walach, "High efficiency, multiplication free approximation of arithmetic coding," in Proc. IEEE Data Compression Conf., Apr. 1991, pp. 43–52.

[8] M. Slattery and J. Mitchell, "The Qx-coder," IBM J. Res, Devel., vol. 42, no. 6, pp. 767–784, Nov. 1998.

[9] E.Belyaev, A.Turlikov, K.Egiazarian and M.Gabbouj,"An efficient adaptive binary arithmetic coder with low memory requirement,"IEEE Journal of Selected Topics in Signal Processing. Special Issue on Video Coding: HEVC and beyond, vol.7, iss.6, pp.1053–1061, 2013.

[10] G. Feygin, P. G. Gulak, and P. Chow, "Minimizing error and VLSI complexity in the multiplication free approximation of arithmetic coding," in Proc. IEEE Data Compression Conf., Snowbird, UT, Mar. 1993, pp. 118–127.

[11] L. Huynh, "Multiplication and division free adaptive arithmetic coding techniques for bi-level images," in Proc. IEEE Data Compression Conf., Snowbird, UT, Mar. 1994, pp. 264–273.

[12] P. G. Howard and J. S. Vitter, "Arithmetic coding for data compression," Proc. IEEE, vol. 82, pp. 857–865, June 1994.

[13] R. Arps, T. Truong, D. Lu, R. Pasco, and T. Friedman, "A multi-purpose VLSI chip for adaptive data compression of bilevel images," IBM J. Res. Develop., vol. 32, no. 6, pp. 775–794, Nov. 1988.

[14] W. B. Pennebaker, J. L. Mitchell, G. G. Langdon, and R. B. Arps, "An overview of the basic principles of the Q-coder adaptive binary arithmetic coder," IBM J. Res. Develop., vol. 32, no. 6, pp. 717–725, Nov. 1988.

[15] G. Feygin, P. G. Gulak, and P. Chow, "Architectural advances in the VLSI implementation of arithmetic coding for binary image compression," in Proc. IEEE Data Compression Conf., Snowbird, UT, Mar. 19.

[16] W. Pennebaker and J. Mitchell, JPEG Still Image Data Compression Standard. New York: Van Nostrand Reinhold, 1993.94, pp. 254–263.

[17] B. Fu and K. K. Parhi, "Two VLSI design advances in arithmetic coding," in Proc. ISCAS, Seattle, WA, Apr. 1995, pp. 1440–1443.

[18] E. Belyaev, K. Liu, M. Gabbouj and Y. Li, "An Efficient Adaptive Binary Range Coder and Its VLSI Architecture," in IEEE Transactions on Circuits and Systems for Video Technology, vol. 25, no. 8, pp. 1435-1446, Aug. 2015.

[19] Shiann-RongKuang, Jer-Min Jou and Yuh-Lin Chen, "The design of an adaptive on-line binary arithmetic-coding chip," in IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications, vol. 45, no. 7, pp. 693-706, July 1998.

[20] J. -. Lin and K. K. Parhi, "Parallelization of Context-Based Adaptive Binary Arithmetic Coders," in IEEE Transactions on Signal Processing, vol. 54, no. 10, pp. 3702-3711, Oct. 2006.

[21] B. Ryabko, "Imaginary sliding window as a tool for data compression", Problems of Information Transmission, pp. 156–163, 1996.

[22] D. Taubman and M. Marcellin, "JPEG2000: Image Compression, Fundamentals, Standards, and Practice", Kluwer Academic Publishers, 2002.

[23] Wheeler FW, Pearlman WA. SPIHT image compression without lists. In: IEEE 2000 International Conference on Acoustics, Speech, and Signal Processing; 5–9 June 2000; İstanbul, Turkey. New York, NY, USA: IEEE. pp.2047-2050.

[24] Fry TW, Hauck SA. SPIHT image compression on FPGAs. IEEE T CircSyst Vid 2005; 15: 1138-1147

[25] Kim S, Lee D, Kim JS, Lee HJ. A high-throughput hardware design of a one-dimensional SPIHT algorithm. IEEET Multimedia 2016; 18: 392-404.

[26] Jin Y, Lee HJ. A block-based pass-parallel SPIHT algorithm. IEEE T CircSyst Vid 2012; 22: 1064-1075.

[27] Wallace G. The JPEG still picture compression standard. IEEE T ConsumElectr 1992; 38: 18-34.

[28] Kaddachi ML, Soudani A, Lecuire V, Torki K, Makkaoui L, Moureaux JM. Low power hardware-based image compression solution for wireless camera sensor networks. Comp Stand Inter 2012; 34: 14-23.

[29] Rafi LONE, Mohd& HAKIM, Najeeb-ud-Din. (2018). FPGA implementation of a low-power and area-efficient state-table-based compression algorithm for DSLR cameras. TURKISH JOURNAL OF ELECTRICAL ENGINEERING & COMPUTER SCIENCES. 26. 2928-2943. 10.3906/elk-1804-208.

[30] K.Liu, Y.Zhou, Y. Song Li, J. Feng Ma, "A high performance MQ encoder architecture in JPEG2000", INTEGRATION, the VLSI journal, vol.43, no.3, pp.305–317, 2010.

[31] I.Shcherbakov, N.Wehn, "A Parallel Adaptive Range Coding Compressor: Algorithm, FPGA Prototype, Evaluation", Data Compression Conference, 2012.