Microprocessors and Microsystems 71 (2019) 102901

Contents lists available at ScienceDirect

Microprocessors and Microsystems

journal homepage: www.elsevier.com/locate/micpro

# M-ABRC (Adaptive Binary Range Coder) using Virtual Sliding Window technique and its VLSI implementation

S.T. Mrudula[a,*], K.E. Srinivasa Murthy[b], M.N. Giri Prasad[a]

[a] Department of ECE, JNTUA, Ananthapuramu, AP, India
[b] Department of ECE, G. Pullaiah College of Engineering and Technology, Kurnool, AP, India

ARTICLE INFO

*Article history:*
Received 26 May 2019
Revised 12 August 2019
Accepted 30 September 2019
Available online 1 October 2019

*Keywords:*
Arithmetic coding
M-ABRC
VSW
VLSI

ABSTRACT

Arithmetic coding is the data compression techniques, which encodes the data by generating the code string that represents a functional value between 0 and 1. In this paper, we propose a modified-Adaptive Binary-RC (Range Coder) or M-ABRC. Our algorithm minimizes the multiplication bit capacity through introducing the VLSI architecture, proposed algorithm uses the LUP (Look UP Table)-VSW (Virtual Sliding Window) for the probability estimation. In order to achieve the higher compression rate, our method M-ABRC has been implemented, this in terms provides the better adoption probability in encoding phase and also gives the absolute estimation of low-EBS(entropy binary sources). Moreover In order to evaluate the algorithm we have compared with the several existing technique, comparison takes place based on the two parameter i.e. device utilization and the power dissipation (static and dynamic).



## 1. Introduction

Arithmetic coding provides the coding of efficient variable length, which reaches the limit of entropy. Arithmetic coding [1] manages the various symbols with the adoptive models. This method requires the arithmetic operation such as multiplication and division is used for computing the symbol probabilities along with the interval. In case if there are two symbols involved, we use binary arithmetic coder, several methods are available such as QM-Coder [2], this method only needs the operations of bit shift and subtraction for computing the probability interval. In order to calculate the adaptive distribution [3], later the division is removed by using the state machine.

To encode the multi-symbol data using the BAC (Binary Arithmetic coding), the symbol is converted to the binary format. In BAC, the recursive portioning takes place, which ranges accordingly along with the occurrence probabilities of the given two input symbols [4]. Let's consider an example that there are certain number of symbols, each of them represents the binary number. Here, each bits are encoded using the BAC(Binary arithmetic coding) with their own probability state. In other words, there would be certain number of independent contexts for estimation of probability.

In a theoretical manner, each and every possible data set holds the "code-word" which is assigned by the arithmetic codes. The code words consists of unit-interval and subinterval (half open) denoted by 0 and 1. These intervals are expressed through defining the particular bits for differentiating the subinterval data from the all-possible sub-intervals. Moreover, arithmetic codes provides much better compression than the prefix codes. However, the direct correspondence between the input data set and coded output file. Arithmetic coding is nothing but the particular form of entropy encoding which is used in lossless data compression. Lossless data compression is part of data compression methodology that allows the reconstruction of original data from the compressed data [5]. Arithmetic coding was developed almost a decade ago, this method offers extensive efficiency when compared to the other VC (video Coding) standards namely MPEG-2 [6], H.263 [7]. Actually, these methods mainly relied on the coding method of Huffman [8] to entropy the steps of coding for compression. Due to its several advantage and high efficiency, arithmetic coding has drawn the attention of many researchers. ABAC has played one of the eminent role when it comes to video coding and for compressing the data and it achieves the good result in software part, however it lacks from the hardware implementation as it uses the multiplication in internal division part. Providing the high compression is essential as the media files (compressed) is increasing day by day from many applications such as from music players, digital cameras and internet.

Hence, we have designed and developed the modified version of ABRC; our research work contribution can be listed as below:

* Corresponding author.
*E-mail addresses:* stmrudula.jntu@gmail.com (S.T. Mrudula), hod.ece@gpcet.ac.in (K.E. Srinivasa Murthy), giriprasadmn.ece@jntua.ac.in (M.N. Giri Prasad).

https://doi.org/10.1016/j.micpro.2019.102901
0141-9331/

- Modified ABRC can be used for the probability estimation and it does not require any LUP (Look up Table) instead it uses VSW (virtual Sliding Window).
- Moreover, in this research work, we propose a model in order to obtain higher compression; this algorithm is based on the M-ARBC our algorithm gives the much faster probability adoption; this is done at the encoding stage.
- Our method is very much suitable for the hardware implementation; this highlights the reduction of bit capacity multiplication.
- Our algorithm completely elaborates the reduction required in the given interval part and suggest how loop usage can be avoided.
- Later VLSI architecture of our model is presented which shows the comparative analysis with the existing technique.

## 2. Literature survey

The first standard AEC (Arithmetic Entropy Coder) within the hybrid video coder, which is based on the block design is introduced by [8]. However, the main problem with this method was that the arithmetic coder used in this method was in considerable amount and this would not fulfil the criteria of most application. Mostly non-block based VC (Video Coding) uses the ECS (Entropy coding scheme) based on the arithmetic coding [9]. In this particular research, AB-(Adaptive Binary)AC(Arithmetic Coder) is proposed, basically, this method was free of multiplication and also there was no need of LUP(Look Up Table). In order to obtain this, probability estimation of both which is based on the VSW (Virtual Sliding Window). Moreover, simple operation is done in order to calculate the next approximation and this is done after each binary symbol is encoded.

[10] In this paper, an adaptive BAC (Binary Arithmetic coder) was proposed, when compared to the M-coder, it achieves the higher compression efficiency and possess the faster performance also it does not require the look up table. This method was based on VSW (Virtual Sliding Window). In case of non-stationary binary sources, it also obtains the tradeoff between the precision of probability estimation and adaption speed. However, it was preferable only to the non-standardized codecs.

[11] In this paper, an efficient multiplication-free, multi-alphabet AAC (Adaptive Arithmetic Coder) is proposed. Here, at first probability estimation is generalized by using VSW in case of multialphabet case, this clearly indicates that this do not require the trade-off among the probability estimation and adoption speed of the same. Later, how VSW (Virtual Sliding Window) is generalized, this shows the elimination of division and multiplication.

[12] Later in order to estimate the LB (locally Biased) probabilities and recover the source, SWBP methodology has been proposed, to set the SW(Sliding Window)-size in SWBP, another novel mechanism is proposed. This scheme is insensitive to the given initial setting; this makes the scheme to implement in more practicality. However, decoding part was not favourable. Hence, In paper [13] a method is proposed which requires the less memory and less complexity and this in terms helps in increasing the good compression efficiency. In order to compute the coefficient of DWT, fractional WF (Wavelet Filter) is employed by the encoder. The algorithm named as LMB (Low Memory Block)-tree coding is implemented, this algorithm comes under the category of listless WBTC. It helps in reducing the memory requirement.

[14] In this paper, modified version of ARC (Adaptive Range Coding) is presented in case of modern parallel hardware. This algorithm combines the higher compression ratio with Huffman coding; this explores the utilization of FPGA, compression ratio, performance and given built in LZSS + PARC. Hence, [15] proposed an architecture of parallel hardware is presented for the estimation of rate in the HEVC for increasing the parallelism so that the computational time can be reduce, this highly parallel architecture provides flexibility with the CABAC, however it ignores the certain syntax element.

[16] In this paper, both binary arithmetic coding and context modelling is combined to achieve the high degree adaption and minimized redundancy is achieved. The framework of CABAC includes the methodology of low complexity for the probability estimation and arithmetic coding; this in terms is very much applicable for the software and hardware implementation. However, still lot of improvement is required since it is still in infancy stage.

This work is organized in such a way that in our first section we discuss about the introduction part, second section deals with the existing methodology that has helped us in designing our model. Third section shows the proposed methodology with pictorial presentation and mathematical notation and algorithm. Evaluation of algorithm is depicted in fourth section, last but not the least section concludes our research.

## 3. Proposed methodology

In this section, we introduce the M-ABRC (Modified ABRC) that is very suitable for the implementation of hardware. Also, we represents the multiplication bit capacity required in the given interval part and suggest how loop usage can be avoided. The proposed M-ABRC utilizes the VSW for estimation of probability that does not need the look-up-tables. In next section, we reviews the BAC based integer implementation, Estimation of probability based on VSW and Byte Re-Norm in the range coders.

### 3.1. Generalized ABRC

#### 3.1.1. BAC (Binary Arithmetic Coding) based Integer implementation

We assume that $\rho$ *is probability* of $1'$swith the stationary discrete memoryless of binary source. In BAE (Binary Arithmetic Encoding), a code word for the BS (binary-sequence) $A^N = \{a_1, a_2, \ldots, a_N\}, a_t \in \{0, 1\}$ is denoted as the number of bits $\lceil -\log_2 PB(A^N) + 1 \rceil$.

$$CP\left(A^N\right) + 0.5 \times PB\left(A^N\right) \tag{1}$$

where, $PB(A^N)$ = probability and $CP(A^N)$ = cumulative probability of the sequence $A^N$ that can be computedby the help of re-current relations:

If $a_t = 0$, then

$$\begin{cases} CP\left(A^t\right) \leftarrow CP\left(A^{t-1}\right) \\ PB\left(A^t\right) \leftarrow PB\left(A^{t-1}\right)(1-\rho) \end{cases} \tag{2}$$

If $a_t = 1$, then

$$\begin{cases} CP\left(A^t\right) \leftarrow CP\left(A^{t-1}\right) + PB\left(A^{t-1}\right)(1-\rho\ ) \\ PB\left(A^t\right) \leftarrow PB\left(A^{t-1}\right)\rho \end{cases} \tag{3}$$

This paper utilizes "$\leftarrow$" = assignment operator, $\mathbb{S}_L$ indicated the left shift and $\mathbb{S}_R$ indicated the right shift, $\oplus$ = XOR operation and "!"=bitwise not operation.

The arithmetic encoder's of integer implementation is based on given two registers such as X and Y with b bits of size, which is mentioned in Fig. 1. The register X and Y which corresponds to $CP(A^N)$ and $PB(A^N)$. The accuracy need to represent the two registers X and Y that grows with maximize of N. In order to minimize the coding latency and avoid registers under-flow, the Re-Norm method is used for each symbol of output, which is shown in Fig. 2.

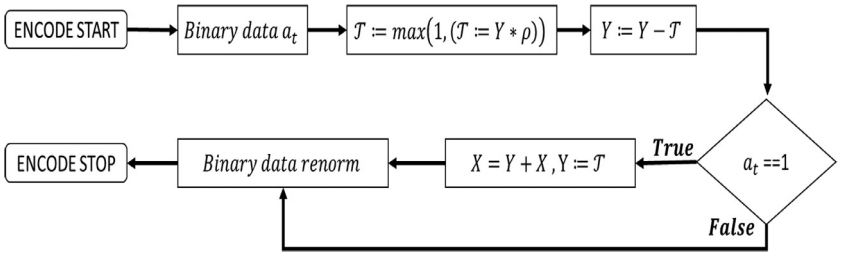


**Fig. 1.** Flow chart of encode procedure $a_t$ of Binary data.

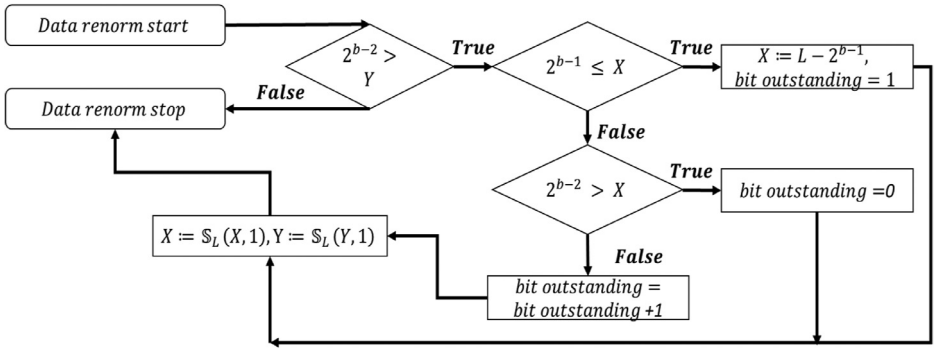


**Fig. 2.** flow chart of Bit Re-Norm.

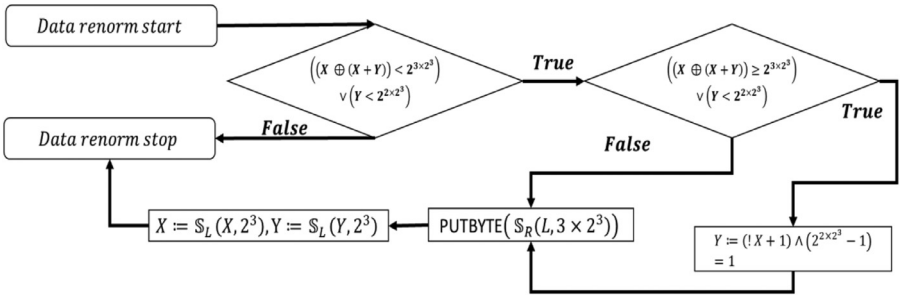


**Fig. 3.** Flow chart of byte Re-Norm.

*3.1.2. Estimation of probability based on the VSW (Virtual Sliding Window)*

In the real applications, the ones probability is unknown. Here, $a_t$ is denoted as the binary symbol of input and $\hat{\rho}_t$ denotes the estimation probability which is used and calculated in Fig. 1. instead of $\rho$. The well-known probability estimation is based of SW (sliding window). The source of probability symbol is analyzed by the concept of special buffer. Previously, the encoded symbol WL keeps the buffer, where WL is denoted as the length buffer. Afterwards, to encode the every symbol the buffer's content is shifted by one position, a new symbol is written to free the cell and the earliest buffer symbol is removed which is disccused in Fig. 3.

According to Krichevsky–Trofimov, $\hat{\rho}_{t+1}$ is denoted as the probability for the binary symbol with $t+1$ index that can be estimated as:

$$\hat{\rho}_{t+1} = \frac{sw_n + \frac{1}{2}}{WL + 1} \tag{4}$$

where, $sw_n$ is denoted as thenumber of 1′s in SW before encoding the symbol with $t+1$ index. The estimation of SW (Sliding window) can obtain the source statistics of more accurate evaluation and the fast adaption to statistic source by maximizing *WL*. In both the decoder and encoder, the SW must be stored, that can incur the problem of memory with maximizing the size of *WL*. To resolve the two-step of method for calculating the number of ones after encoding and proposed the $a_t$ symbol.

Step-1: Remove the average number of 1′s from the window.

$$sw_{n+1} \leftarrow sw_n - \frac{sw_n}{WL} \tag{5}$$

Step-2: Join the symbol from source

$$sw_{n+1} \leftarrow sw_{n+1} + a_t \tag{6}$$

By integrating (6) and (5), the rules for calculating the number of 1′s, which can be assumed as, follows:

$$sw_{n+1} = \left(1 - \frac{1}{WL}\right).sw_n + a_t \tag{7}$$

Based on (7), the estimation of probability utilizing the VSW was introduced. Let us multiply by both sides of (7) by WL:

$$sw_{n+1}{}' = \left(1 - \frac{1}{WL}\right).sw_n + WLa_t \tag{8}$$

where, $sw_{n+1}{}' = WLa_t$. Let us assume $WL = 2^{wl}$, where the positive value of integer is *wl*. Later an integer rounding of Eq. (8), then we get

$$sw_{n+1}{}' = \begin{cases} sw_n{}' + \lfloor \frac{2^{2ws} - sw_n{}' + 2^{ws-1}}{2^{ws}} \rfloor, & if\ a_t = 1 \\ sw_n{}' - \lfloor \frac{sw_n{}' + 2^{ws-1}}{2^{ws}} \rfloor, & if\ a_t = 0, \end{cases} \tag{9}$$

The probability of 1′s is evaluated as:

$$\hat{\rho}_t = \frac{sw_n{}'}{2^{2ws}} \tag{10}$$

Thus, the algorithm of ABAC is based on the VSW (Virtual Sliding Window).

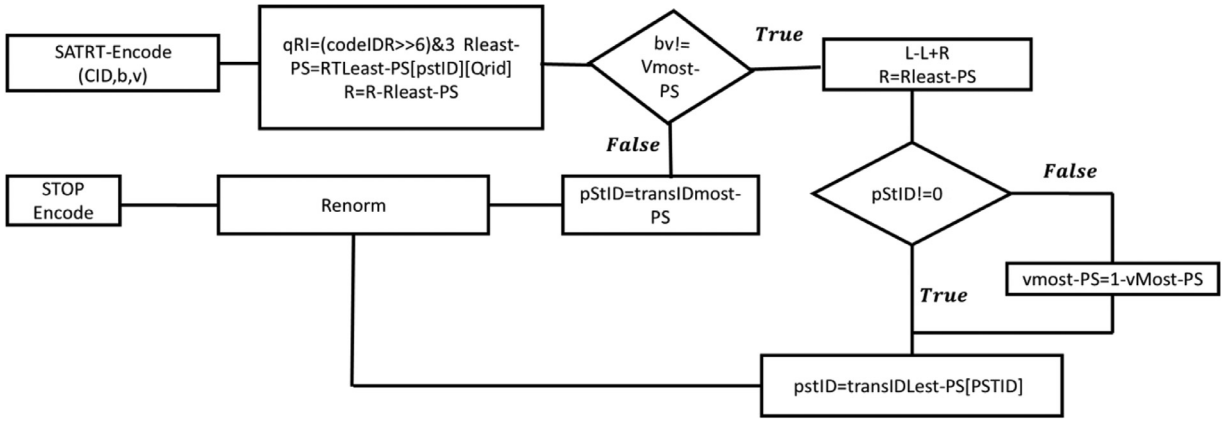


**Fig. 4.** Encoding process.

*3.1.3. Byte Re-Norm (Renormalization)*

the range coders and arithmetic coders utilize the bytes as an output of bit stream of elements and perform the bytes Re-Norm at the time. The byte Re-Norm is represented in flowchart 3. Here X and Y are 32 bit of registers, $2^{3\times2^3}=$ TOP and $2^{2\times2^3}=$ BOTTOM.in next section, we introduces the hardware-efficient M-ABRC and encoding process.

### 3.2. M-ABRC (Modified-ABRC)

*3.2.1. BRC (Bit Reduction Capacity)*

The disadvantage of range coder is needed to utilize the multiplication in the internal division of part. The value represents the bit capacity of $Y\times s$ multiplication, which depends on the *wl* values. We can see that the windows $w=2^3,\ldots\ldots,2^{10}$ need from $32\times6$ *to* $32\times20$bits capacity. In first, these multiplications of bit capacities are very high for the implementation of hardware, specifically for the larger windows. In second, every single window needs its self-precise multiplier, which generates the hardware of architecture with more than one window complex even more than before. To solve these disadvantages we introduced the following method.

Therefore, we can evaluate the register $\mathcal{T}_1 \leftarrow Y \gg 2^3,\ \mathcal{T}_1 > 0$ and utilize in place of $Y$ in multiplication. Here the register $\mathcal{T}_1$ has $3\times2^3$-bit size, in place of $2^5$. Second, we evaluate the register $\mathcal{T}_2 \leftarrow st \gg (2ws-2^3)$, whose size is $2^3$ bits and utilize in place of $st$ in multiplication. As an outcome, for all the windows the multiplication of bit capacity is minimized to $3\times2^3\times2^3$ multiplier, which is required to implement the hardware of architecture with more than one windows. However, we cannot ensure that $\mathcal{T}_2 > 0$ for all possible windows $2^{ws}$ and the states ($st$). This concept can be illuminated in the following way.

From (9) if it satisfies then represents the initial value of $st_0$

$$2^{ws-1}-1 \le st_0 \le 2^{2ws}-2^{ws-1}+1 \tag{11}$$

Then overall-operational time, the $st$ value of low possible is

$$st_{low}=2^{ws-1} \tag{12}$$

*3.2.2. Encoding process*

The arithmetic coding in AVC/H.264 is consists of two parts such as encoding process and Re-Norm. The Encoding process defines the new range. Fig. 4 demonstrates the flowchart of encoding process. When encodes one symbol then the range value is smaller and low value is bigger or equal. For utilizing the arithmetic coding of integer, the value is re-normalized later, each symbol is encoding. The low value is Re-Norm with the help of range value. To update the context information after encoding the symbol.

*3.2.3. Architectural representation of encoding*

The basic architecture of block diagram is shown in Fig. 4, which consists 3 pipeline phases. In this architecture, we separate the operations of low and range into phase1, phase 2. In phase 3 the unit of byte packing can pack the outcomes into the format of byte. Therefore, the architecture can support the 2 encoding modes. As shown in Fig. 4 the phase 1 computes the range value and update the context in probability model. In phase 2 calculates, the low value and generates the output in bits. The phase 3 groups the bits from the output of phase2 and packs them in byte-by-byte manner. Additionally, the bit stuff is done in phase 3 which is explained in Fig. 5.

The block diagram of phase 1 is given below in Fig. 6. The phase 1 has 3 input signals, three intermediate signals and one output for the next phase. The meaning of every symbol is explained as follows:

- S (Symbol)= encode symbol and *st*=state
- Cont (context) = the model of context probability consists the Most-PS and Qe.
- EM (Encoding mode) = specifies the coding in bypass and regular mode.
- The 2 signals is passing to the next phase that means the value which will enhance with low number of output in the process of encoding.
- Cont update= the output is utilized to update the cont information.

For support, the mode of bypass in phase 1, the sig "Add-to-low" and range of the register is controlled by the signal-EM. When bypass the EM, the range will remain un-changed and the signal "Add-to-low" will take the range value. Then the signal of encoding mode will pass to the next phase. In phase 1 the main operation consists the Re-Norm and computation. The experimental outcomes representing the benefits of Modified-ABRC are given in next section of Fig. 7.

## 4. Result analysis

In this section, our methodology is evaluated by comparing the Modified ABRC with the existing methodology. The main intention of our work is to minimize the dynamic power dissipation and area (FPGA). Our proposed methodology is simulated by using the Xilinx version of 14.7 and the code is written using the VHDL. Moreover for the evaluation of Modified ABRC, we have compared it with several existing methodologies mentioned later in the same section and various parameter and constraint has been considered and in each case our methodology outperforms the existing one The result section is parted into two section i.e. power and device

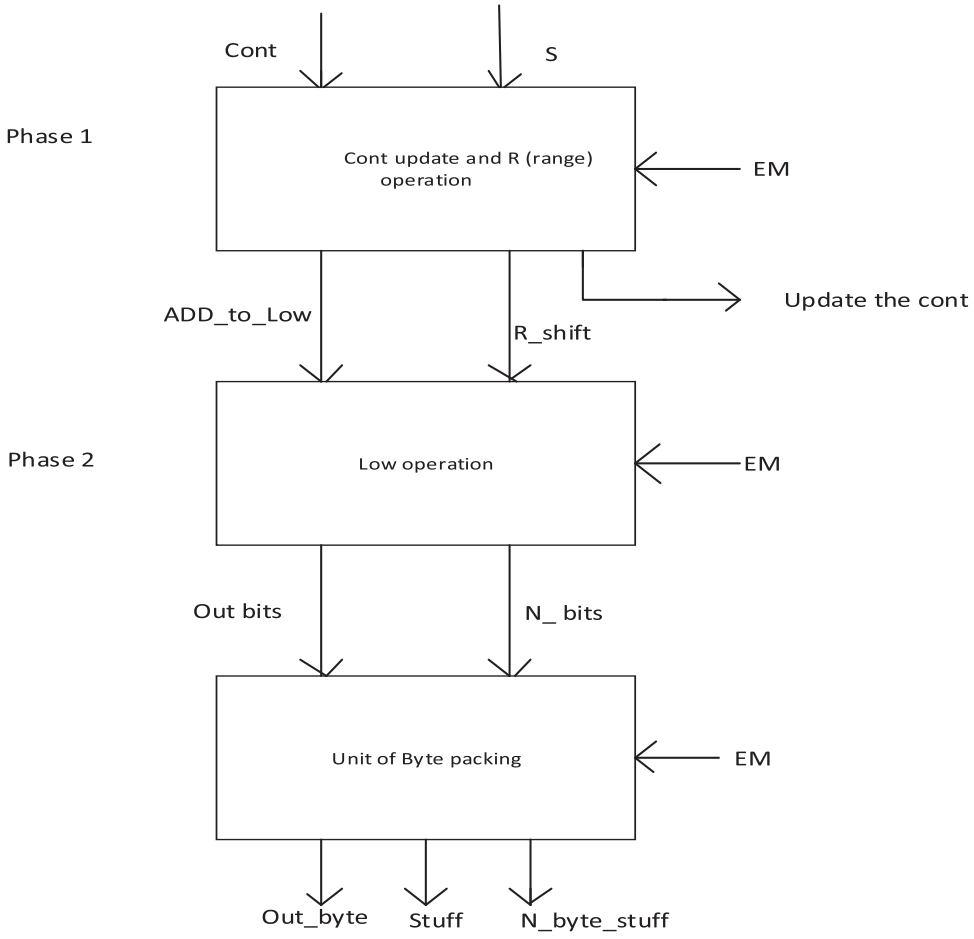


**Fig. 5.** EM architecture of phase-1 symbol.

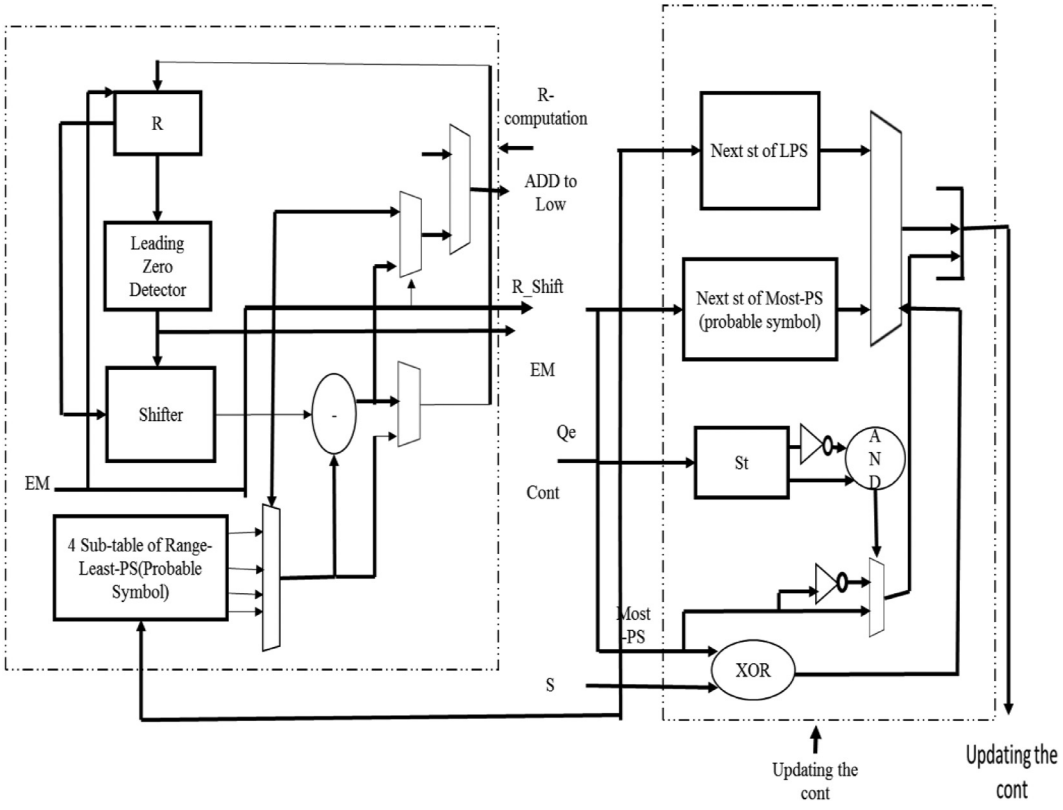


**Fig. 6.** Block diagram of range of EM.

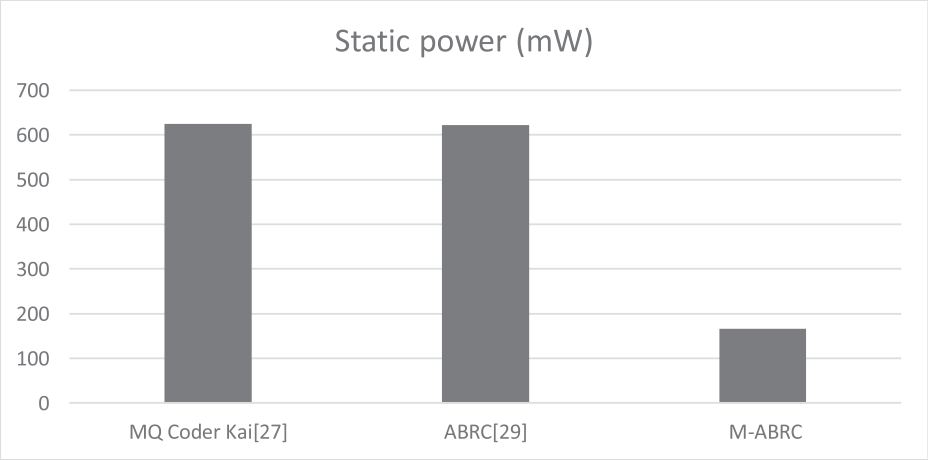


**Fig. 7.** Static Power Comparison.

**Table 1**
Analysis of proposed resizing architecture.

| Architecture | Technology | Logic cell (no.) | Memory (Kbit) |
|---|---|---|---|
| NLS [17] | Xilinx Virtex 2-4 | 10,125 | 432 |
| Bit-plane Parallel SPIHT [18] | Xilinx Virtex 2000E | 83,808 | N/A |
| 1D SPIHT [19] | Xilinx Virtex 6-LX75T | 16,621 | 0 |
| BPS [20] | Xilinx Virtex 5-LX330 | 22,996 | 8.3 |
| JPEG [21] | Xilinx Spartan 3-S200 | 2711 | N/A |
| CL-DCT [22] | Xilinx Spartan 3-S200 | 2385 | N/A |
| STS [23] | Xilinx Zynq Z-7020 | 1017 | 0 |
| MQ Coder [24] | Xilinx Virtex 4-LX80 | 15,692 | 4.17 |
| MQ Coder Dyer [25] | Altera Stratix | 761 | 2675 |
| MQ Coder Dyer [26] | Altera Stratix | 1596 | 8192 |
| MQ Coder Kai [27] | Xilinx Virtex 4- XC4VL | 6974 | 4269 |
| ABRC Shcherbakov [28] | Xilinx Virtex 5- ML507 | 1544 | 552,960 |
| ABRC [29] | Xilinx Virtex 4-LX80 | 1688 | 0 |
| ABRC [30] | Altera Stratix | 1296 | 0 |
| Proposed ABRC | Xilinx Virtex 4- XC4VF | 885 | 0 |

**Table 2**
Power dissipation Comparison with various architecture.

| Architecture | MQ Coder | CL-DCT | STS | ABRC | M-ABRC |
|---|---|---|---|---|---|
| Frequency (MHz) | 48.30 | 66.4 | 96 | 105.92 | 142.95 |
| Dynamic power (mW) | 488.67 | 96 | 74 | 127.05 | 16.75 |
| Normalized power (mW/MHz) | 10.117 | 1.45 | 0.77 | 1.19 | 0.117 |
| Power density ($\mu$W/(MHz $\times$ Logic cell no.)) | 0.65 | 0.60 | 0.75 | 0.71 | 0.13 |

utilization. Device utilization is discussed through the Table 1 and the power is influenced using the Table 2.

### 4.1. Device utilization

The below table i.e. Table 1 gives the outcomes of M-ABRC when FPGA implementation takes place, here the different architecture is compared. Through this table, we see that first column depicts the various architecture, whereas second column depicts the technology that has been used. All these architecture given in Table 1 has been implemented on Xilinx FPGA. Similarly, third column gives the Logic cell or the area, which is used. Logic cell is one of the best parameter that has been used in order to compare the various resources which is basically used by the FPGA technologies. Moreover, it is observed that any modification in the image size have no impact on the given hardware resource of M-ABRC at given and fixed block size. From the Table 1 it is clear that M-ABRC (Modified ABRC) consumes less amount of resources when compared to the architecture based on JPEG [21]and SPIHT [18]. We see that the logic cell number required by the JPEG is almost three times more than the M-ABRC. Whereas in case of NLS [17], NLS requires more than 11 times of the area compared to the M-ABRC. Similarly, we see that MQ-Coder [24], MQ-Coder [25] and MQ Coder [26] requires almost 2 times, 8 times and 1.5 times more logic cell than the M-ABRC respectively. Bit-PP (plane Parallel) SPIHT [18], 1DSPIHT [19] and BPS [20] requires the amount of area when compared to the M-ABRC that are described in Figs. 8 and 9.

### 4.2. Power dissipation

Table 2 shows the comparative analysis between the MQ Coder, CL-DCT, STS, ABRC and M-ABRC. In Table 2, second row depicts the maximum amount of operating frequency for each methodology. Maximum operating frequency for MQ Coder is 48.30(in MHZ), 66.4 MHz for CL-DCT, 96 for ABRC and for the proposed method (M-ABRC) the maximum operating frequency is 142.95 MHz which is comparatively high than the other, this shows the efficiency of M-ABRC. In second row, the comparative analysis is given on the parameter dynamic power. MQ Coder, CL-DCT, STS, ABRC Possesses the Dynamic Power of 488.67 mW, 96 mW, 74 mW, 127.0 mW, whereas our proposed model i.e. M-ABRC possesses only 16.75 mw of dynamic power. The less the dynamic power, the more efficient the model is, hence our methodology outcast the other methodol-

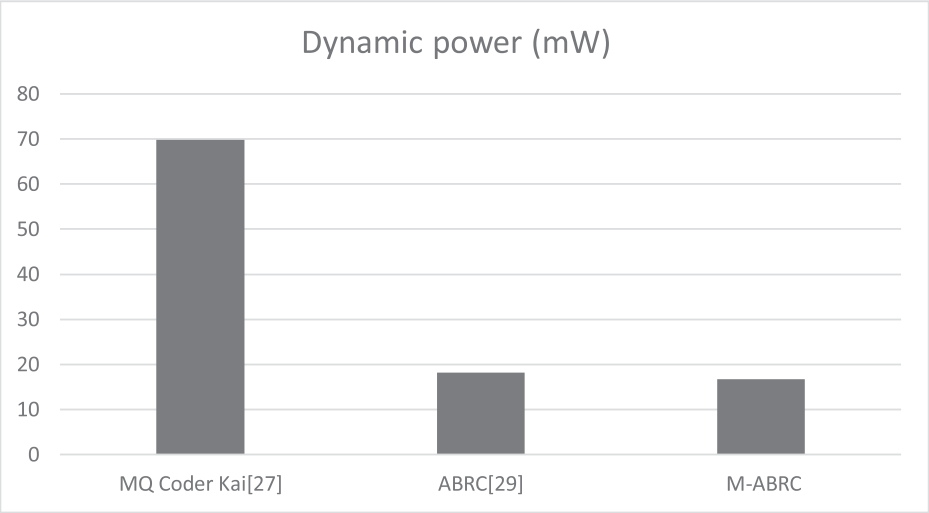


**Fig. 8.** Dynamc PowerComparison.

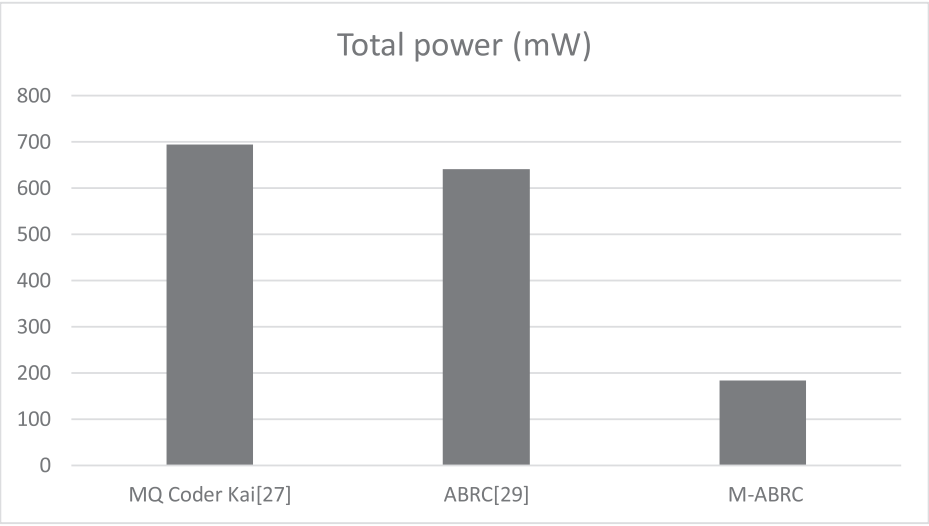


**Fig. 9.** Total Power Comparison.

ogy in terms of Dynamic power. Third column shows the normalized power and it should be evident that the normalized power possession should be as less as possible. Here the comparison analysis shows that M-ABRC possesses 0.117 MHZ which is comparatively lesser than the other model, such as MQ Coder, CL-DCT, STS and ABRC possesses 10.117, 1.45,0.7,1.19 (in MHZ) respectively. This shows that our model has lower normalized power dissipation. At last, the comparison analysis can be done based on power density, which is given in fifth row of Table 2. We see that MQ Coder requires the 0.65 µW, whereas CL-DCT, STS, ABRC requires the 0.60, 0.75 and 0.715(in mW) respectively, whereas M-ABRC possesses 0.13 $\mu$W. The comparative analysis of power dissipation shows that ABRC performs better than MQ Coder, CL-DCT, STS, and ABRC; however, we observe that our model performs slightly marginally better than these entire model including the M-ABRC.

### 4.3. Static power

Static power is defined as the power consumed when no activity takes place in the circuit. The low static power of any particular model indicates that the model is better, i.e. the low the static power dissipation, the more efficient model it is. In order to prove the efficiency of our model, we have considered this parameter which is given in below figure. For the comparison analysis we have taken two best model i.e. MQCoder Kai [27] and ABRC [29] and it is compared with our model M-ABRC. Moreover, the static power dissipation of the other two model is 624.68 and 622.55 respectively. When these model are compared, observation can be made that ABRC performs slightly better whereas proposed model is marginally better than the other two. Static power dissipation of our proposed model is 166.71 mW.

### 4.4. Dynamic power

Dynamic power is defined as the power, which is consumed when the given inputs are in active state. Dynamic power Dissipation is considered as one of the key parameter while comparing the model. In here when the below figure is observed we see that MQ-Coder [27] possesses the 69.81 mW and ABRC possesses 18.15 mW, this shows the marginal improvement in the ABRC model. However, M-ABRC performs slightly better than the ABRC i.e. it possesses 16.75 mw.

At last, the comparative analysis takes place based on total power required to perform the task. For the model to be efficient, the total power requirement should be as less as possible. In the above graph of total power we observe that the two existing model i.e. MQ Coder Kai [22] and ABRC possesses 694.49 mW and 640.7 mW of total power respectively, whereas our model needs

only 183.46 mW, which shows the efficiency of our model and observation can be made that our model simply outperforms the other two models.

## 5. Conclusion

In this paper, we have presented a model known as M-ABRC (Modified ABRC), this paper presented the model along with the hardware architecture. The proposed model is LUP (Look UP Table) based which helps in reduction of bit capacity multiplication, this model helps in achieving the higher compression performance and it provides comparatively faster adaptation at the initial stage (both encoding as well as decoding). In order to evaluate our method we compared our method with the existing methodology in terms of device utilization and power dissipation. We see that in device utilization, the area (FPGA) required by the JPEG is almost three times more than the M-ABRC. Whereas in case of NLS, NLS requires more than 11 times of the area compared to the M-ABRC. Similarly, we see that several methodology of MQCoder requires almost 2 times, 8 times and 1.5 times more logic cell than the M-ABRC respectively. Bit-PP (plane Parallel) SPIHT [20], 1DSPIHT [22] and BPS [21] requires the amount of area when compared to the M-ABRC. We see that MQ Coder requires the 0.65 µW, whereas CL-DCT, STS, ABRC requires the 0.60, 0.75 and 0.715 (in mW) respectively, whereas M-ABRC possesses 0.13 µW. In terms of static power, we see that MQ Coder requires the 0.65 µW, whereas CL-DCT, STS, ABRC requires the 0.60, 0.75 and 0.715 (in mW) respectively, whereas M-ABRC possesses 0.13 µW. In terms of Dynamic power, M-ABRC performs slightly better than the ABRC, i.e. it possesses 16.75 mw. MQ Coder Kai [22] and ABRC possesses 694.49 mW and 640.7 mW of total power respectively, whereas our model needs only 183.46 mW, which shows the efficiency of our model.

## Declaration of Competing Interest

The authors have no affiliation with any organization with a direct or indirect financial interest in the subject matter discussed in the manuscript.

## Supplementary materials

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.micpro.2019.102901.

## CRediT authorship contribution statement

**S.T. Mrudula:** Conceptualization, Data curation, Validation, Writing - original draft, Writing - review & editing. **K.E. Srinivasa Murthy:** Conceptualization, Data curation, Validation, Writing - original draft, Writing - review & editing. **M.N. Giri Prasad:** Conceptualization, Data curation, Validation, Writing - original draft, Writing - review & editing.

## References


[1] I.H. Witten, R.M. Neal, J.G. Cleary, Arithmetic coding for data compression, Commun. ACM 30 (1987) 520–540.
[2] J.L. Mitchell, W.B. Pennebaker, IPEG Still Image Data Compression Standard, Van Nostrand Reinhold, New York, 1993.
[3] Z. Huang, F. Huang, J. Huang, Detection of double compression with the same bit rate in MPEG-2 videos, in: 2014 IEEE China Summit & International Conference on Signal and Information Processing (ChinaSIP), Xi'an, 2014, pp. 306–309.
[4] Jiangtao Wen, Hyungjin Kim, J.D. Villasenor, Binary arithmetic coding with key-based interval splitting, IEEE Signal Process Lett 13 (Feb. (2)) (2006) 69–72.
[5] A. Rodríguez, L. Santos, R. Sarmiento and E. de la Torre, Scalable Hardware-Based on-Board Processing for Run-Time Adaptive Lossless Hyperspectral Compression, in IEEE Access.
[6] Z. Huang, F. Huang, J. Huang, Detection of double compression with the same bit rate in MPEG-2 videos, in: 2014 IEEE China Summit & International Conference on Signal and Information Processing (ChinaSIP), Xi'an, 2014, pp. 306–309.
[7] T. Bernatin, G. Sundari, Video compression based on hybrid transform and quantization with Huffman coding for video codec, in: 2014 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT), Kanyakumari, 2014, pp. 452–456.
[8] Video Coding for Low Bitrate Communications, Version 1, 1995 ITU-T, ITU-T Recommendation H.263.
[9] E. Belyaev, A. Turlikov, K. Egiazarian, M. Gabbouj, An efficient adaptive binary arithmetic coder with low memory requirement, IEEE J. Sel. Top. Signal Process. 7 (6) (2013) 1053–1061.
[10] E. Belyaev, A. Turlikov, K. Egiazarian, M. Gabbouj, An efficient multiplication-free and look-up table-free adaptive binary arithmetic coder, in: 2012 19th IEEE International Conference on Image Processing, Orlando , FL, 2012, pp. 701–704.
[11] E. Belyaev, S. Forchhammer, K. Liu, An adaptive multialphabet arithmetic coding based on generalized virtual sliding window, IEEE Signal Process. Lett. 24 (July (7)) (2017) 1034–1038.
[12] Y. Fang, LDPC-Based lossless compression of nonstationary binary sources using sliding-window belief propagation, IEEE Trans. Commun. 60 (November (11)) (2012) 3161–3166.
[13] M. Tausif, N.R. Kidwai, E. Khan, M. Reisslein, FrWF-Based LMBTC: memory–Efficient image coding for visual sensors, IEEE Sens. J. 15 (Nov. (11)) (2015) 6218–6228.
[14] Shcherbakov, N. Wehn, A parallel adaptive range coding compressor: algorithm, FPGA prototype, evaluation, in: 2012 Data Compression Conference, Snowbird, UT, 2012, pp. 119–128.
[15] T. Strutz, Lossless intra compression of screen content based on soft context formation, IEEE J. Emerg. Select. Top. Circ. Syst. 6 (Dec. (4)) (2016) 508–516.
[16] D. Marpe, H. Schwarz, T. Wiegand, Context-based adaptive binary arithmetic coding in the H.264/AVC video compression standard, IEEE Trans. Circ. Syst. Video Technol. 13 (7) (July 2003) 620–636.
[17] F.W. Wheeler, W.A Pearlman, SPIHT image compression without lists, in: IEEE 2000 International Conferenceon Acoustics, Speech, and Signal Processing, İstanbul, Turkey. New York, NY, USA, IEEE, June 2000, pp. 2047–2050.
[18] T.W. Fry, S.A. Hauck, SPIHT image compression on FPGAs, IEEE T. Circ. Syst. Vid. 15 (2005) 1138–1147.
[19] S. Kim, D. Lee, J.S. Kim, H.J Lee, A high-throughput hardware design of a one-dimensional SPIHT algorithm, IEEET Multimed. 18 (2016) 392–404.
[20] Y. Jin, H.J Lee, A block-based pass-parallel SPIHT algorithm, IEEE T. Circ. Syst. Vid. 22 (2012) 1064–1075.
[21] G. Wallace, The JPEG still picture compression standard, IEEE T. Consum. Electr. 38 (1992) 18–34.
[22] M.L. Kaddachi, A. Soudani, V. Lecuire, K. Torki, L. Makkaoui, J.M Moureaux, Low power hardware-based image compression solution for wireless camera sensor networks, Comp. Stand. Inter. 34 (2012) 14–23.
[23] L.O.N.E. Rafi, Mohd, Najeeb-ud-Din HAKIM, FPGA implementation of a low-power and area-efficient state-table-based compression algorithm for DSLR cameras, Turkish J. Electrical Eng. Comput. Sci. 26 (6) (2018) 2928–2943, doi:10.3906/elk-1804-208.
[24] K. Liu, Y. Zhou, Y.S. Li, J.F Ma, A high performance MQ encoder architecture in JPEG 2000, Integration 43 (3) (2010) 305–317.
[25] M. Dyer, D. Taubman, S. Nooshabadi, A. Gupta, Concurrency techniques for arithmetic coding in JPEG2000, IEEE Trans. Circ. Syst. I 53 (6) (2006) 1203–1213.
[26] M. Dyer, D. Taubman, S. Nooshabadi, A. Gupta, Concurrency techniques for arithmetic coding in JPEG2000", IEEE Trans. Circ. Syst. I 53 (6) (2006) 1203–1213.
[27] K. Liu, Y. Zhou, Y. Song Li, J. Feng Ma, A high performance MQ encoder architecture in JPEG2000, Integr. VLSI J. 43 (3) (2010) 305–317.
[28] I. Shcherbakov, N. Wehn, A parallel adaptive range coding compressor: algorithm, FPGA prototype, evaluation, Data Compression Conference, 2012.
[29] E. Belyaev, K. Liu, M. Gabbouj, Y. Li, An efficient adaptive binary range coder and its VLSI architecture, IEEE Trans. Circ. Syst. Video Technol. 25 (Aug. (8)) (2015) 1435–1446.
[30] E. Belyaev, K. Liu, M. Gabbouj, Y. Li, An efficient adaptive binary range coder and its VLSI architecture, IEEE Trans. Circ. Syst. Video Technol. 25 (Aug. (8)) (2015) 1435–1446.


**S.T. Mrudula** is working as an Assistant Professor in the department of Electronics and Communication Engineering of Intell Engineering College, Anantapur, Andhra Pradesh, India. She has Eight years of teaching experience. She has received her B.Tech degree in 2008, M.Tech degree in 2012. Currently she is pursuing PhD in the area of VLSI and signal processing, JNT University, Anantapur, Andhra Pradesh, India. Her areas of interest includes VLSI, signal processing. She has published several papers in national and international journals. She is a life time member in IETE.

**K. E. Srinivasa Murthy** is currently working as Head of the Department, Dept of E.C.E, G Pullaiah college of engineering and Technology, Kurnool, Andhra Pradesh, India. K. E. Srinivasa Murthy received his B.Tech from S.V.University, Andhra Pradesh in 1989, M.Tech degree from S. V. University, Andhra Pradesh in 1992 and PhD degree from S.K.University, Andhra Pradesh in 2003 .He has more than 25 years of teaching experience .His areas of interest include Embedded systems, Microcontrollers. He authored several national, international journals and conference manuscripts. He is life time member of ISTE and Instrumentation Society of India. He is member of IEE and IETE.

**M. N. Giri Prasad** received his B. Tech degree from JNTU College of Engineering, Anantapur, Andhra Pradesh, India in 1982, M. Tech degree from Sri Venkateswara University, Tirupati, Andhra Pradesh, India in 1994, and PhD degree from J.N.T University, Hyderabad, Andhra Pradesh, India in 2003. He is having more than 30 years of teaching experience. Presently he is working as Director of Admissions, JNTUA, and Professor in the Department of Electronics and Communication Engineering at JNTUA College of Engineering, Anantapur, Andhra Pradesh, India. His research areas are Wireless Communications, Biomedical Instrumentation, signal processing, Image processing, embedded systems and microcontrollers. He has published more than 25 papers in national and international conferences. Around 50 papers published in national and international journals. He is a life member of ISTE, IEI and NAFEN.