Indonesian Journal of Electrical Engineering and Computer Science
Vol. 10, No. 1, April 2018, pp. 176~183
ISSN: 2502-4752, DOI: 10.11591/ijeecs.v10.i1.pp176-183

Comparison of Entropy Coding mechanism on IEEE1857.2 Lossless Audio Compression Standard

Fathiah Abdul Muin[1], Teddy Surya Gunawan[2], Mira Kartiwi[3], Elsheikh M.A. Elsheikh[4]
[1,2,4] Department of Electrical and Computer Engineering, Kulliyyah of Engineering, Malaysia
[3] Department of Information Systems, Kulliyyah of Information and Communication Technology
International Islamic University Malaysia, Jalan Gombak, Kuala Lumpur, Selangor, Malaysia

Article Info

Article history:

Received Jan 7, 2018
Revised Mar 9, 2018
Accepted Mar 25, 2018

Keywords:

Arithmetic entropy algorithms
Entropy coding
Golomb-Rice Coding

ABSTRACT

This paper has two objectives. First, we aim to review and analyze the performance of the IEEE 1857.2 standard, focusing is on the Golomb-Rice and Arithmetic entropy algorithms as well as the effect of the pre-processing block on these entropy blocks. The pre-processing block normalizes the error residue of the Linear Predictive encoder, which then is passed to Entropy block, where the selector chooses the entropy encoder to use. The second objective is to present results from experimenting different existing algorithms available to benchmark it'. The results are discussed, and comparisons are made to identify the effect on compression ratio and encoding speed of the lossless encoder. As well as this, comparison is made to analyze effects of enabling and disabling the pre-processing output to the Entropy Coding block. We concluded that pre-processing block works well to flatten the output at lower predictor order for all the sound types, but works best at improving the residual output for music sound type.

Corresponding Author:

Teddy Surya Gunawan,
Department of Electrical and Computer Engineering, Kulliyyah of Engineering,
International Islamic University Malaysia,
Jalan Gombak, Kuala Lumpur, Selangor, Malaysia.
Email: tsgunawan@iium.edu.my

1. INTRODUCTION

Moving forward from the first innovation of audio compression, the demand for higher quality data has risen throught various fields like medical innovations such as ECG analysis or heat beat analysis, in terms of audio data storage and audio streaming, this inevitably leads to a demand of lossless audio compression as generally high-quality data requires not only high data storage, but also high data rate for transmission and analysis [1]. The idea behind lossless audio compression is to remove redundant bits of data without any loss of quality, by using an encoder containing a predictor which can predict the signal as close as possible and calculating the error of this predicted signal. This prediction residue is finally encoded with both its predictor and various other necessary parameters. When the user receives this encoded data, it uses a decoder that can reconstruct the original signal. On top of this, the encoder will encode the residual error using an entropy coding, which is a part of information theory to further reduce the number of redundant bits [2]. The smaller the residual error given from the predictor, the more efficient the entropy coding is and the smaller the compressed signal as it will contain more zeroes in its error residual.

Overall, the predictor and entropy coding are essential parts of any lossless compression mechanism. In entropy coding, a bitstream may be compressed by removing redundant bits through a generalized algorithm. In tthe IEEE1857.2 lossless audio compression standard, both Golomb-Rice and arithmetic coding is mechanism is defined, thus in this paper, we wish to compare the performance of these compressor with respect to the IEEE1857.2 predictor algorithm and the other algorithms.

In most studies and description of lossless audio compression, the focus is more on prediction block, however, entropy coding is just as important in this case as it used as a tool to write the intended compressed bitstream through its algorithm. Generally, the purpose of entropy coding is not limited to audio per say but as a universal coding mechanism to compress bitstream data. Nevertheless, through this paper we will compare two entropy coding mechanism specifically designed for the IEEE1857.2 lossless audio compression and to find which universal entropy coding may be more suited for its application. In the previous paper, we had evaluated the performance of the IEEE1857.2 Linear Predictor Coding (LPC) block and found interesting relationships of the LPC block with the preprocessing block as well, thus this study of the preprocessing block is also extended into this paper to it's effect on the different entropy mechanism of the IEEE1857.2 standard [3].

In the rest of this paper, for section 2, we will discuss Golomb-Rice Coding mechanism and subsequently Arithmetic Coding in section 3. Then, in section 4, we will divide into 4 subsections, were we will detail components of the IEEE1857.2, by revisiting the pre-processing block cocept, entropy selection, Golomb-Rice encoding, and arithmetic encoding algorithms used. Following that, we will describe the experimental setup and measurement in section 5, as well as the results and discussion of this experimentation in section 6. Then finally conclude the paper in section 7.

2. GOLOMB RICE CODING

Golomb-Rice Coding is widely used for various Lossless Audio Compression popular tools and standards, such as FLAC and MPEG-4 ALS [4],[5]. The reason being is that Golomb-Rice coding is a derivation of Huffman Coding which is suited for time dependant applications, thus useful for improving the encoding speed, but with the expense of compression ratio [6]. In the later chapters, we will verify whether this is applicable to lossless audio compression applications.

Firstly, the way this method is executed is by giving a unique parameter m; Then a positive integer n, which we wish to encode is divided into a remainder of $n \, mod \, m$ and quotient, n/m [7]. If $m = 2^k$, the code word for n will consists of k-least significant bits of n, followed by the number formed by the remaining most significant bits of n in unary representation and a stop bit [2]. The length of this code word is,

$$k + 1 + (n/2^k) \quad (1)$$

During the earlier invention of lossless audio compression, the estimation for the parameter k is given in and is used in AudioPaK and it is based on the expectation $E(|\mathrm{e}(\mathrm{n})|)$ already computed in the predictor block where,

$$k = log_2 E(|e(n)|) \quad (2)$$

The parameter k is defined to be constant over an entire frame and takes values between 0 and (b −1) in the case of b bit audio samples [2]. This concept is still widely used in current existing codecs and standards. Nevertheless, in terms of the lossless audio compression block, it is important to bear in mind that the Golomb codes defined to be optimal for exponentially decaying probability distributions of positive integers and because the prediction residuals may not all positive, it maybe required to map the error residual to an unsigned value as defined in the equation below [8]:

$$e_i(n) = \begin{cases} 2e_i(n), & e_i \geq 0 \\ 2e_i(n), & Otherwise \end{cases} \quad (3)$$

3. ARITHMETIC CODING

Another alternative for universal encoding mechanism in Lossless Compression is arithmetic coding. Firstly, in a nutshell, arithmetic coding calculates the probabilities of occurance of each symbol in a message over a finite alphabhet. This method does so by incorporating two vaariables, L and R, where L is the smallest binary value consistent with the code representing the symbols found so far, whilst R is the product of probabilities of the symbols found. The simple mechanism of arithmetic coding can be described as the following sequence, with steps 2 to 4 occuring recursively as new symbols is processed [9].

1) Initialization of $L = 0$ and $R = 1$
2) Encode next symbol (jth of the alphabet) by $L = L + R \sum_{i=1}^{j-1} p_i$
3) New $R = R \times p_j$
4) Output sequence bit c, for each bit sequence between L and $L + R$.

Where p_i and p_j is the probability of the ith (current) and jth (next) symbol in the alphabet respectively.

4. IEEE 1857.2 STANDARD

4.1. Preprocessing Block

As mentioned in the previous paper, one of the advantages of the IEEE 1857.2 lossless compression is that the bit error introduced during transmission can be minimized by the pre-processing block [3]. This is because the audio frames can be decoded at one frame interval since the information of a frame are independent of each other. However, the prediction residues will be bigger compared to the rest of the frame content, resulting in the increase of the prediction residues dynamic range. To compensate for this, the entropy encoder then increases both the calculation complexity and the alphabet size.

The pre-processor block intends to overcome this situation by taking the prediction residues and downshift them to a certain degree. This operation allows the amplitude to decrease and the envelope of the prediction residues is flattened by adaptive normalization, which was also proved to improve the compression rate on MPEG 4-ALS [8]. The residual sample are quantized into power of 2 using the PARCOR coefficient k to make sure that it is integers for lossless coding as well normalized by down-shifted with E_p as followed [10], [11]:

$$\frac{E_i}{E_p} = \prod_{p=i+1}^{P} \frac{1}{1-k_p^2} \tag{4}$$

$$e(i) = \left\lfloor \sum_{j=i+1}^{P} \log_2 \frac{1}{1-k_p^2} + 0.5 \right\rfloor \tag{5}$$

The term within the summation are pre-computed by RA_shift and RA_shift12 fixed table, which were provided by the standard for fast computation and device portability.

4.2. Entropy Coding Selection

As mentioned previously, the IEEE1857.2 standard has a selection of two types of enthropy coding, which are Golomb-Rice Coding and Arithmetic Coding and it utilizes one bit in the output bitstream to switch between the two methods for the decoder to recognize which method was used in the encoding process [12]. This is illustrated in the Figure 1 below, so the coder will select the entropy type to use and encode this selection as well. In the standard it mentions that this selection occurs depending on the environment of the users, but for simplicity in this study, we explicitly select the entropy type by parsing an argument to the tool.

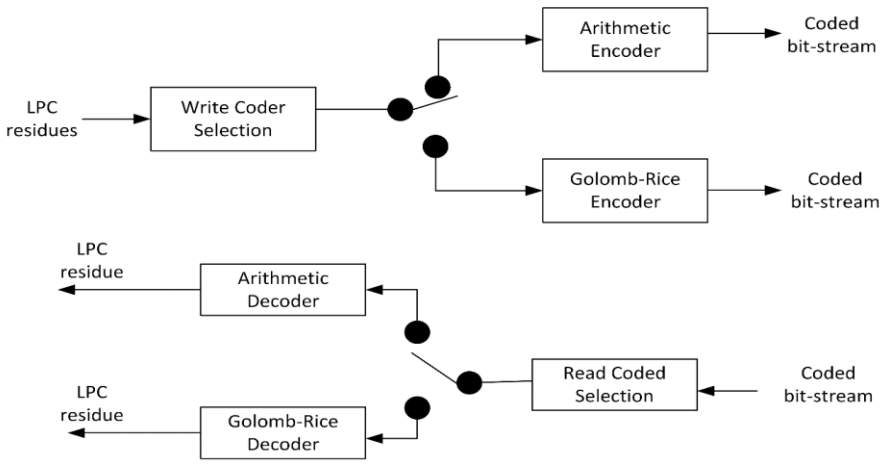

Figure 1. Encoder Coder Selection (above) and Decoder Coder Selection (below)

4.3. Adaptive Golomb Encoding

For the Golomb-Rice Entrypy selection, the standard utilizes an adaptive Golomb-Rice mechanism, an advanced method, whereby unlike it predecessors, it doesn't only update the Rice parameter (leadzero, division and remainder bits), but it also updates the m value after each block is processed, which is fixed in the original Golomb-Rice method.

Firstly, in the encoder, the LPC residue is divided into subblocks of 2^x, where x can be selected to be between 1 and up to 5. Then initial m is calculated by the mean (μ) of the LPC residual error, by the following equation,

$$m = \begin{cases} log_2(\mu) + 0.5, & mean > 1 \\ 0, & mean \leq 1 \end{cases} \quad (6)$$

when each subblock is encoded, the m is updated based on the parameter RICE_NUM_MUL = 32 after the Golomb-Rice Encoder. Figure 1 above, illustrates this updating process of m and the following equations describes how m is updated each time by each subblock,

$$m = \begin{cases} 0, & sum = 0 \\ m, & ricevalue2 \leq sum \leq ricevalue2 \\ m + log_2(2.sum - 1) - log_2(ricevalue2), & sum > ricevalue2 \\ m - min(m, log_2(sum + ricevalue2)), & 0 < sum < ricevalue2 \end{cases} \quad (7)$$

$$ricevalue1 = 2^m.RICE\_NUM\_MUL \quad (8)$$

$$ricevalue2 = 2^{m+1}.RICE\_NUM\_MUL \quad (9)$$

The sum in this case is the Golomb-Rice cumulative sum which is calculated based on the each subblock residual cumulative sum that is processed at that time.

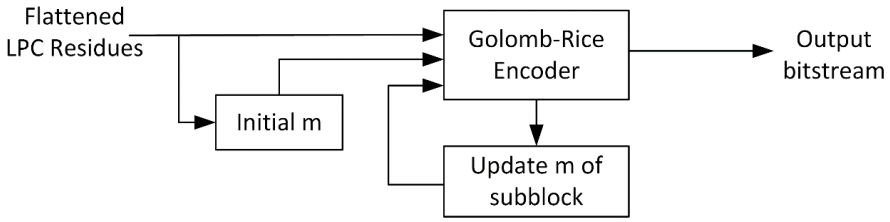

Figure 2. Golomb-Rice Encoder Process

4.4. Arithmetic Encoding

Next, for the Arithmetic Encoding selection of IEEE1857.2 block it is also similar in it's subblocking of the LPC residue, but x only goes up to 3. Firstly, in this block, the mean of the flattened residue is computed, which then is logarithmicly quantized by the following equation,

$$index = log_2 \mu + 0.5 \quad (10)$$

After this, this quatized mean index is locally dequantized and then used to generate the Probabability table from a set of probability template. The following equation is used in this scaling procedure,

$$p(s) = f(\lfloor s/\bar{\mu} + 0.5 \rfloor) \quad (11)$$

The probability template is essentially a set of probability density values from a trained audio data, which is approximately a Gaussion function (mean -0.1 and standard deviation 0.6), but unique to the IEEE1857.2 defined standard. Previous work has shown that this trained table can achieve a lossless compression performace better than that of the Gaussion function, thus this table used [10].

Figure 3 describes the overall process of the Arithmetic encoder. Also defined in this process is the MSB/LSB split block, which is an advantage to the Golomb-rice method. This is because the Arithmetic

Encoder can further split the LPC residue to MSB and LSB parts so that in can further increase the efficiency of the Arithmetic Encoding Algorithm, in the case of trailing bits.

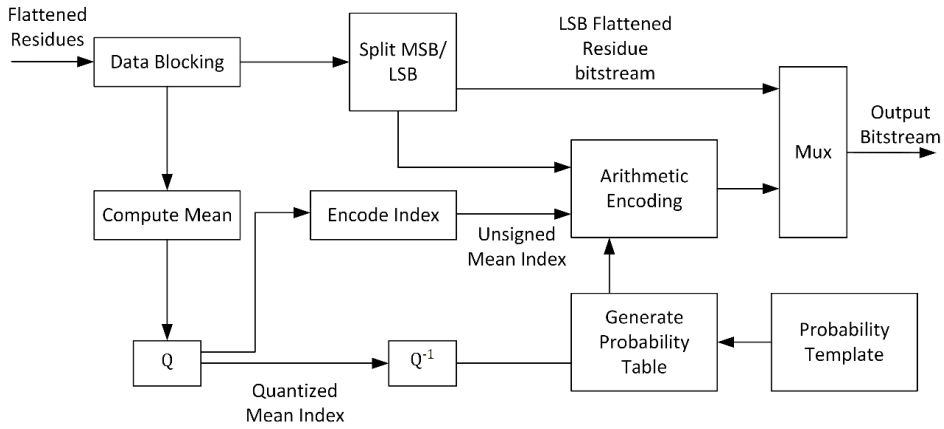

Figure 1. Arithmetic Coding Encoder Block Diagram

5. EXPERIMENTAL SETUP AND MEASUREMENT

The analysis was conducted in a combination of MATLAB application and (.exe) file compiled from C on a DELL laptop, which was running with Intel core i7 processor to measure the performance of each tool. The database consisted of Audio Books which were atleast 1 hour long with a combination of English and Arabic book recordings. Table 1 shows a sample of the Audio books database which were used in the investigation.

Table 1. Sample Audio Book Database

Type	Filename	Playback Length
English Audio Book	ArtOfWar8k.wav	1:12:18
Arabic Audio Book	Book (Amin), 20161220.wav	1:00:43
Qur'anic Recitation	Al-Kahf(Amin), 20161219.wav	1:02:07

For the measurements, one of the basic measurement for a lossless audio codec is the compression ratio, which is measured by the comparison of the output file to the original raw file with the following formula:

$$Compression\ ratio\ = \frac{Input\ Size}{Output\ Size} \tag{12}$$

This compressibility determined the proportionality of bits in which were compressed, the larger this ratio, the more bits were compressed [5]. As well as this, it is also important to ensure that the rate of compressing is efficient, thus encoding speed will also need to be measured. The encoding/decoding speeds are measured in terms of how fast the encoder processes relative to the total length of the Audio file in MB. This is defined by the following equations:

$$Encoding\ Speed\ = (\frac{Total\ length\ of\ file\ in\ MB}{Average\ Encoding\ Time})\ MB/s \tag{13}$$

6. RESULTS AND ANALYSIS

In this experiment, the internal settings of the IEEE1857.2 entropy tool was compared by explicitly adding a parsed option to the tool selecting either Golomb Rice (GR) or Arithmetic Coding (AC), as well as enabling and disabling of the Preprocessing for each Entropy Coder. Additionally, it is also benchmarked with the latest FLAC and MPEG-ALS with varying predictor order to find how well each algorithm works

with higher precision. For a fair comparison, the tool was recreated in C code on top of AVS China Codec and verified by comparing the output file of the decoder to the original file.

The following are the options with its description which are parsed to each of the tool:

- FLAC:
  - f: Force overwrite if output file name exists
  - l <max_predictor_order>: Set max predictor order of fixed model
- MPEG-4 ALS:
  - n<frame_size>: Set frame size
  - o<predictor_order>: Set predictor order
- IEEE 1857.2:
  - f<1>: Audio Storage File output format
  - p<predictor_order>: set predictor order

From the Graph in Figure 5, we can see that the preprocessing block does improve the performance of Entropy coding for the Golomb-Rice Block with higher predictor order by atleast 1.29% from predictor order 14. This improvement increases with predictor order value. However, the same can't be concluded for Arithmetic Coding, as there is no clear pattern on whether the preprocessing method improves the Arithmetic Coding or not.

In terms of Compression ratio, the preprocessing block does improves the compression ratio by atleat 0.37% for the Golomb-Rice Coding in Figure 4, however very insignificant improvement is seen with the Arithmetic block as well as this the performance somehow worsen at predictor order 16 and 20.

Comparing the values of the encoding speed and compression ratio for Arithmetic Coding and Golomb-Rice Coding itself, the Golomb-Rice method has a much faster encoding speed compared to the Arithmetic Coding mechanism, where as the Arithmetic Coding has better compression ratio overall to the Golomb-Rice Coding for lossless audio compression application. This is consistent to the previous work which compares the two methods for image compression application.

In addition to this comparison, another interesting finding was found when comparing the database to the other Algorithms. As seen in the Figure 6, despite previous work [10], the MPEG-ALS has the the best compression ratio, where as the FLAC fastest in terms of encoding speed. Overall FLAC and MPEG-4 ALS algorithms surpasses the IEEE1857.2 compression ratio, as well as the encoding speed as well. One possibility for this occurrence could be related to the fact that the database used are all Audio books files, where as the previous work were all music files, with shorter length. This explains why MPEG-4 ALS is the best as the standard uses long term LPC residue prediction which is more suited for speech files, where as none of there other algorithms contain this [13]. The FLAC comes in a close second, but it also contains an audio detection which switches between different types of speech modelling other than LPC, thus allowing better error residue compared to IEEE1857.2. Additionally, from the FLAC code, it uses its own syscall POSIX interface for read/write protocal, which allows faster read and write speed to the stdlib interface. This could be the reason behind it's significant difference in encoding speed to the others [4].

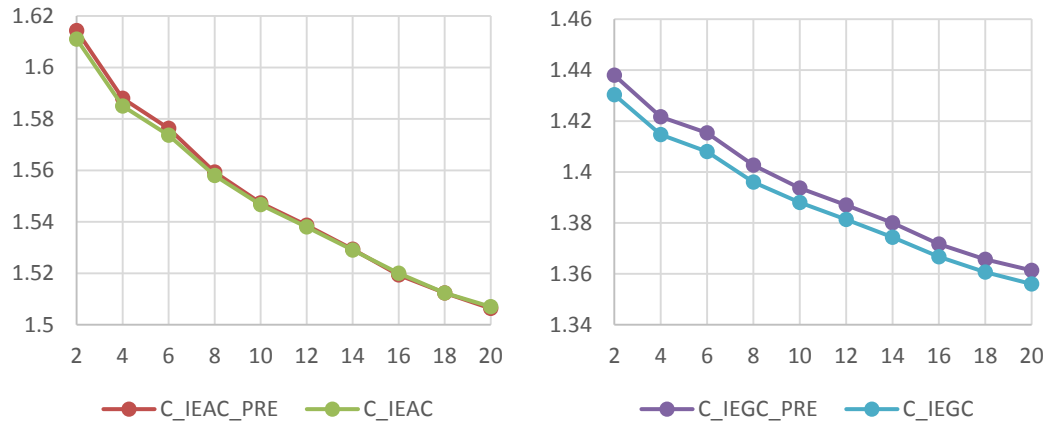

Figure 2. Comparison of Arithmetic Coding (left) and Golomb Rice Coding (right) Compression Ratio by Enabling and Disabling Preprocessing Block

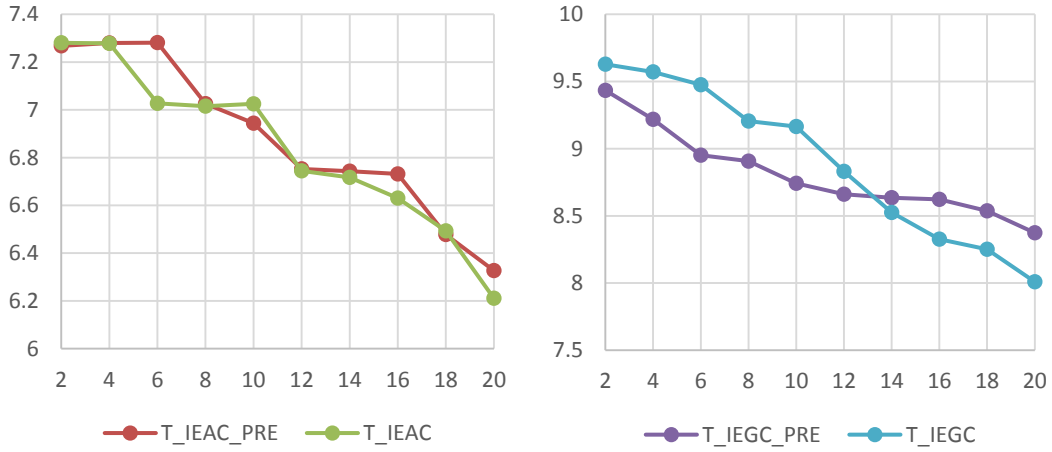

Figure 3. Comparison of Arithmetic Coding (left) and Golomb Rice Coding (right) Encoding Speed by Enabling and Disabling Preprocessing Block

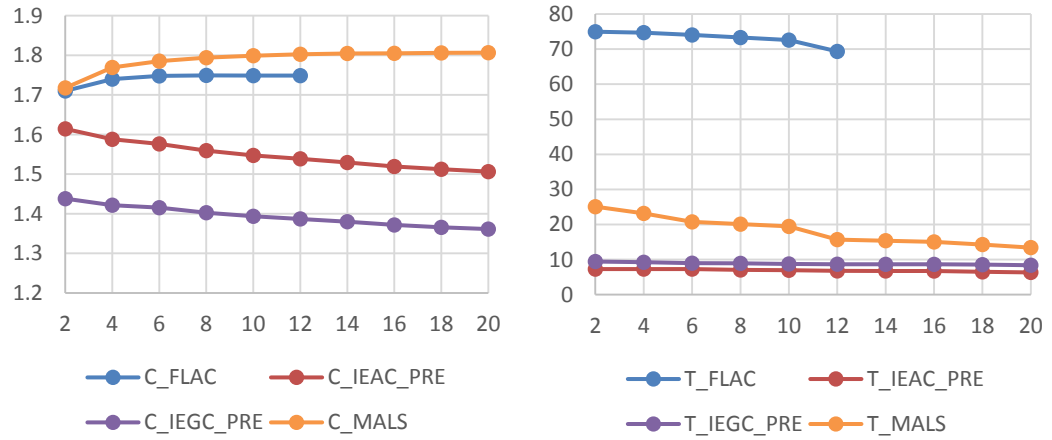

Figure 4. Comparison of Compression Ratio (left) and Encoding Speed (right) for Various Algorithms

7. CONCLUSION

Overall, the experiment shows that for lossless audio compression, the arithemetic coding has a higher compression ratio to golomb rice coding in the IEEE1857.2, with the expense of encoding speed and the pre-processing block does improve the compression ratio of entropy coding especially Golomb-Rice coding, but may worse it's encoding speed for higher predictor order. Nevertheless, when comparing it to other methods, it still falls behind to MPEG-4 and FLAC in terms of compression ratio and encoding speed especially in the case of audio books. Ultimately, IEEE1857.2 may be more suited for music files, but further improvement could be done by detection of speech files and long-term prediction mechanism. In terms of speed, the IEEE1857.2 and MPEG-4 ALS could have a better comparison with the FLAC codec by using FLAC syscall POSIX interface read/write interface.

ACKNOWLEDGEMENT

The authors have gratefully acknowledged that this research has been supported by Ministry of Higher Education Malaysia Research Fund, FRGS15-194-0435.

REFERENCES

[1] Moriya Research Lab, "Lossless Compression of Audio Data (MPEG-4 ALS ad its Application)," 2003.
[2] Hans M. and Schafer R. W., "Lossless compression of digital audio," *IEEE Signal Process Mag*, vol. 18, pp. 21–32, 2001.

[3] Muin F. A., *et al.*, "Performance Analysis of IEEE 1857.2 Lossless Audio Compression Linear Predictor Algorithm," *ICSIMA*, pp. 6, 2017.
[4] Coalson J., "FLAC - Free Lossless Audio Codec," 2017.
[5] Liebchen T., "An Introduction To Mpeg-4 Audio Lossless Coding," *IEEE Int. Conf. Acoust. Speech Signal Process*, vol. 3, pp. 1012–5, 2004.
[6] Shahbahrami A., *et al.*, "Evaluation of Huffman and Arithmetic Algorithms for Multimedia Compression Standards," *Int J Comput Sci Eng Appl*., vol. 1, pp. 34–47, 2011.
[7] Saloman D. M. G., "Handbook of Data Compression," 2010.
[8] Reznik Y. A., "Coding of prediction residual in MPEG-4 standard for lossless audio coding (MPEG-4 ALS)," *IEEE Int. Conf. Acoust. Speech, Signal Process*, vol. 3, pp. 1024–7, 2004.
[9] Moffat A., *et al.*, "Arithmetic Coding Revisited," *ACM Trans Inf Syst*., vol. 16, pp. 256–94, 1998.
[10] Huang H., *et al.*, "Lossless audio compression in the new IEEE Standard for Advanced Audio Coding," *IEEE Int. Conf. Acoust. Speech Signal Process*, pp. 6934–8, 2014.
[11] T. S. Gunawan, *et al.*, "Investigation of Lossless Audio Compression using IEEE 1857.2 Advanced Audio Coding," *Indonesian Journal of Electrical Engineering and Computer Science*, vol/issue: 6(2), 2017.
[12] "IEEE 1857.2. IEEE Standard for Systems of Advanced Audio and Video Coding," 2013.
[13] Harada N., *et al.*, "MPEG-4 ALS: Performance, applications, and related standardization activities," 2007.