Indonesian Journal of Electrical Engineering and Computer Science
Vol. 6, No. 2, May 2017, pp. 422 ~ 430
DOI: 10.11591/ijeecs.v6.i2.pp422-430

Investigation of Lossless Audio Compression using IEEE 1857.2 Advanced Audio Coding

Teddy Surya Gunawan*[1], M. Khalif Mat Zain[2], Fathiah Abdul Muin[3], Mira Kartiwi[4]
[1,2,3]Department of Electrical and Computer Engineering, Kulliyyah of Engineering
[4]Department of Information Systems, Kulliyyah of ICT, International Islamic University Malaysia
Jalan Gombak, 53100 Kuala Lumpur, (+603) 6196 4521
*Corresponding author, e-mail: tsgunawan@iium.edu.my, tsgunawan@gmail.com

Abstract

Audio compression is a method of reducing the space demand and aid transmission of the source file which then can be categorized by lossy and lossless compression. Lossless audio compression was considered to be a luxury previously due to the limited storage space. However, as storage technology progresses, lossless audio files can be seen as the only plausible choice for those seeking the ultimate audio quality experience. There are a lot of commonly used lossless codecs are FLAC, Wavpack, ALAC, Monkey Audio, True Audio, etc. The IEEE Standard for Advanced Audio Coding (IEEE 1857.2) is a new standard approved by IEEE in 2013 that covers both lossy and lossless audio compression tools. A lot of research has been done on this standard, but this paper will focus more on whether the IEEE 1857.2 lossless audio codec to be a viable alternative to other existing codecs in its current state. Therefore, the objective of this paper is to investigate the codec's operation as initial measurements performed by researchers show that the lossless compression performance of the IEEE compressor is better than any traditional encoders, while the encoding speed is slower which can be further optimized.

Keywords*: advancedaudiocoding; IEEE 1857.2; audiocompression; losslessaudio*

1. Introduction

The growth of the Internet have assisted in shifting the trend of storing audio files in the digital format which is usually has been compressed for convenience. Audio compression can be categorized to lossless and lossy, in which the latter sacrifices quality for mobility by removing less important or audible data. Lossy compression is ideal for when space constraint and data transmission is a concern [1]. Depending on the situation, lossy audio compression can achieve up to ten times or more compression ratio which appeals to mobile devices user that prioritizes portability over anything else. However, the constant improvement of storage density and cost allows for audio files that is stored to be bigger. Since that lossless compression does not discard any audio data, the quality of recording can be preserved [2]. Currently there are numerous of lossless audio codecs and each of them has their own advantages over one another, for examples: FLAC, Wavpack and ALAC.

Lossless audio compression works by detecting and discarding statistical redundancy in an audio file in order to reduce the bit-rate. This process is usually achieved by using linear prediction for redundancy removal and the entropy coder for coding the predictor output. This method is used by majority of lossless codecs with variations in algorithm used in both predictor and entropy coder block. A new standard was approved by IEEE in 2013 which comprises of both lossy and lossless audio compression tool. The IEEE Standard for Advanced Audio Coding (IEEE 1857.2) performs wavelet transform and employs a new pre-processing block in addition to a LPC predictor and a modified entropy coder. This pre-processing block functions to flatten the amplitude of the generated prediction residue to achieve better compression performance [3].

IEEE 1857.2 is relatively new standard adopted in 2013. The objective of this paper is to investigate the IEEE 1857.2 standard in more details, and to compare its performance in terms of encoding speed and compression ratio with other lossless audio coders. Section 2 describes the IEEE 1857.2 lossless audio compression in details. Section 3 provides the experimental setup and performance evaluation, while Section 4 concludes this paper.

Received February 21, 2017; Revised May 2, 2017; Accepted May 16, 2017

2. IEEE 1857.2 Lossless Audio Compression

Figure 1 represents the block diagram of the IEEE 1857.2 lossless audio compression part, where it decomposes the audio signal (frame-by-frame in frequency domain) by inter-channel decorrelation and performs Integer Wavelet Transform to find its corresponding detail componentand reconstructs this component. Starting with the encoder, first the audio samples are decorrelated to find the relationship of multi-channel audio inputs, then Wavelet Transform is performed to find the subband signals for the predictor to compute the prediction residue [4]. The residue then will be passed to the pre-processor block to undergo amplitude envelope flattening. Then, the flattened prediction residue is coded in the entropy encoder block for compression.

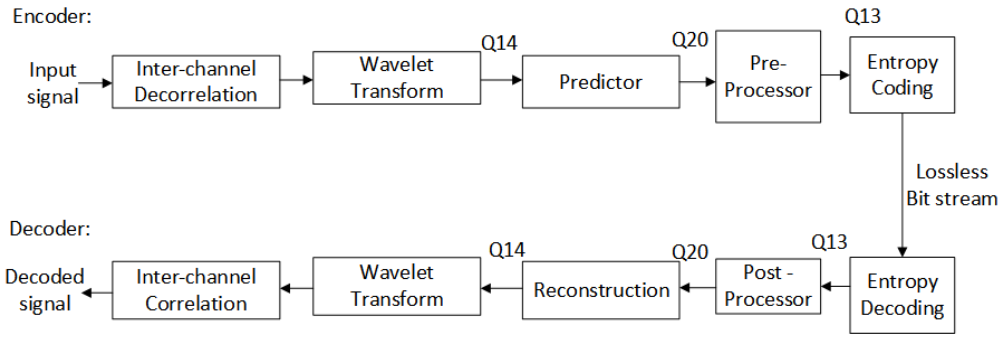

Figure 1. IEEE 1857.2 Lossless Audio Compression Block Diagram

During the decoding stage, the entropy decoder decodes the lossless bit stream obtained from the compressed audio sample to recover the flattened residue which then de-flattened by the post-processor block using up-shifting operation. Finally, the re-constructor blockreconstructs the subband signal component to be further transformed by the wavelet module and correlated to get the original audio signal.

Another interesting feature of the codec is that each output of the module is defined with a fixed-point format representation to improve its encoding and decoding speed. The output of the Wavelet Module has a fixed-point representation of Q14, the predictor; Q20 and the Pre-Processor; Q13. As well as this, the fixed-point format allows portability on no FPU (Floating Point Unit) hardware, which are generally cheaper [5].

2.1. Inter-Channel Decorrelation

Firstly, in order minimize the information of multi-channel audio, this module will remove its redundancy by converting its corresponding Left and Right channels to its corresponding Mid (Sum of L and R channels) and Side (Difference of L and R channels) channels, as shown in Figure 2.

In case LS and RS channels exists, this will be converted exactly as the L and R channels input to SMid and SSide channels, as well as LB and RB channels to its BMid and BSide channels. Centre channel and LFE (Low Frequency Effect) channel will be passed as it is Each channel block will contain the data for the wavelet transform module.

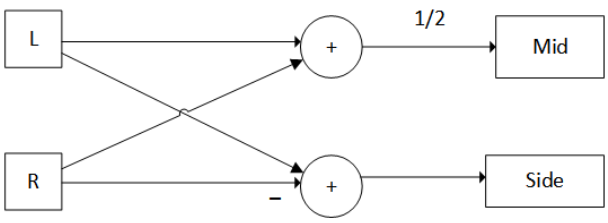

Figure 2. IEEE 1857.2 Channel Decorrelation Module [3]

2.2. Interger Lifting Wavelet Transform

After each channel is decorrelated, then each corresponding channel will be separated by lifting wavelet transform which generates a detail high frequency signal, specified by cd and a scale low frequency signal, specified by ca in the diagram below. The purpose of this module is to prevent round off error effect from quantization.

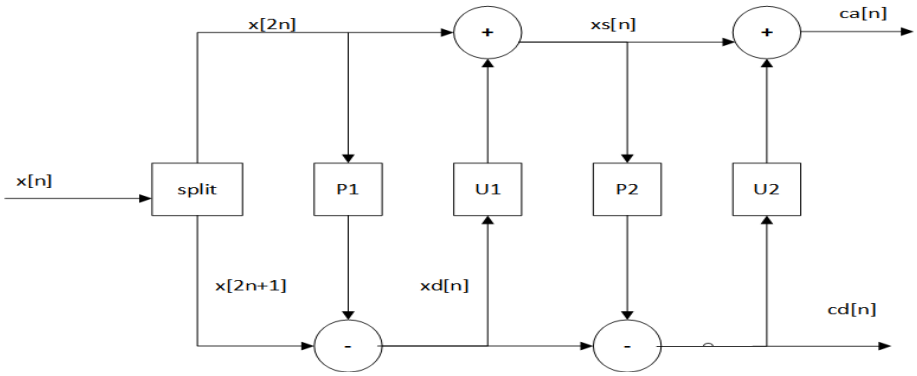

Figure 3. IEEE 1857.2 Integer Lifting Wavelet Transform Module [3] [8]

This module consists of 4 filters, where P1 and P2 are the prediction filters and U1 and U2 are the update filters. Each output stage of the filters, xd andxscan be computed as followed:

$$xd[n] = x[2n+1] - \lfloor 2731 \times (x[2n] + x[2n+2] \rfloor \quad (1)$$

$$xs[n] = x[2n] - \lfloor -9216 \times (xd[n-1] + xd[n] \rfloor \quad (2)$$

Then, the final output signal from the second prediction and update filter can be found by:

$$cd[n] = xd[n] - \lfloor 21845 \times (xs[n] + xs[n+1] \rfloor \quad (3)$$

$$ca[n] = xs[n] - \lfloor 126 \times (ch[n-3] + cd[n+2]) - 938(cd[n-2]) + cd[n+1] + 3372 \times (cd[n-1] + cd[n] \rfloor \quad (4)$$

2.3. Predictor

Only the high frequency audio input sample frame are inputted into the predictor module, where linear predictive coding is then performed on each frame, with partial-correlation (PARCOR) coefficients computed through the Levinson-Durbin algorithm [6]. The PARCOR coefficients are quantized and sent as ancillary information in the lossless bit stream, as shown in Figure 3. The quantized PARCOR coefficients are also locally de-quantized and converted to the LPC coefficients which generates a prediction for each sample in the frame. The signal difference between an input sample and its prediction is called the prediction residue. This prediction residue is encoded into the lossless bit stream for the reconstruction stage, as an extension for lossless bit stream data of the codec.

To obtain the quantized PARCOR coefficients:

$$quantizedPARCOR = \begin{cases} \left[64\left(\ln\left(\frac{2}{3} + \frac{5}{2}\sqrt{\frac{1+PARCORcoeff_1}{2}}\right) \div \ln\left(\frac{3}{2}\right)\right)\right] & k = 1 \\ \left[64\left(\ln\left(\frac{2}{3} + \frac{5}{2}\sqrt{\frac{1-PARCORcoeff_1}{2}}\right) \div \ln\left(\frac{3}{2}\right)\right)\right] & k = 2 \\ [64 \times PARCORcoeff_k] & k = 3, \ldots, LPC\ order \end{cases} \quad (5)$$

For de-quantization of the PARCOR coefficients:

$$dequantizedPARCOR(i) = \{\begin{cases} \left(2.\left(\left(\exp.\left(\frac{quantizedPARCOR(1)}{64log\left(\frac{3}{2}\right)}\right)-\frac{2}{3}\right).\frac{6}{5}\right)^2\right)-1, & i=1 \\ \left(2.\left(\left(\exp.\left(\frac{quantizedPARCOR(1)}{64log\left(\frac{3}{2}\right)}\right)-\frac{2}{3}\right).\frac{6}{5}\right)^2\right)+1, & i=2 \\ \frac{quantizedPARCOR(i)}{64}, & i=3,\dots,PARCOR\ order \end{cases} \quad (6)$$

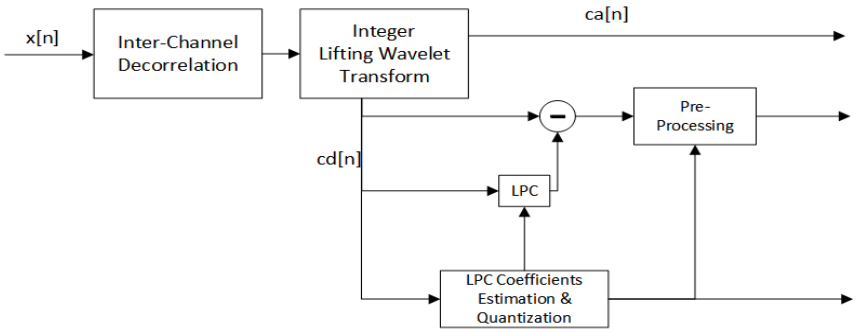

Figure 3. IEEE 1857.2 Predictor Overview [4]

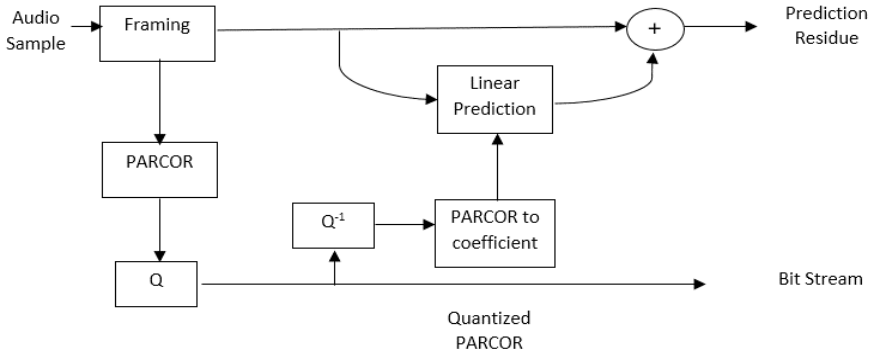

Figure 4. IEEE 1857.2 Predictor Basic Block [3]

To obtain the prediction matrix$y(i)$:

$$y(i) = \begin{cases} \sum_{j=1}^{i} lpc_j{}^{(i)} frames(i-j), 1 \le i \le PARCOR\_order - 1 \\ \sum_{j=1}^{PARCOR\_order} lpc_j{}^{(PARCOR\_order)} x(i-j), i \ge PARCOR\_order \end{cases} \quad (7)$$

$$frames(i), i = 0$$

$$residues(i) = frame(i) + y(i), i = 1,\dots,number\ of\ frames$$

2.4. Pre-processor

One of the advantage of the IEEE 1857.2 lossless compression is that the bit error introduced during transmission can be minimized. 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 increase 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, as shown in Figure 5. This operation allows the amplitude to

decrease and the envelope of the prediction residues is flattened. For the down-shift process, the shift number are calculated from the quantized PARCOR coefficients.

$$shift\,(n) = \begin{cases} \sum_{k=1}^{n+1} RAshift12[quantizedPARCOR(k)] & n = 0,1 \\ \sum_{k=1}^{2} RAshift12[quantizedPARCOR(k)] + \sum_{k=3}^{n+1} RAshift[|quantizedPARCOR(k)|] & 2 \le n \le L-1 \end{cases} \quad (8)$$

The resulting shift matrix (signal length(L) x no of frames) is used for down shifting to obtain 2 new matrix sets (LSB and the flattened data). The LSB data is then transferred to the post processing block. The flattened data goes to the entropy encoder block.

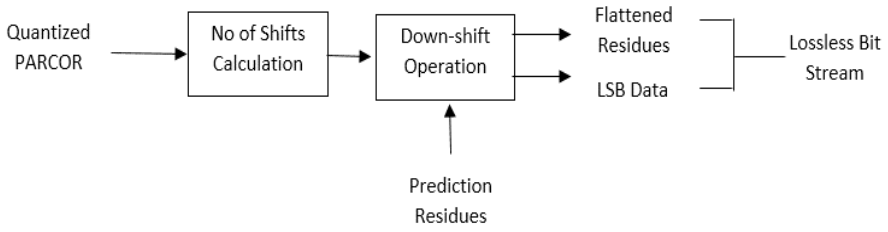

Figure 5. IEEE 1857.2 Pre-Processor Block [4]

2.5. Entropy Coder

Entropy coding, as shown in Figure 6, is an algorithm based on arithmetic encoder that compress frequently occurring pattern with few bits. This means that the frequency of the pattern is inversely proportional to the number of bits used for representation. The output from the pre-processor block that are sent to the entropy encoder is the flattened prediction residue. The flattened data will be used to create an alphabet, lossless data, that will be send to the decoder. Then, the mean value of each of the flattened residue frame is calculated, quantized and locally de-quantized using the following formula:

$$\mu = \frac{\sum_{i=1}^{frame\ length} |flat\_data(i)|}{frame\_length} \quad (9)$$

$$quantized\_\mu = |log_2\mu + 0.5 \quad (10)$$

$$dequantized\_\mu = 2^{q\_u} \quad (11)$$

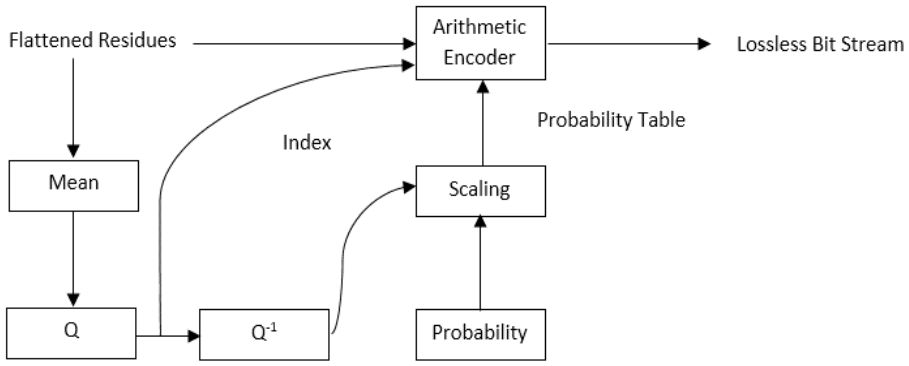

Figure 6. IEEE 1857.2 Entropy Encoder [4]

The de-quantized mean value of the frame is used to scale an existing probability template in order to generate a probability table for the arithmetic coding. The probability template is a table containing a set of probability density values that are obtained by applying Gauss membership function to a set of predetermined value. The Matlab function 'arithenco' generates binary arithmetic code on the flattened data while the probability template serve as a source statistic to achieve compression. This is how the compressed lossless data is obtained [7].

2.6. Entropy Decoder

Entropy decoder operation is based on an arithmetic decoder that will take the lossless data (binary sequence encoded with n bits) and it will decode the binary arithmetic in the vector code to recover the corresponding sequence of flat data. This process is not that different compared to the entropy encoder in which the only major difference is that it is not necessary to calculate and quantize the mean value. The output of this stage should be streams of decoded flattened residue. This is done by applying arithmetic decoding on the lossless data using 'arithdeco' function with the same probability template as the arithmetic encoder.

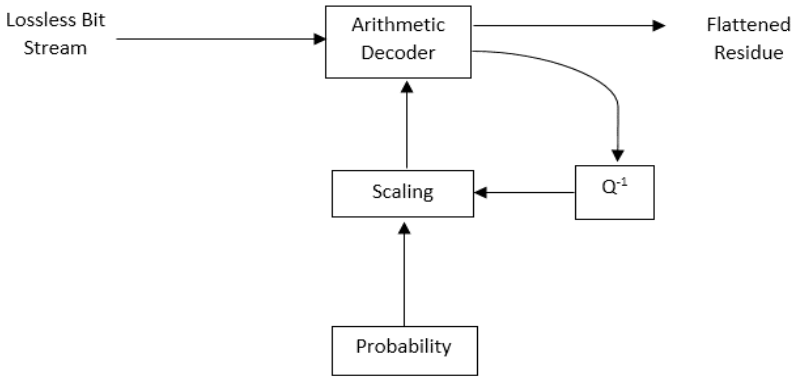

Figure 7. IEEE 1857.2 Entropy Decoder [4]

2.7. Post Processor

During the post-processing stage, the prediction residues is reconstructed using the flattened residues that have been encoded and decoded using entropy coding and the LSB data that have been transmitted directly from the pre-processing. Shifting operation should be performed again in the formed of up-shifting. However, the number of shifts should be calculated first by using the PARCOR quantized coefficients obtained earlier. The output from the up-shift operation which is the prediction residues is then transferred to the reconstructor block for the final step. It is important to note that due to channel and storage devices imperfections, some noise might be introduced. This issue can be countered in the post-processing block with some post-filtering [8].

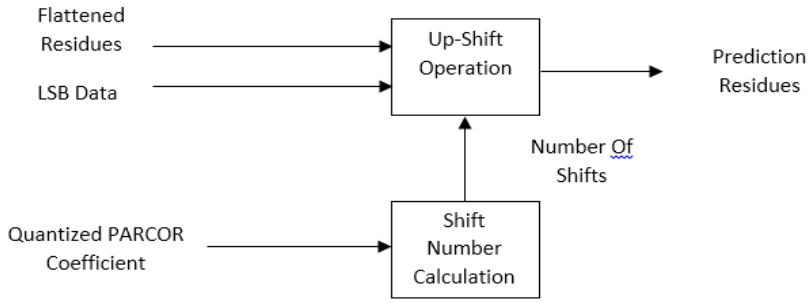

Figure 8. IEEE 1857.2 Post-Processor Block [4]

2.8. Reconstructor

In order to reconstruct the audio signal, the quantized PARCOR coefficients from the post-processor block will be extracted and de-quantized. This coefficients then are converted to LPC coefficients that are identical to those used in the encoder. The linear predictor generates a prediction to be added to the decoded prediction residue in order to reconstruct the high frequency component of the Interger Wavelet Transform.

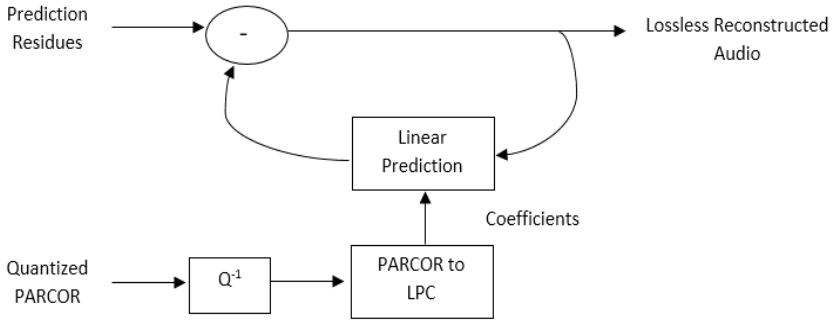

Figure 9. IEEE 1857.2 Reconstructor [4]

3. Performance Evaluation

3.1. Experimental Setup

The experiment was performed on a Windows 10 machine with i5 -6200u CPU and 8GB of DDR4 memory. There are 10 audio files as shown in Table 1. The file is encoded and decoded using all of the available lossless presets for each codecs, as shown in Table 2. The output file is then compared with the original file to check for differences by calculating the mean square error of the two.

Table 1. Audio Database

Audio Files	Duration (s)	Size (kb)	Sample Rate (kHz)	Bitrate (kbps)
Alarm Beep.wav	0.04	81.4	22.05	176
Baby Cry.wav	0.13	140.7	11.025	88
Bird Caw.wav	0.04	43	11.025	88
Crickets.wav	0.1	221.4	22.05	176
Elvis Died.wav	0.12	132.3	11.025	88
Family Man.wav	0.09	93.2	11.025	88
Glass Shatter C.wav	0.01	12.7	11.025	88
No Ways Tired.wav	0.09	95.5	11.025	88
One Small Step .wav	0.06	68.2	11.025	88
Space Oddity.wav	0.03	35.5	11.025	88

Table 2. Lossless Codec Preset Options

Codecs	Performance Preset								
IEEE 1857.2	Default (16 kbps Output Bitrate)								
FLAC	0	1	2	3	4	5	6	7	8
WavPack	Fast		Normal		High		Extra High		
Monkey Audio	Fast		Normal		High		Extra High		Insane
True Audio	Default								

3.2. Encoding Speed

The encoding speed is one of the important performance criteria for lossless audio coding. Table 3 shows the comparison of encoding speed in second for various files and various coders. The encoding time shown is the average of ten repeated experiments for each file and

each coder. The result shows that FLAC has the fastest encoding time, while IEEE 1857.2 has the slowest encoding time. As seen from the results, the IEEE 1857.2 lossless audio encoder is noticeably slower in encoding the audio files compared to other codecs. FLAC is the clear winner here, followed by True Audio, Monkey's Audio, Wavpack and lastly the IEEE 1857.2 codec.

The slower encoding time is understandable due to the codecs early stage of development and higher complexity in compression algorithm (amplitude flattening, probability template coder) when compared to other existing codecs. The other codecs also have years of optimizations and improvements.Note that, the IEEE 1857.2 was compiled with the default parameters and no optimization has been performed on the source code. With the advanced of CPU and GPU computing, the IEEE 1857.2 encoder could be further optimized for speed.

Table 3. Comparison of Encoding Speed (second)

Audio Files	IEEE 1857.2	FLAC	Wavpack	MonkeyAudio	True Audio
Alarm Beep	0.126	0.051	0.058	0.070	0.117
Baby Cry	0.412	0.103	0.064	0.084	0.051
Bird Caw	0.681	0.043	0.058	0.058	0.049
Crickets	0.487	0.068	0.088	0.098	0.055
Elvis Died	0.537	0.050	0.072	0.081	0.051
Family Man	0.126	0.069	0.071	0.092	0.053
Glass ShatterC	0.126	0.048	0.095	0.068	0.047
No Ways Tired	0.562	0.047	0.080	0.065	0.131
One Small Step	0.489	0.055	0.066	0.062	0.050
Space Oddity	0.341	0.051	0.067	0.063	0.047
Average	*0.389*	*0.059*	*0.072*	*0.074*	*0.065*

3.3. Compression Ratio

Compression ratio will be the most important evaluation in audio coder benchmarking. Table 4 shows the compression ratio of various coders and various files. On average, it is found that the IEEE 1857.2 has better compression ratio compared to the other coders. The superior compression performance could be due to the presence of the pre-processor block and the complexity of the algorithm used.

Table 4. Comparison of Compression Ratio

Audio Files	IEEE 1857.2	FLAC	Wavpack	MonkeyAudio	True Audio
Alarm Beep	3.57	2.34	3.23	2.89	2.89
Baby Cry	2.35	1.86	2.14	2.15	1.98
Bird Caw	2.97	1.66	2.46	2.39	2.15
Crickets	3.21	2.53	2.88	3.03	2.91
Elvis Died	1.89	1.51	1.72	1.81	1.74
Family Man	2.42	1.88	2.23	2.24	2.26
Glass ShatterC	1.49	0.71	1.28	1.30	1.23
No Ways Tired	1.71	1.33	1.51	1.59	1.49
One Small Step	2.10	1.52	1.86	1.97	1.87
Space Oddity	1.60	1.07	1.44	1.49	1.41
Average	*2.33*	*1.64*	*2.08*	*2.09*	*1.99*

4. Conclusion

This paper presents the investigation of IEEE 1857.2 lossless audio compression tool which become IEEE standard in 2013. Lossless audio compression is a very interesting topic since there is a limit on how a file can be made smaller without discarding any data. Throughout the years, numerous lossless codecs have surfaced and refined to the point that they delivers comparable performance and ability. The only differentiating factors are the adoption rate and feature set that might appeal to different set of audience. The IEEE 1857.2 lossless audio

compression tool tries to improve the compression ability further by adding pre-processing block to reduce the dynamic range of the prediction residue instead of encoding it directly. This increases the complexity of the algorithm thus affecting the encoding time, but greatly improves the compression ratio in the process. The IEEE 1857.2 audio compression tool promises higher compression performance compared to other existing lossless codecs. Further research could be conducted on the optimization or parallelization of the IEEE 1857.2 to speed up the computation.

Acknowledgement

The Malaysian Ministry of Higher Education (MOHE) kindly provided funding for the research through the Fundamental Research Grant Scheme, FRGS15-194-0435.

References

[1] OA Mahdi, MA Mohammed, AJ Mohamed. Implementing a Novel Approach an Convert Audio Compression to Text Coding via Hybrid Technique. *Int. J. Comput. Sci*. 2012; 9(6) 53–59.
[2] N Nowak, W Zabierowski. *Methods of sound data compression - Comparison of different standards*. 2011 11th International Conference The Experience of Designing and Application of CAD Systems in Microelectronics (CADSM). 2011: 431–434.
[3] H Huang, H Shu, R Yu. *Lossless audio compression in the new IEEE Standard for Advanced Audio Coding*. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). 2014: 6934–6938.
[4] IS Association. IEEE Standard for Systems of Advanced Audio and Video Coding.
[5] D Bismor and K Bartnicki. Real-time Advanced Speech Enhancement using Low-power, Fixed-point Hardware. in *2016 Signal Processing: Algorithms, Architectures, Arrangements, and Applications (SPA).* 2016: 62–67.
[6] SBBU Vagdevi, Y Bineetha. Linear Prediction Analysis. *Int. J. Eng. Sci*. 2013; 2(4): 1–7.
[7] DMB Adrià Romero López. Lossless audio compression in IEEE 1857.2 Standard For Advanced Audio Coding. 2015. [Online]. Available: https://github.com/iamrosmarin/lossless-audio-compression.
[8] FN Auristin. Advanced Audio Compression For Lossless Audio Coding Using Ieee 1857.2. *Int. J. Eng. Comput. Sci.* 2016; 5(9): 18124–18127.