ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

COMPARISON OF BOLDI VIGNA (Z2) ALGORITHM AND ELIAS DELTA CODE ALGORITHM IN AUDIO FILE COMPRESSION

[1]HANDRIZAL, [2]FAUZAN NURAHMADI, [3]MUHAMMAD ABRAR

[1,2,3]Department of Computer Science, Faculty of Computer Science and Information Technology,

Universitas Sumatera Utara, Jl. University No. 9-A, Medan 20155, Indonesia

Email: handrizal@usu.ac.id

ABSTRACT

Information technology nowadays develops quickly and brings a lot of positive impacts on human life. Humans are now able to exchange data and information easily. But the problem is the size of the data tends to be so large and requires a lot of storage and makes the transmission cost so high. To overcome this problem, data processing techniques are required, one of them is through data compression so that the result size can be reduced. WAV is an uncompressed audio saving format and it makes mainly files in (*.wav) extension tend to have a large size and need a lot of space in storage. In this research, the compression test will be done on 8-bit wav audio files using the Boldi-Vigna algorithm and the Elias Delta Code algorithm. Both of these algorithms are included in the lossless compression algorithm which means that it can restore the compressed data to original data through the decompression process. The performance of Boldi-Vigna and Elias Delta Code algorithms will be calculated based on the predefined comparison parameters. Based on the test results, it is obtained that the Boldi-Vigna algorithm is better at compression with an average Ratio of Compression of 69.562%, Compression Ratio of 1.458, and Space Saving of 30.428%. The compression results obtained are influenced by the number of the same digital values contained in the wav audio file that needs to be compressed. Both algorithms can restore the whole audio file like the original audio file through the decompression process.

Keywords: *Compression, Elias Delta Code, Boldi-Vigna.*

1. INTRODUCTION

Information technology nowadays develops quickly. It brings a lot of positive impacts which are very useful for humans. One of the impacts is the information transmission process. Humans are now able to access and obtain information easily. But the problem is the size of the data tends to be so large and requires a lot of storage and makes the transmission cost so high. To overcome this problem, data processing techniques are required, one of them is through data compression so the result size can be reduced.

WAV is an uncompressed audio saving format and it makes mainly files in (*.wav) extension tend to have a relatively large size and need a lot of space in the storage device.

There are some algorithms known that can be used to compress data, one of them is the Elias Delta Code Algorithm. Elias Delta Code Algorithm implements basic Elias Gamma Code coding as the building block. This Piter Elias creation algorithm is included in lossless compression where compressed files can be restored to their original form.

Boldi-Vigna Zeta algorithm was introduced by Paolo Boldi and Sebastiano Vigna as a family of Variable Length Code which is the best choice for compression. Boldi Vigna Zeta Code is started with a positive integer k which shrinks from Code Factor.

Each compression algorithm has different efficiency values in compressing various types of data, therefore a correct algorithm choice will influence the results of the compression.

The Boldi Vigna Algorithm and Elias Delta Code Algorithm method has been widely used to solve various problems such as Analysis of Text Data Compression [1], Data Compression [2], Text Compression [3], Image Files Security And Compression [4], Image Compression And Security [5]. Based on the description above, the writer is willing to do research which title is "Comparison Of Boldi Vigna (Z2) Algorithm And Elias Delta Code Algorithm In Audio File Compression".

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

2. LITERATURE REVIEW

2.1 Digital Audio

Audio is a voice or sound produced from the transformation of molecules in the air which is caused by object movement that creates vibration. The rate at which a vibration occurs in a particular period is called frequency. Back and forth movement is called a cycle. Therefore, the unit of frequency is cps or cycle per second or mainly known as Hertz (Hz) [6].

Digital Audio is the digital form of analog audio that is produced by converting the amplitude of an analog wave to interval time (sample) by using a tool called Analog to Digital Converter (ADC) [6].

To make the audio can be heard again, a tool called Digital to Analog Converter (DAC) is required to convert the digital audio sample to analog audio and then the analog signal will be converted again to air vibration by the speaker so the audio can be heard again by human ears.

2.2 WAV File

WAV is a common format used in the Microsoft Windows operating system to save raw audio and is usually uncompressed because every sample of the audio is saved in the storage device. In this format, digital audio is saved in the waveform so the file will have a wav (wave) extension [7].

WAV files can record audio from different qualities. For example, the 8-bit or 16-bit format with a rate of 11025Hz, 22050Hz, or 44100Hz. For better audio quality, such as 44100Hz, the 16-bit will take up about 150Kb of storage media for every second. A file with a wav extension has a maximum limit of 2 GB and a relatively large size so that's why it is rarely used on the internet [7].

WAV audio files use the standard structure RIFF (Resource Interchange File Format) that is commonly used for multimedia data in Windows. This structure groups the data from the file into parts where each of the data has a header and size, which is called a chunk.

If the WAV Audio file is opened with HEX Editor, it will look like Figure 1 below.

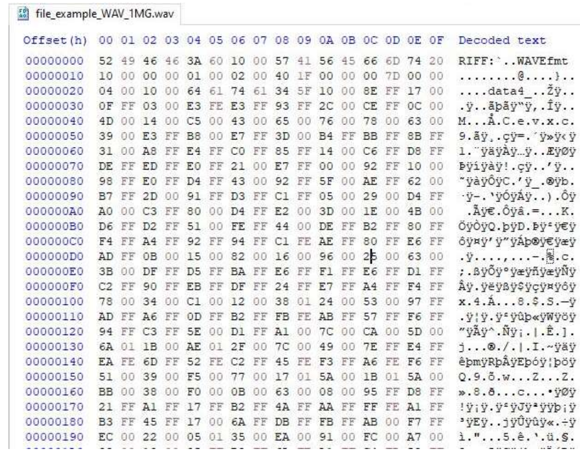

Figure 1. Wav Audio File in Hex Editor

2.3 Compression

Compression is the process of converting data input stream/ original data into a new data stream/ compressed file which has a smaller size [8]. Compression is done to reduce the size of data so that it is more efficient in the storage device and can speed up the data transmission process. In general, compression can be categorized into two types, namely lossless compression and lossy compression.

2.3.1 Lossless compression

Lossless compression is a compression technique that converts the bitstream of the input data into a new data bitstream so that it becomes denser and smaller in size. Lossless compression is more suitable for compressing important data that cannot tolerate the difference between the data after compression and the data after decompression [7]. An illustration of lossless compression is shown in Figure 2.

Figure 2. The Illustration of Lossless Compression

2.3.2 Lossy compression

Lossy compression is a data compression technique that changes the density of the data or reduces some of the bitstreams of the data being compressed so that the resulting data usually experiences a slight decrease meant in quality or resolution. Data that is compressed using a lossy compression technique cannot be decompressed back into the original data. Images, sound, or video are the data type which is usually used in lossy compression[8]. The illustration of lossy compression can be seen in Figure 3.

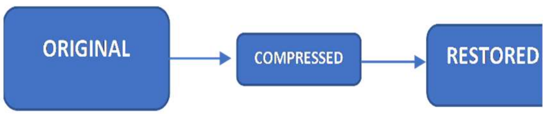

Figure 3: The Illustration of Lossy Compression

2.3.3 Comparison Parameter of Algorithm Performance.

In a specific method like compression algorithm, several things are commonly used as performance measurement parameters to compare each performance, namely:

1. The ratio of Compression (RC)
   The ratio of compression is the comparison between data size after compression process with data size before compression process.

   $$RC = \frac{\text{Data size after compression}}{\text{Data size before compression}} \text{ x } 100\%$$

2. Compression Ratio (CR)
   The compression ratio is the comparison between data size before compression process with data size after compression process. [3].

   $$CR = \frac{Data\ size\ before\ compression}{Data\ size\ after\ compression}$$

3. Space Saving (SS)
   Space Saving is a percentage of the saved storage space. [3].

   $$SS = \left(1 - \frac{Data\ size\ before\ compression}{Data\ size\ before\ compression}\right) \times 100\%$$

3. METHOD

3.1. Boldi-Vigna Algorithm (ζ2)

The Boldi-Vigna (ζ2) algorithm introduced by Paolo Boldi and Sebastiano Vigna is included in a high-level Variable Length Code and is the right choice for compression. The zeta code of Boldi-Vigna is started with the positive integer k which then becomes the code shrinking factor. The set of all positive integers is divided into intervals [2^0, 2^k - 1], [2^k, 2^{2k}- 1], [2^{2k}, 2^{3k}- 1], with the general form [2^{hk}, $2^{(h+1)k}$- 1]. The length of each interval is stated in the form $2^{(h+1)k}$- 2^{hk} [8].

The steps in coding the Boldi-Vigna algorithm (ζ2) are as follows:

1. The set of positive integers is divided into intervals [2^{hk}, $2^{(h+1)k}$- 1].
2. Input value n.
3. Choose K, take K = 2 because it uses the Boldi-Vigna (ζ2) algorithm
4. Find n in the interval, starting at h = 0.
5. Form unary code from h + 1 by changing the value of h + 1 to be unary code reverse.
6. Calculate X using the formula $n - 2^{hk}$
7. Calculate Z using the formula $2^{(h+1)k} - 2^{hk}$.
8. Calculate S using the formula [$\frac{\log_{(Z)}}{\log_{(2)}}$]
9. If X < 2^s- Z, hen X is encoded as a binary code of S-1 digits, but if X ≥ 2^s- Z then (2^s+ X – Z) is encoded as S digits of binary code.
10. Add unary code reverse value with binary code.

Boldi-Vigna Code values n = 1 to 16 can be seen in Table 1 below.

Table 1. Boldi-Vigna code.

N	Boldi-Vigna(ζ2) Codes
1	10
2	110
3	111
4	01000
5	01001
6	01010
7	01011
8	011000
9	011001
10	011010
11	011011
12	011100
13	011101
14	011110
15	011111
16	00100000

3.2. Elias Delta Code Algorithm

The Elias Delta Code algorithm was created by Piter Elias by implementing the gamma code as the basis for coding. In the gamma code Elias adds the length of the code in unary, whereas in the Elias code, he adds the length in binary. So that the Elias Delta code is longer and becomes more complex [7].

The steps in coding the Elias Delta Code algorithm are as follows [6]:

1. Write n in binary. The leftmost bit will be 1.
2. Calculate the number of bits, remove the leftmost bit.
3. Add the calculation in the binary on the left of n, after the leftmost bit of n is removed.
4. Subtract 1 from the calculation in step 2 and add the number of zeros to the code.

Elias Delta Codes for n = 1 to 18 can be seen in Table 2 below [8].

Table 2. *Elias Delta Code*

$1 = 2^0 + 0 \rightarrow \lvert L \rvert = 0 \rightarrow 1$	$10 = 2^3 + 2 \rightarrow \lvert L \rvert = 3 \rightarrow$ 00100*010*
$2 = 2^1 + 0 \rightarrow \lvert L \rvert = 1 \rightarrow$ 010*0*	$11 = 2^3 + 3 \rightarrow \lvert L \rvert = 3 \rightarrow$ 00100*011*
$3 = 2^1 + 1 \rightarrow \lvert L \rvert = 1 \rightarrow$ 010*1*	$12 = 2^3 + 4 \rightarrow \lvert L \rvert = 3 \rightarrow$ 00100*100*
$4 = 2^2 + 0 \rightarrow \lvert L \rvert = 2 \rightarrow$ 011*00*	$13 = 2^3 + 5 \rightarrow \lvert L \rvert = 3 \rightarrow$ 00100*101*
$5 = 2^2 + 1 \rightarrow \lvert L \rvert = 2 \rightarrow$ 011*01*	$14 = 2^3 + 6 \rightarrow \lvert L \rvert = 3 \rightarrow$ 00100*110*
$6 = 2^2 + 2 \rightarrow \lvert L \rvert = 2 \rightarrow$ 011*10*	$15 = 2^3 + 7 \rightarrow \lvert L \rvert = 3 \rightarrow$ 00100*111*
$7 = 2^2 + 3 \rightarrow \lvert L \rvert = 2 \rightarrow$ 011*11*	$16 = 2^4 + 0 \rightarrow \lvert L \rvert = 4 \rightarrow$ 00101*0000*
$8 = 2^3 + 0 \rightarrow \lvert L \rvert = 3 \rightarrow$ 00100*000*	$17 = 2^4 + 1 \rightarrow \lvert L \rvert = 4 \rightarrow$ 00101*0001*
$9 = 2^3 + 1 \rightarrow \lvert L \rvert = 3 \rightarrow$ 00100*001*	$18 = 2^4 + 2 \rightarrow \lvert L \rvert = 4 \rightarrow$ 00101*0010*

3.3 General Architecture

The general architecture is the scheme for system designing that describes the overall flows. General Architecture can also be a guide for making system modeling. The general diagram of the system is shown in Figure 4.

First, the user selects the audio file with * .Wav extension that needs to be compressed then the system will read the byte value from the audio file, next the system will compress according to the algorithm selected by the user, after that the system will output the compressed file and store it into the storage device. If the user wants to reuse the uncompressed wav audio file, the decompression process is carried out according to the selected algorithm, and the system will output the decompression result in the form of an audio file that the user can reuse.

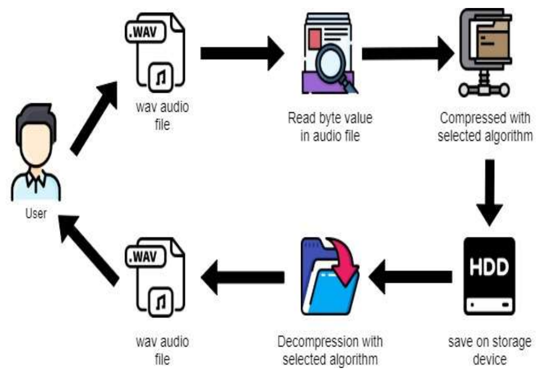

Figure 4. General Architecture System

4. RESULTS AND DISCUSSIONS

The implementation of the system is built using Android Studio IDE using Kotlin programming language, in this application two lossless compression algorithms were applied, namely the Boldi-Vigna algorithm and the Elias Delta Code algorithm to do a comparison based on comparison parameters, namely Ratio of Compression, Compression Ratio and Space Saving to know the effectiveness and efficiency of the two algorithms in compressing the wav audio file. In this research, the writer tested the wav audio file with an 8-bit format.

Journal of Theoretical and Applied Information Technology
15th March 2021. Vol.99. No 5

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

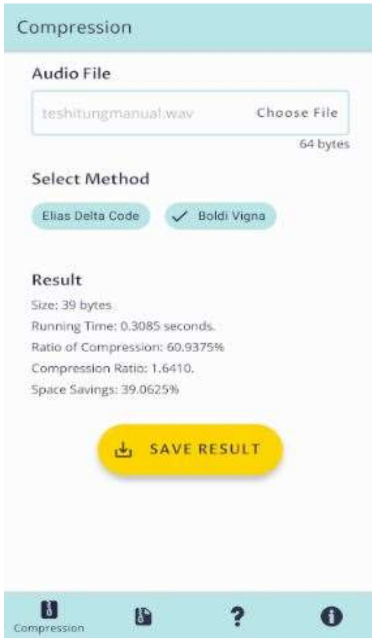

Figure 5. Application and Compression with Boldi-Vigna(ζ2) Algorithm Processes

Compression process using Boldi-Vigna(ζ2) algorithm is as follows:

Digital values contained in the sample audio file with 64 bytes size under test are:

52 49 46 46 BA 78 00 00 57 41 56 45 66 6D 74 20 10 00 00 00 01 00 02 00 44 AC 00 00 88 58 01 00 02 00 08 00 64 61 74 61 96 78 00 00 7F 7F 7F 7F 7F 7F 7F 7E 7E 7E 7E 7E 7E 80 80 80 80 80 80 80.

Then the digital values above will be sorted descending and coding is carried out according to the Boldi-Vigna (ζ2) algorithm code. The calculation process can be seen in Table 3.

Table 3. Calculation of Compression with Boldi-Vigna(ζ2) Algorithm Process

Digital Audio Value (Hex)	Digital Audio Value (Binary)	Frequency	Boldi-Vigna (ζ2)	Bit x Freq
00	00000000	14	10	28
7F	01111111	7	110	21
80	10000000	7	111	21
7E	01111110	6	01000	30
46	01000110	2	01001	10
78	01111000	2	01010	10
74	01110100	2	01011	10
01	00000001	2	011000	12
02	00000010	2	011001	12
61	01100001	2	011010	12
52	01010010	1	011011	6
49	01001001	1	011100	6
BA	10111010	1	011101	6
57	01010111	1	011110	6
41	01000001	1	011111	6
56	01010110	1	00100000	8
45	01000101	1	00100001	8
66	01100110	1	00100010	8
6D	01101101	1	00100011	8
20	00100000	1	00100100	8
10	00010000	1	00100101	8
44	01000100	1	00100110	8
AC	10101100	1	00100111	8
88	10001000	1	00101000	8
58	01011000	1	00101001	8
08	00001000	1	00101010	8
64	01100100	1	00101011	8
96	10010110	1	00101100	8
Total				300

Before the compression result is obtained, the padding bits and flag bits will be added first. This addition is necessary because if compressed bits is divided by 8, it will produce a remainder. The number of the bit string is 300 and if it is divided by 8, it will result in the remainder, it is necessary to add padding bit "0" four times so that it is divisible by 8. Due to the addition of padding bits, the flag bit must also be added to indicate the number of additional padding bits. , in this case, the flag bit added is "00000100".

The compressed bit string using the Boldi-Vigna algorithm is as follows:

01101101110001001010010111010101010100111
100111110010000000010000100100010001000110
10110010010000100100101010011000100110011
00010011000100111101000101000001010010110
00100110011000101010100010101101101001011
01101000101100010101010110110110110110110
11001000010000100001000010000100011111111
1111111111111 0000 00000100.

The number of bit strings obtained is 312, and the size of the compressed file becomes 39 bytes.

$$RC = \frac{Data\ size\ after\ compression}{Data\ size\ before\ compression} \text{ x } 100\%$$

$$= \frac{39}{64} \text{ x } 100\%$$

$$= 60{,}9375\%$$

$$CR = \frac{Data\ size\ before\ compression}{Data\ size\ after\ compression}$$

$$= \frac{64}{39}$$

$$= 1{,}6410$$

$$SS = \left(1 - \frac{Data\ size\ before\ compression}{Data\ size\ before\ compression}\right) \times 100\%$$

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

$$= \left(1 - \frac{39}{64}\right) \times 100\%$$

$$= 39{,}0625$$

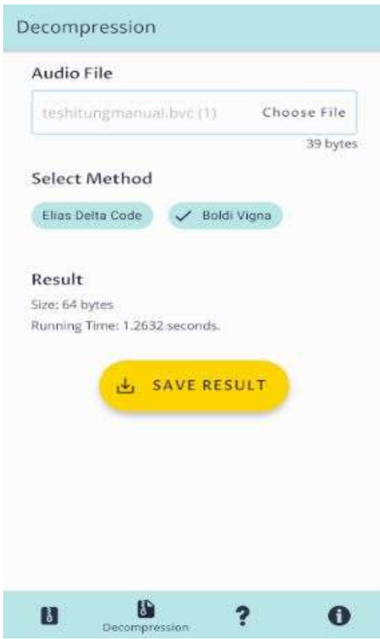

Figure 6. Application of Decompression with Boldi-Vigna(ζ2) Algorithm Processes

The decompression process using Boldi-Vigna(ζ2) algorithm is as follows:

First, the bit string value is read from the compressed file, then the flag bit contained in the file is also read by converting the last 8 bits of the string bit into binary numbers. It is found that the flag bit value is 4, so the next 4 bits will be read.

01101101110001001010010111010101010100111
100111110010000000010000100100010001000110
10110010010000100100101010011000100110011
00010011000100111101000101000001010010110
00100110011000101010100010101101101001011
01101000101100010101010110110110110110110
110010000010000100001000010000100011111111
1111111111111 0000 00000100

After that, the Flag bit and padding bit are removed and the string bit obtained is as follows:

01101101110001001010010111010101010100111
100111110010000000010000100100010001000110
10110010010000100100101010011000100110011
00010011000100111101000101000001010010110
00100110011000101010100010101101101001011
01101000101100010101010110110110110110110
110010000010000100001000010000100011111111
1111111111111

Next, check the bits from the string bit above and then convert them into audio bit values according to the compression table using the Boldi-Vigna algorithm (ζ2) as shown in table 4.

Table 4. Boldi-Vigna(ζ2) Decompression Process.

Nilai Audio (Hexadesimal)	Nilai Audio (Binary)	Boldi-Vigna (ζ2) Code
00	00000000	10
7F	01111111	110
80	10000000	111
7E	01111110	01000
46	01000110	01001
78	01111000	01010
74	01110100	01011
01	00000001	011000
02	00000010	011001
61	01100001	011010
52	01010010	011011
49	01001001	011100
BA	10111010	011101
57	01010111	011110
41	01000001	011111
56	01010110	00100000
45	01000101	00100001
66	01100110	00100010
6D	01101101	00100011
20	00100000	00100100
10	00010000	00100101
44	01000100	00100110
AC	10101100	00100111
88	10001000	00101000
58	01011000	00101001
08	00001000	00101010
64	01100100	00101011
96	10010110	00101100

After the conversion is done according to the table, the audio file value obtained is as follows:

52 49 46 46 BA 78 00 00 57 41 56 45 66 6D 74 20 10 00 00 00 01 00 02 00 44 AC 00 00 88 58 01 00 02 00 08 00 64 61 74 61 96 78 00 00 7F 7F 7F 7F 7F 7F 7F 7E 7E 7E 7E 7E 7E 80 80 80 80 80 80 80.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

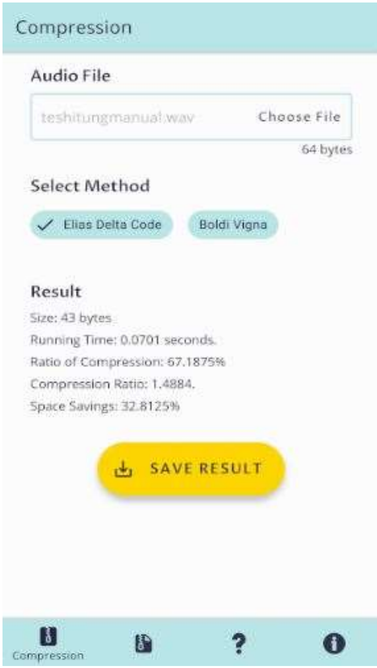

Figure 7. Application of Compression with Elias Delta Code Algorithm Processes

The compression process using Elias Delta Code Algorithm is as follows:

Digital values contained in the sample audio file with 64 bytes size under test are:

52 49 46 46 BA 78 00 00 57 41 56 45 66 6D 74 20 10 00 00 00 01 00 02 00 44 AC 00 00 88 58 01 00 02 00 08 00 64 61 74 61 96 78 00 00 7F 7F 7F 7F 7F 7F 7F 7E 7E 7E 7E 7E 7E 80 80 80 80 80 80 80

Then the digital values above will be sorted descending and coding is carried out according to the Elias Delta Code algorithm. The calculation process can be seen in Table 5.

Table 5. Calculation of Compression with Elias Delta Code Process

Digital Audio Value (Hex)	Digital Audio Value (Binary)	Frequency	Elias Delta Code	Bit x Freq
00	00000000	14	1	14
7F	01111111	7	0100	28
80	10000000	7	0101	28
7E	01111110	6	01100	30
46	01000110	2	01101	10
78	01111000	2	01110	10
74	01110100	2	01111	10
01	00000001	2	00100000	16
02	00000010	2	00100001	16
61	01100001	2	00100010	16
52	01010010	1	00100011	8
49	01001001	1	00100100	8
BA	10111010	1	00100101	8
57	01010111	1	00100110	8
41	01000001	1	00100111	8
56	01010110	1	001010000	9
45	01000101	1	001010001	9
66	01100110	1	001010010	9
6D	01101101	1	001010011	9
20	00100000	1	001010100	9
10	00010000	1	001010101	9
44	01000100	1	001010110	9
AC	10101100	1	001010111	9
88	10001000	1	001011000	9
58	01011000	1	001011001	9
08	00001000	1	001011010	9
64	01100100	1	001011011	9
96	10010110	1	001011100	9
Total				335

From the calculation result above, it is found that the number of the bit string is 335 and if it is divided by 8 it will produce remainder, it is necessary to add padding bit "0" once, and padding flag "00000001".

The compressed bit string using Elias Delta Code algorithm is as follows:

0010001100100100011010110100100101011101 1
0010011000100111001010000000101000100101 00
1000101001101111001010100001010101111001 0
0000100100001100101011000101011111001011 0
0000101100100100000100100001100101101010 0
1011011001000010111100100001001011100011 1
0110100010001000100010001000100011000110 0
0110001100011000110001010101010101010101 0
1010101 0 00000001

The number of compressed bit strings is 344, and the compressed file size is 43 bytes.

$$RC = \frac{Data\ size\ after\ compression}{Data\ size\ before\ compression} \text{ x } 100\%$$

$$= \frac{43}{64} \text{ x } 100\%$$

$$= 67{,}1875\%$$

$$CR = \frac{Data\ size\ before\ compression}{Data\ size\ after\ compressio}$$

$$= \frac{64}{43}$$

$$= 1{,}4884$$

$$SS = \left(1 - \frac{Data\ size\ before\ compression}{Data\ size\ before\ compression}\right) \times 100\%$$

$$= \left(1 - \frac{43}{64}\right) \times 100\%$$

$$= 32{,}8125\%$$

Journal of Theoretical and Applied Information Technology
15th March 2021. Vol.99. No 5

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

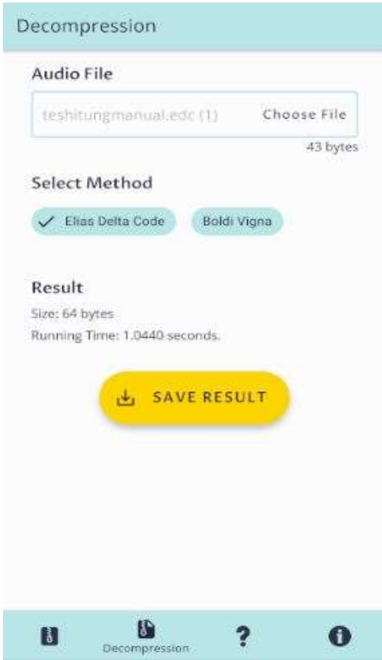

Figure 8. Application of Decompression with Elias Delta Code Algorithm Processes

The decompression process using the Boldi-Vigna (ζ2) algorithm is as follows:

First, read the entire bit string of the compressed file, then read the bit flag and bit padding, the bit flag value obtained is 1, then read the next bit as many as 1 bit.

00100011001001000110101101001001010111011
001001100010011100101000000010100010010100
10001010011011110010101000010101011110010
00001001000011001010110001010111110010110
000010110010010000001001000011001011010100
101101100100000101111001000010010111000111
01101000100010001000100010001000110001100
01100011000110001100010101010101010101010
1010101 0 00000001

Then remove the bit flags and bit padding so that the bit string becomes as follows:

00100011001001000110101101001001010111011
001001100010011100101000000010100010010100
10001010011011110010101000010101011110010
00001001000011001010110001010111110010110
000010110010010000001001000011001011010100
101101100100000101111001000010010111000111
01101000100010001000100010001000110001100
01100011000110001100010101010101010101010
1010101

Then check the bits from the bit string above and then convert them into audio bit according to the table that has been formed for the compression process using the Elias Delta Code algorithm as shown in table 6.

Table 6. *Process of Elias Delta Code Decompression*

Audio Value (Hex)	Audio Value (Binary)	Elias Delta Code
00	00000000	1
7F	01111111	0100
80	10000000	0101
7E	01111110	01100
46	01000110	01101
78	01111000	01110
74	01110100	01111
01	00000001	00100000
02	00000010	00100001
61	01100001	00100010
52	01010010	00100011
49	01001001	00100100
BA	10111010	00100101
57	01010111	00100110
41	01000001	00100111
56	01010110	001010000
45	01000101	001010001
66	01100110	001010010
6D	01101101	001010011
20	00100000	001010100
10	00010000	001010101
44	01000100	001010110
AC	10101100	001010111
88	10001000	001011000
58	01011000	001011001
08	00001000	001011010
64	01100100	001011011
96	10010110	001011100

After changing the bits according to the table above, the results of the bit string are as follows:

52 49 46 46 BA 78 00 00 57 41 56 45 66 6D 74 20 10 00 00 00 01 00 02 00 44 AC 00 00 88 58 01 00 02 00 08 00 64 61 74 61 96 78 00 00 7F 7F 7F 7F 7F 7F 7F 7E 7E 7E 7E 7E 7E 80 80 80 80 80 80 80

The system testing is done with several samples of audio files in *.wav extension with the 8-bit format. The audio files sizes are 1.62Mb, 2.61Mb, 7.08Mb, 9.27Mb, and 12.13Mb. The following table is the result of compression testing using the Boldi-Vigna Algorithm (ζ2) and the Elias Delta Code Algorithm as shown in Table 7.

Table 7. Test Result

Filename	Data size before compression (byte)	The data size after compressed (byte)	
		Boldi-Vigna (ζ2)	Elias Delta Code
record9_sample_stereo_44100.Wav	1.628.326	1.239.239	1.412.548
rec_music_8bit_441.Wav	2.610.880	1.937.831	2.236.220
record4_sample_stereo_44100.Wav	7.087.832	4.426.580	4.978.144
record6_sample_stereo_44100.Wav	9.277.838	7.364.960	8.346.480
record5_sample_stereo_44100.Wav	12.133.550	6.754.400	7.659.585

The calculation result of the performance comparison parameter from the compression algorithm in the form of Ratio of Compression, Compression Ratio, and Space Saving of each algorithm can be seen in Table 8.

Table 8. The results of the calculation of the compression algorithm performance measurement parameters.

Filename	Boldi-Vigna (ζ2)			Elias Delta Code		
	RC	CR	SS	RC	CR	SS
record9_sample_stereo_44100.Wav	76.10%	1.31	23.89%	86.74%	1.15	13.25%
rec_music_8bit_441.Wav	74.22%	1.34	25.77%	85.65%	1.16	14.35%
record4_sample_stereo_44100.Wav	62.45%	1.60	37.54%	70.23%	1.42	29.76%
record6_sample_stereo_44100.Wav	79.38%	1.25	20.61%	89.96%	1.11	10.03%
record5_sample_stereo_44100.Wav	55.66%	1.79	44.33%	63.12%	1.58	36.87%

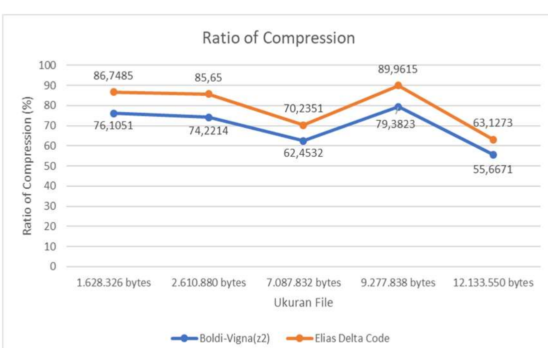

Figure 9. Comparison Ratio of Compression

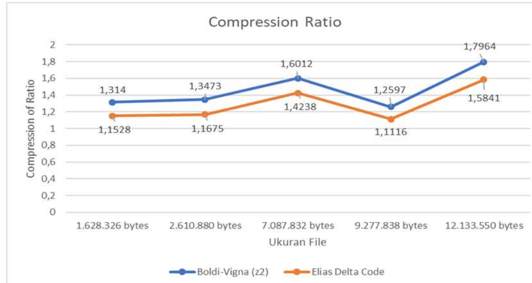

Figure 10. Comparison of Compression Ratio

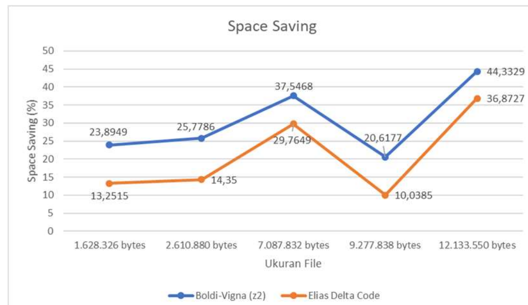

Figure 11. Comparison of Space Saving

5. CONCLUSION

Based on research that has been completed at the analysis stage and the system testing stage, the result shows that the Boldi-Vigna algorithm (ζ2) and the Elias Delta Code algorithm can compress audio files with the * .Wav 8-bit format extension, based on the ratio of Compression, Compression Ratio, and Space Saving, it can be concluded that the Boldi-Vigna algorithm is better at reducing the size of audio files * .Wav in 8-bit format, with an average value of 79.14%, 1.284, and 20.852%. The compression results obtained depend on the number of repetitions of characters or digital audio values contained in an audio file.

REFERENCES:

[1] Antoni, E. B. Nababan dan M. Zarlis, "Results Analysis of Text Data Compression on Elias Gamma Code, Elias Delta Code, and Levenstein Code," *International Journal of Science and Advanced Technology*, vol. 4, no. 9, pp. 17-24, 2014.

[2] Leni Marlina, Andysah Putera Utama Siahaan, Heri Kurniawan, Indri Sulistianingsih, "Data Compression Using Elias Delta Code" *International Journal of Recent Trends in Engineering & Research (IJRTER)* Vol. 03, Issue 08; August - 2017.

[3] Budiman, M. A., & Rachmawati, D. 2017. "On Using Goldbach G0 Codes and Even-Rodeh Codes for Text Compression." *IOP Conference Series: Materials Science and Engineering*. 180(1).

[4] Dian Rachmawati, Amalia, Aktualitas Gulo, "An Implementation Of Rabin Cryptography And Fibonacci Codes Algorithm In Image Files Security And Compression" *Journal Of Theoretical And Applied Information Technology* 15th July 2020. Vol.98. No 13.

[5] Dian Rachmawati, Budiman, M. A., & Saffiera, C. A. 2018. "An Implementation Of Elias Delta Code And ElGamal Algorithm In Image Compression And Security." *IOP Conference Series: Materials Science and Engineering.* 300(1).

[6] Salomon, D. 2007. *Data Compression Fourth Edition. London*: Spring.

[7] Salomon, D. & Giovanni Motta. 2010. *Handbook of Data Compression Data Fifth Edition.* London: Springer.

[8] Sayood, Khalid. 2006. *Introduction to Data Compression Third Edition.* Elsevier