Communications on Applied Electronics (CAE) – ISSN : 2394-4714
Foundation of Computer Science FCS, New York, USA
Volume 7– No. 35, April 2021 – www.caeaccess.org

An Improve Shannon Fano Data Compression Algorithm using Residue Number System

T.D. Lawal
Computer Science Department
Federal Polytechnic Offa
Nigeria

L.O. Olatunbosun
Department of Computer Science
Federal University of Agriculture
Abeokuta, Nigeria

K.A. Gbolagade
Department of Computer Science
Kwara State University Malete
Nigeria

ABSTRACT

The last two decades has witness the rapid development of hardware and software due to technological advancement. This has equally facilitated an increase in generation of information for storage and spread through the internet around the world. The rate at which storage and bandwidth facilities are being develop has not been able to match the rate at which information are been produce for storage and transmission. This has resulted in the researchers looking in the area of data compression. Many Data compression algorithms such as Shannon Fano, Huffman, Lempel Ziv, Arithmetic etc. have been develop. Shannon Fano was found to be one of the best compression algorithms. It is however having the challenges of low compression ratio, high compression factor, low amount of space saved and low saving percentage. In this paper, Residue Number System was embedded in the Shannon Fano algorithm to enhance its performance. File documents of various sizes was compressed using both Shannon Fano and RNS-Shannon Fano algorithms. The results show a significant improvement performance over the traditional Shannon Fano compression algorithm.

Keywords Embedded Shannon Fano (ESF), Residue Number System (RNS), Compression Ratio (CR), Compression Factor (CF)

Keywords

Data Compression Algorithm, Residue Number System

1. INTRODUCTION

Our world has been greatly transformed by information technology. The rapid development of hardware and software due to technological advancement has facilitated the storage and spread of information through the internet around the world. The rate of increase in the number of bytes of data store, processed and transmitted is tremendous. This happened as a result of growing internet technology and rapid development of mobile communication. The rate at which storage and bandwidth facilities developed however has not been able to match the rate at which information are produce. Data compression is the technology that is used to compliment the efforts of storage and bandwidth development. For effective storage, memory and bandwidth management, hardware and software technological development should be accompanying by data compression [1].

Data compression is a process of resizing a file or document to be smaller in size. It is the art of representing information in a compact form rather than its original or uncompressed form [2]. The major importance of data compression is that it enables data to be stored in a format that occupies less space than the original form thus reducing the amount of space require for storage or transmission [3].

The main aim of data compression is to eliminate redundancy in the data through different efficient methodology so that the reduced data can save space to store the data, time to transmit the data and cost to maintain the data. To eliminate the redundancy, the original file is represented with some coded notation and this coded file is known as ‘encrypted file’[4]. Data compression is broadly divided into lossy and lossless depending on whether you can get the exact original data back when the compressed data is later decompressed.

In a Lossless compression algorithm such as Huffman, Shannon Fano and Lempel Ziv, the reconstruction of the original data from the original file does not result in any loss of data. Therefore, the information does not change during the compression and decompression processes. These kinds of compression algorithms are called reversible compressions since the original message is reconstructed by the decompression process. Lossless compression techniques are used to compress medical images, text and images preserved for legal reasons, computer executable file and so on. Lossy compression algorithms on the other hand, results in reconstruction of the original message with loss of some information. It is not possible to reconstruct the original message using the decoding process and is called irreversible compression. The decompression process results in an approximate reconstruction. This technique is desirable in a situation when data of some ranges which could not be recognized by the human brain can be discarded. Lossy techniques could be used for multimedia images, video and audio to achieve more compact data compression Kodituwakku and Amarasinghe (2004).

2. BACKGROUND OF SHANNON FANO ALGORITHM

Shannon Fano compression algorithm is the type of lossless algorithm construct a prefix code based on the set of symbols and their probabilities or frequency. Different symbols are sorted according to their frequencies after which the sorted symbols with their occurrence are group into two parts according to their frequency [6]. It is an entropy encoding algorithm. Unlike Huffman, Shannon Fano algorithm follows a top down tree construction and code assigning approach with a symbol by symbol encoding and it does not achieve the lowest possible expected code word length [7]. Shannon Fano algorithm encode information depending on the frequency of occurrence by allotting small number of bits for information with large frequency and large number of bits for information with less frequency [8]. The Shannon Fano compression algorithm is achieved by the following steps:

Communications on Applied Electronics (CAE) – ISSN : 2394-4714
Foundation of Computer Science FCS, New York, USA
Volume 7– No. 35, April 2021 – www.caeaccess.org

1. determine the frequencies or probability of a given list of symbols, so that each symbol's relative frequency of occurrence is known.
2. Sort the table according to the frequency, with the most frequently occurring symbol at the left and the least common at the right.
3. Divide the list into two parts, with the total frequency counts of the left part being as close to the total of the right as much as possible.
4. The left part of the list is assigned the binary digit 0, and the right part is assigned the digit 1. This means that the codes for the symbols in the first part will all start with 0, and the codes in the second part will all start with 1.
5. Recursively apply the steps 3 and 4 to each of the two halves, subdividing groups and adding bits to the codes until each symbol has become a corresponding leaf on the tree.

This is represented in figure 1 below.
For instance, given the following message:
AAABBBBBBBBBBBBCCCCCCDDDDDDDEEEEFF FFFFFFFF (as presented in Table 1)
Total number of characters is 40.
The ASCII code size of a character is 8 bits.
Total size of the data is 8 * 40 = 320 bits

Table 1: Characters and their frequency

Character	A	B	C	D	E	F
Frequency	3	12	6	5	4	10

Using Shannon Fano compression algorithm, the message compression process is as shown below in Figure 1.

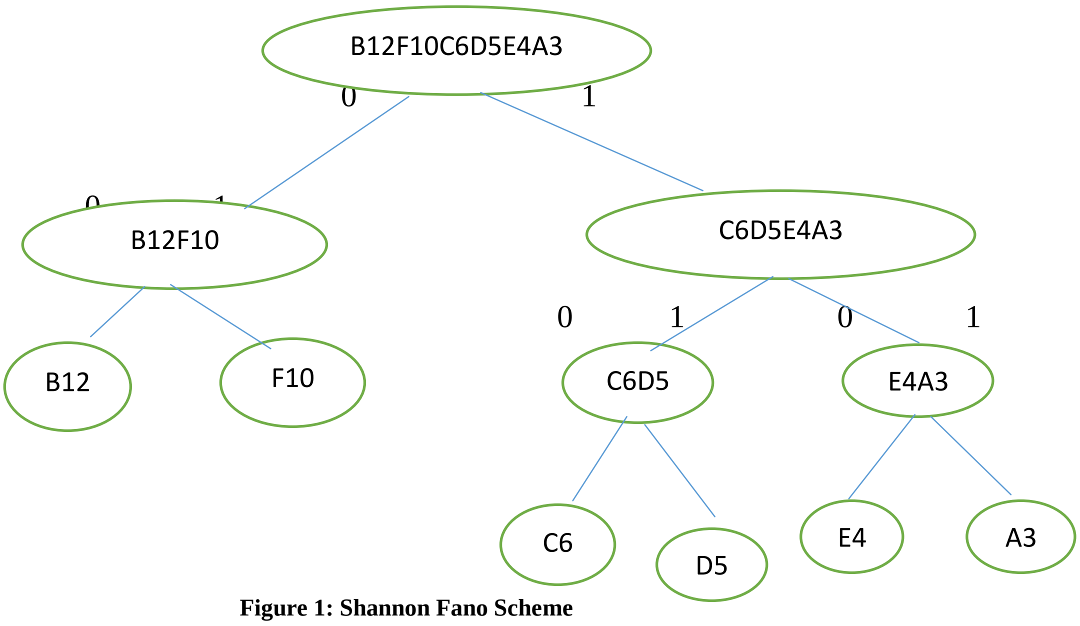

Figure 1: Shannon Fano Scheme

Table 2: Shannon Fano compression logic

Character	No of bit	Freq	bits*Freq
A	111	3	9
B	00	12	24
C	100	6	18
D	101	5	15
E	110	4	12
F	01	10	20
	16		98 bits

From Table 2 above, the size of the message is 98 bits.
The size of the chart is the total number of bits which is 16 bits The ASCII code of the alphabet is the number of different characters multiply by 8 (6 * 8) = 48 bits.
The total size of the message is 98 + 16 + 48 = 162 bits
This means that Shannon Fano has compressed the message from 320 bits to 162 bits i.e., the message has been more than 50% compressed.

3. PROPOSED SCHEME

A highly compressed data for storage and/or transmission is being proposed. This is achieved by the introduction of residue number system (RNS). Residue number system (RNS) is a non-weighted, non-positional number system which can be represented by specifying its base. RNS is different from Decimal or Binary Number System as it does not have a single fixed radix. Their bases are represented by a u tuple of integers (m_1, m_2, m_3,, ., ., m_u,) where each of these bases are called modulus. Any given integer in RNS is represented by a set of residues obtained by modulo which are relatively prime to each other, dividing the integer with a moduli set [9].

One of the main reasons why computing hardware based on Weighted Number System (WNS) cannot be speed up beyond certain bounds is carry propagation. The major challenge therefore in improving the computer arithmetic unit performance is the reduction or elimination of carry chain [10]. RNS has been the most challengeable alternative number system in computer arithmetic for more than half a century [11]. Its ability to perform addition, subtraction and multiplication without carry-propagation between residues make it stand out. RNS is a numeral system representing integers by their values modulo several pairwise coprime integers called moduli. The inherent features such as carry free operations, parallelism, modularity and fault tolerant possessed by RNS conferred very great advantages over conventional binary number system. These advantages make its application wide in Digital Signal Processing applications. However, RNS have challenges in sign detection, magnitude comparison, overflow detection and division, Sousa 2007. The most critical aspect of RNS applications is the choice of moduli set and data conversion. The complexity and the speed of the conversion algorithm depend on the choice of moduli set, [13]. Any integer X in the dynamic range, M = mi, m_2, m_3,...,m_n, is represented by N-tuple (x_1, x_2, x_3,...,x_n), where x_i is the residue of X in moduli m_i for i = 1,2,3,...,n. To revert residues number back to its weighted equivalent value, Chinese Remainder Theorem (CRT) was used [14].

Using CRT, the number X can be computed from its residue by the following expression:

$$X = | \sum_{i=1}^{n} M_i |M_i^{-1}|m_i * x_i|M$$

Such that

$M = \Pi_{i=1}^{n} m_i = m_1 * m_2 *...* m_n$

$M_i = M \ / \ m_i = (m_1 * m_2 * ... m_n) / m_i$

$_M^{-1}$ is the multiplicative inverse of M_i with respect to modulus m_i

x_i is the set of residue X with respect to moduli i

The following is the framework for the proposed Embedded Shannon Fano (ESF) algorithm:

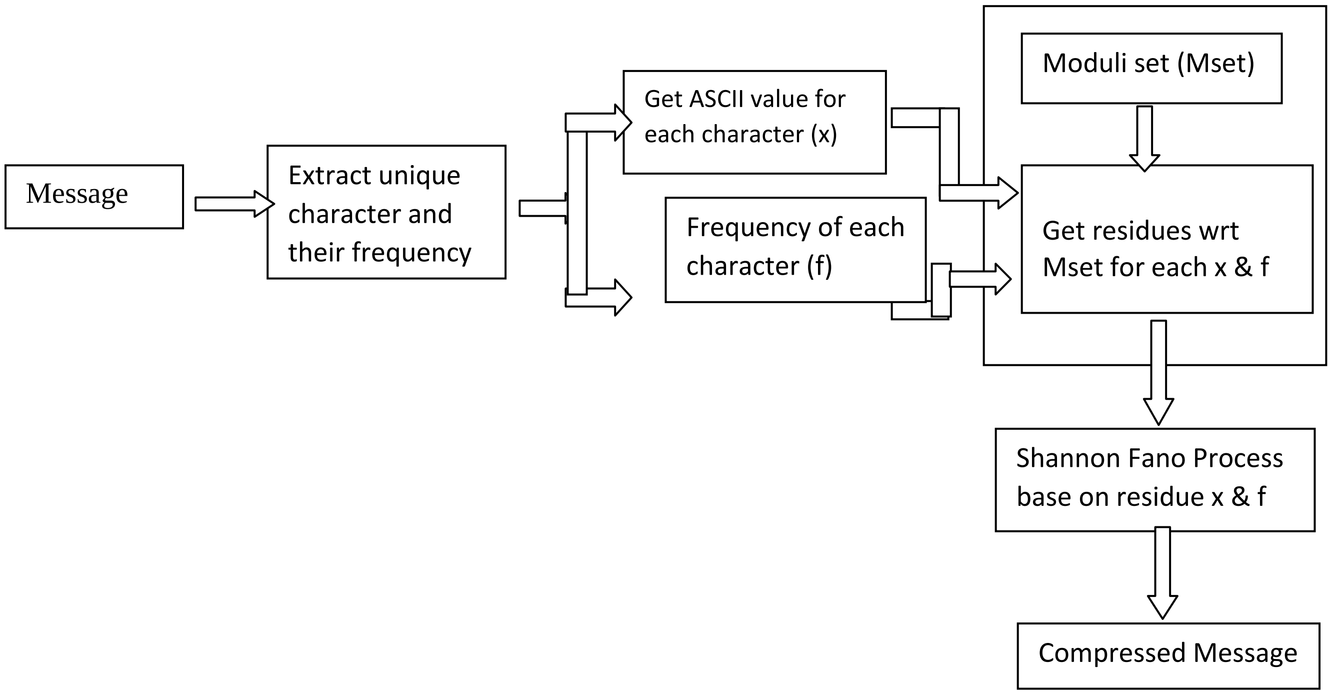
Message
Extract unique character and their frequency
Get ASCII value for each character (x)
Frequency of each character (f)
Moduli set (Mset)
Get residues wrt Mset for each x & f
Shannon Fano Process base on residue x & f
Compressed Message

Figure 2: proposed RNS ESF framework

In figure 2, the characters that form the message and their frequencies of occurrence were extracted along with their ASCII value. Using traditional moduli set, given a character X, the forward conversion to their equivalent residues is given as

$x_i = X \ mod \ m_i = |X|_{m_i}, 0 \leq x_i < m_i$.

Thus $X = (x_1, x_2, x_3, ..., x_n)$ for moduli set $\{m_1, m_2, m_3, ... \ m_{n-1}, m_n\}$. As shown in Table 3. The operation is then performed on the residues instead of decimal or binary values. The result of the frequency multiplication by Shanon Fano code in RNS is shown in Table 4. The result of the compressed message in both Shanon Fano and RNS ESF code is shown in Table 5.

Table 3: forward conversion

Character	**ASCII Code**	**Binary Code**	**RNS with Moduli set**			**Bits Space**
			3	4	5	
A	65	8	2	1	0	4
B	66	8	0	2	1	4

C	67	8	1	3	2	5
D	68	8	2	0	3	5
E	69	8	0	1	4	5
F	70	8	1	2	0	4
		48				27

Table 4: Frequency multiplication by Shannon Fano code in RNS

Character	freq	RNS with moduli set			Shannon Fano code	RNS with moduli set			Freq* Shannon Fano code	RNS with moduli set			Bits Space
		3	4	5		3	4	5		3	4	5	
A	3	0	3	3	111=3	0	3	3	3*3=9	0	1	4	5
B	12	0	0	2	00=2	2	2	2	12*2=24	0	2	4	6
C	6	0	2	1	100=3	0	3	3	6*3=18	0	2	3	5
D	5	2	1	0	101=3	0	3	3	5*3=15	0	3	0	4
E	4	1	0	4	110=3	0	3	3	4*3=12	0	0	2	4
F	10	1	2	0	01=2	2	2	2	10*2=20	2	0	0	4
	40				16 bits				108 bits				28 bits

Table 5: compressed message in Shannon Fano and RNS ESF code

	Shannon Fano codes	RSN Codes
Message	98 bits	28 bits
ASCII binary codes	48 bits	26 bits
Table or chart	16 bits	16 bits
	162 bits	70 bits

4. RESULT ANALYSIS AND DISCUSSION

A file document of 3,888, 4224, 7976, 8984, 11056 and 13120 bits messages were compressed using both Shannon Fano and proposed RNS ESF compression algorithms. The results were analyzed using Compression Ratio (CR), Compression Factor (CF), Saving Percentage (SP) and Amount of Spaced saved.

.

$$Compression\ Ratio = \frac{size\ before\ compression}{size\ after\ compression}$$

$$Compression\ Factor = \frac{size\ after\ compression}{size\ before\ compression}$$

$$Saving\ Percentage = \frac{size\ before\ compression - saving\ after\ co\ mpression}{size\ before\ compression} * 100$$

$$Amount\ of\ Space\ save = 1 - \frac{size\ after\ compression}{size\ beforer\ compression}$$

Table 6: original message size and their corresponding Shannon Fano

Original Message size (bits)	Shannon Fano size (bits)	RNS-EFS size (bits)
3888	3006	654
4224	3316	693
7976	6069	972
8984	6439	969
11056	8404	1130
13120	10210	1203

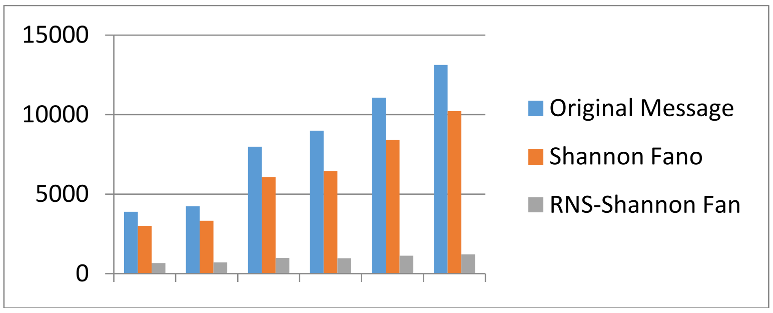

Figure 3: original message size and their Shannon Fano and RNS-ESF sizes

From table 6 and figure 3, the original messages were better compressed by embedded RNS-Shannon Fano than traditional Shannon Fano

Table 7: Compression Ratio

Original Message size (bits)	Shannon Fano size (bits)	RNS-Shannon Fano size (bits)
3888	1.6322418136020151	23.563636363636363
4224	1.5909604519774012	25.75609756097561
7976	1.5556855861127366	38.90731707317073
8984	1.6304900181488202	44.03921568627451
11056	1.5112083105522143	48.27947598253275
13120	1.4489232468249587	54.21487603305785

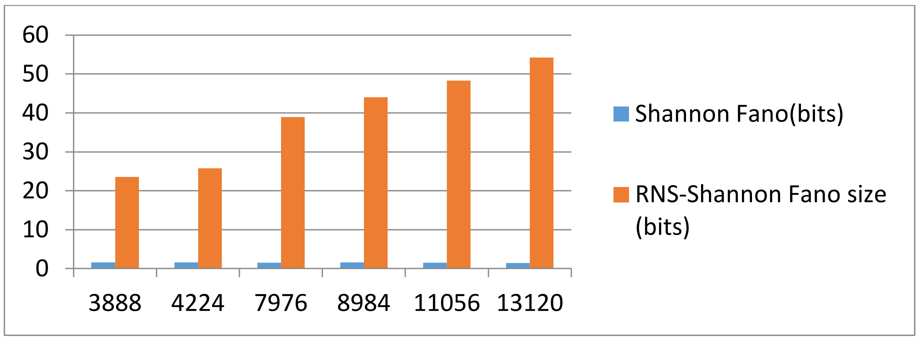

Figure 4: compression ratio

From table 7 and figure 4, it is observed that compression ratio of RNS-ESF are significantly higher than that of Traditional Shannon Fano.

Table 8: Compression Factor

Original Message size (bits)	Shannon Fano size (bits)	RNS-Shannon Fano size (bits)
3888	0.6126543209876543	0.04243827160493827
4224	0.6285511363636364	0.038825757575757576
7976	0.6428034102306921	0.02570210631895687
8984	0.6133125556544969	0.022707034728406055
11056	0.6617221418234442	0.02071273516642547
13120	0.6901676829268293	0.01844512195121951

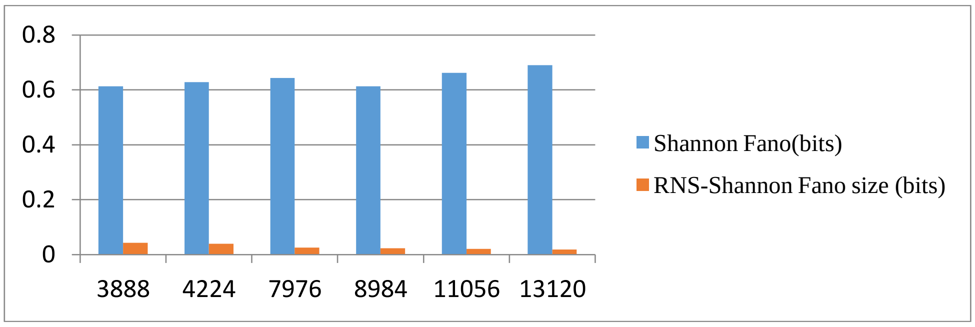

Figure 5: Compression factor

From table 8 and Figure 5, it is observed that compression factor of embedded RNS-ESF is significantly lower than that of Traditional Shannon Fano.

Table 9: Amount of Space Save

Original Message size (bits)	Shannon Fano size (bits)	RNS-Shannon Fano size (bits)
3888	0.38734567901234573	0.9575617283950617
4224	0.37144886363636365	0.9611742424242424
7976	0.3571965897693079	0.9742978936810431
8984	0.38668744434550306	0.9772929652715939
11056	0.33827785817655576	0.9792872648335745
13120	0.3098323170731707	0.9815548780487805

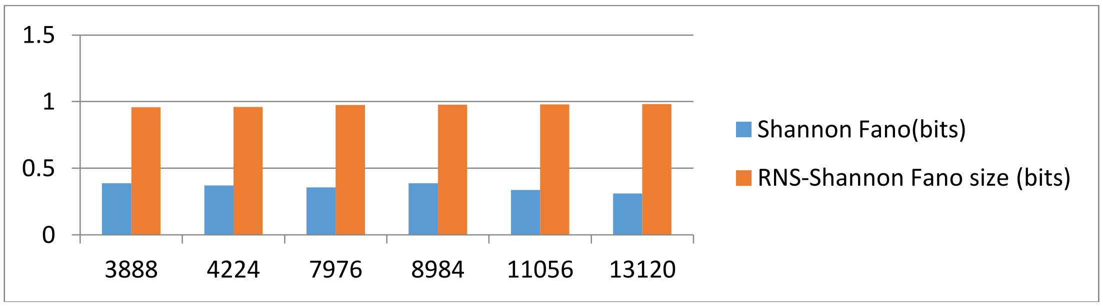

Figure 6: Amount Space Saved

From Table 9 and Figure 6, it is observed that amount of space saved by RNS-ESF are significantly higher than that of Traditional Shannon Fano.

Table 10: Saving Percentage

Original Message size (bits)	Shannon Fano size (bits)	RNS-Shannon Fano size (bits)
3888	38.73456790123457	95.75617283950618
4224	37.14488636363637	96.11742424242425
7976	35.71965897693079	97.42978936810431
8984	38.66874443455031	97.72929652715939
11056	33.82778581765557	97.92872648335745
13120	30.98323170731707	98.15548780487805

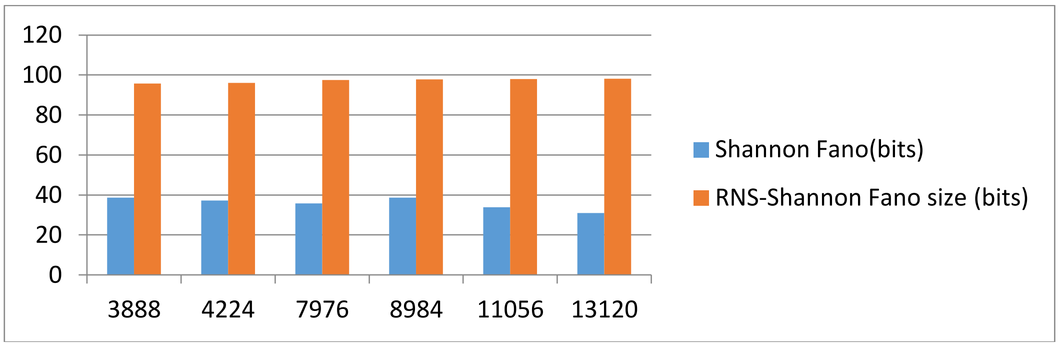

Figure 7: Saving Percentage

From Table 10 and Figure 7, it is observed that saving of RNS-ESF are significantly higher than that of Traditional Shannon Fano

5. CONCLUSION

The improve technology of storage and bandwidth facilities development has not been able to match huge amount of information that is daily churn out for storage and transmission. These challenges rekindle the interest of researchers to continuously focus on the area of data compression. Shannon Fano algorithm has proved to be one of the effective means of compression, however, the algorithm has the challenges of low compression ratio, high compression factor, low amount of space saved and low saving percentage. In this paper, we propose a new Shannon Fano compression algorithm using RNS with traditional moduli set. With the introduction of RNS to Shannon Fano algorithm, there was a significant improvement in ESF performance in term of these metrics as shown in Tables 6, 7, 8, 9 and 10, and Figures 3, 4, 5, 6 and 7. It is evident that our proposed method produced a better result than the traditional method.

6. REFERENCES

[1] Rachesti, D. A., Purboyo, T. W. and Prasasti, A. L. (2017): *Comparison of Text Data Compression Using Huffman, Shannon-Fano, Run Length Encoding, and Tunstall Methods.* International Journal of Applied Engineering Research ISSN 0973-4562 Volume 12, Number 23 (2017) pp. 13618-13622

[2] Pu, I. M., (2006): Fundamental Data Compression, Elsevier, Britain.

[3] Jacob, N. Somvanshi P. and Tornekar R. (2012): *Comparative Analysis of Lossless Text Compression Techniques.* International Journal of Computer Applications (0975– 8887) Volume 56– No.3,

[4] Bhattacharjee, A. K., Bej, T. and Agarwal S. (2013): Comparison Study of Lossless Data Compression Algorithms for Text Data. IOSR Journal of Computer Engineering (IOSR-JCE) e-ISSN: 2278-0661, p- ISSN: 2278-8727Volume 11, Issue 6 PP 15-19

[5] Kodituwakku, S.R. and Amarasinghe, U. S. (2004) Comparison of Lossless Data Compression Algorithms For Text Data. Indian Journal of Computer Science and Engineering Vol 1 No 4 416-426

[6] Mahesh V., Ekjot S. W. and Aditya G. (2014) Data Compression Using Shannon-Fano Algorithm Implemented by VHDL IEEE International Conference on Advances in Engineering & Technology Research (ICAETR - 2014), August 01-02, 2014,

[7] Komal S. and Kunal G. (2017) Lossless Data Compression Techniques and Their Performance. International Conference on Computing, Communication and Automation (ICCCA2017)

[8]Shanmugasundaram S. and Lourdusamy, R. (2011): A Comparative Study Of Text Compression Algorithms. International Journal of Wisdom Based Computing, Vol. 1 (3), December 2011.

[9] James J. and Pe, A. (2015): Error Correction based on Redundant Residue Number System 978-1-4799-9985-9/15/$31.00 ©2015 IEEE

[10] Gbolagade, K. A.; Cotofana, S. D. (2008). *Residue Number System Operands to Decimal Conversion for 3-Moduli Sets [IEEE 2008 51st IEEE International Midwest Symposium on Circuits and Systems (MWSCAS) - Knoxville, TN, USA (2008.08.10-2008.08.13)] 2008 51st Midwest Symposium on Circuits and Systems - Residue Number System operands to decimal conversion for 3-moduli sets. , (), 791–794.*

[11] Molahosseini, A. S. and Sousa L. (2017): Introduction to Residue Number System: Structure and Teaching Methodology.

[12] Sousa, L. Efficient method for magnitude comparison in RNS based on two pairs of conjugate moduli. Proceedings of the 18th IEEE Symposium on Computer Arithmetic, 2007.

[13] Gbolagade, K. A.; Chaves, R.; Sousa, L.; Cotofana, S. D. (2010). An Improved RNS Reverse Converter for the $\{2^{2n+1}-1, 2^n, 2^n-1\}$ Moduli Set, *[IEEE 2010 IEEE International Symposium on Circuits and Systems - ISCAS 2010 - Paris, France.*

[14] Aremu, I. A. and Gbolagade, K. A. (2017): Redundant Residue Number System Based Multiple Error Detection and Correction Using Chinese Remainder Theorem. Software Engineering. ISSN2376-8029(Print) 2376-8037 (online)5(5):pg 72-80.