A Novel and Efficient LZW-RNS Scheme for Enhanced Information Compression and Security

Abdul-Barik Alhassan, Kazeem A. Gbolagade, and Edem K. Bankas

***Abstract*—** Encryption and compression of data is an aspect of information theory in which the core objective is to reduce the amount of data to be stored or transmitted. In transmitting data through high-speed media, high-speed and secured data compression is desired. The Lempel-Ziv-Welch algorithm is one of the compression algorithms that capitalises on the reoccurrence or repetition of words or phrases in a file for compression, reducing the number of bits or the execution time is therefore crucial in this paper. The inherent features of RNS such as carry free, parallelism, and fault tolerant capabilities is of great essence in this research. RNS has been efficiently applied to data compression using the moduli set $\{2^n-1,\ 2^n,\ 2^n+1\}$ which is an adaptation of the LZW algorithm that codes strings of characters with codes of a fixed number of bits. A number of strategies have been employed in the development of this new scheme. Firstly, the LZW algorithm has been modified by applying RNS resulting in new efficient data encryption and decryption schemes, new encoder and decoder pairs which also allows for conversion from one number representation to the other. Secondly, the output of the LZW algorithm has been modified to allow for secrete order bit stream or channel or residual archiving or transmission of data through networks so that network intruders cannot make meaning of intercepted data. Finally, simulations have been done using MatLab for some documents (of varying sizes) a number of times and the averages CPU times taken to determine the efficiency of the modified or proposed LZW-RNS scheme. This proposed scheme has led to the development of new data encoding and decoding schemes as well as enhanced security, storage capacity, and speed of compression and transmission than the traditional LZW and equivalent state of the art LZW compression schemes.

***Index Terms*—**Compression, Decoder, Encoder, Encryption, Lempel-Ziv-Welch, Residue Number System (RNS).

I. INTRODUCTION

Data compression is the technique of reducing the amount of information to be stored or transmitted by removing excess information without the loss of the ability to reconstruct the original data. Data compression algorithms are employed in different file formats including video, sound, image, and text, and can be classified as either lossy or lossless, dictionary or non-dictionary based [4-5], [11].

Alhassan [7] researched on improving the traditional Huffman's data encoding algorithm by applying RNS. The research presented an enhanced data compression with the traditional Huffman's encoding where the frequency of occurrences of each character are used to generate binary codes. A similar research is done in [8] on enhancing the security of a digital image using RNS and the method of Arnold Transforms for image encoding. They stated that digital images have many applications, and hence the need to secure it in transmitting through networks in particular.

There exist a lot of literature on improving the Lempel-Ziv-Welch algorithm (LZW). In [9] a proposed hardware approach to the LZW algorithm for binary data compression is presented where they noted that software implementations are often not fast enough in transmitting high speed data through high speed media and therefore the essence of their research. Kaur [10] modified the dictionary of the LZW as content based addressable memory (CAM) array which utilizes less bits than its ASCII code. Data stored on disk or tapes or transferred over communication media in commercial computer systems generally contains significant redundancy. Several encountered problems have hindered the use of automatic data compression, a new scheme whose principle is not found in general commercial methods is presented in [11]. In [12] a comparative study of text compression algorithms is done where the LZW is found to be the least performing in terms of bits per compression (BPC). Parthasarathy [13] in their research to determine the existence of secrete information hidden within an image, the LZW is used to manipulate 128-bit input using flipping, substitution, and permutation to achieve encryption and decryption. Mahyar [14] recognised that 3G networks can revolutionise data and communication exchanges amongst people in an unprecedented fashion than 2G and 2.5G networks, recognising the fact that 3G are built using KASUMI block cipher which has error detection and correction as a major challenge, and hence the use of RNS. A method premised on modulus projection approach where the algorithm reduces considerably the computation overhead for RRNS codes decoding because of the employed hybrid method involving integer recovery process is proposed in [15].

Data compression is a defined method or encoding technique which reduces data size substantially based on an existing law, and that important applications are in storage systems and communication networks which should be highly secured [16-18].

Barati et al. in [19], applied Redundant Residue Number System (RRNS) to data distribution for mobile systems and wireless networks based on peer-to-peer protocol. Its error detection and correction capability was used together with multi-level RNS to propose a new scheme which showed better performance.

The focus of this paper is to apply RNS to the LZW algorithm using the traditional moduli set for efficient and secured data encoding and decoding.

II. MATERIALS AND METHODS

A. The LZ Compression Algorithms

Data compression algorithms are classified as either lossless

ISSN: 2278 – 1323

or lossy. It is lossless only if it is possible to exactly reconstruct the original data from the compressed version or lossy where an approximated data can be reconstructed because of the removal of some part of the data [4-5], [11-12].

Most lossless compression techniques though few nowadays, are based on dictionary or probability and entropy, utilising the occurrence of the same character or string in the data to achieve compression. The Dictionary based compression technique known as the Lempel-Ziv scheme, divided into two families, including those derived from LZ77 (LZ77, LZSS, LZH and LZB) and those derived from LZ78 (LZ78, LZW and LZFG).

Dictionary coding techniques rely upon the observation that there are correlations between parts of data. The basic idea is to replace those repetitions by (shorter) references to a "dictionary" containing the original [4-5], [11-12].

1.1 The Lempel-Ziv Algorithms

The lossless Lempel-Ziv algorithm is not a single algorithm, but a whole family of algorithms, stemming from the two algorithms proposed by Jacob Ziv and Abraham Lempel in their landmark papers in 1977 and 1978 as shown below:

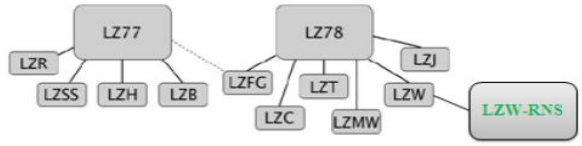

Fig.1 Classification of LZ Algorithms

1.1.1 LZ77 Algorithms

Jacob Ziv and Abraham Lempel in 1977 presented their dictionary-based scheme for lossless data compression which exploits the repetition of words and phrases within a text file. These repetitions can be encoded as a pointer to an earlier occurrence, with the pointer accompanied by the number of characters to be matched. It requires no prior knowledge of the source and assumptions about the characteristics of the source [4-5].

1.1.2 LZ78 Algorithms

Jacob Ziv and Abraham Lempel again in 1978 presented their dictionary based scheme known as LZ78, which is a dictionary based compression algorithm that maintains an explicit dictionary built both at the encoding and decoding side, and must follow the same rules to ensure that they use an identical dictionary.

Even though LZ78 has a serious drawback of unbound growth of the dictionary, it however has the ability to capture patterns and hold them indefinitely. The unbound drawback of LZ78 can be dealt with by using static dictionary encoder or by clearing old entries [4-5], [11-12].

1.1.3 LZW Algorithm

LZW algorithm is a universal lossless data compression algorithm created by Abraham Lempel, Jacob Ziv, and Terry Welch. It was published by Welch in 1984 as an improved implementation of the LZ78 algorithm published by Lempel and Ziv in 1978, which is simple to implement, and has the potential for very high throughput in hardware implementations.

The scenario described by Welch (1984), encodes sequences of 8-bit data as fixed-length 12-bit codes. The codes from 0 to 255 represent 1-character sequences consisting of the corresponding 8-bit character, and the codes 256 through 4095 are created in a dictionary for sequences encountered in the data as it is encoded [4-5], [11-12].

B. The Residue Number System

RNS is defined by a basis consisting of a set of co-prime numbers, called moduli $\{m_1, m_2, m_3 \dots m_n\}$, where the greatest common divisor (GCD) between any two moduli is one, that is GCD$(m_i, m_j) = 1, \forall\, i \neq j$. An integer X is represented by n-tuple $(r_1, r_2, \dots, r_n)$ in RNS where the residue $r_i = |X|_{m_i}$ for $i = 1, 2, \dots, n$, and $|X|_{m_i}$ is defined as $X \bmod m_i$. The Dynamic range of the RNS is given by $M = \prod_{i=1}^{n} m_i$. Using the Chinese Remainder Theorem (CRT), an integer X can be calculated from its residue digits $(r_1, r_2, \dots, r_n)$ as follows;

$$X = \left| \left| \sum_{i=1}^{k} M_i^{-1} r_i \right|_{m_i} M_i \right|_M \qquad (1)$$

Where $M = \prod_{i=1}^{n} mi$, $Mi = \frac{M}{m_i}$, and M_i^{-1} is the multiplicative inverse of M_i with respect to m_i. [1-2], [6].

1.1 The Conversion Process

Convertors are used to change from one number representation to the other. A forward convertor converts from Decimal to RNS, and the reverse convertor converts from RNS to Decimal representation.

1.1.1 Forward Conversion Process for the Moduli Set $\{2^n - 1, 2^n, 2^n + 1\}$

Let $m_1 = 2^n + 1$, $m_2 = 2^n$, and $m_3 = 2^n - 1$; Then $M = [0, 2^{3n} - 2^n - 1]$ (where the upper end of the range $(m_1 m_2 m_3)$, is uniquely defined by the residue set $\{r_1, r_2, r_3\}$).

X is a $3n$-bit number which can be represented as [2];

$$X = x_{3n-1} x_{3n-2} \dots x_1 x_0 \qquad (2)$$

Since $r_i = |X|_{m_i}$, the $r_i's$ can be computed as follows;

r_2 is the n least significant bit (LSB) of X in binary.

For r_1 and r_3, we partition X into three (3) n-bit blocks B_1, B_2, and B_3 where;

$B_1 = \sum_{j=2n}^{3n-1} x_j 2^{j-2n}$, $B_2 = \sum_{j=n}^{2n-1} x_j 2^{j-n}$, and

$B_3 = \sum_{j=0}^{n-1} x_j 2^j$ } (3)

which implies $X = B_1 2^{2n} + B_2 2^n + B_3$ (4)

Therefore,

$r_1 = |X|_{2^n+1} = |B_1 2^{2n} + B_2 2^n + B_3|_{2^n+1}$,

and simplifying further,

$r_1 = |B_1 - B_2 + B_3|_{2^n+1}$

Similarly, $r_3 = |B_1 + B_2 + B_3|_{2^n-1}$

For example, given the moduli set $\{2^n - 1, 2^n, 2^n + 1\}$, $n = 3$, and X representing a character of a message to be encoded, with ASCII character representation $X = 50$. Then the conversion is as follows;

$50 = 110010 = 000110010$ (9 -bits, since X is a $3n - \text{bit}$ number)

Since $n = 3$, we partition X into 3-bits blocks. Thus, $B_1 = 000$, $B_2 = 110$, *and* $B_3 = 010$

Therefore; $|50|_{2^3+1} = |50|_9 = |0 - 6 + 2|_9 = 5$, and

$|50|_{2^3-1} = |50|_7 = |0 + 6 + 2|_7 = 1$.

The hardware realisation of the forward conversion is

ISSN: 2278 – 1323

represented below;

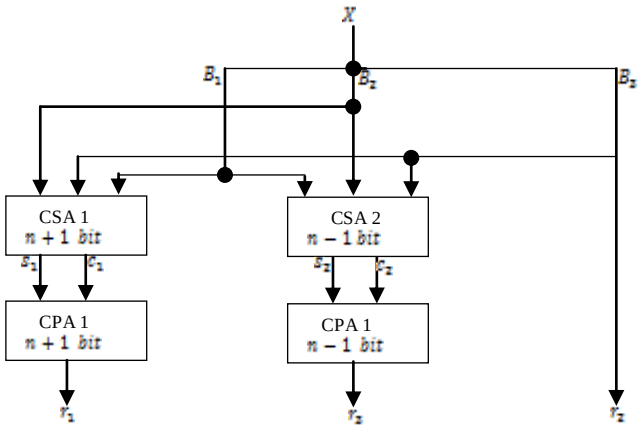

Fig. 2 Forward Convertor

1.1.2 Reverse Conversion Process for the Moduli Set $\{2^n-1,\ 2^n,\ 2^n+1\}$

The reverse conversion (RNS to Decimal) is done using the CRT formulated as;

Given the RNS numbers $X=(x_1,x_2,x_3)$ with respect to the moduli set $\{2^n-1, 2^n,\ 2^n+1\}$, where $m_1=2^n-1$, $m_2=2^n$ and $m_3=2^n+1$.

Then, $M_1=2^n(2^n+1)$; $M_2=(2^{2n}-1)$, and $M_3=2^n(2^n-1)\}$ (5)

Theorem 1*:* For the given moduli set, we have

$$\left.\begin{aligned}|M_1^{-1}|_{m_1}&=|2^{n-1}|_{m_1}\\ |M_2^{-1}|_{m_2}&=|-1|_{m_2}\\ |M_3^{-1}|_{m_3}&=|-2^{n-1}|_{m_3}\end{aligned}\right\} \quad (6)$$

Theorem 2: For the given moduli set, any RNS number X can be represented as;

$$X=2^n\xi+x_2 \quad (7)$$

where,

$$\xi=\left|\begin{matrix}|(2^nx_1+x_1)2^{n-1}|_{2^{2n}-1}+|-2^nx_2|_{2^{2n}-1}\\ +|-x_3|_{2^{2n}-1}+|+2^{n-1}x_3|_{2^{2n}-1}\end{matrix}\right|_{2^{2n}-1} \quad (8)$$

Proof: Substituting equations (5) and (6) into (1) and factorizing out 2^n, (7) is obtained.

Hardware Implementation

Equation (7) can further be simplified as follows

$$\xi=|\varphi_1+\varphi_2+\varphi_3+\varphi_4|_{2^{2n}-1} \quad (9)$$

where

$$\varphi_1=|(2^nx_1+x_1)2^{n-1}|_{2^{2n}-1} \quad (10)$$

$$\varphi_2=|-2^nx_2|_{2^{2n}-1} \quad (11)$$

$$\varphi_3=|-x_3|_{2^{2n}-1} \quad (12)$$

$$\varphi_4=|2^{n-1}x_3|_{2^{2n}-1} \quad (13)$$

Now, we consider (9)-(13) and simplify them for implementation in a VLSI system. It is necessary to note that $x_{i,j}$ means the *j*-th bit of x_i.

Evaluation of φ_1

The residue x_1 can be represented as follows;

$$x_1=\ x_{1,n-1}\cdots x_{1,1}x_{1,0} \quad (14)$$

Thus,

$$|(2^nx_1+x_1)2^{n-1}|_{2^{2n}-1}$$

$$=\left|2^{n-1}\left(\underbrace{x_{1,n-1}\cdots x_{1,1}x_{1,0}}_{2n-bits}\overbrace{0\ldots0}^{n\ Bits}+\underbrace{\overbrace{0\ldots0}^{n\ Bits}x_{1,n-1}\cdots x_{1,0}}_{2n}\right)\right|_{2^{2n}-1}$$

$$=\left|2^{n-1}\left(\underbrace{x_{1,n-1}\cdots x_{1,1}x_{1,0}x_{1,n-1}\cdots x_{1,1}x_{1,0}}_{2n-bits}\right)\right|_{2^{2n}-1}$$

$$=\overbrace{x_{1,0}x_{1,n-1}\cdots x_{1,1}x_{1,0}}^{n+1\ Bits}\underbrace{x_{1,n-1}\cdots x_{1,n+2}x_{1,1}}_{n-1} \quad (15)$$

Similarly, φ_2, φ_3, and φ_4 can be evaluated as follows;

The residue x_2 can be represented as;

$$\varphi_2=|-2^nx_2|_{2^{2n}-1}=\overline{x}_{2,n-1}\ldots\overline{x}_{2,1}\overline{x}_{2,0}\overbrace{11\ldots11}^{n\ Bits} \quad (16)$$

$$\varphi_3=|-x_3|_{2^{2n}-1}=\overbrace{11\ldots11}^{n\ Bits}\underbrace{\overline{x}_{3,n-1}\ldots\overline{x}_{3,1}\overline{x}_{3,0}}_{n\ Bits} \quad (17)$$

$$\varphi_4=|2^{n-1}x_3|_{2^{2n}-1}=\underbrace{0x_{3,n}\ldots x_{3,1}x_{3,0}}_{n+1\ Bits}\overbrace{00\ldots00}^{n-1\ Bits} \quad (18)$$

Proposed Architecture

ξ is computed according to equation (7) where all the parameters are defined in equations (14)–(18). As shown in *Fig* 1.3, ξ is computed using CSAs, CPAs and a MUX. The final computation for (7) is a concatenation which does not require any hardware.

The schematic diagram for the proposed scheme is shown below:

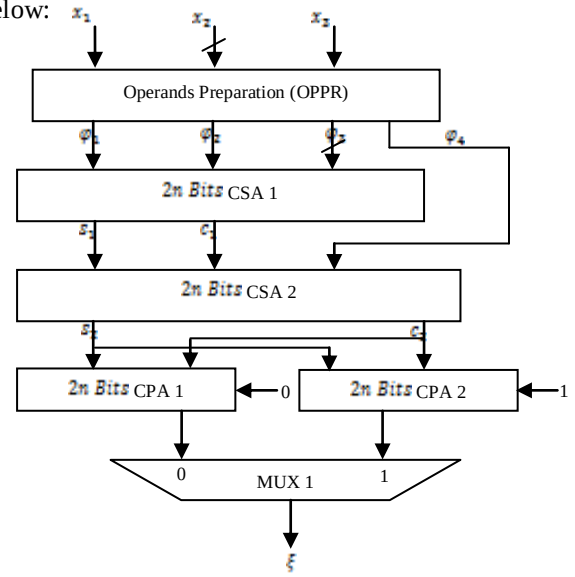

Fig. 3 Reverse Convertor

For example, given the moduli set $\{2^n-1, 2^n, 2^n+1\}$, $n=3$, and X representing a character of a message to be encoded, with ASCII character representation $X=50$. Then the conversion is as follows;

The moduli set for n=3 is {9, 8, 7}; then $M_1=56, M_2=63, M_3=72$;

$|M_1^{-1}|_{m_1}=|56^{-1}|_9=5,\ |M_2^{-1}|_{m_2}=|63^{-1}|_8=7,\ and\ |M_3^{-1}|_{m_3}=$

Therefore, from (1.0),

$X=||(56\times5\times5)|_{504}+|(63\times7\times2)|_{504}|(72\times4\times1)|_{504}|_{504}$

$$=|392+378+288|_{504}\ =|50|_{504}=50$$

III. RESULTS AND DISCUSSIONS

The proposed scheme consists of a modified encoder and decoder pair with enhanced security, speed, and compression ratio. RNS is applied to the ASCII character with decimal representation X which is used in the encryption and decryption process using the LZW algorithm.

ISSN: 2278 – 1323

1.1 The Proposed LZW-RNS Encoder

The initial dictionary with single character codes in decimals is created and then converted into its residues in a process termed forward conversion. The encoding process is continued with the modified algorithm in residues. The compressed or encoded message is then transmitted in three bit stream channels or residuals in a particular secrete order as shown in Fig. 6.

1.2 The Proposed LZW-RNS Decoder

The secrete order transmitted three bit stream channel message is received, reorganised by the decoder pair and then converted to its decimal representations by a process known as reverse conversion with the moduli set $\{2^n - 1, 2^n, 2^n + 1\}$. The modified LZW decoding process continues until the original message is acquired back as shown in Fig. 6

1.3 Performance Analysis of the LZW and Proposed LZW-RNS (3-Moduli set) Scheme

The goodness and efficiency of a data compression algorithm is measured by its simplicity to implement, the compression ratio (reducing storage space), data transmission speed, and enhanced security (Kinsner and Greenfield, 1991 and Ammar et. al., 2001). The performance analysis of the LZW and RNS-LZW applied algorithm is done on compression and speed of execution.

1.3.1 Performance Analysis on the Speed of Compression using MATLAB

The performance analysis on the speed of compression is done using different Microsoft word documents of varying sizes. The LZW and RNS-LZW applied methods are run on the file using Matlab on a 1.3GH Processor three times and the average CPU time is taken for both the Encoder and Decoder as well as compressed files as shown in Table I below.

From Table I, notice that there is an overall gain in execution time of the Proposed LZW-RNS (3-Moduli set) of 2.8645**s**, a reduction in compression of over 15% (15.6250**kb**), and over 40% (37**kb**) gain to the original file size. The time and compression efficiencies are shown Fig. 4 and Fig. 5 respectively.

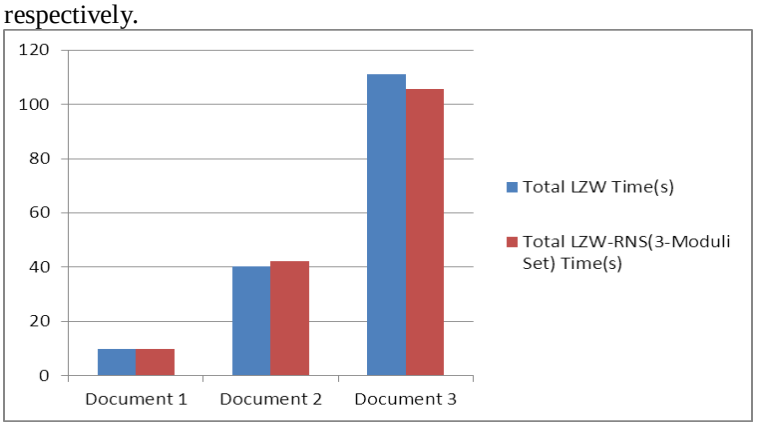

Fig. 4: The Execution Time of LZW and Proposed LZW-RNS (3-Moduli set) Scheme

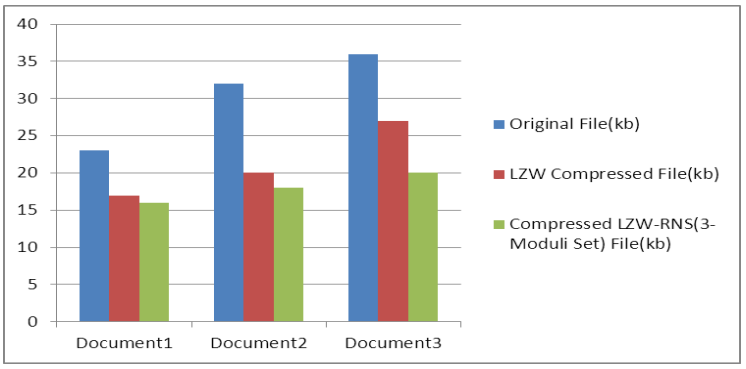

Fig. 5: Compression Efficiency of LZW and Proposed LZW-RNS (3-Moduli set) Scheme

Table I: Simulation Results of LZW and LZW-RNS Scheme for $\{2^n - 1, 2^n, 2^n + 1\}$

Documents(kb)/ Operations(s)	**Original File (kb)**	**LZW Compression Algorithm**			**Proposed LZW-RNS(3-Moduli Set) Algorithm.**		
		Encoder Speed (s)	**Decoder Speed (s)**	**Compressed File (kb)**	**Encoder Speed (s)**	**Decoder Speed (s)**	**Compressed File (kb)**
Document1	23	6.2969	3.5781	17	5.7656	3.125	16
Document2	32	32.1406	8.1406	20	31.0469	7.1719	18
Document3	36	90.5469	20.625	27	89.2188	16.4063	20
Totals	**91**	**42.9948**	**10.781233**	**64**	**42.010433**	**8.9010667**	**54**
Total Execution Time				**53.7760**			**50.9115**

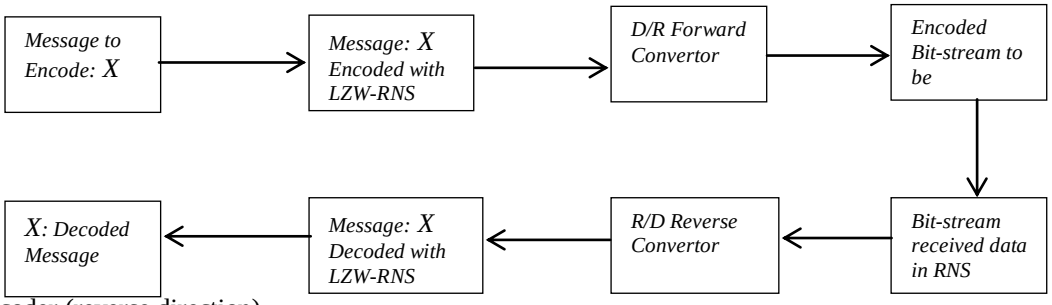

Fig. 6 Encoder (reverse direction)

1.4 Schematic Representation of the Proposed Scheme

The message X (with its ASCII representation) to be encoded is inputted and then encoded with the Proposed LZW-RNS encoding algorithm. The encoded message is converted to RNS (with the forward convertor) and represented in channels. This channel message is then sent in bit streams or

ISSN: 2278 – 1323

residuals in a secrete order. The secrete order bit stream sent message is received in RNS and reorganised into its proper representation. This is then converted to decimal representation using the reverse conversion process and decoded to get back the original message X as in Fig. 6.

IV. CONCLUSION

In this paper, RNS has been applied to the LZW algorithm using the moduli set $\{2^n - 1, 2^n, 2^n + 1\}$ which results in an encoder and decoder pair. The decimal representations of the ASCII codes are converted to its residues in a process known as forward conversion and used in the encoding process. The encoded message is then transmitted in a three bit stream channels in a secrete order.

The decoder pair receives, reorganises the secrete order sent bit stream message and converts it back to the decimal representations through a process known as reverse conversion. The process continues with the proposed LZW-RNS algorithm until the original message is acquired back.

Finally, simulations have been done using MATLAB for some documents to determine the efficiency of the proposed LZW-RNS algorithm which shows better performance of an overall gain of over 15% (15.6250kb) in data size reduction, over 40% (37kb) gain in the original file size, and a gain of 2.8645s in execution time than the original method.

This research has led to the development of a new data encoding and decoding scheme as well as enhanced security and speed of compression and transmission than in [12].

V. FUTURE WORK

Undoubtedly, the proposed system is efficient in terms of security, reduced bits requirement for storage and transmission, and speed of execution of the algorithm. It will however be interesting to investigate the impact of the convertor type on the speed and compression efficiency of the LZW algorithm.

REFERENCES

[1] K. A. Gbolagade, "Effective Reverse Conversion in Residue Number System Processors", PhD thesis, Delft University of Technology (TU-Delft), The Netherlands, Pp 1-187, 2010.

[2] A. Omondi and B. Pumkumar, "Residue Number Systems: Theory and Implementation", Imperial College Press, 57 Shelton street, Covent Garden, London WC2H 9HE, Pp 5-900, 2007.

[3] B. Pahami, "Computer Arithmetic Algorithms and Hardware Design", Department of Electrical and Computer Engineering, University of California, Santa Barabara, Oxford University Press, New York, Pp 2-200, 2000.

[4] J. Amit, "Comparative Study of Dictionary Based Compression Algorithms on Text Data". International Journal of Computer Engineering and Applications, Vol I, Issue II, Pg.1-11, India.

[5] H. Jane, J. Trivedi, "A Survey on Different Compression Techniques Algorithm for Data Compression", International Journal of Advanced Research in Computer Science and Technology, Vol II, Issue III, Pg1-5, 2014.

[6] Mi Lu, "Arithmetic and Logic in Computer System", John Wily and Sons, Inc, Hoboken, New Jersey, 2004.

[7] A. Alhassan, I. Saeed, and P.A. Agbedemnab, "The Huffman's Method of Secured Data Encoding and Error Correction using Residue Number System (RNS)", Communication on Applied Electronics (CAE) Journal, Foundation of Computer Science (FCS), New York, USA, 2015.

[8] S. Alhassan, and K. A Gbolagade, "Enhancement of the Security of a Digital Image using the Moduli set $\{2^n\text{-}1, 2^n, 2^n\text{+}1\}$", International Journal of Advanced Research in Computer Engineering and Technology (IJARCET), Vol. (2), Issue (7), Pp 2223-2229, 2013.

[9] Md. Arif Sobhan, Md. Mamun, A. B. Ahmad Ashrif, and H. Hafizah, "Hardware Approach of Lempel-Ziv-Welch Algorithm for Binary Data Compression", World Applied Sciences Journal, Vol. 22(1), Pp 133-139, 2013.

[10] S. Kaur, and S. Verma, "Design and Implementation of LZW Data Compression Algorithm", International Journal of Information Sciences and Techniques (IJIST), Vol. 2(4), Pp 71-81, 2012.

[11] T. A. Welch, "A Technique for High Performance Data Compression", IEEE, Sperry, Research Centre, Pp 8-19, 1984.

[12] S. Shammugasundaram and R. Lourdusamy, "A Comparative Study of Text Compression Algorithm", International Journal of Wisdom Based Computing, Vol. 1(3), India, Pp 68-76, 2012.

[13] C. Parthasaraty, G. Kalpana, and V. Gnanachandran, "LZW Data Compression for ISP Algorithm", International Journal of Advanced Information Technology, Vol. 2(5), Pp 25-36, 2012.

[14] H. Mahyar, "Reliable and High Speed KASUMI Block Cipher by Residue Number System Code", World Applied Sciences Journal, Vol. 17(9), Pp 1149-1158, 2012.

[15] K. A. Amusa and E.O. Nwoye, "Novel Algorithm for Decoding Redundant Residue Number System (RRNS) Codes", IJRRAS, Vol. (12), Issue (1), Pp 158-163, 2012.

[16] N. Saud, N. Rameez, R. Raja Ali, and S. Faisal, "Optimized RTL Design and Implementation of LZW Algorithm for High Bandwidth Application", Pp 279-285, 2011.

[17] A. Ammar, A. Al-Kabbany, M. Youssef, and A. Emam, "A Secure Image Coding Scheme using Residue Number System, Proceedings of the 18th National Radio Science Conference, (NRCS`01), Egypt, Pp: 339-405, 2001.

[18] V. Cappellini, "Data Compression and Error Control Techniques with Applications", Third Edition, Academic Press, London, ISBN: 0-8194-2427-7, pp: 9-37, 1989.

[19] A. Barati, M. Dehgan, A. Movaghar, and H. Barati, " Improving Fault Tolerance in Ad-Hoc Networks by using Residue Number Systems", Journal of Applied Sciences, Vol 8, Pp 3272-3278, 2008.

A. Alhassan, Department of Computer Science, Faculty of Mathematical Sciences, University for Development Studies, Navrongo, Ghana.

Professor K. A. Gbolagade, Provost, Collage of Information and Communication Technology, Kwara State University, Malate, Ilorin, Kwara State, Nigeria.

Dr. Edem K. Bankas, Department of Computer Science, Faculty of Mathematical Sciences, University for Development Studies, Navrongo, Ghana.

ISSN: 2278 – 1323