# A Novel Approach of Modified Run Length Encoding Scheme for High Speed Data Communication Application

**S. Joseph[1], N. Srikanth[2], J. E. N. Abhilash[3]**

Swarnandra College of Engineering and Technology

**Abstract:** ***This paper presents a Modified Run Length Encoding (RLE) Scheme for High Speed Data Compression. Compression is efficient technique to reduce the memory occupancy and to improve the performance of the system from many available techniques to occupancy of memory. RLE is one in substantial compression algorithm which can reduce memory use. The major improvement in compression rate and limitations and improving the speed of the system performance in RLE scheme, are discussed in this research paper. This is the new and more reliable technique for data compression. In previous traditional RLE algorithms the compression rate is limited. In order to improve compression rate with a compromise between Data recovery and allowable distance a new technique is implemented. Random sequences have been analyzed using proposed scheme, and the compression is observed. In order to store input sequences and output sequences FIFOs are used. Complete system is designed using VHDL language and is implemented on Spartan 3 FPGA.***

**Keywords:** Compression Rate, Data Compression, Data Recovery, Run Length Encoding (RLE), VHDL

## 1. Introduction

Data compression implies sending or storing a smaller number of bits. Data compression is a process that reduces the amount of data in order to reduce data transmitted and decreases transfer time because the size of the data is reduced [1]. Data compression is commonly used in modern database systems. Compression can be utilized for different reasons including:1) Reducing storage/archival costs, which is particularly important for large data warehouses.2) Improving query workload performance by reducing the I/O costs[2]. In database systems, the disk I/O is one of the important factor. Any Data Compression technique in database is for reducing memory space. Compression is best approach to improve the system performance. Compression has been around nearly as long as there has been research in database and has been much worked in this field. [4][5][6].Data compression involves transforming a string of characters in some representation (such as ASCII) into a new string which contains the same information but with smallest possible length [3]. Data compression has important application in the areas of data transmission and data storage. Compressing data reduces storage and communication costs. Similarly, compressing a file to half of its original size is equivalent to doubling the capacity of the storage medium. Data compression is rapidly becoming a standard component of communications hardware and data storage devices. Data compression implies sending or storing a smaller number of bits. Although many methods are used for this purpose, in general these methods can be divided into two broad categories: lossless and lossy methods.

In lossless data compression, the integrity of the data is preserved. The original data and the data after compression and decompression are exactly the same because, in these methods, the compression and decompression algorithms are exact inverses of each other: no part of the data is lost in the process. Redundant data is removed in compression and added during decompression. Lossless compression methods are normally used when we cannot afford to lose any data. Our eyes and ears cannot distinguish subtle changes. In such cases, we can use a lossy data compression method. These methods are cheaper they take less time and space when it comes to sending millions of bits per second for images and video. Several methods have been developed using lossy compression techniques. JPEG (Joint Photographic Experts Group) encoding is used to compress pictures and graphics, MPEG (Moving Picture Experts Group) encoding is used to compress video, and MP3 (MPEG audio layer 3) for audio compression.

## 2. Run Length Encoding

Run-length encoding (RLE) is probably very simplest form of data compression in which runs of data (that is, sequences in which the same data value occurs in many consecutive data elements) are stored as a single data value and count, rather than as the original run[3][4]. This is most useful on data that contains many such runs: for example, simple graphic images such as icons, line drawings, and animations. It is not useful with files that don't have many runs as it could greatly increase the file size. It can be used to compress data made of any combination of symbols [6]. It does not need to know the frequency of occurrence of symbols and can be very efficient if data is represented as 0s and 1s.The general idea behind this method is to replace consecutive repeating occurrences of a symbol by one occurrence of the symbol followed by the number of occurrences.

The method can be even more efficient if the data uses only two symbols (for example 0 and 1) in its bit pattern and one symbol is more frequent than the other RLE may also be used to refer to an early graphics file format supported by CompuServe for compressing black and white images, but was widely supplanted by their later Graphics Interchange Format. RLE also refers to a little-used image format in Windows 3.x, with the extension rle, which is a Run Length Encoded Bitmap, used to compress the Windows 3.x startup

screen. Typical applications of this encoding are when the source information comprises long substrings of the same character or binary digit.

The RLE algorithm performs a lossless compression of input data based on sequences of identical values (runs). It is a historical technique, originally exploited by fax machine and later adopted in image processing. The algorithm is quite easy: each run, instead of being represented explicitly, is translated by the encoding algorithm in a pair (l,v) where l is the length of the run and v is the value of the run elements[3]. The longer the run in the Sequence to be compressed, the better is the compression ratio [3][7]. Run-length encoding (RLE) packs consecutive same values into a (value, length) pair to compress the data. For example, sequence '22, 22, 22, 2, 2' will be encoded into '(22, 3), (2, 2)'. When there exists many runs of the same values, RLE can lead to a very high compression ratio. At the same time, RLE is very light-weighted in terms of both the compression and decompression performance [4][8].

## 3. Modified Run Length Encoding Scheme

As mention in the above, to improve the system performance, increase the compression rate and decrease the memory occupancy of RLE technique. We have proposed some modifications in run length encoding scheme. These modifications are specially designed to counter the for this proposed method. The modified run length encoding scheme gives a major improvement in compression ratio for any kind of data. The above mentioned problem in Run Length Encoding Scheme can be overcome by bright compression. Analyzing the input data is the first and core step. We analyze data to highlight if there are any largest numbers of sequences that may increase the number of bits to represent the length of each run. In proposed method if input data contains nearest value with its adjacent data then both value are considered as same data.

Examples1: RLE Compressor

>> Input data :{ 54 54 54 53 64 64 64 73 73 12 12 12 12 13 13 13}
>>Compressed output :{ 54 3 53 1 64 3 73 2 12 4 13 3}
Example2: Modified method
>>Input data :{ 54 54 54 53 64 64 64 73 73 12 12 12 12 13 13 13}
>>Compressed output :{ 54 4 64 3 73 2 12 7}

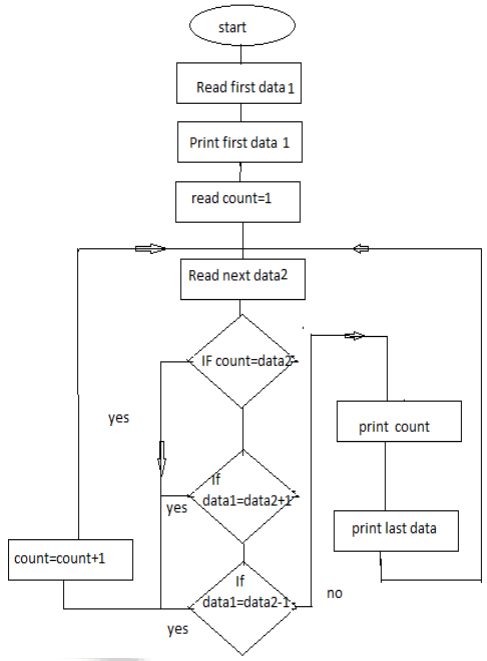


**Figure 1:** Flowchart of modified RLE compressor

Compression algorithm can be explained as follows:

1) Read first data1 from consecutive run length input of data.
2) Print that in next step.
3) Read Count is equal to 1 of consecutive input.
4) Read next data also from the given input.
5) In this stage condition is if data1 value is equal to data 2 then go to step 6.
6) Count increment i.e. count=count+1.
7) In this step need when the statement5 is false.
In this step if input data1=data2+1 then go to step 6, otherwise go step 8). In this step data1=data2-1 then go to step6 in this statement is false go to step 8.
8) In this step print total count.
9) Print last data.

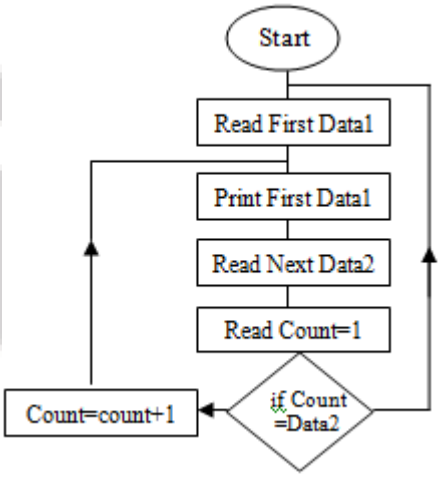


**Figure 2:** Flowchart of Modified RLE de-compressor

From the decompression flow chart,
1. Read first data1.
2. Print data1.
3. Read next data2.
4. Read count=1.

5. If count=data2 then go to step1 otherwise go to count count increment.
6. Count=count+1.
7. Go to step2.

With reference of the above two flow chats of compression and de-compression implemented the proposed method this algorithm is simple to implement the compression & de-compression and reduce the limitations in RLE i.e. smaller sequence of single data, single zero and single ones, double zero and double ones that may result in expansion of data instead of compression [3][4].

## 4. HDL Implementation

For these modifications are specially designed to counter the above mentioned problems. In the implementation of Modified Run Length Encoding Scheme have three blocks (two FIFO memories, compressor) in both compression and decompression implementation in VHDL. HDL description is simulated using Xilinx ISE simulation tools.

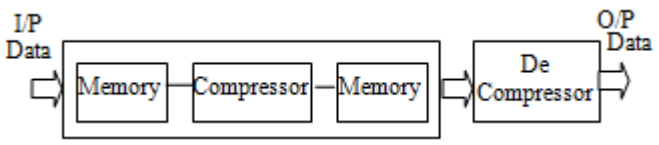


**Figure 3:** Block Diagram of RLE Compression Technique.

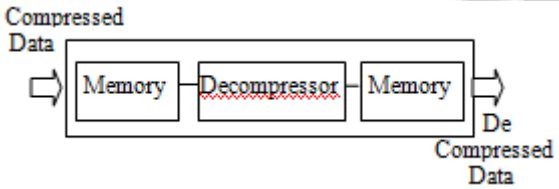


**Figure 4:** Block Diagram of RLE Decompression Technique

From the block diagram of RLE compressor input data consecutive sequences are stored in FIFO Memory. Compressor reads data from memory randomly and performs compression. Compressed output stored in second FIFO memory. In the decompression, de-compressor read data from FIFO memory, then perform decompression and output is stored in output FIFO memory.

In proposed method if input data contains nearest value with its adjacent data then both value are considered as same data. By this implementation in proposed method some information may be loosed, But losing information may not affect the original data. So compression rate is increasing, memory space is decreasing. This technique is more useful in where the redundancy data is high. For example redundancy data has been occurs in video and image compression. Star sports channel receiving time lesser than DD sports receiving time, the difference is 40 seconds. But in DD sports everything has been covered, loss of information is nil. In star sports video channel unnecessary data is removed, so transmitting speed has increased. By using this modified RLE technique efficient compression is possible, this can be applicable in both video and image compression. So this approach is more use full in high redundancy in database or video or image compression

## 5. Simulation Results

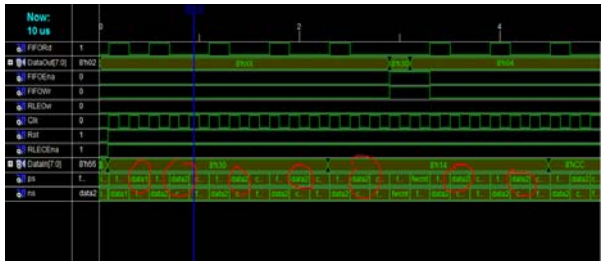

**Figure 5: RLE Compression Scheme**

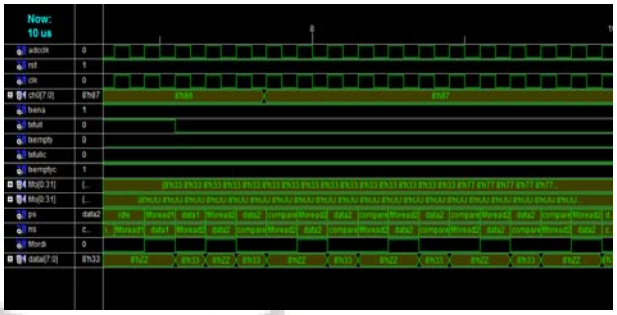

**Figure 6:** Modified Compression Scheme

In the simulation of compressor, at the input side clock signal is applied, reset is 1 applied and RLEC signal is enabled. Compressor read data from FIFO memory and writes it. Input data is applied at corresponding signal. That data is stored in fifo32x8 . Data out signal is compressed output.

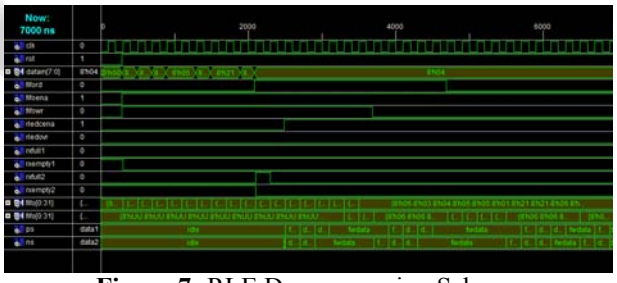

**Figure 7:** RLE Decompression Scheme

In the decompression simulation, compressed data read from FIFO memory, clock signal is applied and reset is 1.FIFORd signal is enabled, and FIFOWr also enabled. RLEDC signal is high decompressed output is stored in fifo32x8 memory

## 6. Conclusion

In This paper provides a novel and more reliable technique for data compression. It solves the limitations present in Run Length Encoding Scheme. In this method compression rate is increased and memory requirement is decreased than Run Length Encoding Scheme .This technique applicable in where the redundancy is more like image compression, data compression and video compression. Simulation results show that the speed of transmission also improved. With this technique the storage space required also reduces. In order to improve the performance of this compression scheme the decompressor should be modified so that the decompressed data is exact replica of the original data that was compressed.

## References

[1] Eug enePamba Capo-Chichi,Herv eGuyennet,Jean-Michel Friedt, …A new Data Compression Algorithm for wireless Sensor Network,…in proc Third International Conference on Sensor Technologies and Applications,2009,pp,1-6 DOI 10,1109/SENSORCOMM 2009.
[2] Stratosldreos, Raghav Kaushik, Vivek Narasayya, Ravi shankar Ramamurthy,…Estimating the Compression Fraction of an Index using sampling,…in Proc International Conference on data Engineering(ICDE),2010.doi. 109/ICDE2010.5447694
[3] Asjad Amin,Haseeb Ahmad Qureshi,Mahammad junaid, Mahammad Yasir Habib,Waqas Anjum,…Modified Run Length Encoding Scheme with Introduction of Bit Stuffing for efficient Data Compression…in Proc 6th Internatational Conference on internet Technology and Secured transactions,11-14 December 2011,IEEE2011
[4] Daniel J, Abadi, Samuel R, Madden, and Miguel C, Ferreira ,..Integrating Compression and execution in column-oriented database Systems…, Proc.SIGMOD 2006 , ACM, 2006, pp.671-682
[5] G. V. Cormack,..Data Compression on Database system,…Commun. ACM, 28(12):1336-1342,1985.
[6] Mingyuan An ,…Column-Based RLE in Row-oriented database…2009 IEEE
[7] Martin H, Weik, …Computer Science and Communication Dictionary… 2000, Volume1,p.129
[8] Allison L, Holloway, and David J, DeWitt….Read-optimized Database, in depth… Proc.VLDB 2008, Morgan Kaufmann,2008,pp 502-51

## Author Profile

**S. Joseph,** perusing his Master of Technology (M.Tech) in VLSI from the Jawaharlal Nehru Technological University. His research interests are VLSI, Embedded systems and Communications.

**N. Srikanth,** working as Associate Professor in Swarnandhra College of Engineering and Technology, Narsapur, India. He has seven years of teaching experience. He has guided many undergraduate and post graduate students. His research interests are VLSI, Embedded Systems and Image Processing.

**J. E. N. Abhilash,** working as Associate Professor in Swarnandhra College of Engineering and Technology, Narsapur, India. He has seven years of teaching experience. He has guided many undergraduate and post graduate students. His research interests are VLSI, Embedded Systems and Image Processing.