International Journal of Electronics Signals and Systems

Volume 4 | Issue 2 Article 6

October 2014

A LOW POWER NEW DATA COMPRESSION ALGORITHM FOR WIRE/WIRELESS SENSOR NETWORKS USING K-RLE

V. KRISHNAN
Dept. of ECE, SIR C.R. Reddy college of Engineering, v_krishnan@yahoo.com

R. TRINADH
Dept. of ECE, SIR C.R. Reddy college of Engineering, r.trinadh12@gmail.com

A LOW POWER NEW DATA COMPRESSION ALGORITHM FOR WIRE/WIRELESS SENSOR NETWORKS USING K-RLE

V.KRISHNAN[1], MR. R.TRINADH[2]

[1]Student, M. Tech Student, [2]M.Tech, Assistant Professor, Dept. of ECE, SIR C.R. Reddy college of Engineering

Abstract- In this project, we propose and evaluate a new data compression algorithm inspired from Run Length Encoding called K-RLE which means RLE with a K-Precision. This increases the ratio compression compared to RLE. In order to improve the compression results with different statistics of data sources. Here we want to introduce in-network processing technique in order to save energy. In-network processing techniques allow the reduction of the amount of data to be transmitted. The well known in-network processing technique is data compression and/or data aggregation. Data compression is a process that reduces the amount of data in order to reduce data transmitted and/or decreases transfer time because the size of the data is reduced.

Keywords- Wsn,Rle,K-rle.....

1. INTRODUTION

The most common technique for saving energy is the use of sleep mode where significant parts of the sensor's transceiver are switched off. As described in, in most cases, the radio transceiver on board sensor nodes is the main cause of energy consumption. Hence, it is important to keep the transceiver most of the time in switched off mode to save energy. Nevertheless, using the sleep mode reduces data transmission/reception rate and thus communication in the network. The question is how to keep the same data rate sent to the base station by reducing the number of transmission.

This article introduced in-network processing technique in order to save energy. In-network processing techniques allow the reduction of the amount of data to be transmitted. The well known in-network processing technique is data compression and/or data aggregation. Data compression is a process that reduces the amount of data in order to reduce data to be transmitted and/or decreases transfer time because the size of the data is reduced. However, the limited resources of sensor nodes such as processor abilities or RAM have resulted in the adaption of existing compression algorithm to WSN's constraint. Two main kinds of compression algorithms are available: lossless and lossy. Algorithm for WSN is S-LZW. Nevertheless, S-LZW which is an adaption for WSN of the popular LZW. Data compression algorithm is a dictionary-based algorithm.Compression algorithms based on dictionary require extensive use of RAM: such algorithms cannot be applied to most sensor platform configurations due to limited RAM. We hence introduce a generic data compression algorithm usable by several sensor platforms. In this article, we study the adaptation of a basic compression algorithm called Run Length Encoding (RLE). The best known is lossless compression.

1.1 Over view of k-rle data compression technique: A new data compression technique that is the K-RLE. In general the run length encoding algorithm is data compression algorithm and basic idea is, if a data item 'd' occurs 'n' consecutive times in the input stream, we replace the n occurrences with the single pair 'nd'. In my proposed system, K-RLE algorithm is very efficient compression technique which decreases the ratio of compression when compared to traditional RLE compression technique. Where as in the proposed algorithm let 'K' be a number, If a data item 'd' or data between d+K and d-K occur 'n' consecutive times in the input stream, we replace the n occurrences with the single pair 'nd'.

1.2 CODING:

Coding is the job of taking probabilities for messages and generating bit strings based on these probabilities e.g:- each character or word in a text. To be consistent with the terminology in the previous section, which consider each of these components a message on its own and use the term message sequence for the larger message made up of these components. In general each little message can be of a different type and come from its own probability distribution

1.3 SHANNON THEORY OF INFORMATION CODING:

Shannon also developed the theory of lossy data compression. This is better known as rate-distortion theory. In lossy data compression, the decompressed data does not have to be exactly the same as the original data. Instead, some amount of distortion, D, is tolerated. Shannon showed that, for a given source (with all its statistical properties known) and a given distortion measure, there is a function, R(D), called the rate-distortion function. The theory says that if D is the tolerable amount of distortion, then R(D) is the best possible compression rate. When the compression is lossless (i.e., no distortion or D=0), the best possible compression rate is R(0)=H (for a

finite alphabet source).In this sense, rate-distortion theory is a generalization of lossless data compression theory, where it starts from no distortion (D=0) to some distortion (D>0).

1.4 HUFFMAN CODES:

The algorithm is now probably the most prevalently used component of compression algorithms, used as the back end of GZIP, JPEG and many other utilities.

The Huffman algorithm is very simple and is most easily described in terms of how it generates the prefix-code tree. Even though Huffman codes are optimal relative to other pre fix codes, prefix codes can be quite inefficient relative to the entropy. In particular H (S) could be much less than 1 and so the extra 1 in H(S) +1 could be very significant. The entropy lower bound is 2.88 bits/character

An example of how this is done is shown below

I. It does not matter how the characters are arranged. After arranging it as above, the final code tree looks nice and neat.
II. Label the final code tree with upper branches 0's and the lower branches 1's.
III. There may be cases where there is a tie for the two least probable characters. In such Huffman codes are not unique.
IV. Huffman codes are optimal in the sense that no other lossless fixed-to-variable length code has a lower average rate.
V. The rate of the above code is 2.94 bits/character.
VI. cases, any tie-breaking procedure is acceptable.

2. ALGORITHM LEMPEL-ZIV

The Lempel-Ziv algorithm is a variable-to-fixed length code. Basically, there are two versions of the algorithm presented in the literature: the theoretical version and the practical version.

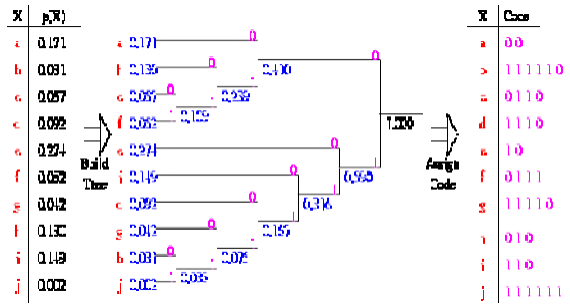


Figure 2 The final static code tree of Huffman coding

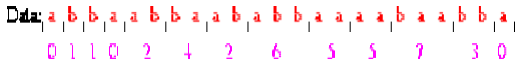
Theoretically, both versions perform essentially the same. However, the proof of the asymptotic optimality of the theoretical version is easier. In practice, the practical version is easier to implement and is slightly more efficient. The basic idea is to parse the input sequence into non-overlapping blocks of different lengths while constructing a dictionary of blocks seen thus far.

2.1 ARITHMETIC CODING:

Arithmetic coding is a technique for coding that allows the information from the messages in a message sequence to be combined to share the same bits.

Encoding Algorithm

Initialize the dictionary to contain all blocks of length one (D= {a,b}). Search for the longest block W which has appeared in the dictionary. Encode W by its index in the dictionary. Add W followed by the first symbol of the next block to the dictionary, Go to Step 2. The following example illustrates how the encoding is performed



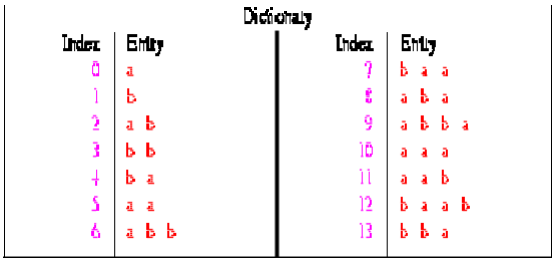


Table 2.1 Lempel-Ziv algorithm

Theoretically, the size of the dictionary can grow infinitely large. In practice, the dictionary size is limited. Once the limit is reached, no more entries are added. Welch had recommended a dictionary of size 4096. This corresponds to 12 bits per index. The length of the index may vary.Many popular programs (e.g. UNIX compress and uncompress, gzip and gunzip, and Windows WinZip) are based on the Lempel-Zivalgorithm.

3. DATA COMPRESSION ALGORITHMS:

Compression is useful because it helps reduce the consumption of expensive resources, such as hard disk space or transmission bandwidth on the downside, compressed data must be decompressed to be used, and this extra processing may be detrimental to some applications.

Lossy Compression (vs) Lossless Compression

Lossy image compression is used in digital cameras, to increase storage capacities with minimal degradation of picture quality. Similarly, DVDs use the lossy MPEG-2 Video codec for video compression.

3.1 Lossless Compression:

The Lempel-Ziv (LZ) compression methods are among the most popular algorithms for lossless storage. DEFLATE is a variation on LZ which is optimized for decompression speed and compression ratio, therefore compression can be slow. DEFLATE is used in PKZIP, gzip and PNG. LZW (Lempel-Ziv-Welch) is used in GIF images. The very best compressors use probabilistic models, in which predictions are coupled to an algorithm called arithmetic coding. Arithmetic coding, invented by Jorma Rissanen, and turned into a practical method by Witten, Neal, and Cleary, achieves superior compression to the better-known Huffman algorithm, and lends itself especially well to adaptive data compression tasks where the predictions are strongly context-dependent. Arithmetic coding is used in the bi-level image-compression standard JBIG, and the document-compression standard DjVu. The text entry system, Dasher, is an inverse-arithmetic-coder.

3.2Run-Length Encoding:

Run-Length Encoding (RLE) is a very simple 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.

For example, consider a screen containing plain black text on a solid white background. There will be many long runs of white pixels in the blank space, and many short runs of black pixels within the text. Let us take a hypothetical single scan line, with B representing a black pixel and W representing white:
WWWWWWWWWWWWBWWWWWWWWWW
WWBBBWWWWWWWWWWWWWWWWWW
WWWWWWBWWWWWWWWWWWWWW

Apply the run-length encoding (RLE) data compression algorithm to the above hypothetical scan line, the encoded text is as follows:
12W1B12W3B24W1B14WInterpret this as twelve W's, one B, twelve W's, three B's, etc.

The run-length code represents the original 67 characters in only 18. Of course, the actual format used for the storage of images is generally binary rather than ASCII characters like this, but the principle remains the same. Run-Length Encoding (RLE) is a basic compression algorithm. As described on, the simple idea behind this algorithm is this: If a data item, d occurs ‘n’ consecutive times in the input stream, then replace the n occurrences with the single pair ‘nd’. The graphical representation of the RLE algorithm applied on temperature readings. However, because RLE is based on the same consecutive input stream, its results depend on the data source. In this way, in order to perform RLE results with different data sources statistics, introducing a new compression algorithm which is inspired from RLE named K-RLE which means RLE with a K-Precision.

4. IMPLEMENTATION OF K-RLE ALGORITHM

Run-Length Encoding (RLE) is a basic compression algorithm. Data item d occurs ‘n’ consecutive times in the input stream, we replace the n occurrences with the single pair nd. K-RLE-The idea behind this new algorithm is this let K be a number, If a data item ‘d’ or data between d+K and d-K occur n consecutive times in the input stream, replace the ‘n’ occurrences with the single pair ‘nd’.We introduce a parameter K which is a precision. K is defined as:δ= σ/ KWith a minimum estimate of the Allan standard deviation i.e. σ is a representative of the instrument measurement noise below which the precision is no longer significant. If K = 0, K-RLE is RLE. K has the same unit as the dataset values, in thiscasedegree. However, the change on RLE using the K-precision introduces data modified. Indeed, while RLE is a lossless compression algorithm K-RLE is a lossy compression algorithm.

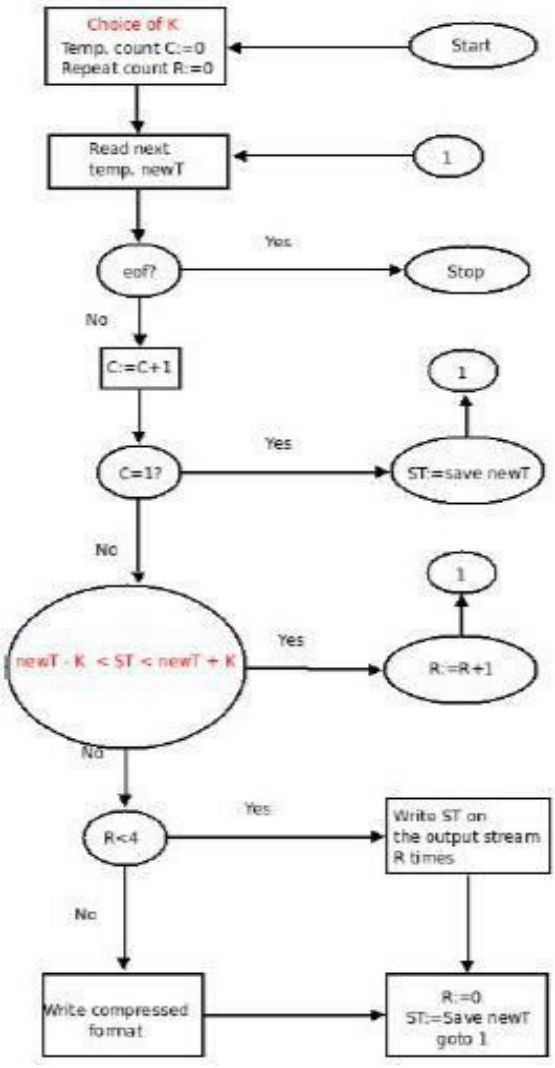


Figure 4 K-RLE compression algorithms

4.1 K-RLE COMPRESSION &DECOMPRESSION BLOCKS

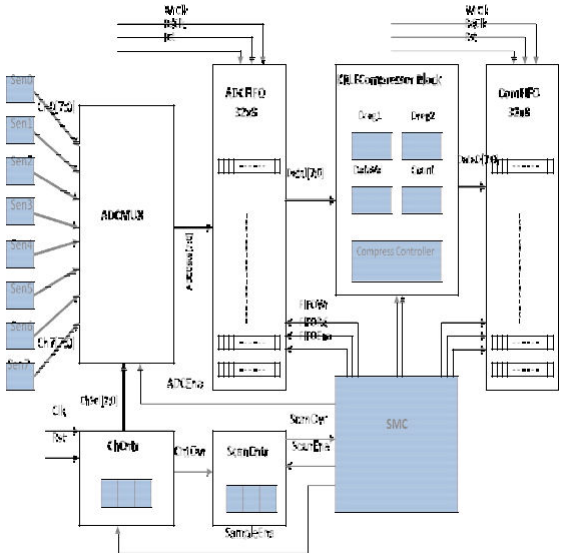


Figure 4.1:-Bloc Diagram for K-RLE Compressor

The idea behind this new algorithm is this: let K be a number, If a data item 'd' or data between d+K and d-K occur n consecutive times in the input stream, replace the 'n' occurrences with the single pair 'nd'. Introduce a parameter K which is a precision. K is defined as:

$\Delta=k/\sigma$ with σ a minimum estimate of the Allan standard deviation [10]; i.e. σ is a representative of the instrument measurement noise below which the precision is no longer significant.If K = 0, K-RLE is RLE. K has the same unit as the dataset values, in this case degree.

However, the change on RLE using the K-precision introduces data modified. Indeed, while RLE is a lossless compression algorithm K-RLE is a lossy compression algorithm. This algorithm is lossless at the user level because it chooses K considering that there is no difference between the data item d, d+K or d-K according to the application.

Above is the block diagram showing hardware requirement to achieve this type of compression. Multiple numbers of sensors used fortaking the raw data. Sensors giving data through different channels may be selected using Channel counter. Channel counter controls the data input from the sensors. After receiving required data bytes from the sensors, Channel counter sets a status signal "CntOvr". After receiving CntOvr signal from channel counter, SMC controller activates the K-RLE compressor. Data received from the sensors is captured in FIFO using ADC mux. After capturing data from the sensors SMC controller activates K-RLE compressors do the compression using K values supplied by K-input here in this K-RLE compressor have provided the compression technique with K-value 0 or 1 or 2. In order to store compressed data we have used another FIFO.Decompress or is a hardware block for recovering original data. Here in decompression logic there are three logic blocks one input FIFO, one decompress or and another output FIFO. Input FIFO for holding compressed data. Decompress or block have to select the K-value, after recovering original data that must be loaded in output FIFO. Below is the description regarding each block of K-RLE compressor and K-RLE decompress or.

4.2 Channel Counter: In this project 32 sensors are used which may be selected using a channel counter and ADC multiplexer. Channel counter is triggered by clock signal in order to select channels one by one we have used five bit binary counter logic Cntovr is the signal which is active when capturing data from the sensors is completed or when the FIFO is full.

4.3 FIFO:

FIFO is a logic block which is used twice in compression logic. One for capturing input data. Another is for storing compressed data. Data first entered into the FIFO is the data which is retrieved from the FIFO. Input FIFO and output FIFO enabled, FIFO disable, FIFO Read and FIFO Write are the operations by SMC controller

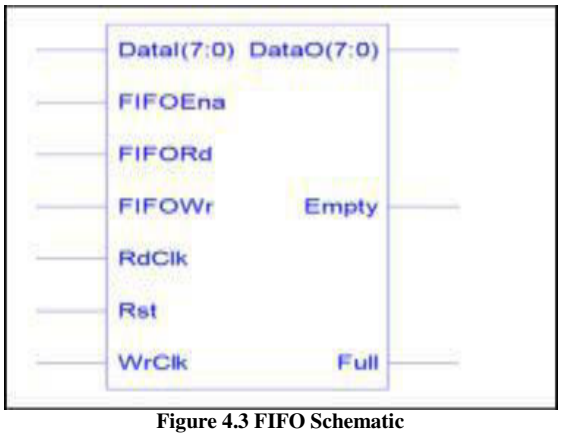


Figure 4.3 FIFO Schematic

4.4 TRANSMITTER STATE MACHINES:

Figure 4.4 Transmitter State Machine

Channel counter use to receive data from the sensors after completion of data receiving from the sensor channel counter sets “cntovr” signal. This is the input signal for Tx state machine (chcntovr). Upon activation of the signal Tx state machine enables FIFO using “FIFO Ena” signal and to the write operation uses “FIFO WR”. Then input FIFO is an enable and data received from the sensors is loaded in input FIFO. After loading the entire data, RLE compressor do the compression of FIFO data. RLE compressor sends the data signal, “RLEOVR” used to tell the Tx State machine that the compression process is completed. After completion of compression process Tx state machine enables output FIFO and do the write operation of compressed data on output FIFO.

4.5 K-RLE COMPRESSOR:

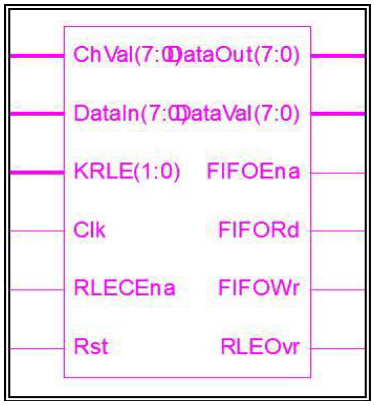


Figure 4.5 K-RLE Compressor

Tx state machine do the control of all the logic blocks such ADC mux input FIFO, output FIFO and K-RLE compressor upon completion of data loading into input waveform. Tx state machine enables the K-RLE using ”RLEC En” signal. By selecting K value using K-RLE signal, K-RLEcompressor do the compression for K value 0 or1 or 2. After completion compressor process K-RLE compressor sends the status using the signal ‘RLEOvr”. Compressed data is loaded in output FIFO this operation si carried out by Tx state machine.

4.6 K-RLE Decompressor:

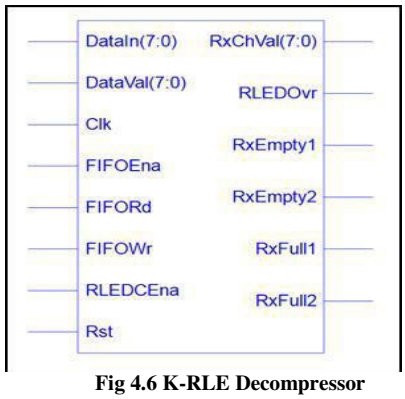


Fig 4.6 K-RLE Decompressor

By receiving “RLE DCEna”, which is enable signal for K-RLE decompressor, K-RLE decompressor start the de compression process by reading data from the input FIFO. The decompressed data is loaded in output FIFO by enabling OUTPUT FIFO using “FIFOEna” signal and do the write operation into output FIFO using “FIFOWR”. The decompressed data can observe in output FIFO.

5. SIMULATION RESULTS

Below shown the schematic diagram for k-RLE compressor here in this the received data from the 32 channels such as 08, 08, 08, ..., 02 is loaded in input FIFO in order to store large amount of samples we need a lot of memory. If we use compression process the compressed data can be stored I less amount of memory. In this technique we have choose K-RLE algorithm to compress samples for 32 sensors. Here we have to chosen K a value 2 then the range of data (-d+2) to (d+2) is considered as “d”. in this example 08, 09, 07 all are considered as of 08. Then 08 counts 1C times in the present example. 0A and 0B both are considered as 0B then 0B comes three times than 02 times only one time. Samples taken from 32 sensors are located in output FIFO.

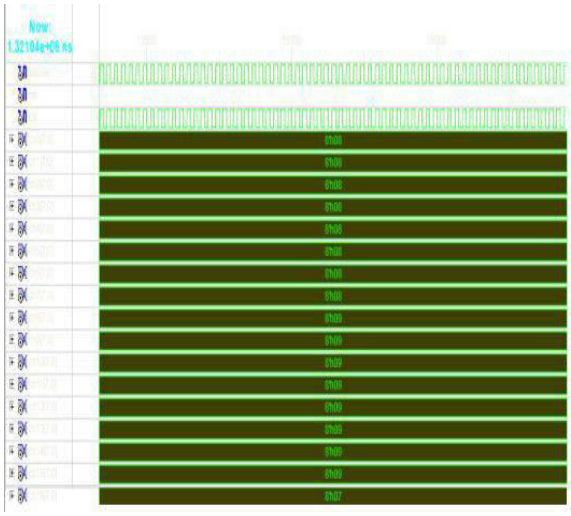


Fig5A: Simulation Waveform for K-RLE Compressor

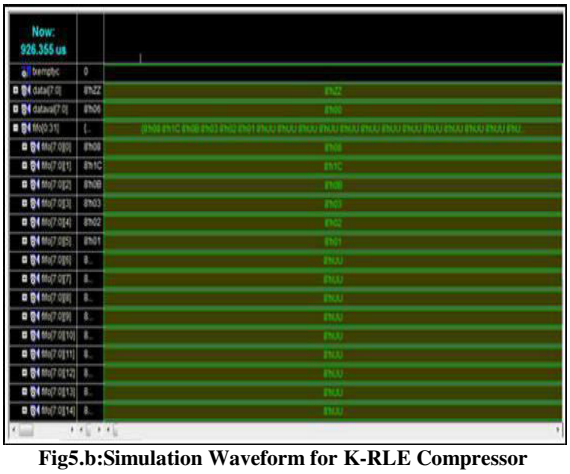
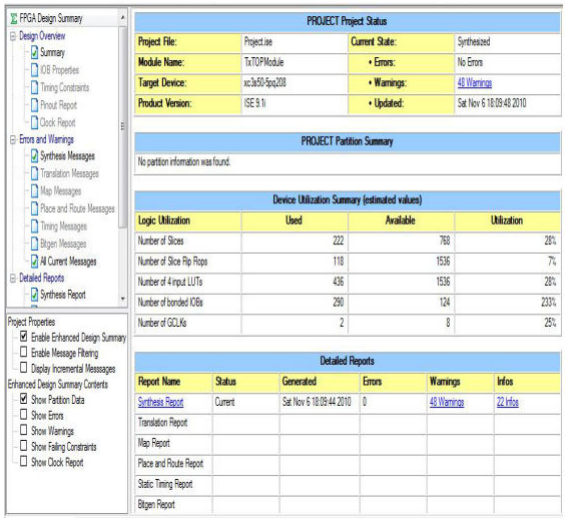


Fig5.b:Simulation Waveform for K-RLE Compressor

5.2 KRLE COMPRESSION SYNTHESIS REPORT:



Final Report *

5.1 K-RLE DECOMPRESSOR:

Below is the schematic diagram for K-RLE decompressor in this diagram we can observe three logic blocks

1. Input FIFO
2. K-RLE decompressor
3. Output FIFO

Compressor data is loaded in input FIFO after loading all the data bytes in the input FIFO K-RLE d compressor is enabling using "RLEDCEna". The decompress or de compresses input FIFO data using reverse of the K-RLE algorithm and the decompressed data is loaded in output FIFO here in this example we have taken compressed data 08,1C; 0B,03; 02,01. This compressed data is decompressed by the "K-RLE DC" in the output FIFO we can observe decompressed data 08 in 1C times 0B in 03 times 02 in 01 time. By observing input FIFO and output FIFO decompression is possible by the K-RLE decompression algorithm

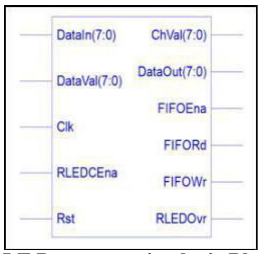


Fig 5.1: K-RLE Decompression logic Block Diagram

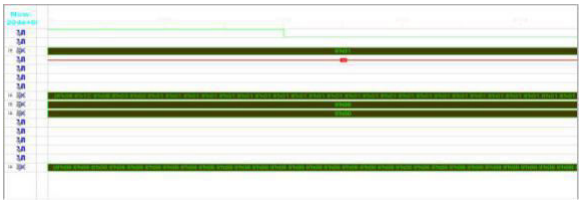


Fig 5.1.A Simulation Results for K-RLE Decompression

Fig 5.1.B: Simulation Results for K-RLE Decompression

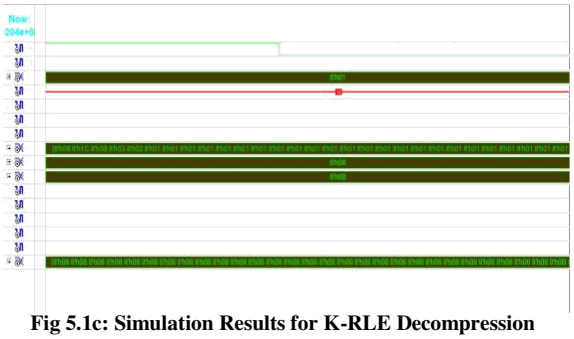


Fig 5.1c: Simulation Results for K-RLE Decompression

5.4 K-RLE DECOMPRESSOR SYNTHESIS REPORT

Comparison of Compression ratio:

RLE	65.625%
K-RLE 0	65.625%
K-RLE 1	84.375%
K-RLE 2	84.375%
K-RLE 3	90.625%

CONCLUSION

The project presented a new compression algorithm for data compression. This data compression algorithm is a low power Compression algorithm. K-RLE is a lossy compression algorithm. It is lossless at the user level, because it chooses K considering that there is no difference between the data item d, d+k, d-k . This algorithm inspired from RLE named K-RLE which increases the compression ratio compared to RLE. For K=2 this algorithm increases the ratio by 40% compared to RLE. With this approach a fast transmission of data with minimum hardware requirement and less power consumption is possible.

FUTURE SCOPE

This project implemented an efficient compression process. Since RLE doesn't have great compression ratio K-RLE uses less energy and high compression efficiency compared to RLE. Future work of this project is modifying algorithm for lossless compression with equal compression efficiency and low power consumption

REFERENCES:

[1] Yang-Kiefer algorithms for data compression En-hui ,Yang Da-ke He.

[2] A Generalized Method for Encoding and Decoding Run-Length-Limited Binary Sequences G. F. M. BEENKER AND K. A. SCHOUHAMER IMMINK.

[3] N. Kimura and S. Latifi. A survey on data compression in wireless sensor networks. In Information Technology: Coding and Computing.

[4] F. Marcelloni and M. Vecchio. A simple algorithm for data compression in wireless sensor networks. Communications Letters, IEEE, 12(6):411–413, June 2008.

[5] Croce, Silvio, Marcelloni, Francesco, Vecchio, and Massimo. Reducing power consumption in wireless sensor networks using a novel approach to data aggregation. Computer Journal, 51(2):227–239, March 2008.

[6] B. Krishnamachari, D. Estrin, and S. B.Wicker. The impact of data aggregation in wireless sensor network. In ICDCSW '02: Proceedings of the 22nd International Conference on Distributed Computing Systems, IEEE Computer Society. p.p575–578, may 2002.

[7] C. M. Sadler and M. Martonosi. Data compression algorithms for energy-constrained devices in delay tolerant networks. In SenSys, p.p265–278, march 2006.

[8] http://www.consecolo.org/vol1/iss1/art4.

[9] http://www.hn.is.uec.ac.jp/~arimura/compression_links.html.

R. Trinadh received the M.Tech. Degree in embedded System from the gudlavalleru engineering college affiliated to JNTU Kakinada in 2011.He is currently working as an Assistant Professor with the Department of Electronics & communication Engineering, SIR.C.R.Reddy college of Engineering, Eluru .His research interests in embedded real time operating system in relation with power consume and defilation .

V. Krishnan received the B.Tech Degree in Electronics and Communication Engineering from JNTU KKD in 2009.Pursuing M.Tech in SIR CRR COLLEGE ELURU.