International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-2 Issue-6, August 2013

CCSDS Lossless Data Compression Algorithm in FPGA for Space Applications

Rakhee Sasi

Abstract*—Lossless data compression has been suggested for many space science exploration mission applications either to increase the science return or to reduce the requirement for on-board memory, station contact time, and data archival volume. This paper presents a study and implementation of a lossless data compression system, based on the extended_Rice or e_Rice algorithm, as recommended by the Consultative Committee for Space Data Systems (CCSDS), which is implemented on FPGAs (Field Programmable Gate Arrays). A major feature of the e_rice algorithm is that it requires no look- up tables. A simple modification is suggested for e_Rice data compression system which improves its compression performance and thus mainly focus on the reduction of memory and data archival volume. Also the performance parameters of modified e_Rice is compared with Huffman algorithm. The FPGA implementation consists of (a)the received flight mission data decompressed and retrieve the original samples, (b)then original samples are encoded and compared with the received data.

Index Terms- CCSDS, e_Rice algorithm, FPGA, Huffman algorithm, Loseless Data Compression.

I. INTRODUCTION

With the development of high-resolution satellite in earth observing technology, more and more data are transported from space to ground. In some cases, the space-to ground communication link cannot afford so much data. Then, it's necessary to make efficient compression on the raw satellite data, which can reduce the bandwidths taken up by transmission, save the storage and shorten the transmission time. Based on this demand, the CCSDS recommended the standard of lossless data compression. Because the CCSDS recommendation is easy for implementation, more and more space systems begin to adapt this recommendation. A set of requirements were first formulated for selecting a lossless compression algorithm: (a) The algorithm has to adapt to the changes in data statistics to maximize compression performance, (b) The algorithm can be easily implemented with few processing steps in real time with small memory and little power usage, (c) The algorithm can be interfaced with a packet data system such that each packet can be independently decoded without requiring information from other packets. A Lossless compression technique guarantees full reconstruction of the original data without incurring any distortion in the process. It may be necessary to rearrange the data into appropriate sequence before applying the data compression algorithm, inorder to improve the compression ratio. The e_Rice algorithm exploits a set of variable-length codes to achieve compression.

Manuscript published on 30 August 2013.
* Correspondence Author (s)
Rakhee Sasi*, M-Tech Student, Dept. of ECE, Mangalam College of Engineering, Kottayam, India.

After compression has been performed, the resulting variable-length data structure is then packetized and transported through a space-to-ground communication link from the source on-board a space vehicle to a data sink on the ground using a packet data structure.

Further study on the algorithm brought out a parallel architecture compared to the original rice architecture. An extension to low-entropy data was also devised and incorporated in the original Rice architecture. The algorithm is then referred to as the extended_Rice, or e_Rice algorithm. In this work, a simple modification is suggested for e_Rice which improves its compression performance. So main focus of this paper is on reduction of onboard memory, data archival volume etc. Various test data samples are taken for e_Rice including the real time data samples such as voltage, acceleration, pressure etc. Simulations are done in ModelSim SE 6.3f and synthesis performed using Xilinx ISE 8.1i Suite. Based on these results, a comparison with Huffman algorithm made. The FPGA implementation consists of: (a)received flight mission data decompressed and retrieve original data samples, (b)then original samples encoded and compared with the received data.

II. THE E_RICE ALGORITHM ARCHITECTURE

The e_Rice algorithm exploits a set of variable-length codes to achieve compression. Each code is nearly optimal for a particular geometrically distributed source. Variable-length codes, such as Huffman codes and the codes used by the Rice algorithm, compress data by assigning shorter code words to symbols that are expected to occur with higher frequency.

A block diagram of the e_Rice algorithm encoder is shown in Figure 1. It consists of two functional parts: the preprocessor and the adaptive entropy coder. The input data to the preprocessor, *x*, is a *J-sample* block of *n-bit* samples:

$$x = x_1, x_2, \ldots x_J.$$

The preprocessor transforms input data into blocks of preprocessed samples, δ, where:

$$d = \delta_1, \delta_2, \ldots \delta_J.$$

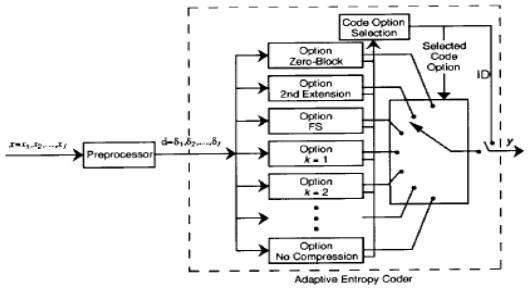


Fig.1. The e_Rice Encoder Architecture

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)

The Adaptive Encoder converts preprocessed samples, δ, into an encoded bit sequence *y*.
The entropy coding module is a collection of variable-length codes operating in parallel on blocks of *J* preprocessed samples. The coding option achieving the highest compression is selected for transmission, along with an ID bit pattern used to identify the option to the decoder. Because a new compression option can be selected for each block, the Rice algorithm can adapt to changing source statistics.The CCSDS Recommendation specifies that the parameter *J* be either 8 or 16 samples per block.

A. Preprocessor

The role of the preprocessor is to transform the data into samples that can be more efficiently compressed by the entropy encoder. To ensure that compression is "lossless", the preprocessor must be reversible. The preprocessing is done by a predictor followed by a prediction error mapper as shown in the Figure 2.

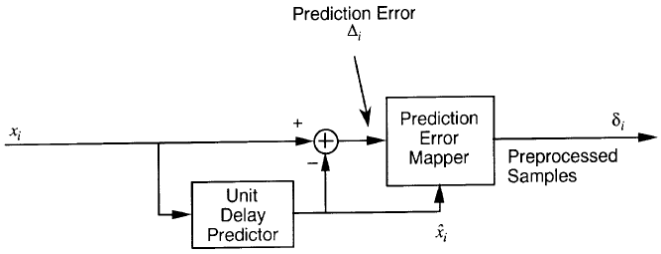


Fig.2. Preprocessor using a Unit- Delay Predictor

1) Predictor

The decorrelation function of the preprocessor can be implemented by a judicious choice of the predictor for an expected set of data samples. One of the simplest predictive coders is a linear first-order unit-delay predictor shown in Figure 2. The output, *Δi*, will be the difference between the input data sample and the preceding data sample. The reference sample is needed to perform the first prediction which is an unaltered data sample upon which successive predictions are based. When required, the reference must be the first sample of a block of *J* input data samples

2) Prediction Error Mapper

Based on the predicted value, *xi^*, the prediction error mapper converts each prediction error value, *Δi*, to an *n-bit* nonnegative integer, *δi*, suitable for processing by the entropy coder.

Sample Value x_i	Predictor Value $\hat{x}_i$	Δ_i	θ_i	δ_i
101	—	—	—	—
101	101	0	101	0
100	101	−1	101	1
101	100	1	100	2
99	101	−2	101	3
101	99	2	99	4
223	101	122	101	223
100	223	−123	32	155

Table 1. Prediction Error Mapper Operation

Consequently, CCSDS recommend the prediction error mapping function as:

$$\delta_i = \begin{cases} 2\Delta_i & 0 \le \Delta_i \le \theta \\ 2|\Delta_i| - 1 & -\theta \le \Delta_i \le 0 \\ \theta + |\Delta_i| & \text{otherwise} \end{cases}$$

where $\theta = \text{minimum}\,(\hat{x}_i - x_{min}, x_{max} - \hat{x}_i)$.

Δi is the prediction error produced by the predictor, *xmin* and *xmax* denote the minimum and maximum values of any input sample *xi.*

B. The Adaptive Entropy Coder

1) Fundamental Sequence Encoding

For the FS code we recognize that each ' 1' digit signals the end of a code word, and the number of preceding zeros identifies which symbol was transmitted. This simple encoding procedure allows FS codewords to be decoded without the use of lookup tables. Table 2 illustrates the FS *codewords* for preprocessed sample values with *n-bit* dynamic range.

2) The Split- Sample Option

Most of the options in the entropy coder are called 'split-sample options'. *The k*th split sample option takes a block of J preprocessed data samples, splits off the *k* least significant bits (LSB) from each sample and encodes the remaining higher order bits with a simple FS codeword before appending the split bits to the encoded FS data stream.

Preprocessed Sample Values, δ_i	FS Codeword
0	1
1	01
2	001
.	.
.	.
.	.
2^n-1	0000 . . . 00001 (2^n-1 zeros)

Table 2. Fundamental Sequence Codewords as a Function Of the Preprocessed Samples.

3) Low Entropy Options

The two code options: the Second-Extension option and the Zero-Block option.

(i) The Second- extension Option

The Second-Extension option is selected, each pair of preprocessed samples in a *J-sample* block is transformed and encoded using an FS codeword. The adjacent sample pairs from a *J* sample preprocessed data block are transformed into a single new symbol y by the following equation:

$$\gamma = (\delta_i + \delta_{i+1})\,(\delta_i + \delta_{i+1} + 1)/2 + \delta_{i+1}$$

The *J/2* transformed symbols in a block are encoded using the FS codeword of Table 2.

(ii) The Zero-Block Option

The ideal of Zero-Block option was first suggested by J. Venbrux at the University of New Mexico's Microelectronics Research Lab. to deal with long runs of "O's". It is selected when one or more blocks of preprocessed samples are all zeros.

4) No Compression

The last option is not to apply any data compression. If it is the selected option, the preprocessed block of samples receives an attached identification field but is otherwise unaltered.

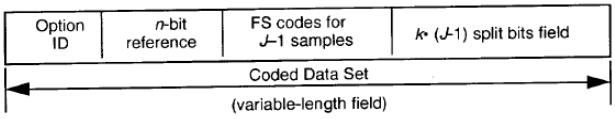
Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)

5) Code Selection

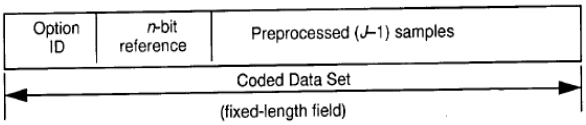
The Adaptive Entropy Coder includes a code selection function, which selects the coding option that performs best on the current block of samples.

C) Coded Data Format

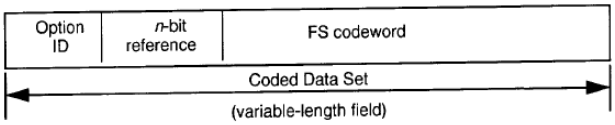
A coded data set (CDS) contains these coded bits from one block. The format of CDS under different conditions are given in Figure 3 (a),(b),(c),(d) for the case when a reference sample is used for encoding.



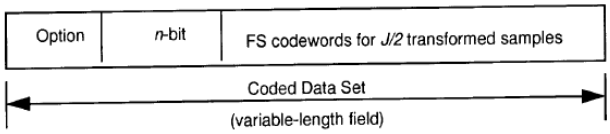
(a) CDS format with sample- splitting and reference



(b) CDS format when no- compression option is Selected



(c) CDS format when Zero- Block option is Selected



(d) CDS format when 2-nd extension option is Selected

Figure 3. Format for Coded Data Set

E) Decoding

The Lossless decoder consists of two separate functional parts, the post processor and the adaptive entropy decoder, as shown in Figure 4. The postprocessor performs both the inverse prediction operation and the inverse of the standard mapper operation.

The first step towards decoding is to set up the configuration parameters so both encoder and decoder operatemission/application specific. Decoding the coded bits is straight forward once the ID bits are interpreted first. Only the second-extension code needs an additional short look up table (LUT) to replace computation. For this option, *J/2* FS codewords will be extracted first. For the Zero-Block option, the single FS codeword following the ID and reference indicates the number of all-zero blocks or the ROS condition.

The postprocessor reverses the mapper function, given the predicted value $\hat{x}_i$.

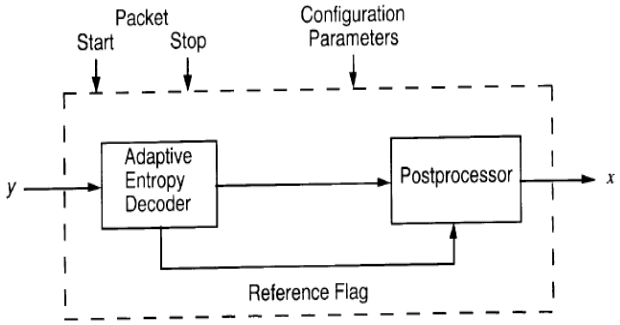


Fig. 4. Decoder Block Diagram

The inverse mapper function can be:

$$\text{if } \delta_i \le 2\theta,$$

$$\Delta_i = \begin{cases} \delta_i/2 & \text{when } \delta_i \text{ is even} \\ -(\delta_i+1)/2 & \text{when } \delta_i \text{ is odd} \end{cases}$$

$$\text{if } \delta_i > 2\theta,$$

$$\Delta_i = \begin{cases} \delta_i - \theta & \text{when } \theta = \hat{x}_i - x_{min} \\ \theta - \delta_i & \text{when } \theta = x_{max} - \hat{x}_i \end{cases}$$

$$\text{where } \theta = \text{minimum } (\hat{x}_i - x_{min}, x_{max} - \hat{x}_i)$$

Finally the postprocessor will rebuild the sample data and save the sample data as reference for the next sample data block.

III. HUFFMAN ALGORITHM

The Huffman algorithm was originally proposed by David A. Huffman in 1952, and published in *A Method for the Construction of Minimum-Redundancy Codes.* It is a lossless data compression algorithm based on variable-length code table, which is established by the estimation of the probability of occurrence for each source symbol. In other words, the higher probability of symbol occurrence, the shorter code word length will be. This makes the length of a whole message shorter.

Huffman coding uses the prefix-free method to choose the representation for each symbol. The Huffman algorithm works by creating a binary tree. The length of the binary tree depends on the number of symbols. The process of creating a binary tree is as follows:

Step 1: List the source symbols in descending order of their probabilities of occurrence.

Step 2: Form a compound symbol with the two symbols. The combination of the two symbols forms a branch in the tree. This step is repeated until all the original symbols have been combined into a single compound symbol.

Step 3: A tree is formed, with the top and bottom stems going from the compound symbol to the symbols which form it, labeled with 0 and 1.

After these three steps, a binary labeled tree is formed. We can get code words for each symbol by reading the labels of the tree stems. Then the Huffman algorithm finishes encoding the source symbols and sends the code table as a file head with the codeword. At the receiver side, the decoder decompresses the codewords into a message according to the code table. Huffman's algorithm is an example of a *greedy* algorithm.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

IV. RELATED WORKS

A.Existing Work

The extended_ rice consists of two low entropy options ie, second- extension and zero block options, which provide more efficient coding than other options when the preprocessed data are highly compressible compared to the original rice architecture.

B. Modified Works

A modification suggested for the improvement of e_Rice data compression performance. In the code option selection process, in the calculation of total bits only $J - 1$ samples are considered for FS Coding and Split- sample options; since in the CDS format reference sample is transmitted as such. The real time negative/ and floating point voltage, acceleration, pressure values etc. also simulated. Simulations and synthesis comparison with Huffman algorithm shows that modified e_Rice adapts to the changes in data statistics because of a number of coding options. The total power, delay and equivalent gate count values also comparatively less for modified e_Rice encoder, obtained by synthesis in Xilinx ISE 8.1i.

V. SIMULATION RESULTS

The Simulations are done in ModelSim SE 6.3f of Xilinx Design Suite.

A. e_Rice Algorithm Outputs

Let the e_Rice Encoder takes for example, a J = 8 – Sample Block of n = 8 bits each i.e., the input sample block x1,x2,…..x(J = 8) is 01100100, 01100110, 01100101, 01101000, 01101011, 01101001, 01101010, 01101100. The encoded output data "enc_data" is shown in Fig. 6. The enc_data appears between the start bit '0' and the stop bit '1'. The rice postprocessor decoded output 'data_decode' is the original input data sample x1,x2,……, xJ.

e_Rice encoder 'x' inputs : 100,102,101,104,107,105,106,108

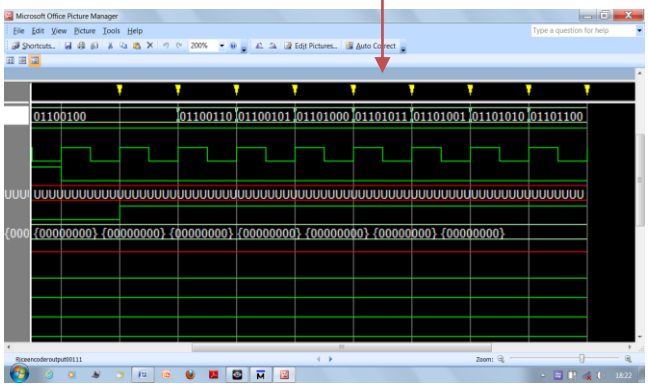


Fig.5. e_Rice Encoder Inputs 'x'

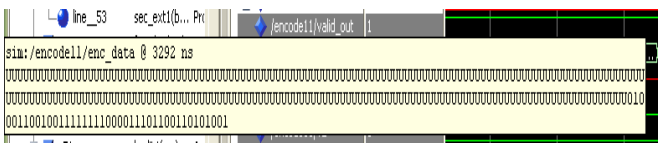


Fig. 6. e_Rice Encoder Output "enc_data"

B. Modified e_Rice Outputs

It takes the same input sample values. The encoded output "enc_data" in Fig. 7 shows that better compression performance achieved compared to e_Rice.

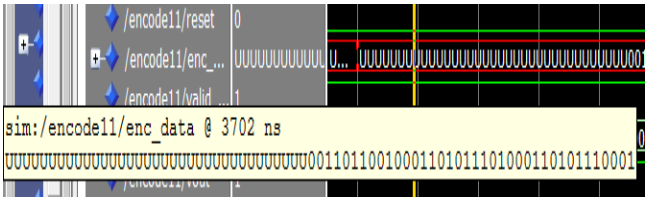


Fig. 7. Modified e_Rice Encoder Output "enc_data"

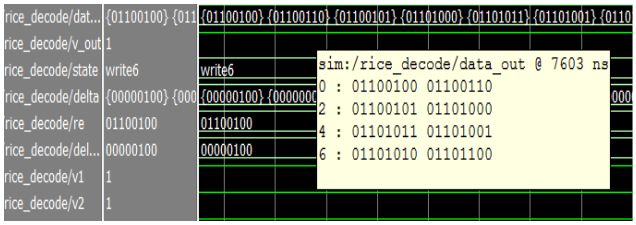


Fig.8. Modified e_Rice Postprocessor Binary decoded output "data_out"

C. Huffman Algorithm Outputs

The Huffman encoder also takes the same data sample input. The encoded data output is shown in Fig. 9The Huffman decoded output will be the original input sample.

Encoder inputs 'data': 100, 102, 101, 104, 107, 105, 106, 108

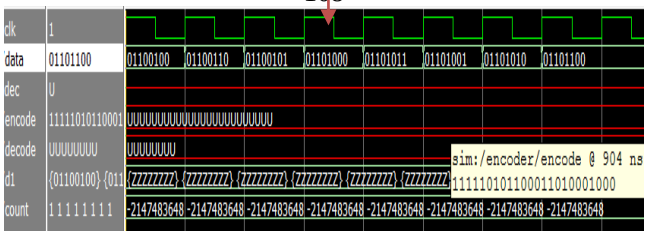


Fig. 9. Huffman Encoder Inputs 'data' and Outputs 'encode'.

Decoder outputs: 100,102,101,104,107,105,106,108.

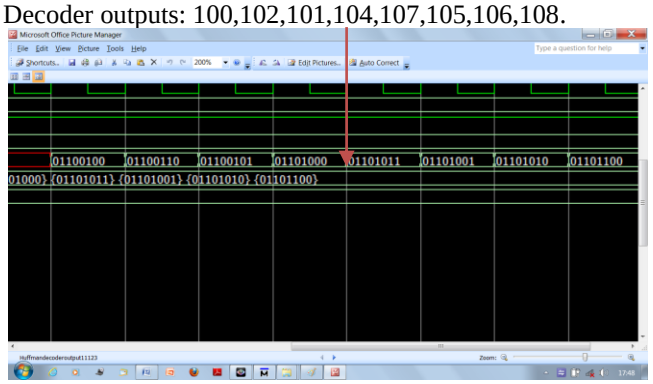
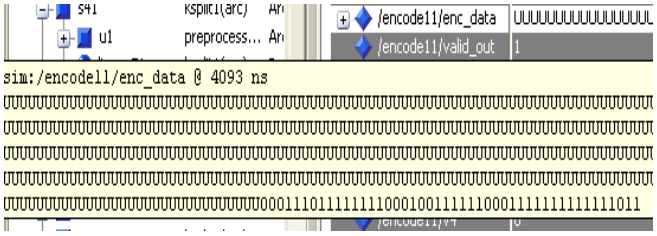


Fig.10. Huffman Decoder Outputs

C. Modified e_Rice real time negative/ and floating point outputs



Encoded output "enc_data"

Fig. 11. Floating Point encoded data output

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

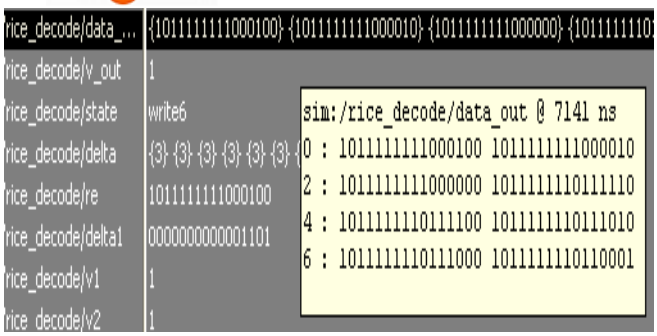


Fig. 12. Floating point decoded data in binary format

Let the modified e_Rice Encoder, for example, takes some real time values such as voltage values for space applications, ie., -1.537760, -1.521376, -1.504992, -1.488608, -1.472224, -1.455847, -1.439456, -1.390304. They are represented using single- precision format here. Only the 16 bits are considered here. That is the sign bit(1 bit), exponent(8 bits) , and the significand or the mantissa part (7 bits). So a total of 1+ 8+ 7 = 16 bits.

VI. COMPARISON RESULTS

Both the algorithms are compared for their performance parameters.

Algorithms / Parameters	Modified e_Rice	Huffman
Compression Ratio	1.86	2.67
Total Power	329mW	334mW
Total Delay	7.026ns	40.137ns
Area(Total equivalent gate count)	60,527	65,204

Table 3. Comparison Results for Performance Parameters

VII. APPLICATIONS

Extended_Rice or e_rice is suggested for many space science exploration mission applications. Besides the obvious benefits to space missions, lately infused the compression technique into ground data distribution and archive facilities. It will reduce not only the archive volume, but also the network connection time needed for distributing science data product over the internet. In an effort to broaden the domain of applications, this algorithm also applied to processed science data such as sea surface temperature, vegetation index in floating point representations. Applications also have found ways into science data archive, seismic data, acoustic data, and particle physics as well.

VIII. CONCLUSIONS

In this work, a CCSDS loseless data compression system based on e_Rice algorithm was introduced. The e_Rice algorithm exploits a set of variable-length codes to achieve compression. A modification suggested for e_Rice which improved its compression performance compared to original e_Rice. Since it has a number of coding options, it adapts to various data statistics to maximize compression performance. The synthesis results shows that power, area consumption and delay etc. is low for modified e_Rice.

The lossless data compression algorithm recommended by CCSDS has benefited many space missions by either reducing bandwidth, onboard storage requirement, or by increasing science data return. The percentage data reduction may not be the best achievable considering all other available techniques; however, its simplicity and adaptivity does allow high-speed space implementation and applicability to loseless data.

REFERENCES

[1] Data Compression and Huffman Encoding, CS106X Handout 25 Autumn 2009 November 2nd, 2009.
[2] Huffman, D.A, "A Method for the Construction of Minimum Redundancy Codes," Proc. IRE, Vol. 40, pp. 1098 1101, 1952.
[3] Lossless Data Compression, Consultative Committee for Space Data Systems CCSDS 121.0-B- 1 Blue Book, May 1997.
[4] Lossless Data Compression, Consultative Committee for Space Data Systems CCSDS 120.0-G-1 Green Book, May 1997.
[5] Pen-Shu Yeh, "The CCSDS Lossless Data Compression Recommendation for Space Applications," NASA/Goddard Space Flight Center Greenbelt, MD 20771.
[6] Rice, R. F., "Practical Universal Noiseless Coding," Proc. of the SPIE Symposium, Vol 207, San Diego, CA, Aug. 1979.
[7] Yeh, P.-S., Rice, F. R. and Miller, W. H., "On the Optimality of A Universal Noiseless Coder," Proc. of the AIAA Computing in Aerospace 9 Conference, San Diego, CA, Oct. 1993.

AUTHOR BIOGRAPHY - RAKHEE SASI was born in Ernakulam, Kerala, India. She has received her B.Tech degree in Electronics and Communication Engineering from Mahatma Gandhi University, Kerala, India and pursuing M.Tech in VLSI &Embedded Systems from Mahatma Gandhi University, Kerala, India. She has completed her project work under the guidance of Dr. Sreelal S., Sci/Engineer-SG (Section Head), BSED/ DSG at Vikram Sarabhai Space Centre (VSSC), Trivandrum.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)