iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 2 **Issue:** XII **Month of publication:** December 2014

DOI:

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)

Modified Huffman Coding Technique for High Secured Data Transmission

G. S. L. Alekhya[1], A. Narayana Kiran[2], M. S. S. Bhargav[3]
[1,2,3]Electronics and Communication Engineering Department
Shri Vishnu Engineering College for Women, India

Abstract*— In recent years, data encoding has become more popular in network access. Providing security for the data being transferred plays a crucial role. The rapid development in wired and wireless digital communication has made the extensive use of the text data. However, there are few researches focusing on encoding data, providing security and memory usage. The basic characteristics of text data like transmission rate, bandwidth, redundancy, bulk capacity and co-relation among text data makes basic compression algorithms mandatory. The incoming data is received using a wireless sensor .Therefore this paper considers the problem of area optimization and providing security to allow low bit rate transmission based on HUFFMAN coding. As for n bit being transmitted it requires 2n memory stack for further increase in the data bits it requires 2n+1 memory stack which is wide waste of memory if there presents redundancy bits. Image transmission has large repeated sequences at some places which can be considered as redundant. The proposed method uses stuffing bits in order to provide high security, speed, area optimizaton and also to avoid retention faults.
Keywords*— Encoding, HUFFMAN coding, Retention faults, Redundancy, Stuffing bits, Wireless sensor etc.

I. INTRODUCTION

In recent decades, compressing the data before transmitting has gained a lot of interest with a rapid growth of multimedia and presence of wide network access, as uses of this compressing of data ranges from mobiles, laptops to high quality satellite communication. Compressed data is the art of presenting data in its compact form. Compression techniques are used to reduce the amount of data that would otherwise be needed to store, handle, and/or transmit the represented content. Using compression technique provides high bandwidth rate, as HUFFMAN coding is a variable length coding, it provides an advantage of increased compression rate. Hence it is widely used as compression technique during transmission of data. In this STUFFING bits are used during compression of data [1][6]. Here using of stuffing bits provides advantage of decreasing the memory size which thereby reduces the cost [2] and retention faults can also be avoided. Without using stuffing bits the memory required should change dynamically. As the sequence of same value bits increases the count value which there by increases the memory width and size. The memory width is the count of the largest sequence of the incoming data, it is waste of memory as rest of the sequence count may not require that much width. The overall cost of the encoder increases and there is a chance of having retention faults. To avoid such disadvantages the concept called bit stuffing is introduced to the encoding technique. As stuffing an opposite bit after the largest allowing sequence allow the available memory to be used efficiently thereby decreasing the overall cost.

II. ENCODING TECHNIQUE

A. Sorting

A sorting algorithm is an algorithm that puts elements of a list in a certain order. The most-used orders are numerical order and lexicographical order. Here Bubble sort is the sorting algorithm performed. Bubble sort, sometimes referred to as sinking sort, is a simple sorting algorithm that works by repeatedly stepping through the list to be sorted, comparing each pair of adjacent items and swapping them if they are in the wrong order. The pass through the list is repeated until no swaps are needed, which indicates that the list is sorted. The algorithm gets its name from the way smaller elements "bubble" to the top of the list. Because it only uses comparisons to operate on elements, it is a comparison sort. Although the algorithm is simple, it is too slow for practical use, even compared to insertion sort.

B. Run 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. This is most useful on data that contains many such runs: for example, relatively 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 potentially double the file

size. As RLE is performed after bubble sorting the incoming data .

C. *Huffman encoding*

The technique works constructing a binary tree of nodes. The size of the tree depends on the number of symbols [8].

The simplest construction algorithm uses a priority queue where the node with lowest probability is given highest priority [3][7]:

1) Create a node called the leaf for each symbol and add it to the priority queue.
2) If there is more than one node in the queue:
   a) The nodes having the highest priority are to be removed (lowest probability) from the queue
   b) Create a new internal node with these two nodes as children and with probability equal to the sum of these two nodes' probabilities.
   c) Add the new node to the queue.
3) The remaining node is the root node and the tree is complete.

	a	b	c	d	e
Frequency	60	5	30	5	10
Fixed length	000	001	110	101	111
Variable length	000	001	010	011	1

Table 1: Huffman coding example.

No. bits for fixed length coding is 150*3=450 and for variable length coding is 60*3+5*3+30*3+5*3+50*1=350

10% memory is saved.

In figure 1 as shown Huffman coding is a variable length coding. In a variable-length code the code words may have different lengths. Here are examples of fixed and variable length codes for our problem (note that a fixed- length code must have at least 3 bits per code word).

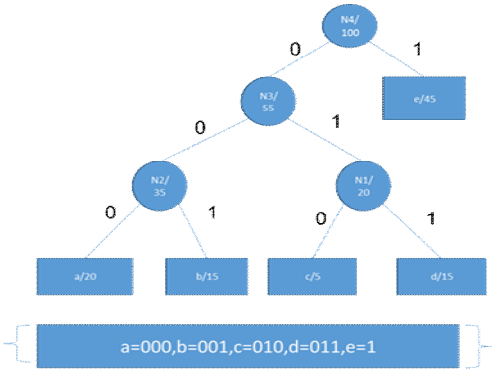


Fig 1: Huffman tree.

D. *Bit Stuffing*

Stuffing bits are mainly used to limit the occurrence of consecutive bits having the same value. To limit this occurrence, a bit of opposite value is inserted after allowing maximum number of consecutive bits [4]. Bit stuffing is mainly used to limit the width of memory and decrease the cost. As memory cannot be changed dynamically and is also a cost issue, increased occurrence of

same value consecutively which exceeds the existing memory width may cause loss of data and increasing the width of the memory also increases the memory size which there by increases the cost and ratentioon faults also arises. To avoid this problem and for efficiently using the already existing memory bit stuffing plays a prominent role.

Therefore, increased speed of transmission, reduced cost of memory and efficient usage of available memory are achieved by bit stuffing.

E. HDL Implementation of Encoder

FIFO is an acronym for First In, First Out, a method for organizing and manipulating a data buffer, where the oldest (first) entry, or 'head' of the queue, is processed first. It is analogous to processing a queue with first-come, first-served (FCFS) behavior where the people leave the queue in the order in which they arrive.

The incoming pixel values i.e., data is stored in FIFO which is given as input to the sorting block.

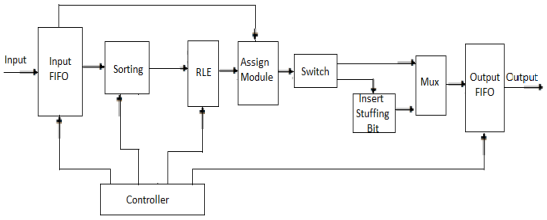


Fig2: Block diagram of encoder.

The Encoder block again contains sub modules such as a) FIFO b) sorting module c)RLE module d) Assign module e)Controller. Here Bubble sort is performed where the data is stored in ascending order. As it makes RLE easy to count the repetition of each pixel value. The output after performing bubble sort is given as input to Run Length Encoder. The data is compressed efficiently and is given to the assign block. Run Length Encoder counts the repetition of each pixel and gives it as input to the Assign module. In assign module Huffman encoding is performed. It allocates minimum number of bits to the maximum repeated pixel values. So, whenever the maximum repeated pixel appears from input instead of sending the pixel value it sends the allocated binary value. Thereby reducing the memory needed and decreasing the transmission band width.

The output of the assigning module is given to the switch which is a de-multiplexer, it decides where the it needs to insert stuffing bit or not. the output of the assign module is in binary format which is stored in output FIFO including stuffing bit where ever required. Controller gives the control signals to the RLE, FIFO, Sorting block like FIFO FULL, FIFO EMPTY, RESET, FIFO ENABLE etc,.

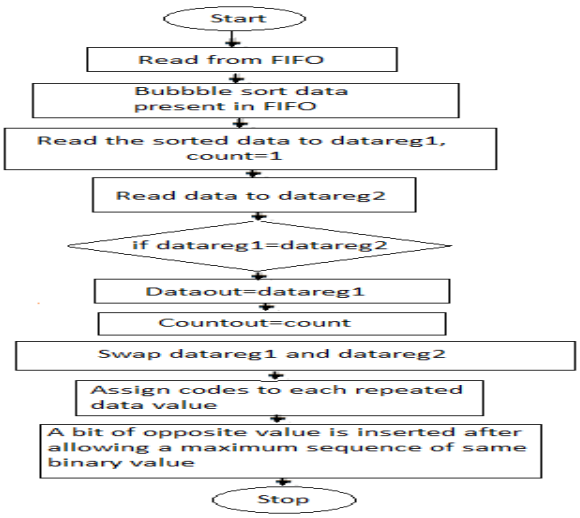


Fig 3: Flow chart of the encoder.

The incoming data which is to be transmitted is stored in FIFO which is read from when data is being encoded during transmission. first data is read and stored in register1 and second data is read and stored in register2. These two information bits ie, binary data are compared if the two binary bits are same then count register is incremented and again the next data is fetched and compared, if not equal send the data to the data out and count to count out. Now the two data register values are swapped and again fetch the data to the second register. Binary codes are assigned to each data value such that whenever a hexa-decimal value occurs at the input it gives corresponding binary code as output. Stuffing bit is added where it is required that is whenever there is a large sequence of same binary value a bit of opposite value is inserted to avoid retention faults.

F. FIFO Stack

FIFO is used to store the incoming data and send to any other module when necessary and to maintain same frequency between transmitter and the compressor. As there may be some frequency difference between transmitter and the compressor unit which may lead to in appropriate working of the unit.

Datai(7:0)	Datao(7:0)
FIFOEna	
FIFORd	Empty
FIFOWr	
RdClk	
RST	
WrClk	FULL

Fig 6: FIFO Schematic.

G.Encoder Algorithm

Step1: Read data from the input FIFO and assign it to Datareg1, again read next data from FIFO and assign it to Datareg2.
Step2:sort the incoming data using bubble sort.
Step3:Compare the data present in datareg1 and datareg2.
Step4: If the binary bits are equal increment the countreg value.
Step5: If count reg value reaches the maximum value, send the data to the output FIFO and also send the count value to output FIFO.
Step6: Now insert a opposite value bit to the one present in data out, which is called the stuffing bit.
Step7: Assign the value of the bit present in datareg2 to datareg1 and fetch next binary value from the input FIFO to datareg2.
Step8:Again compare the binary values present in datareg1 and datareg2 and go to step3.
Step9:Assign codes to each repeated hexadecimal value.
Step10:now insert a stuffing bit to the output after assigning codes where ever necessary.

H.Experimental Results

The incoming hexa-decimal values are stored in FIFO which are used for encoding.

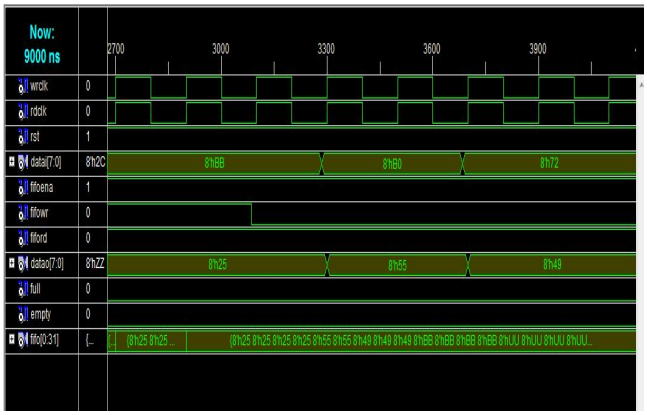


Fig 3:simulation results of FIFO stack.

Working of encoder and queues that are used to store the data can be explained by seeing the simulation results.

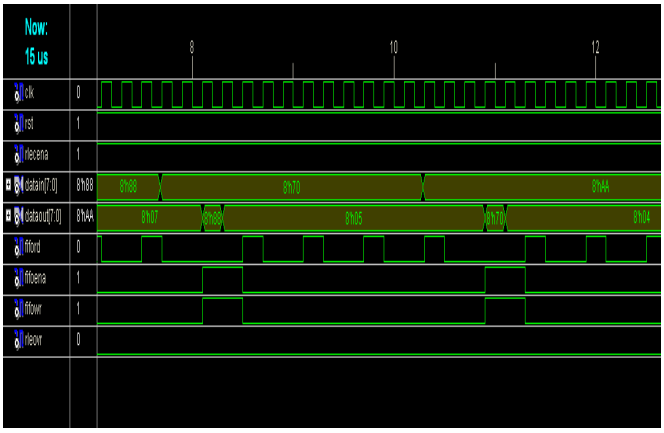


Fig 4:simulation results of RLE.

Incoming serial data is stored in FIFO which is encoded that is RLE counts the repeated sequence of each value after sorting is performed.

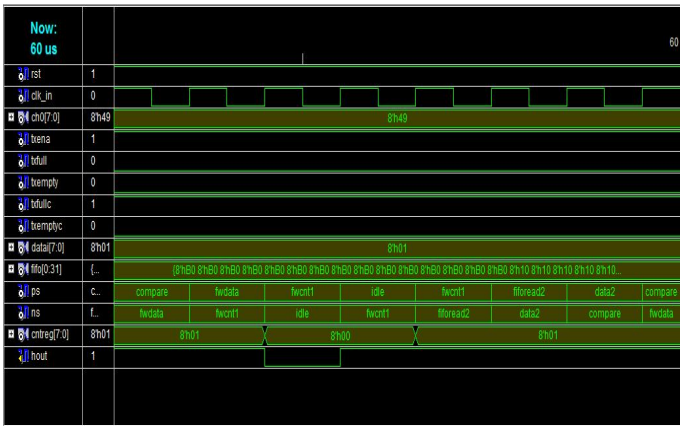


Fig 5:Simulation results of Huffman encoder without stuffing bits.

The encoded data is now sent to the huffman encoder which assigns the binary values.

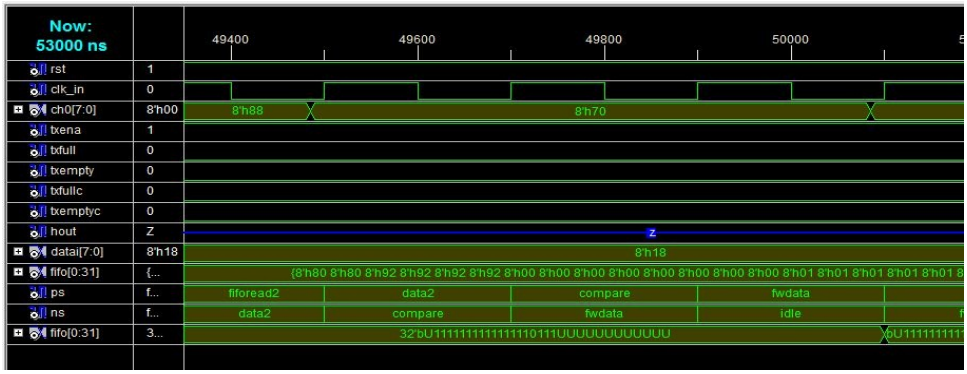


Fig 6: Simulation results of Encoder using Stuffing bits.

Using stuffing bits, as we can see it avoids the repeating of consecutive binary digit of same value and retention faults can be eliminated.

III. CONCLUSIONS

The encoding technique used is highly reliable and Huffman technique used here makes the data uniquely decodable. Using Stuffing bits provides efficient use of memory and increase in speed factor as it helps in breaking the long sequence of same binary value and also to avoid retention faults. With this proposed method of adding stuffing bits there is a small burden on memory because of the extra stuffing bits but the effect of stuffing bits is very much high while reducing the memory required and also cost of the integrated circuit and high security can also be provided for the data being transmitted and stored . Overall memory required to store the complete data sequence reduces. In the case of communication systems, the transmission rate can be increased with the introduction of stuffing bits. The overall cost is also reduced. By adopting some other compression techniques the transmission rate can be further improved.

REFERENCES

[1] Dr. Muhammad Younus Javed and Mr. Abid Nadeem,"Data Compression Through Adaptive Huffman Coding Scheme" , IEEE-2000, Vol. II, pp.187-190.

[2] D.A.Huffman,A method for the constraction of Minimum Redundancy Codes , Proceedings of the IRE, 1952 ,pp. 1098-1101.

[3] Hirschberg , D.S. and Lelewer,D.A. ,Efficient Decoding of prefix codes ,Communication of the ACM, pp.449-458, 1990.

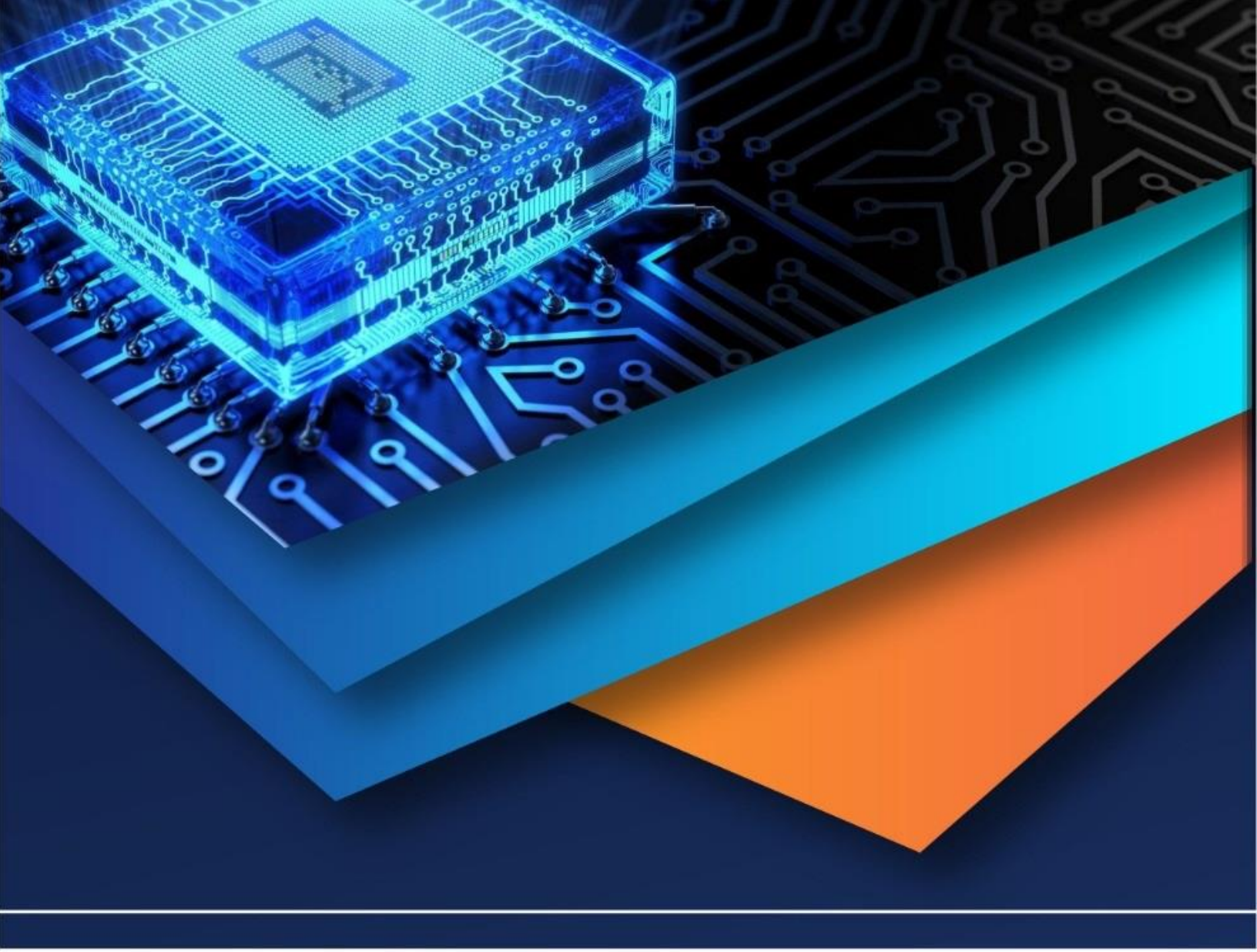
[4] J. Feng and G. Li, " A Test Data Compression Method for System-on-a-Chip" 4th IEEE International Symposium on Electronic Design, Test and Applications, 2008.

[5] Nadeem , A. ,Design and Implementation of Data Compression System , College of Electrical and Mechanical Engineering ,Rawalpindi ,Thesis, 1995.

[6] P. Bender and J. K. Wolf "A universal algorithm for generating optimal and nearly optimal run-length-limited, charge constrained binary sequences", Proc. IEEE Int. Symp. Information Theory, 1993

[7] Shukla, P.K. ; Rusiya, P. ; Agrawal, D. ; Chhablani, L. ; Raghuwanshi, B.S. “Multiple Subgroup Data Compression Technique Based on Huffman Coding” International conference on Computational Intelligence, Communication Systems and Networks, pp. 397- 402, 2009.

[8] Vitter, S. V.,Design and analysis of dynamic Huffman codes ,Journal of the Assocition for Computing Machinery ,Vol. 34, No. 4, pp. 825-845, 1987.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY