International Journal of Current Engineering and Technology
E-ISSN 2277 – 4106, P-ISSN 2347 – 5161

Available at http://inpressco.com/category/ijcet

Research Article

Design and Implementation of Huffman Decoder for Text data Compression

Swapna R†* and Ramesh P.†

†Department of ECE, College of Engineering Munnar, Kerala, India

Accepted 12 June 2015, Available online 17 June 2015, **Vol.5, No.3 (June 2015)**

Abstract

Digital compression of data is important due to the bandwidth limitations inherent in the transmission medium. Data compression is also called as source coding. It is the process of encoding information using fewer bits than an uncoded representation. Compression is a technology for reducing the quantity of data used to represent any content without excessively reducing the quality of the picture. It also reduces the number of bits required to store and/or transmit digital media. Compression is a technique that makes storing easier for large amount of data. There are various techniques available for compression, this paper presents Huffman decoder based on new binary tree method for improving usage of memory and Bandwidth for Text data Compression. The work mainly deals with the implementation of Huffman decoder on a Xilinx 14.7 version, using Verilog Hardware Description Language.

Keywords: *Binary tree, Data compression, Decoding algorithm Huffman decoder, Verilog, FPGA.*

1. Introduction

Compression is a necessity in the current world of technology, which is centered on speed and efficiency. Consequently, large and bulky pieces of information are abandoned for smaller bits of data, which can be shared between peers at faster rates. Data can be broken into smaller pieces or forcefully compressed by programs that are powered by algorithms. The two major types of compression algorithms are lossless compression and lossy compression. Lossless compression is used for applications that require an exact reconstruction of the original data, while lossy compression is used when the user can tolerate some differences between the original and reconstructed representations of the data. Lossy compression techniques involve some loss of information and data are compressed using lossy techniques generally cannot be recovered or reconstructed exactly. In return for accepting this distortion in the reconstruction, we can generally obtain much higher compression ratios than is possible with lossless compression. An important element in the design of data compression algorithms is the modeling of the data. The extremely fast growth of data that needs to be stored and transferred has given rise to the demands of better transmission and storage techniques. Various lossless data compression algorithms have been proposed and used. Huffman Coding, Arithmetic Coding, Shannon Fano Algorithm, Run Length Encoding Algorithm are some of the techniques in use. David Huffman as part of a class assignment developed the Huffman coding algorithm. Huffman codes are prefix codes and are optimum for a set of probabilities. The Huffman code is based on two observations. First, in an optimum code, symbols that occur more frequently (have a higher probability of occurrence) have shorter codewords than symbols that occur less frequently. Second, in an optimum code, the two symbols that occur least frequently have the same length. The Huffman procedure is obtained by adding a simple requirement to these two observations. This requirement is that the codewords corresponding to the two lowest probability symbols differ only in the last bit.

2. Preliminary

A Huffman decoder is implemented for text. The text compression involves its encoding; the text decoder contains the Huffman decoder for obtaining the original text.

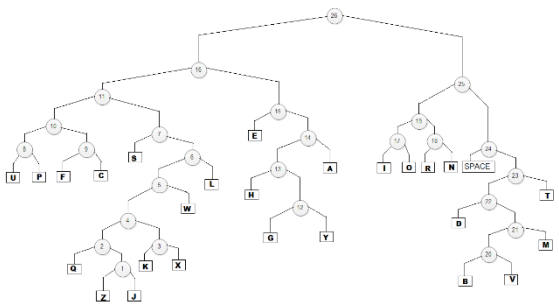

Fig 2.1 Binary trees for Text

*Corresponding author: **Swapna R**

The Huffman tree used by encoder and decoder is shown in Fig2.1.The alphabet consists of the uppercase letters and the space. All left branches are labeled 0, and all right branches are labeled1.This tree is based on the following assumed frequencies E 130 T 93 N 78 R 77 I 74 O 74 A 73 S 63 D 44 H 35 L 35 C 30 F 28 P 27 U 27 M 25 Y 19 G 16 W 16 V 13 B 9 X 5 K 3 Q 3 J 2 Z 1

It is assumed that there are 130 Es and 182 spaces for every 1000 letters. The encoder retrieves the code for each symbol from a map, and shifts it out one bit at the time. The decoder is a finite state machine whose state transition graph is obtained from the tree by adding acs from the leaves back to the top of the tree. Each node uses ten bits for its encoding. The code of the root is 0. If a state is not a leaf of the tree, and its encoding is n, then the encodings of its two children are 2n+1 and 2n+2.

2.1 Implementation of Huffman Encoder

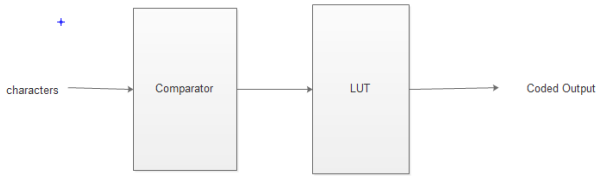

Fig.2.2 Block diagram of Encoder

The Fig 2.2 shows the block diagram of encoder and code for each character which comes from the tree stored in the LUT. Character input which is given to the encoder is stored inside the LUT. Therefore, the output of the encoder block will be these stored values inside the LUT.

2.1 Implementation of Huffman decoder

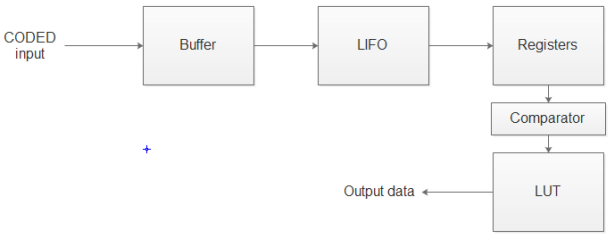

Fig.2.3 Block diagram of Decoder

The fig 2.3 shows the block diagram for the decoder in which the coded value is first stored in the buffer then it is shifted using a LIFO,The shifted value is then stored in the 9 bit temporary register which is then compared with respective codes stored in the LUT and finally the character is decoded.

3. The Proposed Method

In the Proposed method a Huffman tree is implemented using the binary tree which is built upon using the frequencies corresponding to the characters shown in fig2.1 and Fig3.1.Figure 3.1 shows a binary tree in which the branch values are mentioned based on its construction using the characters and respective frequencies given.

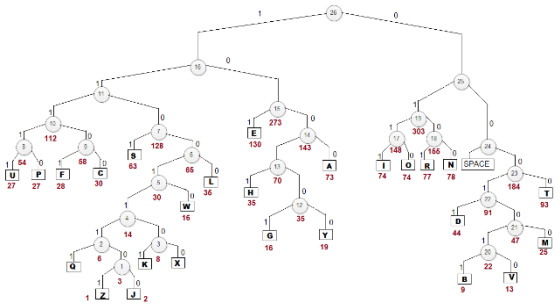

Fig 3.1 binary trees with corresponding codes for Text

3.1 Implementation of Proposed Huffman Encoder

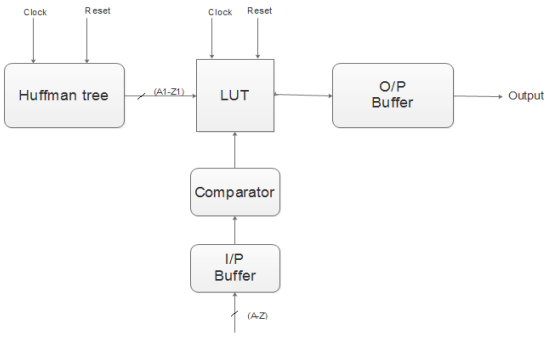

Fig.3.1 Block diagram of Proposed Encoder

In this proposed work the Encoder is implemented using a Huffman tree. A Huffman tree is implemented in Verilog platform using the binary tree shown in the fig 2.1. This preimplemented Huffman tree is stored in the LUT to give the corresponding encoded output in correspondence to the character.

3.2 Implementation of Proposed Huffman Decoder

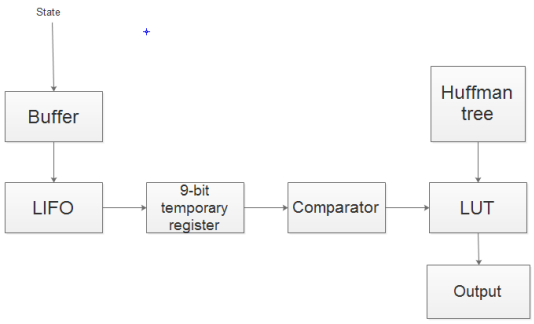

Fig.3.1 Block diagram of Proposed Decoder

Both the encoding and decoding should done with respect to the same tree.so that the data stored in the encoder block is stored in the encoder LUT is stored in the decoder LUT. In the proposed method, inside the

decoder block first a buffer is present in order to store the output of the encoder part. A LIFO is present next to it which will shift the coded values stored inside the Buffer. This shifted code is then stored inside a temporary register of 9-bit size. Both this coded value and the predetermined Huffman tree which is stored inside the LUT is compared to obtain a decoded output with respective to the corresponding coded state.

4. Simulation Results for Text using the Proposed Method

A Huffman Encoder and decoder is designed, described in Verilog and implemented on a Xilinx Virtex 5 FPGA using ISE 14.7. The design aims to achieve high operating frequencies using few logical resources. The functional simulation for the Huffman encoder and decoder block is carried out using the ISE design suite 14.7

The binary tree for the characters based on the assumed frequencies E 130 T 93 N 78 R 77 I 74 O 74 A 73 S 63 D 44 H 35 L 35 C 30 F 28 P 27 U 27 M 25 Y 19 G 16 W 16 V 13 B 9 X 5 K 3 Q 3 J 2 Z 1 is shown below.

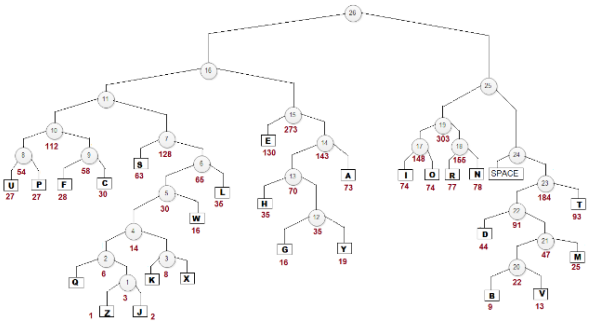

Fig.4.1 Binary tree based on the assumed frequencies

The Simulation results for the tree is obtained .In this the data value corresponding to the branch is shown in fig4.2.

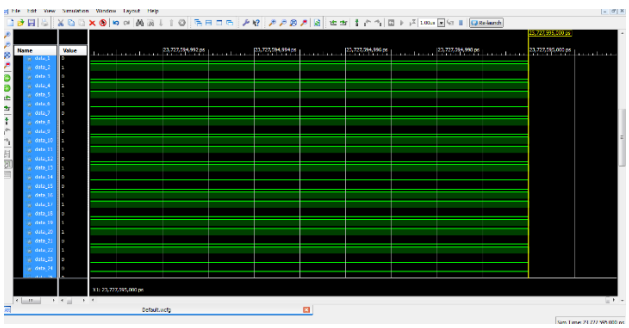

Fig.4.2 Branch value corresponding to the tree

The binary values for the frequencies corresponding to the characters is shown in fig4.3

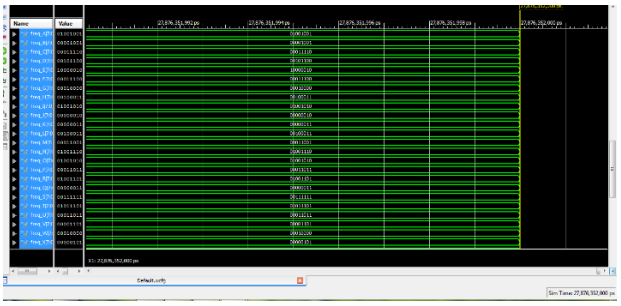

Fig.4.3 Simulated output of the Branch value corresponding to the tree

Fig 4.4 shows the node values of the tree corresponding binary formed on the basis of the assumed frequencies.

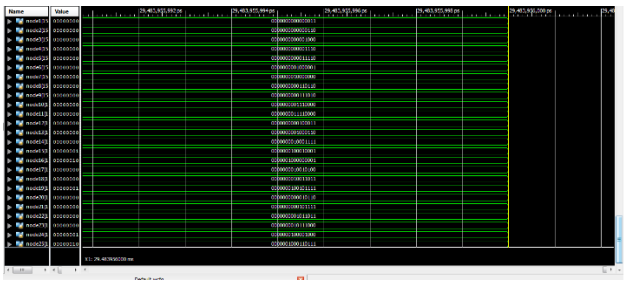

Fig.4.3 Simulated output of the branch values of the binary tree

The simulated result of the Encoder for output of a word is obtained using the Xilinx 14.7

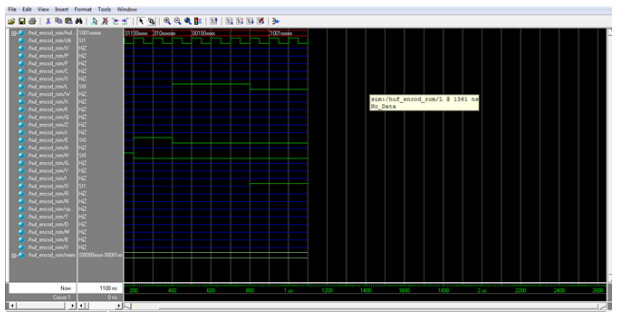

Fig.4.4 Simulated output of the word HELLO

The output of the encoder is given as the input to the buffer in the decoder block the value is compared with the LUT and the decoded output is obtained

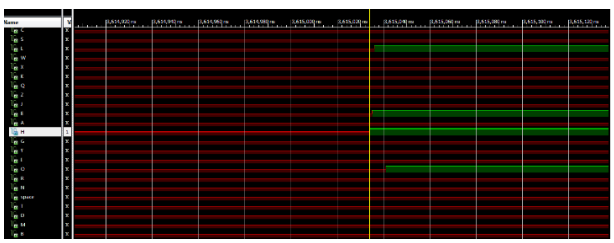

Fig.4.5 Simulated decoded output of the word HELLO

In Fig 4.4, the input signal is a 5-bit input signal which acts as the address to the LUT in the encoder stage which gives corresponding alphabetical outputs. Encode is the serial output stream which given as input to the decoder. Is the decoded character output from the decoder. The encoding and decoding operations are performed for the text HELLO. The simulation results for HELLO text reveal that only 22 bits are required to store it whereas 40 bits are required for the original text. Hence original data can be retrieved easily and requires less memory by using the new binary tree algorithm method

Conclusion

This research will show that the higher data redundancy helps to achieve more compression. The presented new compression and decompression technique based on Huffman tree for scan testing is used to reduce test data size and test application time. At Present Started with designing a Huffman encoder and decoder in Verilog platform. Huffman decoder using Binary tree algorithm was implemented on Verilog and FPGA platforms. The Architecture implemented by VERILOG Design, using XIINX 14.7 version. Future works needs to be carried out to improve the area. On comparing with other different compression techniques, came to a conclusion that Huffman coding is efficient technique for image compression and decompression to some extent.

References

L. Acasandrei and M. Neag (2000), A fast parallel huffman decoder for fpga implementation, *Acta Tehnica Napocensis Electronics and Telecommunications*,

Z. Aspar, Z. M. Yusof, and I. Suleiman (2000), Parallel huffman decoder with an optimized look up table option on fpga, in *TENCON 2000. Proceedings*, vol.1.IEEE, pp. 73–76

S. Beak, B. Hieu, H. Lee, S. Choi, I. Kim, K. Lee, Y. Lee, and T. Jeong (2012) Novel binary tree huffman decoding algorithm and field programmable gate array implementation for terrestrial-digital multimedia broadcasting mobile handheld, *Science, Measurement & Technology, IET*, vol. 6, no. 6, pp. 527–532.

M. Benes, S. M. Nowick, and A. Wolfe (1998), A fast asynchronous huffman decoder for compressed-code embedded processors, in *Advanced Research in Asynchronous Circuits and Systems, 1998. Proceedings. 1998 Fourth International Symposium on*. IEEE, pp. 43–56.

S. B. Choi and M. H. Lee, High speed pattern matching for a fast Huffman decoder, *Consumer Electronics, IEEE Transactions on*, vol. 41, no. 1, pp. 97– 103, 1995.

R. A. Chowdhury, M. Kaykobad, and I. King (2002), An efficient decoding technique for huffman codes, *Information processing letters*, vol. 81, no. 6, pp. 305–308.

D.-H. Kim, J. H. Song, D.-H. Kim, and S. Lee (2014), Fixed cycle huffman decoding instruction for multi-format decoder, in *Consumer Electronics (ICCE), 2014 IEEE International Conference on*. IEEE, pp. 113–1

C. Kuo-Liang (1999), Efficient huffman decoding, *Information Processing Letters*, vol. 61, no. 2, pp. 97–99

J. Lee (2011), Huffman data compressi

Vijayakumar Suvvari and M.V.H Bhaskara Murthy (2011), VLSI implementation of Huffman tree using binary tree algorithm. *International Journal of Computer Technology & Applications*, vol. 3, no. 1

M. K. Mathur, S. Loonker, and D. Saxena (2012), Lossless huffman coding technique for image compression and reconstruction using binary trees. *International Journal of Computer Technology & Applications*, vol. 3, no.

Riyadh A. Abdulhussein And Abdulkareem S. Abdallah (2013), A Comparison Study Of Non-Binary Tcm-Aided Pam, Qam, Psk Schemes-Based Novel Decoding Algorithm International journal of Electronics and Communication Engineering &Technology (IJECET), Volume 4, Issue 5, pp. 177 - 186,

P. Prasanth Babu, L.Rangaiah And D.Maruthi Kumar (2013), Comparison And Improvement Of Image Compression Using DCT, DWT & Huffman Encoding Techniques, International Journal of Computer Engineering & Technology (IJCET), Volume 4, Issue 1, pp. 5

S. Anandanarayanan and Dr.S.K. Srivatsa (2013), A High Performance Novel Image Compression Technique Using Huffman Coding With Edge Dection, International Journal of Computer Engineering & Technology (IJCET), Volume 4, Issue 2, pp. 17 - 22