International Journal of Emerging Trends & Technology in Computer Science (IJETTCS)
Web Site: www.ijettcs.org Email: editor@ijettcs.org, editorijettcs@gmail.com
Volume 7, Issue 2, March - April 2018 ISSN 2278-6856

HUFFMAN ENCODER AND DECODER USING VERILOG

Abhishek Kumar Jha, Deepak Pathak, Bharat Yadav' Abhishek Bharadwaj, Neerja Singh

ABES Engineering College,
NH-24, Ghaziabad, India

Abstract : ***A binary Huffman encoder has been made using Verilog HDL on tool Xilinx ise14.6 isim simulator.***
The Huffman encoder has been designed and synthesized using a finite state machine. It basically reduces the repeated messages and thus contracts the message. So now the repeated messages will be send only one time.
Keywords: Compression, bandwidth, lossless, lossy.

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 In 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 usinglossy techniques generally cannot be recovered or reconstructed exactly return for accepting this distortion in the reconstruction, we can generally obtain much higher compression ratios than is possible with lossless compression.

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.
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.
Design and Implementation of Huffman Decoder for Text data Compression.
The Huffman tree used by encoder and decoder is shown in.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.

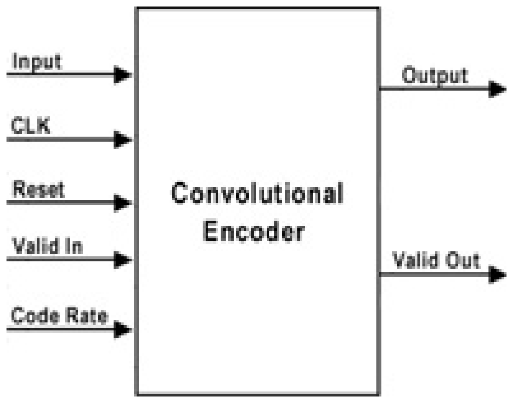

Fig. 1 Encoder

3. Implementation of Huffman Encoder

The Fig 1 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.

The block diagram for the decoder in which the coded value is first stored in the buffer then it is shifted using a LIFO,Theshifted 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.

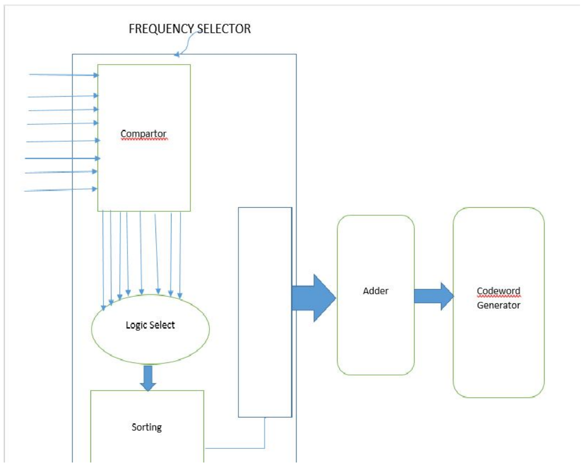

Fig. 2 Flowchart of Coding Process

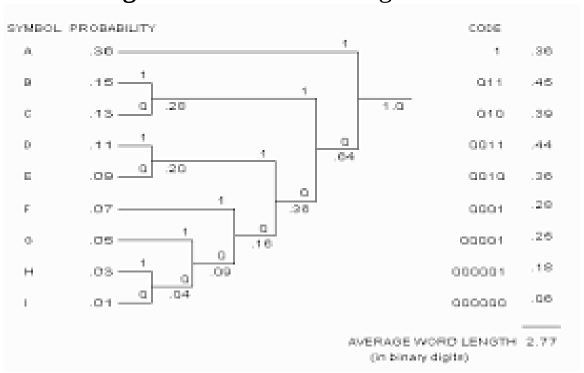

Fig. 3 Assigning of Codeword

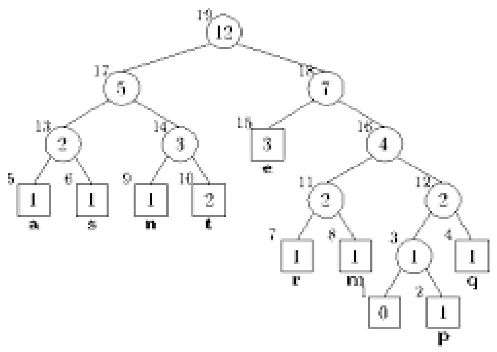

Fig. 4 Binary Tree

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.

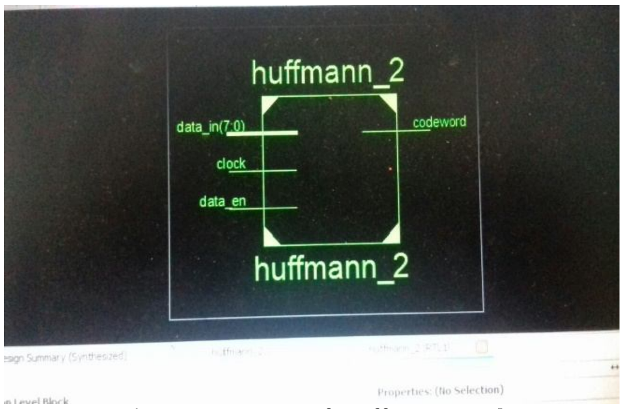

Fig. 5 RTL View of Huffman encoder

RTL View is a Register Transfer Level graphical representation of your design. This representation (.ngr file produced by Xilinx Synthesis Technology (XST)) is generated by the synthesis tool at earlier stages of a synthesis process when technology mapping is not yet completed.

The above shown Fig. 5 shows the Register Transistor Logic (RTL) View of the Huffman encoder. The RTL view describes the complete circuit into a single block in which there are multiples transistors used to complete an encoder circuit.

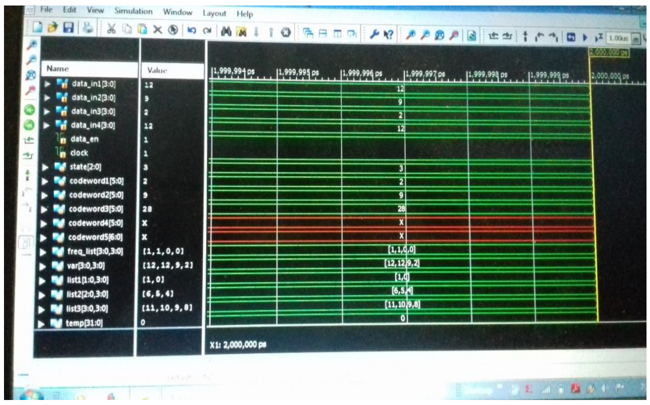

Fig. 6.1 Simulation result

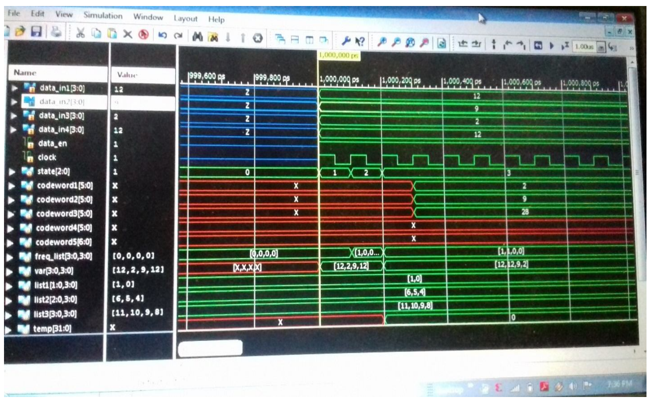

Fig. 6.2 Simulation result

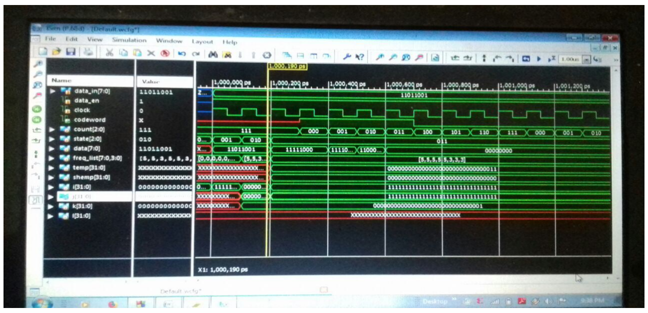

Fig. 7 Simulation report of Huffman encoder

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.

5. 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 versions. 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.

6. References

[1] Sameer palnitkar, Verilog HDL design
[2] BP Lathi, Principles of communication
[3] IEEE Std 1364-2005 – The official standard for Verilog 2005
[4] schawmz series, Data structure
[5] M.V.H Bhaskara Murthy (2011), VLSI
[6] (1998) The Google website. [Online]. Available: http://www.google.com
[7] (2010) The quora website. [Online]. Available: http://www.quora.com
[8] (2010) The GitHub website. [Online]. Available: http://www.github.com
[9] (2010) The edaboard website. [Online]. Available: http://www.edaboard.com
[10] (2010) The researchgate website. [Online]. Available: http://www.researchgate.com

AUTHORS

Abhishek Kumar Jha is pursuing his Engineering in Electronics and Communication from ABES Engineering College (2014-2018) batch.
He is a certified HDL Programmer and has worked on multiple projects with his team.

Deepak Pathak is pursuing his Engineering in Electronics and Communication from ABES Engineering College (2014-2018) batch.
He is a certified HDL Programmer and has worked on multiple projects with his team.

Bharat Yadav is pursuing his Engineering in Electronics and Communication from ABES Engineering College (2014-2018) batch.
He is a certified HDL Programmer and has worked on multiple projects with his team.

Abhishek Bharadwaj is pursuing his Engineering in Electronics and Communication from ABES Engineering College (2015-2018) batch.
He has also completed his Diploma from Lovely Professional University.

Neerja Singh is an Asst. Professor in the Department of Electronics and Communication Engineering in ABES Engineering College, Ghaziabad, Uttar Pradesh, India. She has received her M.Tech. in VLSI Design and B. Tech in Electronics and Communication Engineering in the year of 2012 and 2010 respectively. Her main research interests are in Low power VLSI Design for low power Electronics product.