https://doi.org/10.26682/csjuod.2023.26.2.37

Journal of University of Duhok., Vol. 26, No.2 (Pure and Engineering Sciences), Pp 400-406, 2023
4th International Conference on Recent Innovations in Engineering (ICRIE 2023) (Special issue)

HIGH PERFORMANCE OF CODING AND DECODING PROCESS FOR DATA COMPRESSION USING VARIABLE IN LENGTH CODING

MAAN **H**AMEED* , **A**HMED **L**ATEEF **H**AMEED **, **N**ASEER **AL**WAN **H**USSEIN***
*Ministry of Water Resources, State Commission for Reservoirs and Dams, Diyala- Iraq
**Diyala Education Directorate, Ministry of Education, Diyala- Iraq
***Computer of engineering Networks, College of fine Arts, University of Diyala- Iraq

(Accepted for Publication: November 27, 2023)

ABSTRACT

Data compression using Huffman coding refers to decreasing the quantity of data without decreasing the quality of original file. Besides that, it can retrieve original data in decompression process without losing any details. In this research, an 9bit/8bit encoding and decoding process divide the block design. The input transmission code consists of 9-bit which are variable in length coding and can be suitable for high-speed applications. Coding and decoding blocks were designed separately. The encoder module gets the 9-bit data used as input and delivers the 8-bit coded-output from encoder design. this output data used as input to the decoder module to get the 8-bit as output form decoder design. In this research, the proposed design includes encoder and decoder were achieved Compression Ratio up to 52% from original data size and saving percentage up to 47.95%. The suggested design was implemented by using ASIC and FPGA design methodologies to execute the compression and decompression architectures. The architecture of coding and decoding process has been created using Verilog HDL language. Quartus II 11.1 Web Edition (32-Bit). In addition, simulated using ModelSim-Altera 10.0c (Quartus II 11.1) Starter Edition. And it is implemented using Altera FPGA (DE2) for real time implementation. Finally, all of the blocks were combined together to have an integrated system.

KEYWORD: Data compression, Symbols, Encoding, Delays, Decoding

INTRODUCTION

Using Huffman coding, lossless data compression reduces the amount of data used to represent a file without lowering the quality of the data source [1]. Furthermore, it may retrieve original information during decompression without missing any data. This study will proceed by means of the strategy used for designing and putting the coding and decoding process into practice over the course of the work. This project was well managed and organized. By creating a Huffman architecture, which is implemented based on the binary tree to extract the codewords, the process for creating this project was generalized. The HDL language Verilog was used for creating the code for Huffman, and every module's construction were validated. Following that, to accomplish real-time layout execution, an FPGA DE2 board was utilized execution of the system [2]. It was described how to develop a Huffman design. It was also addressed how to implement an FPGA and how to validate module-level and top-level designs. Basically, there are numerous stages to the design technique [3]. The first two steps concern Huffman construction, whereas the third step is unique to FPGA Huffman implementation. Before the final ASIC implementation, all ASIC designers have consistently employed FPGAs as a tool to test and validate their designs. A matrix of reconfigurable gate array logic circuitry is found inside an FPGA. The internal circuitry of an FPGA is coupled in such a way during configuration that a hardware implementation of the software program is produced. FPGAs employ specialized hardware for using this approach, ideas that process logic but lack an operating system can be tested and verified in hardware without having to go through the drawn-out manufacturing process of a custom ASIC design [4]. It is one of the greatest techniques to simulate designs to obtain the actual system behavior, and to make it more thorough, it is confirmed by using an FPGA implementation. Additionally, designers have access to ASIC integrated circuit technology for the implementation of digital logic. The final four steps concern the investigation and improvement of the Huffman design's power usage [5].

Steps of Building the Tree

The Huffman code generation process, which uses the technique of exchanging fixed length codes for input data ASCII by VLC, has been shown to be advantageous in cutting down on the total length of the information. The Huffman coding algorithm is simple and may be understood using the Huffman code tree. By simply traversing the binary tree from root to node after creating a Huffman tree, the algorithm generates a codeword for each symbol from the text. It gives right hand branches a value of 1 and left hand branches a value of 0. For each symbol in the text, the static Huffman method as shown above needs a specific number of frequencies. It is necessary to record the output codes, Huffman tree, and data frequency. This information is saved in the compressed file's header and is used throughout the decompression process. The first step in creating a Huffman tree is to arrange the text data according to each character's frequency in order to extract its code. Each node starts out with a symbol and its probability [6].

Queue of Data

Both the encoder and the decoder use the Huffman tree. The uppercase letters and the space make up the alphabet. The following assumed frequencies form the basis of the Huffman tree. The frequency of each character is represented by each number. 180 spaces are present every 1000 letters. For the purpose of creating a Huffman tree, arrange the alphabetic characters in ascending order by their frequencies. Each leaf on a list of available leaves should be assigned to an alphabetic symbol and listed in ascending order of frequency. The binary tree with branches labeled with bits 0 and 1 is created during the coding process. Figures 1 depict the node's block construction. Figure 2 shows the top-level Huffman design in RTL.

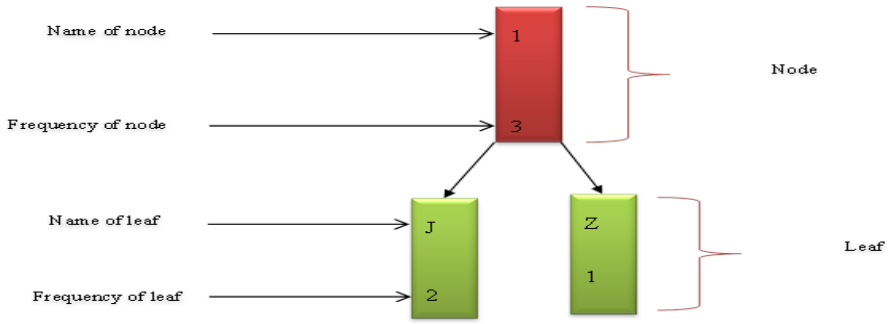

Fig.(1): -Explanation a construction of nodes

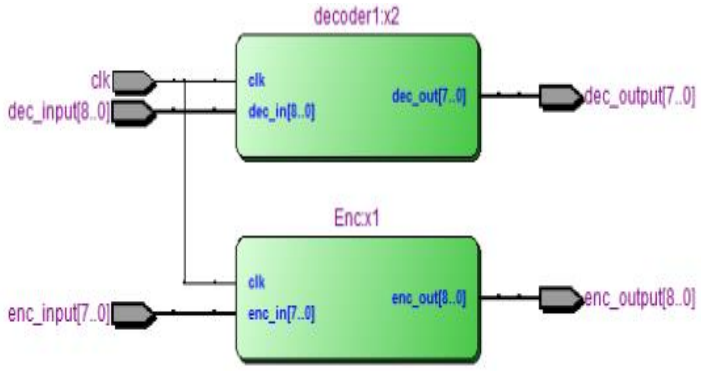

Fig.(2):-RTL viewer of top-level Huffman design

4.2. Codewords Generating

The best compression rate for compressed block length magnitude is achieved by the Huffman coding procedure, which generates Variable Length Codes (VLC) [7]. The recommended approach is to indicate a bigger number of bits for codewords that occur infrequently and a smaller number of bits for those that do. It is possible to create Huffman codes by creating a Huffman tree, which displays the Huffman tree for a sequence of data (all English alphabets), starting from the highest frequency (E=130), and ending with the lowest frequency (Z=1). The tree has annotated frequencies, and the generated codewords of the Huffman encoder are displayed in Table 1.

Table (1): -Codewords size

No	Characters	Frequencies	Codeword	Codeword bits	New size
1	Space	180	110	3	540
2	E	130	010	3	390
3	T	93	1111	4	372
4	N	78	1011	4	312
5	R	77	1010	4	308
6	I	74	1000	4	296
7	O	74	1001	4	296
8	A	73	0111	4	292
9	S	63	0010	4	252
10	D	44	11100	5	220
11	H	35	01100	5	175
12	L	35	00111	5	175
13	C	30	00011	5	150
14	F	28	00010	5	140
15	P	27	00001	5	135
16	U	27	00000	5	135
17	M	25	111011	6	150
18	Y	19	011011	6	114
19	G	16	011010	6	96
20	W	16	001101	6	96
21	V	13	1110101	7	91
22	B	9	1110100	7	63
23	X	5	00110011	8	40
24	K	3	00110010	8	24
25	Q	3	00110000	8	24
26	J	2	001100011	9	18
27	Z	1	001100010	9	9

4.2.1. Specifications of Huffman Tree

Huffman design based on the binary tree used to generating codeword bits for compressed text data. Binary tree consist of number of nodes and leaves, each node connected by two leaves or nodes. This is illustrated in Table 2.

Table (2): -Specifications of Huffman tree

Levels	Depth	Nodes	Characters
Level 1	0	26	_
Level 2	1	16, 25	_
Level 3	2	11, 15, 19, 24	_
Level 4	3	10, 7, 14, 17, 18, 23	E, SPEACE
Level 5	4	8, 9, 6, 13, 22	S, A, I, O, R, N, T
Level 6	5	5, 12, 21	U, P, F, C, L, H, D
Level 7	6	4, 20	W, G, Y, M
Level 8	7	2, 3	B, V
Level 9	8	1	Q, K, X,
Level 10	9	_	Z, J

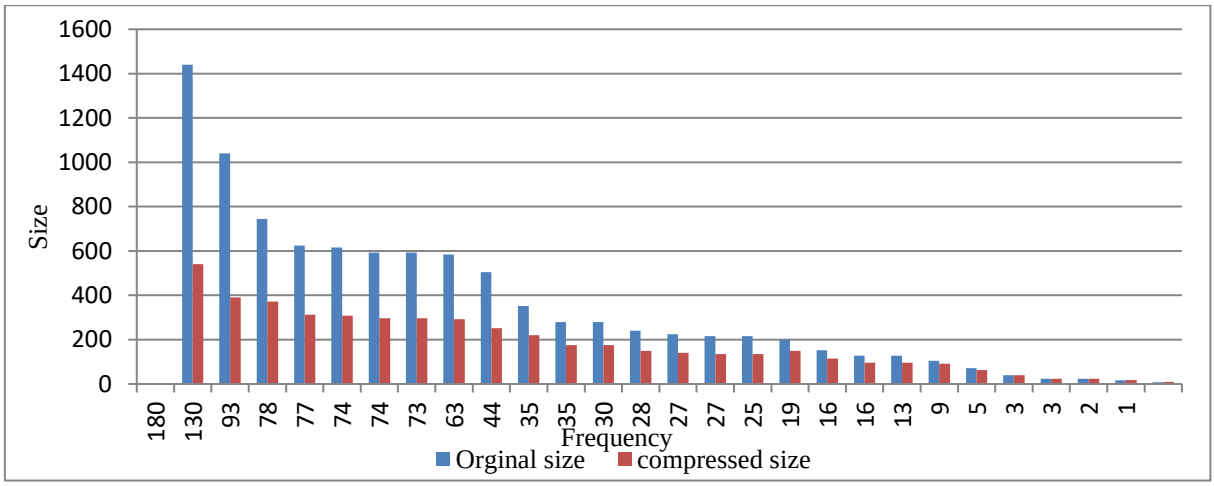

Fig.(3):- Comparison of compressed and uncompressed data sizes

As it could be summarized from Table 2, Huffman tree consists of 26 nodes, 27 leaves and 10 levels with different depths. The minimum depth was 0 while the maximum was 9-segments. All nodes and leaves are distributed in these 10 levels in such a way that high compression ratio could be achieved. Moreover, the difference in data sizes before and after compression is clear in Figure 3. The saving percentage in high frequency is increased by increasing the frequency of characters and vice versa.

Measuring Performance of Compression Process

There are various models for gauging the execution of the coding process, depending on the nature of the implementation [8]. Space efficiency will be the key consideration when judging the quality. It is challenging to gauge the general effectiveness of a coding method because compression behavior is dependent on the recurrence of characters in the source text. the degree of quality determined by the model and the source of the compressed data. Additionally, coding mode is determined by the type of compression used. By calculating the size of the uncompressed data (original data), the compressed size was calculated. This is the same as adding all of the table 1's letter frequencies. The process of determining the overall size for original then follows.

Total Frequencies = 180 +130 +93 +78 +77 +74 +74 +73 +63 +44 +35 +35 +30 +28 +27 +27 +25 +19 +16 +16 +13 +9 +5 +3 +3 +2 +1 = 1180

The size of original data is now multiplied by 8-bit ASCII to generate the representatives of the characters in Table 1 as follows:

Original Data Size = Total Frequency *ASCII Length (1)

Original Data Size: 9440 bits (1180 x 8).

For compression of the data in Table 1, the following is done:

Compressed Data Size = Codeword Bits * Frequency (2)

= 540+ 390+ 372+ 312+ 308+ 296+ 296+ 292+ 252+ 220+ 175+ 175+ 150+ 140+ 135+ 135+ 150+ 114+ 96+ 96+ 91+ 63+ 40+ 24+ 24+ 18+ 9.

Compressed Data Size= 4913 bits.

Next are the measurements that utilized to estimate the quality of lossless Huffman coding [9].

Compression Ratio: It is defined as the ratio of the compression file's and the initial file's information sizes.

$$\textbf{Compression Ratio} = \left(\frac{\text{Size after compression}}{\text{Size before compression}}\right) * 100\% \quad (3)$$

$$= \frac{4913}{9440}$$

= 52.04%

Compression Factor: it defines as the inverse of the data compression ratio. Represent the ratio between original file and the size of new compressed file.

$$\textbf{Compression Factor} = \left(\frac{\text{Size before compression}}{\text{Size after compression}}\right) \quad (4)$$

$$= \frac{9440}{4913}$$

= 1.92143

proportion of savings This is obtained by computing the source file's percentage shrinkage as shown below. It is commonly recognized as a percentage-based indicator of how effective a coding method performs

.

$$\textbf{Saving Percentage} = \left(\frac{\text{Size before compression} - \text{Size after compression}}{\text{Size before compression}}\right) 100\% \quad (5)$$

$$= \left(\frac{9440 - 4913}{9440}\right) 100\%$$

= 47.95%

A significant amount of data size can be saved with the Huffman design method, up to 47.95%.

VLC which is capable of achieving the shortest possible code word length for a particular symbol which can be greater than its entropy [10]. Using balancing Huffman binary tree and dictionary-based coding. The proposed technique based on the type and structure of the input data. Variable length codes for compressed size have been derived in a particular way based on the depth of binary tree proposed, entropy based coding, and non-balancing tree.

The computed chance of happening for any potential value of the source file has been used to determine how to get variable length codes for compressed size. Additionally, constructing a Huffman tree to generate codewords and evaluating the performance of the suggested approach for compression based on the kind and structure of the input data.

Simulation of Huffman Results

A Clk signal and 8-bit ASCII are used as encoder inputs to represent input data. 9-bit output data with variable length coding was produced by Huffman. As shown in Figure 2, the range of codeword lengths for representing encoder outputs ranged from 3-bit codewords for the highest frequency to 9-bit codewords for the lowest frequency. In contrast, the decoder module used a Clk signal with 9-bits of data as input to produce 8-bit ASCII to represent the decoder. on the basis of the validation waveform in Figure 4. The output of the encoder and decoder modules' Huffman approximated output is turned off when the e-d signal is set to a high level. In this instance, the frequency scaling state affects the encoder output signal.

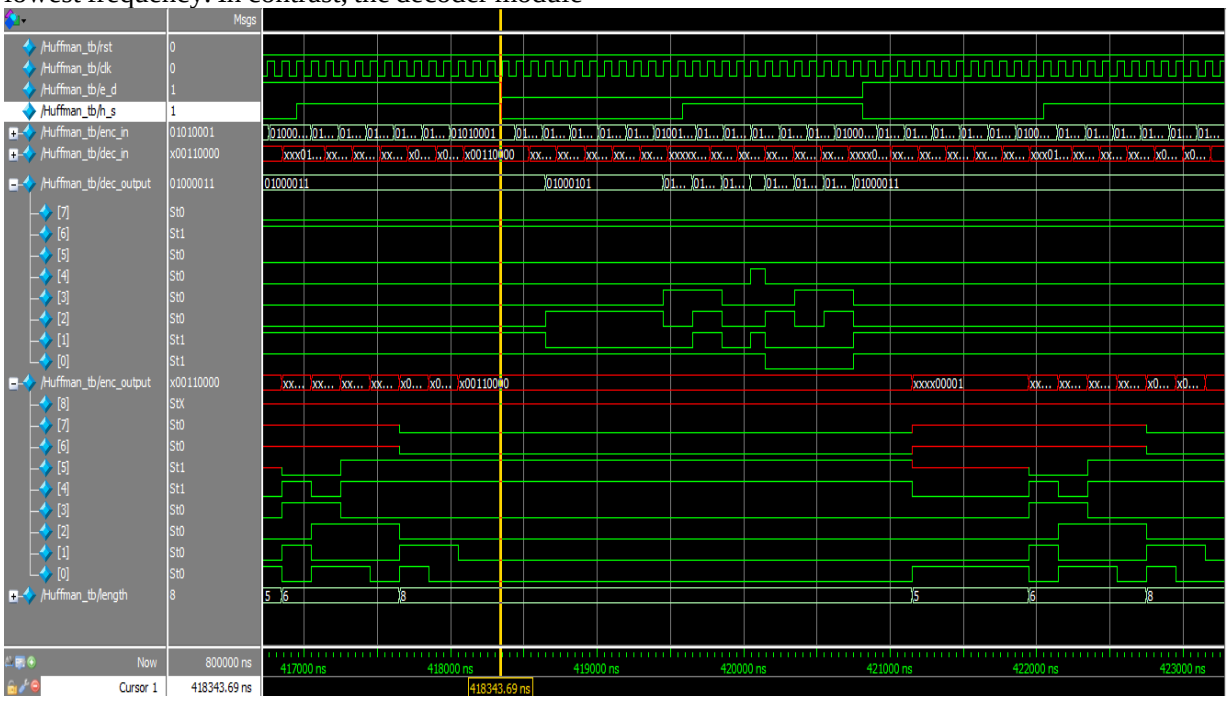

Fig.(4): -Huffman waveform validations

CONCLUSION

A Huffman coding for data compression has been proposed using binary trees for compressing and uncompressing text data without sacrificing any features in order to fully utilize lower size in data compression. The frequency of the input data determines the length of the codewords produced by the coding process, which produces 3-bit codewords for the highest frequencies across all series of data frequencies. Following that, increase the frequency till the codeword length reached 9 bits at the lowest data frequency. As can be seen, static Huffman coding was used to reduce data from 9440 bits to 4913 bits, which resulted in a reduction of 47.95% from the original size with a compression ratio of 52.04% and a compression factor of 1.9214. The length and Huffman algorithm were accomplished.

REFERENCES

M. Hameed., A. Khmag., F.Z. Rokhani., A. R. Ramli, "VLSI Implementation of Huffman Design Using FPGA with A Comprehensive Analysis of Power Restrictions", International Journal of Advanced Research in Computer Science and Software Engineering (IJARCSSE), 5(6): 49-54. (2015).

M. Hameed, "Low Power Approach for Implementation of Huffman Coding", ISBN-13: 978-620-2-31711-5 & EAN: 9786202317115 & Book language: English & Publishing house: Scholars' Press & Website: http://www.scholars-press.com & Number of pages: 56 & published on: 2018-09-20. (2018).

R. Stasinski and G. Ulacha, "Mixed Huffman codes for on-line and off-line applications," 2022 Data Compression Conference (DCC), Snowbird, UT, USA, 2022, pp. 483-483, doi: 10.1109/DCC52660.2022.00094.

A. Fruchtman, Y. Gross, S. T. Klein and D. Shapira, "Weighted Adaptive Huffman Coding," 2020 Data Compression Conference (DCC),

Snowbird, UT, USA, 2020, pp. 368-368, doi: 10.1109/DCC47342.2020.00059.

M. Hameed., A. Khmag., F.Z. Rokhani., A. R. Ramli. "A New Lossless Method of Huffman Coding for Text Data Compression and Decompression Process with FPGA Implementation", Paper presented at the International Conference of Computer Science Engineering and Technologi (COMSCET), (2016).

Chen, C.-Y., Pai, Y.-T., &Ruan, S.-J.(2006). *Low power Huffman coding for high performance data transmission*. Paper presented at the 2006 ICHIT'06, International Conference on Hybrid Information Technology: 71 – 77.

M. Hameed., H. Shakor., I. Razak, "Low Power Text Compression for Huffman Coding using Altera FPGA with Power Management Controller", Paper submitted at the 1st International Scientific Conference of Engineering Sciences - 3rd Scientific Conference of Engineering Science (ISCES). 978-1-5386-1498-3/ 18/31.00$©2018 IEE. . (2018).c

J. Yan and L. Wang, "The Novel Improving Algorithms on DRA Audio Entropy Coding," 2020 IEEE International Conference on Signal Processing, Communications and Computing (ICSPCC), Macau, China, 2020, pp. 1-4, doi: 10.1109/ICSPCC50002.2020.9259450.

R. Pal, "Speech Compression with Wavelet Transform and Huffman Coding," 2021 International Conference on Communication information and Computing Technology (ICCICT), Mumbai, India, 2021, pp. 1-4, doi: 10.1109/ICCICT50803.2021.9510116.

Y. Jianjun and L. ChunQuan, "Research and Improvement of Huffman Compression Method Based on Linear Linked Forest," 2021 International Conference on Education, Information Management and Service Science (EIMSS), Xi'an, China, 2021, pp. 495-499, doi: 10.1109/EIMSS53851.2021.00112.

M Hameed, F Zaman, AR Ramli, A Khmag." CMOS technology using clock gating techniques with tri-state buffer", Walailak Journal of Science and Technology (WJST), V(14)4, 2017. [1]