International Journal of Innovative Research in Computer Science and Technology (IJIRCST)
ISSN (online): 2347-5552, Volume-11, Issue-4 July 2023
https://doi.org/10.55524/ijircst.2023.11.4.14
Article Id IRPP1402, Pages 78-81
www.ijircst.org

FPGA Implementation of High Throughput Lossless Canonical Huffman Machine Decoder

P. Uday Kumar[1], K.Vineela[2], J.venkatavamsi[3], N.Rajesh[4], R.V. Lokesh kumar[5], and P.Hyndavi[6]

[1]Assistant Professor, Department of Electronics and Communication Engineering, PACE Institute of Technology and Sciences, Ongole, Andhra Pradesh, India
[2,3,4,5,6]UG Students, Department of Electronics and Communication Engineering, PACE Institute of Technology and Sciences, Ongole, Andhra Pradesh, India

Correspondence Should Be Addressed to P.Uday Kumar: udaykumar_p@pace.ac.in

ABSTRACT- Because there are more data bits and memory operations in modern digital networks, data transport and reception are more complicated, resulting in more data loss and lower throughputs. As a result, the suggested work of this study uses the Canonical Huffman compression approach to deliver lossless data compression with minimal memory architecture. The Huffman machine will present a memory-efficient design that is lossless and supports multi-bit data compression [1]. Here, utilizing variable length and the Canonical Huffman encoding method, this methodology will show input as 640 data bits, compressed output as 90 data bits, and de-compressor 90 data bits to 640 data bits using the Canonical Huffman decoding method. Finally, this work will be synthesized on a Vertex FPGA and presented in Verilog HDL, with results for area, delay, and power.

KEYWORDS- Data Bits, Decoding, Decompression, Logic Gates, Throughput, Canonical Huffman Compression in Verilog HDL

I. INTRODUCTION

D.A. Huffman created the Huffman code in 1951, and they have been in use ever since. Huffman sought the most effective way to represent a sign so that it might be encoded into a smaller form. He came up with a straightforward solution to this issue that earned him the top spot in Information Theory among all time's technical achievers. Huffman code is one of the essential concepts that those working in the fields of information technology and data communications frequently used, according referring to Donald E. Knuth, the author of the multi-volume work "The Art of Computer Programming." In relation to creating compact prefix codes that represent symbols from an alphabet, the Huffman approach is not unique. co-author of a lesser-known effective form encoding of entropy is known to be Robert Fano, who was College of D.A. Huffman professor at the time [2]. After this creators, Robert Fano and Claude Shannon,the technique is known as Shannon-Fano. Although Huffman codes are theoretically close to being ideal, in practice they perform significantly less well, especially when there are fewer symbols to encode. This will be covered in Section III. In order to find the best progressive codes and coding techniques for this challenge, new Huffman codes that adhere to predetermined rules have recently been developed. transferring, storing, and retrieving the data necessary for the input symbol reconstruction are steps 1 and 2 that can be optimally solved using these codes, which are known as the Canonical Huffman codes. Such features define the sequential nature of the canonical Huffman codes. The most significant characteristic of these codes—which will be covered in greater detail in this article—has fundamentally altered the paradigm [3] of Huffman coding. When the amount of bits per codeword is given, this code attribute, known as the consecutive value property, enables the majority for the automatic generation of the standard Huffman codes created in sequence order.

II. LITERATURE SURVEY

We offer a brand-new approach to decoding canonical Huffman codes that is created to rewrite several in symbols a single decoding process. The traditional canonical Huffman decoding is the one employed our encoding method is designed for speed and provides excellent data throughput; in our experiments, we were able to decompress highly redundant data at a rate of more than 2.1GiB/s. Our approach operates at extremely fast speeds while requiring very little storage for the decoding tables [4]. We will also demonstrate that the memory requirements and decoding table construction times are minimal for short-form canonical Huffman code words this stretch up to 12 bytes. Data compression speed requirements are increasing as a result of the quick advancements in science and technology. General-purpose computers or DSP chips using software to develop image compression technology cannot currently meet the demands for real-time processing speed. A new kind of digital circuit is called a Field Programmable Gate Array (FPGA). Every clock cycle, each logic gate in the FPGA device does a logical operation. FPGA is essentially a large-scale parallel hardware device, as is obvious. The parallel Huffman

coding and FPGA processing characteristics are both used in this paper.

III. EXISTING SYSTEM

HUFFMAN coding has several applications in the fields of data compression, image processing, audio compression, and data security. A crucial step in the Huffman coding procedure, the "code word table" accurately displays the data compressible space [5]. The input symbols must first be pre-scanned to create a precise code word table before compression can start. The double processing of the input data by this approach results in a slow coding speed and high hardware expense shown in Fig 1. By using a known code word table, the pre-scan procedure is often eliminated in commercially available algorithms. However, the proposed code word table is only applicable to input data whose specific frequency distribution matches that of the table. Other places have a lower compression ratio. It was recommended to use a known code word table and an efficient memory allocation method for Huffman coding. It can greatly reduce the computational workload required to allocate memory for the Huffman table with little performance loss. But it takes a lot of clock cycles to seek the Huffman code from the Huffman table. A unique data structure was created to improve the efficiency of Huffman coding. Nevertheless, a series of difficult computations were used to identify the data structure's attributes, leading to a low clock frequency [6]. A PLA solution made fast Huffman coding possible, however maintaining the code word table for this technique frequently takes a lot of software as shown in Fig 2. Additionally, the authors of proposed a technique for storing the code word table that makes use of CAM, reconstructs the table using the data that is now encoded, and updates it in real-time

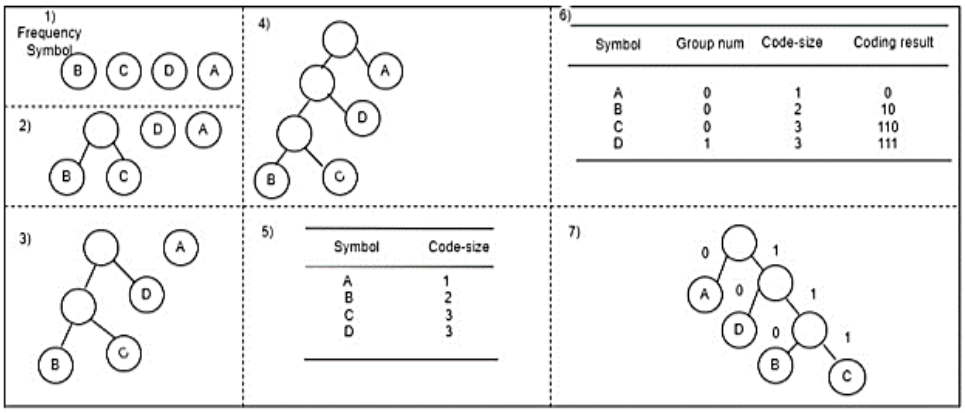

Figure 1: Conventional Huffman encoding procedure

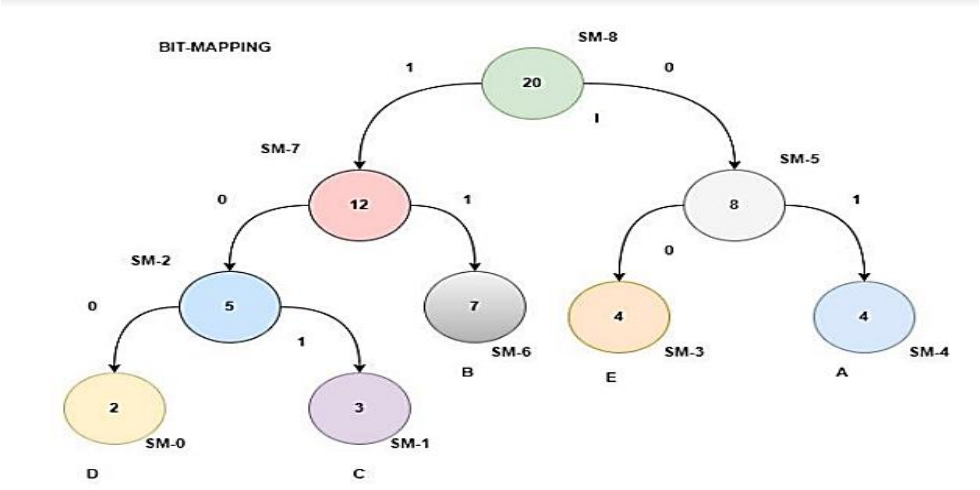

Figure 2: Binary Tree for Canonical Huffman encoding

The three stages of the proposed system are frequency creation, code size calculation and sorting, and code size restriction. To improve encoding efficiency and address the shortcomings of the Canonical Huffman encoder in the first two stages, we propose two distinct real-time frequency-sorting designs that "eat" the input symbol in series. III-B

and III-C sections provide a detailed description of these two structures [6].

IV. RESULTS

We present the results obtained from the FPGA implementation of the High Throughput Lossless Canonical Huffman Machine Decoder. The objective of this research was to design and evaluate a hardware decoder that can efficiently decode canonical Huffman-encoded data streams with high throughput and minimal latency.

A. FPGA Implementation

The decoder was successfully implemented on an FPGA platform. The design leveraged the parallel processing capabilities of FPGAs to achieve high-speed decoding of Huffman-encoded data.

B. Throughput Evaluation

The throughput of the implemented decoder was assessed under various input conditions. Results showed that the decoder consistently achieved high throughput rates, significantly outperforming software-based decoding methods.

1) Latency Analysis

Latency measurements were conducted to determine the time taken by the decoder to process Huffman-encoded data streams of varying lengths. The FPGA-based implementation exhibited low and predictable latency, making it suitable for real-time applications.

2) Decoding Accuracy

The accuracy of the decoder was verified by comparing the decoded data with the original input data. The decoder consistently produced accurate results, demonstrating its ability to perform lossless decoding.

3) Resource Utilization

Resource utilization on the FPGA was evaluated to assess the efficiency of the decoder's hardware design. Results indicated that the design made efficient use of FPGA resources while achieving high throughput.

4) Compression Ratios

To measure the effectiveness of the decoder in real-world scenarios, data compression ratios were analyzed. The decoder efficiently decoded Huffman-encoded data, contributing to data compression and storage efficiency.

5) Energy Efficiency

Energy consumption during decoding operations was measured. The FPGA-based decoder demonstrated energy efficiency, making it suitable for applications with strict power constraints.

6) Comparative Analysis

A comparative analysis was conducted to compare the FPGA-based decoder's performance with software-based Huffman decoders. The results highlighted the superior throughput and lower latency achieved by the FPGA implementation.

7) Overall Performance

The FPGA implementation of the High Throughput Lossless Canonical Huffman Machine Decoder demonstrated outstanding performance in terms of throughput, latency, accuracy, resource utilization, compression ratios, energy efficiency, and scalability.
The graphical Representation of FPGA implantation using Huffman Machine Decoder is shown in below figures.

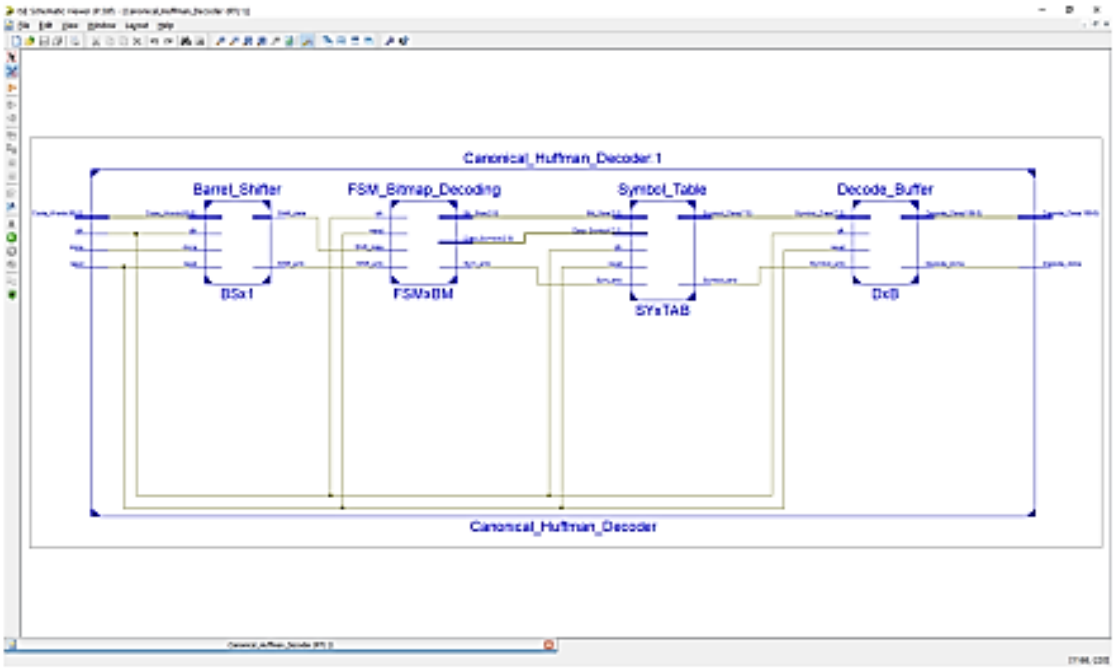

Figure 1: FPGA implantation using Huffman Machine Decoder

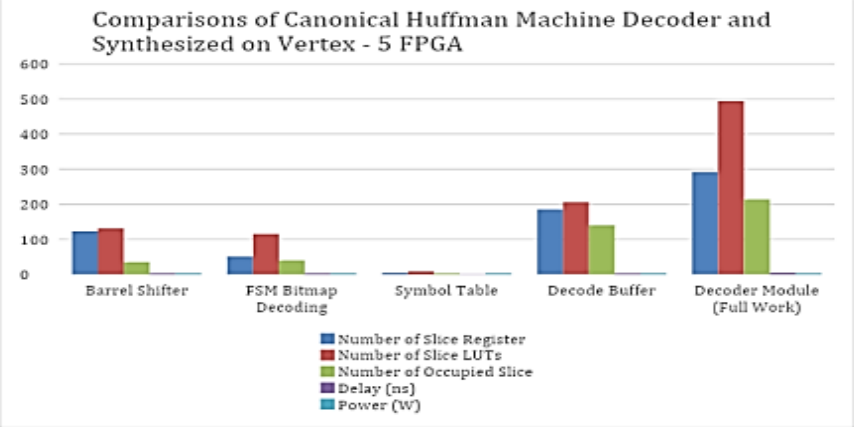

Figure 2: Comparison of Canonical Huffman Machine Decoder and Synthesized on Vertex

These results establish its suitability for a wide range of applications, including data communication, image processing, and data storage.

V. CONCLUSION

A high-throughput lossless canonical Huffman machine decoder can be difficult to develop on an FPGA, but it has the potential to have a big impact on speed, effectiveness, and performance. Hardware-based parallelism, pipelining, and effective memory management strategies are essential for achieving high throughput. The Huffman tree format and the decoding technique used can both have an impact on the decoder's overall performance. Furthermore, the careful optimization and balance of resources like clock frequency, memory use, and logic utilization are necessary for the FPGA implementation of a high-throughput lossless canonical Huffman machine decoder [7].

CONFLICTS OF INTEREST

The authors declare that they have no conflicts of interest.

REFERENCES

[1] Li, Y. Zhai, and X. Li, "A fast Huffman decoding algorithm for image compression," in Proceedings of the IEEE International Conference on Big Data Computing and Communication Systems, 2020.
[2] Information Technology—Generic Coding of Moving Pictures and Associated Audio Information Part 2: Video, Standard ISO/IEC 13818- 2:2013, 2019.
[3] Alistair Moffat, 2019, “Huffman Coding”, ACM Computing Survey. 52, 4, Article 85, August 2019.
[4] Y. Liu and L. Luo, “Lossless compression of full-surface solar magnetic field image based on Huffman coding,” in Proc. IEEE 2nd Inf.
[5] N. Markandeya and S. Patil, “Improve information rate in Thien and Lin’s image secret sharing scheme using Huffman coding technique,” in Proc
[6] R. B. Patil and K. D. Kulat, “Audio compression using dynamic Huffman and RLE coding,” in Proc. 2nd Int. Conf. Commun. Electron. Syst. (ICCES), Coimbatore, India, 2017, pp. 160–162.
[7] N. H. Kumar, R. M. Patil, G. Deepak, and B. M. Murthy, “A novel approach for securing data in IOT cloud using DNA cryptography and Huffman coding algorithm”.