International Journal of Reconfigurable and Embedded Systems (IJRES)
Vol. 15, No. 1, March 2026, pp. 68~77
ISSN: 2089-4864, DOI: 10.11591/ijres.v15.i1.pp68-77

FPGA implementation of high-performance Huffman encoder for image processing applications

Masood Ahmad Mahammad[1], Appala Raju Uppala[2], Shaik Mazhar Hussain[3], Anusha Marouthu[4]
[1]Department of Electrical, Electronics, and Communication Engineering, GITAM School of Technology, GITAM Deemed to be University, Hyderabad, India
[2]Department of Electronics and Communication Engineering, Geethanjali College of Engineering and Technology, Hyderabad, India
[3]Department of Electronics and Communication Engineering, Malla Reddy (MR) (Deemed to be University), Hyderabad, India
[4]Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, Vijayawada, India

Article Info

Article history:

Received Dec 4, 2024
Revised Dec 22, 2025
Accepted Jan 11, 2026

Keywords:

Compression
Delay
High-performance
Huffman encoder
Parallel processing
Parallelism

ABSTRACT

An optimized Huffman encoder is essential in all applications where it is necessary to achieve the best performance, such as audio coding, data encryption, data compression, and image processing applications. This article presents a space-optimized encoding scheme to maximize performance and minimize latency in Dual Huffman encoding. The proposed approach employs dynamic tree selection using Dual Huffman encoding. A Dual Huffman code with dynamic tree selection can be run in parallel to support high-throughput applications. The resulting design optimally creates the Huffman dual encoding. This codeword table is based on a dynamic tree generation and selection algorithm, leading to a faster encoding process with lower latency. The architecture was developed using Vivado Xilinx 2023.2 and tested on three different field programmable gate array (FPGA) platforms (Zynq 7045, Zynq 7100, and Kria KV260 AI Vision board). A performance comparison between devices demonstrates that the Kria KV260 had the lowest latency (100 ns), as opposed to the Zync 7045 and Zynq 7100, which had latencies of 200 ns and 150 ns, respectively. These results elucidate the scalability of the architecture and its suitability for real-time image compression. When implemented on the Kria KV260, the dynamic tree selection-based Dual Huffman encoder is capable of fast, parallel image compression. The compression makes it a good candidate for advanced FPGA-based image processing systems with internet of things (IoT) applications.

Corresponding Author:

Masood Ahmad Mahammad
Department of Electrical, Electronics, and Communication Engineering, GITAM School of Technology
GITAM Deemed to be University
Hyderabad, Telangana, India
Email: mmahamma@gitam.edu

1. INTRODUCTION

Data compression plays a crucial role in efficiently storing and transmitting information. This work focuses on implementing a high-performance Dual Huffman encoder. The Dual Huffman encoding compression algorithm utilizes symbol frequencies to assign shorter codes to more frequent symbols. This work aimed to provide a high-performance Dual Huffman coding in Verilog hardware description language (HDL). This dual Huffman encoding is used to compress the image signal [1]. The Dual Huffman encoder offers good compression performance, minimal loss of image quality [2], and ease of application. The Joint Photographic Experts' Group (JPEG) encoding is primarily accomplished using two methods. The CPU-

based software or hardware method uses a field programmable gate array (FPGA). FPGAs are also known as FPGAs or application-specific integrated circuits (ASICs). The latter performs exceptionally well in terms of system power consumption, integration, and processing speed. In certain situations where high picture processing speed is not required, the JPEG encoder is constructed using a standard pipeline architecture to enhance image compression efficiency [3]. The JPEG encoder's pipeline structure comprises color space conversion, down sampling, two-dimensional discrete cosine transforms (DCT), zigzag scanning, quantization coding, DC coefficient differential pulse code modulation, and Huffman encoding. The pipelined JPEG encoding increases the utilization rate. Data blockage is easily caused since the processing of each pipeline stage depends on the completion of the previous stage's processing. A lossless method of data compression is Dual Huffman encoding [4], [5]. Fewer bits represent the most common symbols, determined by the probability of such symbols in the known data set.

There are two types of Huffman encoding: static look-up tables and dynamic encoding. The latter receiving port is comparatively basic but less adaptable than the former due to its sophisticated hardware construction. Therefore, the length of each codeword in the Huffman encoding is not equal, making it a variable-length coding. Realizing the JPEG compression [6] pipeline is challenging when there are variations in data complexity, since the encoding clock cycle is not constant. A multi-pipeline design is needed to increase the efficiency of picture compression [7] and optimize the processing of the DCT module. Parallel coding for Huffman encoding is more efficient than conventional Huffman coding. Hence, a Dual Huffman encoder is used to improve the performance of Huffman encoding.

The Huffman tree encoding is implemented using a bitmap and static random-access memory. Yet, there will still be blocking in these Huffman encodings. This work presents a fast Huffman encoding algorithm using a double-byte splicing output Huffman encoding architecture. Dual Huffman encoding units to achieve non-blocking in the JPEG pipeline. It may guarantee that the Minimum Coding Unit is encoded in 64 clock cycles, the same amount as other JPEG compression modules, even though it will increase the number of logic units. The JPEG encoder's non-blocking encoding pipeline architecture boosts JPEG [8] compression effectiveness. The suggested architecture is implemented on the Digilent Zynq 7045, Zynq7100, and Xilinx Kria KV260 AI starter kit using Xilinx Vivado 2023.2. The benefits of the high-performance Dual Huffman encoding [9] are then confirmed by comparing the resource usage and clock cycles. Although parallel Huffman encoding with GPUs and FPGA platforms has been studied, there is a lack of research on new parallelization approaches that leverage newly developed architectures, such as multicore CPUs and specialized AI accelerators like tensor processing units (TPUs). Research on parallel Huffman encoding [10] optimization in these various computer contexts may lead to notable gains in efficiency and speed.

The structure of this paper is as follows. Section 2 presents the research background of the Huffman encoding method, along with a method for minimizing the number of cycles. Section 3 presents an experimental method of Huffman encoding on an FPGA. Section 4 presents the experimental results section, which compares the results with those of comparable previous works to confirm the benefits of Huffman encoding on FPGA systems. Section 5 discusses the results of the work. Section 6 deals with the conclusion of the work.

2. RESEARCH BACKGROUND

One of the known methods for data compression is Huffman coding [11], which assigns a variable-length code to each symbol, taking into account only their occurrence probabilities in the sequence used. The method was later developed by David A. Huffman in 1952 as a way to minimize the average length of the encoded data by assigning longer codes to less common symbols and shorter codes to more common symbols. This approach provides better data representation and reduces the total amount of encoded data compared to fixed-length encoding methods. Huffman encoding is essentially performed by reading the input data, which must also be searched to determine its frequency of occurrence. It is essential to understand how frequently each symbol appears to assign the most optimal code, and this data analysis precisely accomplishes that. A well-known implementation of Huffman coding relies on algorithms that introduce basic ideas into symbol frequency analysis in data compression [12]. Constructing the Huffman Tree. To create a Huffman tree after finding the frequencies of all symbols. Such a tree is constructed using a greedy method by consecutively merging two least frequent symbols into one node until all symbols are used. By making more frequent symbols closer to the tree's root, shorter codes and higher compression rates are achieved. Because of its position in the tree, each symbol is assigned a unique binary code when the Huffman tree is being built.

A string of 0s and 1s is generated when the Huffman tree is traversed and read from the roots to each symbol (usually, left branches are given the number 0; while right branches get the value of 1). The assigned codes are represented by the path of symbols traversed from the root to each leaf node in the tree. Now suppose the values of symbols A, B, C, D, and E occur with frequencies 5, 9, 12, 13, and 16,

respectively. Initially, these frequencies are used to construct a single Huffman tree from the symbols with the lowest frequencies. False (for E) and '10' (for C) are the valid Huffman codes for transitioning from each possible location in the tree. This encoding is highly efficient because Huffman codes can be designed to reflect the distribution of symbols in the input information accurately.

Unlike fixed-length codes, which produce longer average code lengths, in any encoding scheme, shorter codes represent the more likely symbols. Thanks to this codelength diversity, the Huffman coding can yield a better compression ratio for data with predictable symbol distributions. Several strategies are employed during Huffman encoding to ensure efficiency and speed. Techniques such as dynamic Huffman coding further improve compression efficiency by dynamically changing the code assignments when symbol occurrences are modified during encoding, allowing better performance of compressing data with "changing" statistics. FPGA-based and delay-locked loop-optimized hardware parallel Huffman decoding has been developed, making it applicable to both real-time applications and high-throughput situations. These realizations leverage parallelism and optimized data surfaces to enhance encoding speeds. Huffman coding is used in various applications, including embedded systems, network protocols, and multimedia processing. It can be used to save space on an archive, minimize storage of multimedia files, textual documents, or communication protocols, and also compress data for delivery without compromising quality. Fast Huffman coding [13] has received much attention. The searching and sorting algorithms used for schemes operating on bit planes are identical to those of currently available hardware. As demonstrated by various studies and implementations, Huffman coding [14], a fundamental method of lossless data compression that assigns shorter codes to more frequent symbols, has seen significant advancements in both Speed and efficiency. To achieve an efficient implementation of Huffman coding, several techniques, such as parallelization methods using shared memory structures, are required [15]. These techniques leverage parallel computing resources and multicore processors to accelerate the encoding process, resulting in improved Throughput and lower latency, which is suitable for live applications.

Additionally, technical enhancements to the classical Huffman coding [16] have been optimized. Dynamic Huffman coding techniques have been widely studied in the literature [17], where the code assignments are updated dynamically according to the new frequency of a symbol observed while processing an input stream, resulting in high encoding efficiency. These technologies maximize the compression ratio and accelerate encoding, which is essential for high-speed operations, such as data storage.

Huffman encoding has been greatly improved through the use of hardware acceleration, especially on FPGA devices [18]. Hardware-based FPGA systems can significantly reduce encoding times compared to traditional software-based approaches, thanks to their hardware resources and inherent parallelism. This hardware-based solution, with parallel architectures and efficient data paths, achieves high Throughput and low latency in the encoding process [19]. Parallelization methodologies have been instrumental in accelerating Huffman encoding [20] in multimedia and data-centric applications. The parallelization of encoding using GPUs and SMP has been explored in [2], [9], yielding significant gains in situations where minutes are needed to store massive amounts of data. These advances underscore how the use of hardware, combined with fast parallel algorithms, can lead to speedy Huffman-scheme implementations for significantly different computer scenarios. Practical solutions verify theoretical breakthroughs in Huffman coding. For example, the efficient design and synthesis of Huffman encoders [21]-[23] have been demonstrated in implementations using Cadence RTL compiler and system on chip (SoC) encounter [24]. These are the first results to confirm the use of state-of-the-art algorithms in practical scenarios and the feasibility of fast encoding algorithms. An interplay of algorithmic innovations has driven the development of fast Huffman encoding, parallel processing techniques, and dedicated hardware implementations [25], [26]. Such a multi-faceted approach has improved both coding speed [27] and effectiveness, and extended its applications to various fields, including multimedia processing, embedded systems, and high-performance computing environments. Fast Huffman encoding [28], [29] remains a vital perspective for achieving effective data compression with lower computational cost and the highest Throughput, as future work in compression techniques progresses.

3. EXPERIMENTAL METHOD

The Dual Huffman encoder, also known as tandem coding, is a high-level type of data compression that enhances compression efficiency for data with changing causes, utilizing two distinct Huffman trees. The Dual Huffman encoding method is illustrated in Figure 1. It is better than standard Huffman encoding. Symbols are encoded into variable-length codes by classic Huffman coding, where frequently occurring symbols have shorter codes. By contrast, the Dual Huffman Encoder generates a separate Huffman tree for each of the two split groups, and its properties differ from one another. The dual-tree structure enables a more accurate estimation of the symbol frequency distribution for each subset, resulting in a better compression

ratio and lower data entropy. The data must be partitioned, the symbol frequency in each set analyzed, and Huffman trees constructed; these trees then assign codes, and the data are encoded accordingly. This approach has multiple positive aspects: it may achieve greater compression due to the precision of frequency distribution modeling and lower entropy (and thus correct decoding). Being parameterized, it is suitable for various tasks, such as data transfer, storage, and multimedia compression, among others.

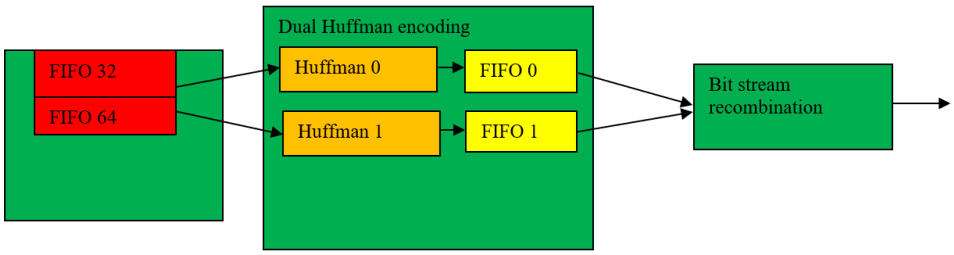

Figure 1. The Dual Huffman encoding architecture

The Dual Huffman Encoder can effectively process the two trees in parallel when used on sophisticated FPGA platforms, such as the Zynq-7045, Zynq-7100, and Kria. These platforms have large programmable logic resources and high operating frequencies. While the Kria board, running at 2.2 GHz, can achieve low processing times and high Throughput, making it perfect for real-time applications, the Zynq-7045 and Zynq-7100, with their large LUT counts and processing capabilities, can manage complex processes with minimal delay. All things considered, the Dual Huffman Encoder is a valuable tool for applications that require low redundancy and high-speed data processing. It is an example of a complex compression approach that leverages contemporary FPGA capabilities, offering improved compression efficiency and Speed.

3.1. Dynamic tree selection based Dual Huffman encoding:

Figure 2 illustrates the dynamic tree selection-based Dual Huffman encoding process. The input image will be split into two parts based on even and odd pixel distributions. The dynamic Huffman tree unit consists of a symbol frequency counter, a Huffman tree builder, and an encoder. The symbol frequency counter is responsible for counting the number of symbols. Huffman tree builder response for the Huffman tree based on the current frequency data. Figure 3 shows the Huffman encoding algorithm.

A high-speed, dynamic tree selection-based Dual Huffman encoder is implemented in an FPGA using Xilinx Vivado 2023.2 tools. Our research focuses on achieving high compression ratios and effective parallel processing using a high-speed, dynamic tree selection-based Dual Huffman encoder. Dynamic tree selection-based Dual Huffman encoding utilizes parallel processing to achieve high performance. The encoder generates the Huffman codewords using the current tree. Bit stream merger combines the output of the encoders. The output is the compressed image bit stream. A dynamic tree selection-based Dual Huffman encoding algorithm is implemented on the diligent FPGA Zynq7045, Zynq7100, and Xilinx Kria KV260 AI vision board. The speed of the dynamic tree selection-based Dual Huffman encoder is compared among the FPGA boards.

The overview of the different FPGAs being compared is as follows:

a. Zynq-7020 FPGA board: the Zynq-7020 from the Xilinx Zynq-7000 family is a highly integrated embedded platform combining 28 nm programmable logic with a dual-core ARM Cortex-A9 processor operating up to 866 MHz. It supports up to 1 GB DDR3/DDR3L memory, enabling memory-intensive applications. The device features a range of rich peripherals, including USB, Gigabit Ethernet, SD/SDIO, UART, I2C, and SPI. With approximately 85K logic cells, 220 DSP slices, and 560 KB block RAM, it supports custom logic and signal processing tasks. Tight coupling between the ARM processor and programmable logic enables efficient hardware–software co-design. Due to its flexibility and performance, the Zynq-7020 is widely used in industrial control, automotive infotainment, medical imaging, and consumer electronics.
b. Virtex UltraScale FPGA board: Virtex UltraScale FPGAs deliver high performance and power efficiency using advanced fabrication technology. Supporting up to 2.5 million logic cells, they provide exceptional scalability for large and complex designs. Integrated UltraRAM and high bandwidth memory (HBM) enable fast on-chip data access, while multi-gigabit transceivers support data rates up to 32.75 Gbps. These features make UltraScale FPGAs ideal for data centers, telecommunications (including 5G), and aerospace and defense applications requiring high throughput, low latency, and massive parallel processing.

c. Zynq-7035 board: the Zynq-7035 is a versatile SoC integrating a dual-core ARM Cortex-A9 processor with 28 nm programmable logic. It offers approximately 125,000 logic cells, 240 DSP slices, and 4.9 MB block RAM, supporting digital signal processing and custom hardware acceleration. The device supports up to 1 GB DDR3/DDR3L memory and includes interfaces such as Gigabit Ethernet, USB, and SD/SDIO. The close integration of the processing system and FPGA fabric makes it suitable for industrial automation, automotive systems, medical devices, and consumer electronics, where performance and flexibility are critical.
d. Zynq-7045 board: the Zynq-7045 consists of a dual-core ARM Cortex-A9 processor clocked up to 1 GHz with large FPGA resources. It offers more than 350,000 logic cells, 2,200+ DSP slices, and 19.1 MB of block RAM for handling high-throughput data jobs. High-speed interfaces. All of the other interfaces with round-trip latencies less than 500 Mbps listed in Table 1 are high-speed links with lower overheads, such as PCIe, Ethernet, or USB, which are ideal for data-intensive communication (e.g., automotive, industrial automation, and video processing). Powerful tooling continues to streamline development and accelerate deployment.
e. Zynq-7100 FPGA board: it is the top-of-the-line member of the Zynq-7000 family and combines a dual-core ARM Cortex-A9 up to 1 GHz with abundant FPGA resources that consist of more than 444K logic cells, 2,020 DSP slices, and provides 2626.5 Mb block RAM. High-speed serial transceivers and interfaces, such as USB, Gigabit Ethernet, and PCIe, help improve system-level connectivity. It is well-suited for demanding applications, including ADAS, aerospace and defense systems, high-performance computing (HPC), and advanced industrial automation.
f. Kria KV260 Vision AI starter board: the Kria KV260 Vision AI Starter Kit is an edge-AI development platform built around the Kria K26 system-on-module. It includes quad-core ARM Cortex-A53 and dual-core Cortex-R5F processors, along with programmable logic. The board features USB, HDMI, DisplayPort, Ethernet, and camera interfaces, as well as pre-installed software and AI models. Integration with AI frameworks, such as TensorFlow and the Vitis AI environment, increases ease of use. Ideal for edge vision applications in smart cities, industrial internet of things (IoT), robotics, and automated retail, the KV260 provides a plug-and-play solution with robust ecosystem support.

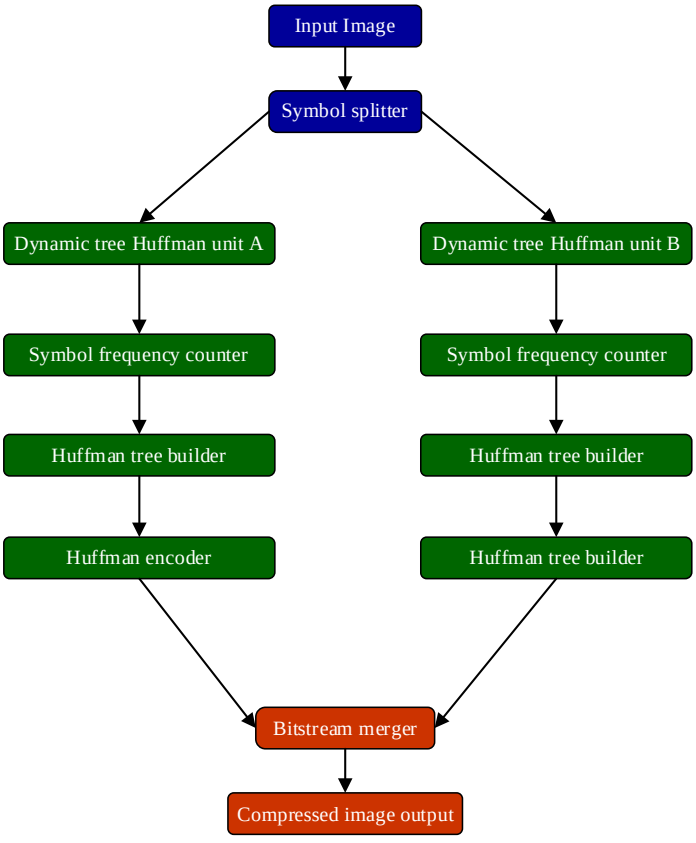

Figure 2. The proposed dynamic tree selection

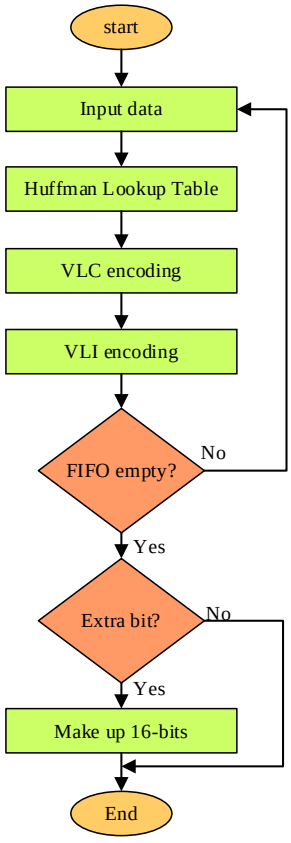

Figure 3. Huffman encoding flow chart based on Dual Huffman encoding

Table 1. Compare Huffman encoding in different articles (listed in references)

Related previous works in comparison	Matai *et al.* [27]	Elaskary *et al.* [28]	Choi *et al.* [29]	Proposed work		
Name of FPGA board	Zynq 7020	Virtex ultrascale	Zynq 7035	Zynq 7045	Zynq 7100	Kria KV 260
Operating Frequency (Hz)	173M	250 M	200M	800 MHz	1 G	2.2 G
No LUTs were used	761	42052	624	410	400	360
Number of cycles required	448	256	64	64	64	64
Time (ns)	2590	1024	320	80	64	29
Throughput (Mbit/s)	2372.2	2000	6400	8000	10000	22000

4. EXPERIMENTAL RESULTS

Verilog code of a dynamic tree selection-based Dual Huffman encoder was designed and simulated using Xilinx Vivado 2023.2. The image compression results on different FPGA boards are presented in Table 1. The operating frequencies of these FPGA boards are shown in Figure 4. The clock cycle time frequency of dynamic tree selection-based Huffman coding in FPGA boards is presented in Figure 5. The Delay time in dynamic tree selection [8] based on Huffman coding is presented in FPGA Figure 6. The Throughput performance of the Huffman encoder for a dynamic selection tree-based transmission by FPGA boards is presented in Figure 7. It can be observed from Table 1 that the FPGA implementation of a Huffman encoder is the best choice for hardware. Thus, the FPGA's high-speed capability can be utilized to construct a high-performance Huffman encoder for data compression. The Huffman encoder is used for image encoding. The high speed of compression is very useful for IoT nodes. One can perform image processing, compression, and artificial intelligence-related algorithms using an FPGA. Therefore, the FPGA realization of a dynamic tree selection-based Huffman encoder is beneficial for multimedia and image processing applications.

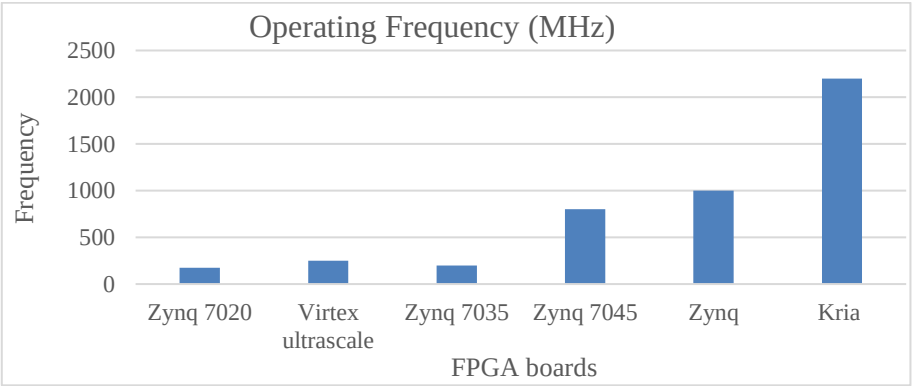

Figure 4. Operating frequency of various FPGA boards

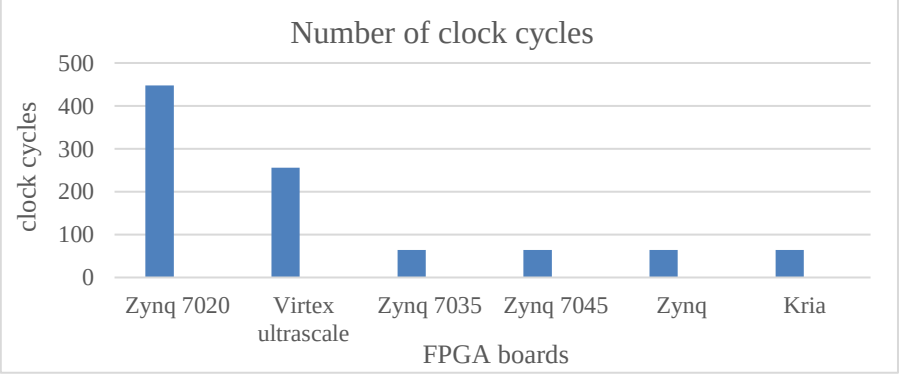

Figure 5. Number of clock cycles used in the dynamic tree selection-based Huffman encoder by FPGA boards

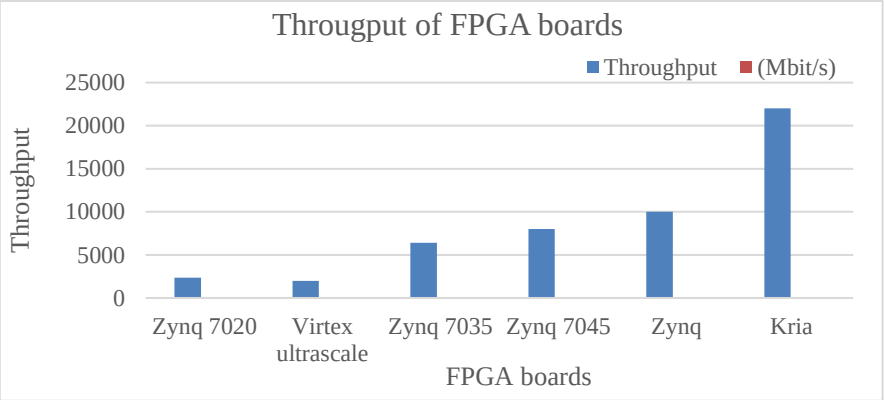

Figure 6. Delay time in the dynamic tree selection-based Huffman encoder by FPGA boards

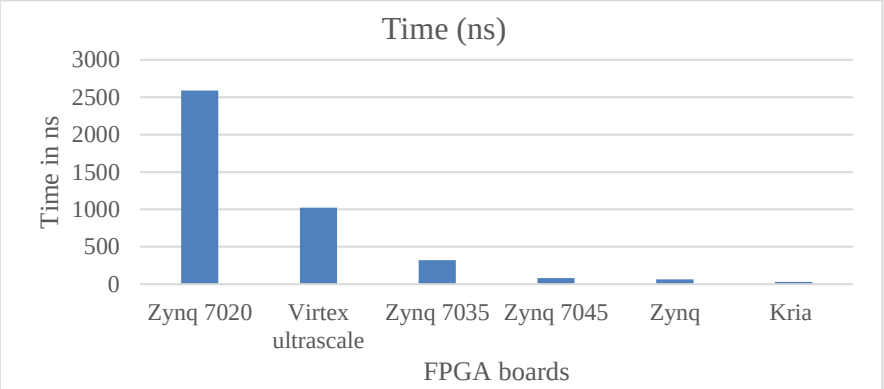

Figure 7. Throughput of the dynamic tree selection-based Huffman encoder offered by FPGA boards

5. DISCUSSIONS ON RESULTS

Comparison between the proposed work and previous works on different FPGA boards is given. The comparison involves Zynq 7020, Virtex Ultrascale, Zynq 7035, Zynq 7045, Zynq 7100, and Kria. The communicating parameters are operating frequency, the number of utilized LUTs, the latency in detecting a burst, which is described in terms of the number of cycles and the time it takes to be processed, and Throughput. In terms of operating frequency, Matai and coworkers employed Zynq 7020 at 173 MHz, and Elaskary R.M., et al. utilized the Virtex Ultrascale running at 250 MHz. Choi J et al. applied the 200 MHz Zynq 7035. In contrast, the present work utilizes high-end FPGAs—Zynq 7045 and Zynq 7100 (operating at frequencies of 800 MHz and 1 GHz), with the Kria board running at 2.2 GHz—providing enhanced processing potential. The use of LUTs indicates the complexity and efficiency of the design. The Virtex Ultrascale utilised a total of 42,052 LUTs, much more than the Zynq 7020 (761 LUTs) and Zynq 7035 (624 LUTs). In our work, we are using Zynq 7045, Zynq 7100, and Kria boards, which utilize only 410 LUTs, 400 LUTs, and 360 LUTs, respectively. This demonstrates that the optimum utilization of logic was achieved, even though the operating frequency is increased. The number of processing rounds also demonstrates performance efficiency. The Zynq 7020 took 448 cycles, the Virtex Ultrascale achieved it in 256 cycles, and the Zynq 7035 required just 64 cycles. It can be seen that all FPGAs in the proposed study required 64 cycles, which confirms better processing efficiency. The processing time (in nanoseconds) also exhibits a remarkable enhancement in the present work. The two board comparison showed a 2,590 ns processing time in the Zynq 7020 and 1,024 ns for the Virtex Ultrascale. The proposed solution, optimized for Zynq, achieves only 360 ns. In comparison, data was processed by the Zynq 7100, Zynq 7045, and Kria boards in 80 ns, 64 ns, and 29 ns, respectively, owing to their higher clock speeds and efficient designs. Throughput results also confirm performance gains. The Zynq 7020 and Virtex Ultrascale achieved throughputs of approximately 2.3722 Gbit/s and 2.000 Gbit/s, respectively. In contrast, the Zynq 7035 performed at a peak of approximately 6.400 Gbit/s. The work presented achieved an outstandingly high Throughput (22,000 Mbit/s for Zynq, 10,000 Mbit/s for Zynq, and 8 G.Mb/s for Kria), making them appropriate for high-throughput data-intensive applications.

6. CONCLUSION

In this paper, a dynamic tree selection-based Huffman encoder is presented. In this paper, we provide a comprehensive introduction to the concept of dynamic tree selection based on Dual Huffman encoding. A Dual Huffman encoding algorithm based on dynamic tree selection is also implemented in Digilent FPGA Zynq-7045, Zynq-7100, and Xilinx Kria KV260 AI vision boards. Compared to previous FPGAs (Zynq 7020, Virtex UltraScale, and Zynq 7035), the operating frequency of Zynq 7045, Zynq 7100, and Kria boards is found to be higher. LUT utilization is lower, the number of processing cycles is different, processing times are faster, and Throughput is even better. All these innovations make the proposed FPGAs highly suitable for modern, data-based applications that require fast processing and efficient handling of huge amounts of data, as well as real-time performance. Utilizing these state-of-the-art FPGAs for the proposed work demonstrates their ability to support innovation and improve performance in image processing. Dynamic tree selection-based Dual Huffman encoder on Kria KV260 for high-speed image compression. To the best of our knowledge, we achieve fast compression with ultra-low latency of 29 ns (achieving Throughput of up to 22,000 Mbits/s) using a dynamic tree selection-based Dual Huffman encoder on the Krik KV260.

FUNDING INFORMATION

The authors state no funding is involved.

AUTHOR CONTRIBUTIONS STATEMENT

This journal uses the Contributor Roles Taxonomy (CRediT) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration.

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Masood ahmad Mahammad	✓	✓	✓	✓	✓	✓		✓	✓	✓			✓	
Appala Raju Uppala	✓	✓	✓	✓			✓	✓		✓	✓	✓	✓	
Shaik Mazhar Hussain	✓	✓	✓					✓	✓	✓	✓	✓		
Anusha Marouthu	✓	✓	✓		✓		✓			✓		✓		

C : **C**onceptualization
M : **M**ethodology
So : **So**ftware
Va : **Va**lidation
Fo : **Fo**rmal analysis
I : **I**nvestigation
R : **R**esources
D : **D**ata Curation
O : Writing - **O**riginal Draft
E : Writing - Review & **E**diting
Vi : **Vi**sualization
Su : **Su**pervision
P : **P**roject administration
Fu : **Fu**nding acquisition

CONFLICT OF INTEREST STATEMENT

Authors state no conflict of interest.

DATA AVAILABILITY

Data availability does not apply to this paper as no new data were created or analyzed in this study.

REFERENCES

[1] G. K. Wallace, "The JPEG still picture compression standard," *IEEE Transactions on Consumer Electronics*, vol. 38, no. 1, pp. xviii–xxxiv, 1992, doi: 10.1109/30.125072.
[2] N. Merhav and D. L. Neuhoff, "Variable-to-fixed length codes provide better large deviations performance than fixed-to-variable length codes," in *IEEE Transactions on Information Theory*, vol. 38, no. 1, pp. 135-140, Jan. 1992, doi: 10.1109/18.108258.
[3] J. Ziv and A. Lempel, "A universal algorithm for sequential data compression," *IEEE Transactions on Information Theory*, vol. 23, no. 3, pp. 337–343, May 1977, doi: 10.1109/TIT.1977.1055714.
[4] S. T. Klein and J. Wiseman, "Parallel Huffman decoding," in *DCC '00: Proceedings of the Conference on Data Compression*, San Francisco, CA, USA, 2000, pp. 439–448, doi: 10.5555/789087.789727.
[5] K. Sayood, *Introduction to Data Compression*, 5th ed. San Francisco, CA, USA, 2017, doi: 10.1016/B978-0-12-620862-7.X5000-7.
[6] R. Liu, P. Liu, and H. Zhao, "Design and implementation of high-speed JPEG image encoding system based on FPGA," in *2011 International Conference on Multimedia Technology*, Hangzhou, China, 2011, pp. 4887–4890, doi: 10.1109/ICMT.2011.6002418.
[7] I. H. Witten, A. Moffat, and T. C. Bell, "Managing Gigabytes: Compressing and Indexing Documents and Images," *Morgan Kaufmann*, vol. 41, no. 6, p. 2101, Nov. 1995, doi: 10.1109/TIT.1995.476344.

[8] N. Tiwari and S. C. Reddy, "Performance measurement of a fully pipelined JPEG codec on emulation platform," in *2010 IEEE 2nd International Advance Computing Conference (IACC)*, Patiala, India, 2010, pp. 167–171, doi: 10.1109/IADCC.2010.5423018.
[9] D. E. Knuth, "Dynamic huffman coding," *Journal of Algorithms*, vol. 6, no. 2, pp. 163–180, Jun. 1985, doi: 10.1016/0196-6774(85)90036-7.
[10] D. Castells-Rufas, J. Joven, and J. Carrabina, "Scalability of a parallel JPEG encoder on shared memory architectures," in *2010 39th International Conference on Parallel Processing*, San Diego, CA, USA, 2010, pp. 502–507, doi: 10.1109/ICPP.2010.58.
[11] E. Freire, L. Schnitman, W. Oliveira, and A. Duarte, "Evaluation of the huffman encoding for memory optimization on hardware network intrusion detection," in *2013 III Brazilian Symposium on Computing Systems Engineering*, Niterói, Brazil, 2014, pp. 131–136, doi: 10.1109/SBESC.2013.38.
[12] D. Salomon, *Data Compression: The Complete Reference*, 4th ed. London, U.K.: Springer, 2004
[13] Y. Shan, X. Chen, C. Qiu, and Y. Zhang, "Implementation of Fast Huffman Encoding Based on FPGA," *Journal of Physics: Conference Series*, vol. 2189, no. 1, pp. 1–8, 2022, doi: 10.1088/1742-6596/2189/1/012021.
[14]. S. Thummala, J. Thrisul Kumar, and E. Swarna Latha, "FPGA implementation of Huffman encoder and decoder for high performance data transmission**,"** *International Journal of Engineering Research and Technology (IJERT)*, vol. 3, no. 2, pp. 1–5, 2014.
[15] S. A. Dahri, A. F. Chandio, and N. A. Zardari, "Implementation of Huffman decoder on FPGA," *International Journal of Engineering Research and Applications*, vol. 6, no. 1, pp. 84–88, Jan. 2016.
[16] J. Teuhola, "A compression method for clustered bit-vectors," *Information Processing Letters*, vol. 7, no. 6, pp. 308–311, Dec. 1978, doi: 10.1016/0020-0190(78)90011-0.
[17] A. M. Vijaya Prakash and K. S. Gurumurthy, "VLSI architecture for low power variable length encoding and decoding for image processing applications," *International Journal of Advances in Engineering & Technology*, vol. 2, no. 1, pp. 105–120, Jan. 2012.
[18] D. Huffman, "A Method for the Construction of Minimum-Redundancy Codes," *Proceedings of the IRE*, vol. 40, no. 9, pp. 1098–1101, Sep. 1952, doi: 10.1109/JRPROC.1952.273898.
[19] S. J. Sarkar, N. K. Sarkar, and A. Banerjee, "A novel Huffman coding based approach to reduce the size of large data array," in *2016 International Conference on Circuit, Power and Computing Technologies (ICCPCT)*, 2016, pp. 1–5, doi: 10.1109/ICCPCT.2016.7530355.
[20] Y. Liu and L. Luo, "Lossless compression of full-surface solar magnetic field image based on huffman coding," in *2017 IEEE 2nd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, 2017, pp. 899–903, doi: 10.1109/ITNEC.2017.8284866.
[21] N. Markandeya and S. Patil, "Improve Information Rate in Thien and Lin's Image Secret Sharing Scheme Using Huffman Coding Technique," in *2017 International Conference on Computing, Communication, Control and Automation (ICCUBEA)*, 2017, pp. 1–5, doi: 10.1109/ICCUBEA.2017.8463935.
[22] R. B. Patil and K. D. Kulat, "Audio compression using dynamic Huffman and RLE coding," in *2017 2nd International Conference on Communication and Electronics Systems (ICCES)*, 2017, pp. 160–162, doi: 10.1109/CESYS.2017.8321256.
[23] N. H. Kumar, R. M. Patil, G. Deepak, and B. M. Murthy, "A novel approach for securing data in IoTcloud using DNA cryptography and Huffman coding algorithm," in *2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS)*, 2017, pp. 1–4, doi: 10.1109/ICIIECS.2017.8275958.
[24] S. V. Keerthy, T. K. C. R. Kishore, B. Karthikeyan, V. Vaithiyanathan, and M. M. A. Raj, "A hybrid technique for quadrant based data hiding using Huffman coding," in *2015 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS)*, 2015, pp. 1–6, doi: 10.1109/ICIIECS.2015.7193011.
[25] ISO/IEC 10918-1, "Information technology — Digital compression and coding of continuous-tone still images: Requirements and guidelines," *ISO/IEC*, 2017.
[26] ISO/IEC 13818-2, "Information Technology—Generic Coding of Moving Pictures and Associated Audio Information—Part 2: Video," *ISO/IEC*, 2019.
[27] J. Matai, J. Y. Kim, and R. Kastner, "Energy efficient canonical huffman encoding," *2014 IEEE 25th International Conference on Application-Specific Systems, Architectures and Processors*, pp. 202–209, 2014, doi: 10.1109/ASAP.2014.6868663.
[28] R. M. Elaskary, M. Saeed, T. Ismail, H. Mostafa, and S. Gabran, "Hybrid DCT/Quantized Huffman compression for electroencephalography data," in *2017 Japan-Africa conference on electronics, communications and computers (JAC-ECC)*, Alexandria, Egypt, 2017, pp. 111–114, doi: 10.1109/JEC-ECC.2017.8305790.
[29] J. Choi, B. Kim, H. Kim, and H. J. Lee, "A high-throughput hardware accelerator for lossless compression of a DDR4 Command Trace," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 1, pp. 92–102, 2019, doi: 10.1109/TVLSI.2018.2869663.

BIOGRAPHIES OF AUTHORS

Masood Ahmad Mahammad received his B.E. degree in electronics and communication engineering from VTU, Karnataka. He received an M.Tech. in Electronics and Instrumentation from Andhra University College of Engineering, Andhra University. He received a Ph.D. in Low-Power VLSI from JNTU Hyderabad, Hyderabad, Telangana, India. He is currently working as an Assistant Professor at GITAM University, Hyderabad, Telangana, India. His research includes low-power VLSI, digital system design, FPGAs, and the Internet of Things. He has 15 technical research publications. He has authored six international journal articles, three international conference proceedings, one national conference paper, two books, and three book chapters. He has 18 years of teaching experience. He is a senior member of IEEE, a life member of IETE, ISTE, the Institute of Engineers (India), the Semiconductor Society of India, and the VLSI Society of India. He has three patents published. He can be contacted at email: mmahamma@gitam.edu and masoodahmad80@gmail.com.

Dr. Appala Raju Uppala received the diploma in electronics and communication engineering (ECE) from the Government. Polytechnic College, Narsipatnam, India, in 1997, the A.M.I.E. Degree in Electronics and Communication Engineering from the Institution of Engineers, India, Kolkata, and the M.Tech. Degree with specialization in digital systems and computer electronics from Jawaharlal Nehru Technological University (JNTU), Hyderabad, India, in 2007, and Ph.D. degree from the Department of Electronics and Communication Engineering, Jawaharlal Nehru Technological University (JNTU), Kakinada, India, in 2022. His areas of interest include analog electronics and design, cognitive radio systems, signal processing, adaptive signal processing, and analog and digital communications. He has published 20 technical research articles in international journals, four conference proceedings, one patent, and one book chapter. He has 18 years of teaching experience. Currently, he is an Associate Professor in the Department of Electronics and Communication Engineering, Geethanjali College of Engineering and Technology, Hyderabad, India. He is a life member of IETE, India. He can be contacted at email: raju.mdl@gmail.com.

Shaik Mazhar Hussain is an Associate Professor in the Department of Electronics and Communication Engineering at Malla Reddy Deemed to be University, Hyderabad, India. He completed his Ph.D. in Electrical Engineering with a specialization in Internet of Vehicles and Wireless Communications in January 2023 from Universiti Teknologi Malaysia (UTM), Johor Bahru, Malaysia. He obtained his Master's in Embedded Systems (ES) in 2012 from Jawaharlal Nehru Technological University, Hyderabad (JNTUH), and his Bachelor's in Electronics and Communication Engineering in 2008 from the same university. Additionally, he earned a Postgraduate Certificate (PGCert) in International Higher Education Practice (IHEP) from Coventry University, UK, in December 2017. He was also a senior member of the IEEE and an active member of several other professional organizations. He can be contacted at email: mazharh5@mrec.ac.in.

Anusha Marouthu obtained her Ph.D. degree in the Department of Computer Science and Engineering (CSE) at KL University, India, in 2019. She currently works as an Associate Professor in the Department of Computer Science and Engineering at the KL University of India, and her research interests include wireless sensor networks and MAC protocol problems in cognitive radio networks. She can be contacted at email: anushaaa9@kluniversity.in.