Efficient compression algorithm for multimedia storage device

S. Cammillus[1] Nithya Kalyani E[2] Sreemathi RT[3] Andal Priyadharsini R[4]

[1]Assistant Professor, Department of ECE, National Engineering College, K.R.Nagar Post, Kovilpatti, Thoothukudi

[2-4]Department of ECE, National Engineering College, K.R. Nagar Post, Kovilpatti, Thoothukudi

[1]cammilus@nec.edu.in, [2]2111009@nec.edu.in, [3]2111010@nec.edu.in, [4]2111030@nec.edu.in

Abstract

Digital Video Recorders (DVRs) have some major problems when it comes to storage, as for the efficiency and the accuracy of data; they contain inadequate storage space, data may get corrupted, and the like. As a solution to the storage problem, there is some research that tries to establish a concrete plan to work out a solution that may involve the identification of better approaches to data management. The main plan is to increase the storage by having additional hard drives and RAID combinations for backup, plus buying the finest gear that does not fail. To avoid such storage problems, this research is experimenting with the 842-compression method to simplify data storage. This technique assists in reducing the data size, thus eliminating the data that is not required and, in the process, creating more space. This is because, applying this approach of continuous compression, DVRs can record such important information for not less than two months, guaranteeing fifteen days of data saving. One should also maintain the system properly, update the software, and improve on the security aspects to improve the performance of the system and to ensure protection of the data. Solving the DVR storage issues, this plan also guarantees the proper functioning of the system with the necessary data longevity.

Introduction

Data compression and encryption techniques are critical in fields such as telecommunications, medical imaging, multimedia, and storage, with each requiring unique solutions to optimize efficiency, security, and quality. The research summarized here provides insights into a variety of innovative compression approaches. In [1], a modern video compression method employs adaptive positional coding with varying codogram lengths in order to increase efficiency while maintaining video quality. [2] which examines compression techniques for radiology in order to strike a balance between compression levels and diagnostic quality. Because medical images frequently require high detail, the paper focuses on controlled compression to avoid loss of diagnostic accuracy. Another hardware-oriented study in [3] describes a fast, secure compression chip intended for high-performance storage, allowing for faster and more reliable data processing. Video compression methods continue to evolve, as evidenced by [4] and [5]. Paper [4] assesses recent compression techniques, focusing on parameters such as computational complexity and image quality, whereas [5] investigates neural network-based methods for image and video compression, noting significant potential in compression quality, albeit at high computational costs. [6] focuses on real-time video processing, evaluating zero-latency compression strategies and demonstrating that effective compression for streaming content can be achieved with appropriate hardware optimization. Another comprehensive review, [7] surveys data compression techniques across fields, discussing their efficiency and reliability in various applications. In [8], segmentation-based compression combines motion and texture models to improve video quality, whereas [9] revisits Huffman coding and proposes a double-tree variant to accelerate encoding by removing low-frequency codes. [10] investigates arithmetic coding, which reduces required bandwidth by optimizing compression. In [11], Huffman encoding is adapted for steganography, increasing the security of hidden data within images while preserving image quality. Data storage optimization is the focus of [12], which proposes methods for increasing data density and extending the lifespan of SSDs. [13] and [14] highlight hardware-based performance improvements, with GPU-based decompression and FPGA-based compression demonstrating significant speed and efficiency gains, respectively. Both approaches are ideal for real-time, high-performance applications requiring low latency.

Hosseini's 2012 overview, discussed in [15] and [16], surveys various data compression algorithms and their applications, comparing lossless and lossy methods for different types of data.

FLOW DIAGRAM:

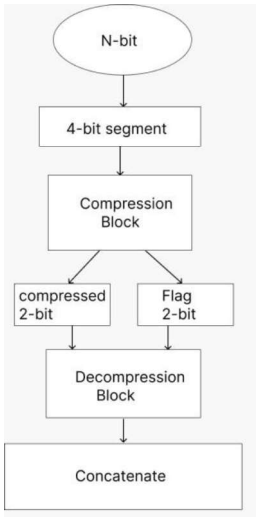


Fig 1.1 Flow diagram of Compression

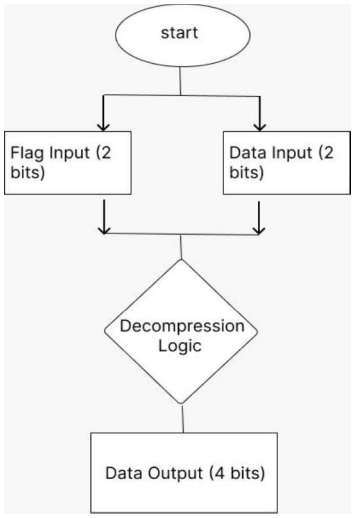


Fig 1.2 Flow diagram of decompression

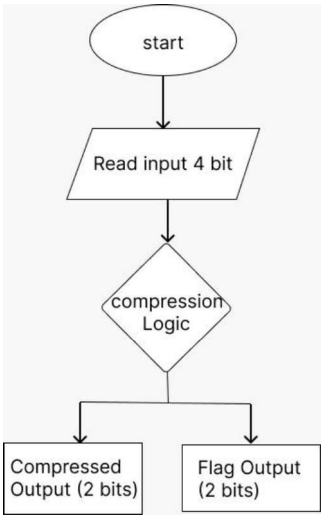


Fig 1.3 Flow diagram of N-bit Compression & Decompression

The Fig 1.1 describes the step-by-step process for compressing 4-bit input data into two outputs: a compressed 2-bit result and a 2-bit flag. It begins with the system reading a 4-bit input, which then moves into the compression logic. This logic likely reduces the size of the data, transforming it into a more efficient format while also generating a flag to accompany the compressed data. The output of this process consists of two distinct parts: a compressed 2-bit representation of the input and a 2-bit flag. The flag could serve as an indicator of the compression method used or signal some specific condition associated with the data.

The Fig 1.2 describes the step-by-step process of a simple data decompression process. It starts with two inputs: a 2-bit flag and a 2-bit data input. The flag is used to guide the decompression process, indicating how the incoming data should be interpreted. Both the flag and data input feed into a decompression logic block, where the actual decompression takes place based on the values of these inputs. The decompression logic expands the 2-bit compressed data into a 4-bit output, producing the original or decompressed form of the data. This process helps reduce data size during storage or the transmission by encoding the data into fewer bits and later restoring it through decompression logic.

In Fig 1.3 the process begins by taking a N-bit input signal, which is then divided into ten smaller segments, each consisting of 4 bits. These 4-bit segments are extracted from the input and assigned to individual wires, essentially breaking down the larger input into more manageable parts. Each of these 4-bit segments is then fed into an instance of a module named compress, which is responsible for performing some specific operations on the segment. The compress module processes each 4-bit chunk and produces two important outputs: a 2-bit result and a 2-bit flag. These outputs reflect some intermediate results that will be used in the next stage of the process.

ALGORITHM:

4-BIT COMPRESSION

The 4-bit input data.

This Verilog code compresses a 4-bit input into a 2-bit output using XOR operations.

The compression uses the most significant two bits XOR-ed together for the first output bit, and the least significant two bits XOR-ed together for the second output.

4-BIT DECOMPRESSION

The flag (2-bit flag input) is used to decompress into a 4-bit output .

Based on the flag, the output data is restored to one of the predefined 4-bit values.

N-BIT COMPRESSION & DECOMPRESSION

N: The bit width (e.g., 160, 120, 80, 60, 40).

M: The number of 4-bit segments = N/4

Input Declaration:

Declare input: input [N-1:0] d;

Intermediate Wires Declaration:

Declare arrays:

result [M-1:0] (2-bit): Intermediate results from masterfile.

flag [M-1:0] (2-bit): Flags from masterfile.

out [M-1:0] (4-bit): Results after de4.

Declare M 4-bit cout wires.

Assign Values to cout:

Split the input bits into 4-bit segments.

For each segment, assign values to cout[i] from the input d.

Compression: Use the derived compression algorithm on the cout segments for data compression.

Decompression: Utilize the compressed output to reconstruct the original data through the corresponding decompression algorithm.

Concatenate Outputs: Assign output by concatenating all elements of the decompressed array in reverse order

RESULTS AND DISCUSSION:

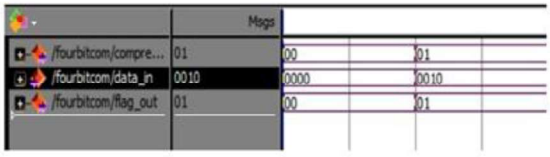


Fig 2.1.1 Output of 4 bit compression

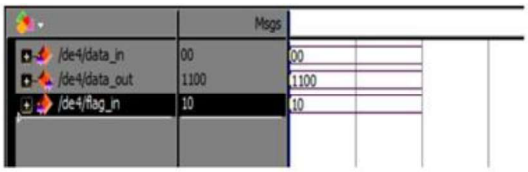
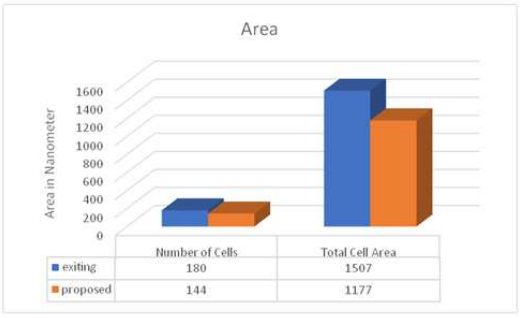


Fig 2.1.2 Output of 4 bit decompression

The compressed output reduces the original 4-bit size, resulting in a more efficient representation. This output can then be used for further processing or concatenated with other compressed segments for a complete data stream.The decompressed output gives the original 4-bit size using decompression logic with respect its input data and flag data. The existing 842 compression algorithm [8] involves more computations and more logical gates for efficient compression.



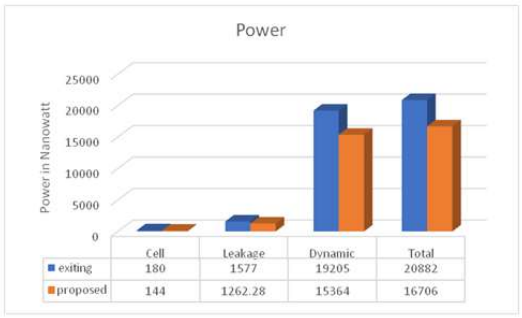
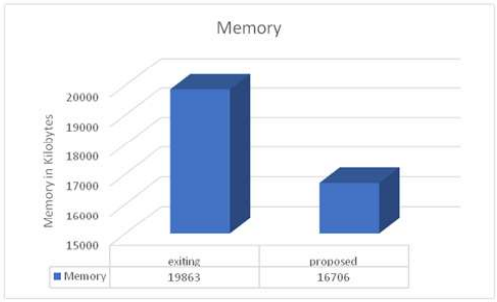


Fig 3.1.1 **Fig 3.1.2**

Fig 3.1.1 shows existing algorithm that consist of 180 cells (gates). The proposed algorithm consumes 144 cells (gates) and minimum cell occupancy in SOC.The Fig 3.1.2 depicts power consumption of the proposed model which is better than the existing model. Approximately 20% of power consumption is reduced in the proposed model due to minimum gates compared to the existing one.



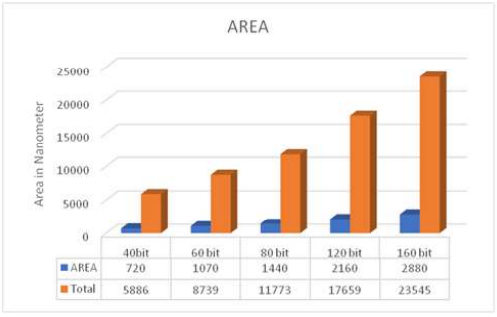


Fig 3.1.3 **Fig 3.1.4**

Fig 3.1.3 clearly says that the proposed model consumes minimum memory space in SOC compared to the existing model by 16%.

The compression and decompression parameter is used in many applications. The results of the compression and decompression algorithms applied for the input streams of 40, 60, 80, 120 and 160 bits of the proposed model are described in Fig 3.1.4.

As the bit stream increases, the number of cells (gates) increases linearly. Considering 40 bit and 80 bit input streams, the number of cells increases by 48%. The increasing in gates as input stream increases is due to the increase in memory components (sequential circuits).

Considering the power consumption for the interest of data (input stream), power consumption increases as the sequential circuit in each case of the interest increases. The proposed model introduces NIL slew rate and the delay is reactive. The power is drained as the leakage voltage is limited to 10% in each case which is shown in Fig 3.1.5

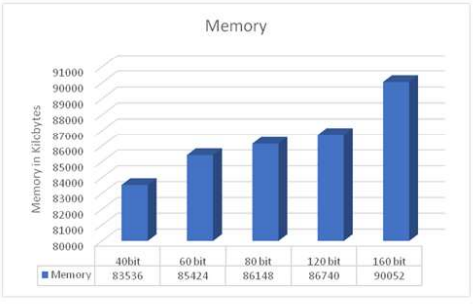


Fig 3.1.5 **Fig 3.1.6**

The memory space consumed in SOC also increases slightly 1% to 3%. The increased consumption of memory space may further be reduced by adapting suitable pick and route algorithm. The detailed comparison for different input streams are given below in Fig 3.1.6.

CONCLUSION:

In this project, each 4-bit data segment is paired with a 2-bit flag, which improves both efficiency and control during compression and decompression. The 2-bit flag is critical to maintaining data integrity because it allows for selective adjustments during compression, ensuring that only necessary segments are compressed without losing accuracy. This structured approach allows for precise tracking of intermediate results, resulting in a decompression process that reliably restores the original data using input and flag data. Overall, this method offers a reliable and adaptable solution for data compression and decompression. Further, the proposed model can be enhanced by adapting pick and route algorithm to reduce the memory space in SOC thereby reducing the power consumption.

REFERENCES:

[1]. Barannik, V., Sidchenko, S., Barannik, D., Yermachenkov, A., Savchuk, M. and Pris, G., 2023. Video images compression method based on floating positional coding with an unequal codograms length. Radioelectronic and Computer Systems, (1), pp.134-146.

[2]. Braunschweig, R., Kaden, I., Schwarzer, J., Sprengel, C. and Klose, K., 2009, July. Image data compression in diagnostic imaging: international literature review and workflow recommendation. In RöFo-Fortschritte auf demGebiet der Röntgenstrahlen und der bildgebendenVerfahren (Vol. 181, No. 07, pp. 629-636). © Georg Thieme Verlag KG Stuttgart· New York.

[3]. Cheng, J.M., Duyanovich, L.M. and Craft, D.J., 1996. A fast, highly reliable data compression chip and algorithm for storage systems. IBM journal of research and development, 40(6), pp.603-613.

[4]. Singh, P. and Singh, Y.P., 2021. Review on Modern Video Data Compression Techniques.

[5]. Ma, S., Zhang, X., Jia, C., Zhao, Z., Wang, S. and Wang, S., 2019. Image and video compression with neural networks: A review. IEEE Transactions on Circuits and Systems for Video Technology, 30(6), pp.1683-1698.

[6]. Westwater, R. and Furht, B., 2007. Real-time video compression: techniques and algorithms (Vol. 376). Springer.

[7]. Fitriya, L.A., Purboyo, T.W. and Prasasti, A.L., 2017. A review of data compression techniques. International Journal of Applied Engineering Research, 12(19), pp.8956-8963.

[8]. Bosch, M., Zhu, F. and Delp, E.J., 2011. Segmentation-based video compression using texture and motion models. IEEE Journal of Selected Topics in Signal Processing, 5(7), pp.1366-1377.

[9]. Tommy, T., 2021. Dual tree huffman encoding implementation to reduce low frequency bit codes. INFOKUM, 9(2, June), pp.436-442.

[10]. Faizal, F., Rachmat, R. and Ibrahim, A., 2021. Aritmathic Code Data Compression In Building Data Communication. E-JURNAL JUSITI: JurnalSistemInformasi dan TeknologiInformasi, 10(2), pp.188-197.

[11]. Das, R. and Tuithung, T., 2012, March. A novel steganography method for image based on Huffman Encoding. In 2012 3rd National Conference on Emerging Trends and Applications in Computer Science (pp. 14-18). IEEE.

[12]. Kong, J., Samsung Electronics Co Ltd, 2013. Method of Storing Data in a Storage Medium and Data Storage Device Including the Storage Medium. U.S. Patent Application 13/615,752.

[13]. Plauth, M. and Polze, A., 2019, November. GPU-based decompression for the 842 algorithm. In 2019 Seventh International Symposium on Computing and Networking Workshops (CANDARW) (pp. 97-102). IEEE.

[14]. Sukhwani, B., Abali, B., Brezzo, B. and Asaad, S., 2011, May. High-throughput, lossless data compresion on FPGAs. In 2011 IEEE 19th Annual International Symposium on Field-Programmable Custom Computing Machines (pp. 113-116). IEEE.

[15]. Hosseini, M., 2012. A survey of data compression algorithms and their applications. Network Systems Laboratory, School of Computing Science, Simon Fraser University, BC, Canada.

[16]. Blelloch, G.E., 2001. Introduction to data compression. Computer Science Department, Carnegie Mellon University, 54.

Biographies

Sreemathi RT pursuing final year bachelors degree in Electronics and Communication Engineering at National Engineering College, Kovilpatti. Similarly co authors Andal priyadharsini R and Nithyakalyani E also pursuing same degree in National Engineering College. Our research interests include digital signal processing, image and video compression, and multimedia storage systems. We were working on developing efficient compression algorithms for multimedia data for the past two years. This paper proposes an efficient compression algorithm for multimedia storage devices.

Email Id: sreemathi943@gmail.com
Phone no: 9566572347