An FPGA Accelerator for Real Time Hyperspectral Image Compression based on JPEG2000 Standard

Refka Ghodhbani
Department of Computer Sciences, Faculty of Computing and Information Technology, Northern Border University, Saudi Arabia | Laboratory of Electronic and Microelectronics, Faculty of Sciences, University of Monastir, Tunisia
refka.ghodhbani@nbu.edu.sa (corresponding author)

Taoufik Saidani
Department of Computer Sciences, Faculty of Computing and Information Technology, Northern Border University, Saudi Arabia
taoufik.saidan@nbu.edu.sa

Layla Horrigue
Laboratory of Electronic and Microelectronics, Faculty of Sciences, University of Monastir, Tunisia
layla.k-12@hotmail.com

Asaad M. Algarni
Department of Computer Sciences, Faculty of Computing and Information Technology, Northern Border University, Saudi Arabia
asaad.algarni@nbu.edu.sa

Muteb Alshammari
Department of Information Technology, Faculty of Computing and Information Technology, Northern Border University, Saudi Arabia
muteb.alshammari@nbu.edu.sa

Received: 1 January 2024 | Revised: 14 January 2024 | Accepted: 15 January 2024

 DOI: https://doi.org/10.48084/etasr.6853

ABSTRACT

Lossless hyperspectral images have the advantage of reducing the data size, hence saving on storage and transmission costs. This study presents a dynamic pipeline hardware design for compressing and decompressing images using the Joint Photographic Experts Group-Lossless (JPEG2000) algorithm. The proposed architecture was specifically tailored for implementation on a Field Programmable Gate Array (FPGA) to accomplish efficient image processing. The introduction of a pipeline pause mechanism effectively resolves the issue of coding errors deriving from parameter modifications. Bit-plane coding was employed to enhance the efficacy of image coding calculations, leading to a reduction of parameter update delays. However, the context and decision creation procedure were streamlined, resulting in a significant enhancement in throughput. A hardware module utilizing the parallel block compression architecture was developed for JPEG2000 compression/decompression, allowing for configurable block size and bringing about enhanced image, compression/decompression, throughput, and reduced times. Verification results were obtained by implementing the proposed JPEG 2000 compression on a Zynq-7000 system-on-chip. The purpose of this system was to enable on-board satellite processing of hyperspectral image cubes with a specific focus on achieving lossless compression. The proposed architecture outperformed previous approaches by using fewer resources and achieving a higher compression ratio and clock frequency.

Keywords-hyperspectral image compression; JPEG2000; EBCOT; MQ coder; FPGA; `

I. INTRODUCTION

Hyperspectral imaging (HSI) is used in a variety of remote sensing applications, including intelligence gathering, commerce, agriculture, the military, and humanitarian efforts. Among these uses are environmental monitoring [1], field observation in agriculture [2], geological mapping [3], and mineral exploration [4], while its growth in recent years has been steady. In 2018-2023, the global market for HSI was expected to expand at a Compound Annual Growth Rate (CAGR) of 14.7%, according to the Business Communications Company, USA. The forecasts indicated that by 2023, the market would have grown from $104.0 million in 2018 to $206.2 million [4].

Hyperspectral image compression is the creation of a smaller version of data from a larger one, usually to make storage or transmission easier. Lossless and lossy compression are the two main types of compression. Whether or not the compressed version can be used to reconstruct the original datastream in its entirety determines this classification [5]. The DC program employs two different image resolutions: 4096×2160 (4K) and 2048×1080 (2K). On top of that, the 2K image can support 24 or 48 fps. However, the 4K image can only support 24 fps. So, for a three-hour film, the amount of raw data is nearly 9 GB. For that reason, a fast JPEG 2000 encoder is required for the real-time compression of these data. This study goes deep into image compression, focusing on the JPEG 2000 standard in particular.

The specific standard was produced by ISO/IEC and introduced in 2000 by the Joint Photographic Experts Group (JPEG) committee. The former uses the Discrete Wavelet Transform (DWT) instead of the Discrete Cosine Transform (DCT) [6]. The incorporation of many features in the JPEG 2000 standard, in addition to its improved compression ratios [7], is what differentiates it from earlier generations of compression methods. Its scalability stands out among the rest. Parts of the compressed JPEG 2000 bitstream can be extracted, together with a configurable spatial locality value between 1 and 5 [6, 8]. Due to this, the possible uses of JPEG 2000 are considerably broadened. Figure 1 shows a schematic of the JPEG 2000 encoding algorithm. Separating the original picture into smaller, more manageable pieces, called tiles and code blocks, allows for more efficient processing and reduces memory utilization. In [8], the techniques implemented in JPEG 2000, which include Embedded Block Coding with Optimum Truncation (EBCOT) and DWT [9], were described. Two processing stages make up the EBCOT Tier-1 encoding scheme: Matrix Quantizer (MQ) and Context Modeling (CM). To generate Context Decision (CX/D) pairings, CM processes the data contained in a CB based on algorithm evaluation. The MQ coder utilizes these CX/Ds pairings to create an embedded bitstream.

II. JPEG2000 ALGORITHM

The JPEG2000 standard employs some picture properties to perform compression sequentially. Its principal use is photo compression, but it has found other uses, such as in electroencephalography, video, and hyperspectral pictures.

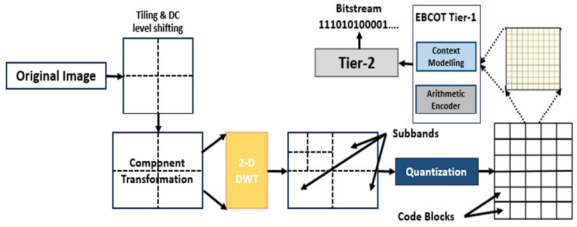

Fig. 1. JPEG 2000 encoder architecture.

An RGB input color space is transformed into a luminance and chrominance model via a pixel-wise color transformation because changes in brightness are more perceptible than color changes. Downsampling the color channels significantly reduces the number of data bits, but there is no noticeable loss in quality [8, 10]. After that, each channel should undergo a wavelet transform [10]. A high-pass filter is applied horizontally to all rows and columns, and a low-pass filter is applied to all columns, to implement a wavelet transform. This can be accomplished adopting either irreversible (lossy) or reversible (lossless) methods. In the end, there is a partitioned channel with zones that compress the data better than the originals due to their distinct patterns. The values that emerge from the wavelet transform are then quantized to produce integers. The lossy wavelet transform is useful but results in some data being lost [11]. The values are encoded at the end. Each block in the image can hold up to 4096 samples. The wavelet transform's patterns and local redundancies are utilized to encode each block independently.

Since hyperspectral compression decorrelates the spectral dimension during dimensionality reduction, the color transform is superfluous. Encoding and wavelet transform operations are the only ones carried out. In JPEG2000, encoding takes up around 70% of the total execution time, while the wavelet transform is much faster [12-14]. Furthermore, it has a major impact on the execution time within JYPEC [15]. This study aims to reduce the method's total execution times by focusing on an FPGA implementation that speeds up the bottleneck and greatly improves encoding.

III. JPEG2000 PROPOSED ARCHITECTURE

The BPC and MQ coders, when combined, provide the complete tier-1 coder. Figure 2 shows the basic organizational structure of the tier-1 coder. The bit plane coder creates CX/D pairs after retrieving data from memory. The MQ coder codes them, and an output stream is produced.

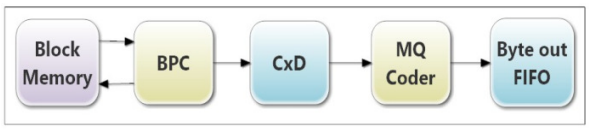

Fig. 2. Tier-1 coder architecture.

A. Bit-Plane Coding

Figure 3 displays the proposed EBCOT architecture. The DWT coefficients are kept in a codebook (CB) with a dimension of 32×9×32, including ten bits. Following the

initialization of state variables (*r*, *r0*, and *g*), the memory controller retrieves bit plane data and sign matrix data from the code block memory, as well as state variable data. These are then written into the stripe generator, which uses the column generator module to construct the columns. The stripe controller carefully picks the stripe to be processed, while considering the boundary criteria. Every column is sequentially encoded using the basic coding in each coding step. The pass detection logic identifies the pass to be executed and activates the required primitive modules. To expedite the intense block EBCOT process and reduce its complexity, all samples in a column are coded simultaneously inside a single clock cycle. In addition, the column of state variables is updated and both of its neighboring variables are made accessible simultaneously [8]. This has the potential to produce anywhere from one to eight CX/D pairs concurrently. Therefore, to enhance the speed of design and minimize the memory needed for BPC, a novel architecture for bit-plane coding is proposed that incorporates a more efficient approach to data organization and memory arrangement. In this proposed design, MRP and CUP are processed concurrently. The BPC controller determines the appropriate pass to execute, chooses a specific coding primitive, and saves context data in the context sequencer based on the conditions generated during runtime. The purpose of the last step is to segregate and store these pairs in the appropriate sequence before they are transmitted to the MQ coder module.

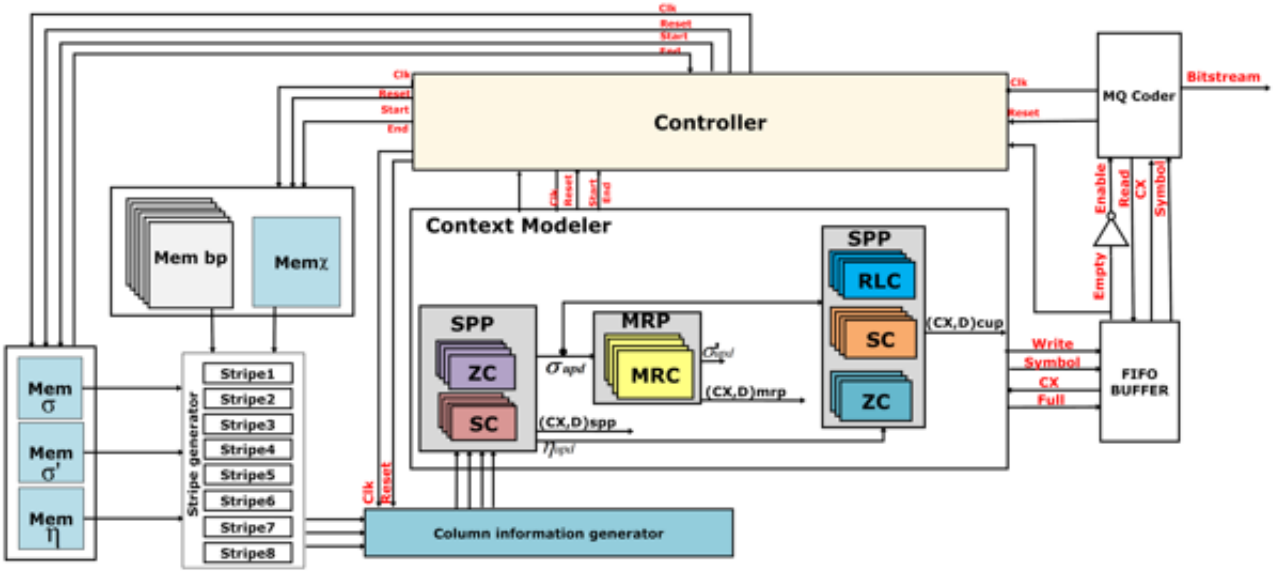

Fig. 3. Top module of the proposed column-parallel EBCOT context modeling.

B. MQ-Coder

The block diagram portrayed in Figure 4 illustrates the proposed design of the encoder architecture. The MQ coder receives the pairs (*C*, *D*) as input and produces a sequence of bytes named *ByteOutReg* as output. This design comprises two components: the prediction component, which uses two RAMs (ICX, MPS) and four ROMs (NMPS, NLPS, Switch, Qe) to calculate the likelihood of the symbol to be coded, and the coding component which consists of a state machine [12]. The four ROMs remain unaltered during the coding process. The pairs (*CX*, *D*) are initially read sequentially. Afterward, the *CX* context will be communicated via the bus address of both the ICX RAM and the MPS RAM. Subsequently, the values of *I*(*CX*) and *MPS*(*CX*) will be retrieved. Subsequently, the *I*(*CX*) index will be transmitted to the four ROMs. The mps_D will be executed with signal *D*, generating the *LPS_en* signal. If the signal is equal to one, the CODELPS state will be executed. Otherwise, the CODEMPS state will be executed. The ICX RAM is primarily dependent on the *Ren_out* signal for updating. This signal will be set to one when renormalization is performed. The MPS RAM will only update if the *LPS_SW* signal is set to one. Figure 4 depicts the architecture for estimating probabilities. The block diagram in Figure 5 illustrates the proposed encoder design. The MQ coder receives the pairs (*C*, *D*) as input and generates a sequence of *b*. This study focuses on the coding aspect of controlling the process of coding finite states. This involves replacing different subalgorithms with states. The outputs depend on the present state and the inputs, and they immediately respond to any alterations in the inputs. The MQ encoder process is described by a set of fourteen states.

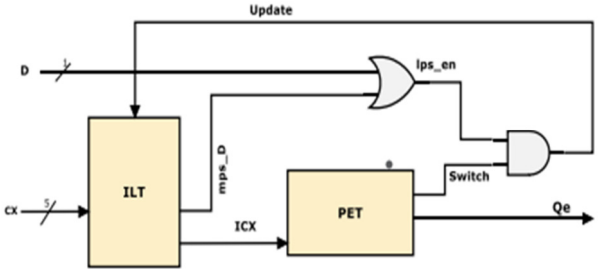

Fig. 4. Architecture for estimating probabilities.

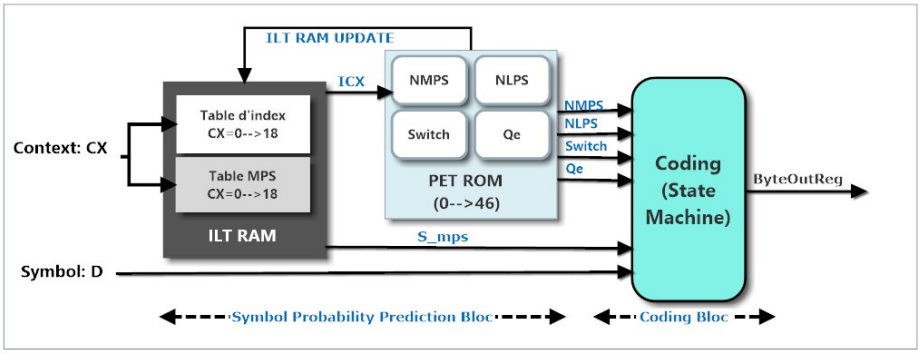

Fig. 5. MQ-coder architecture. The interval updates are fused when possible, having fewer bound updates which could stall the pipeline.

IV. FPGA IMPLEMENTATION

The compression/decompression system was developed on the Zynq7020 platform. The compression and decompression modules were packaged separately as tailored AXI IP cores. The top level of the IP core consists of the instantiation of the compression/decompression module and two BRAMs for storing the original/compressed and compressed/decompressed data. If the size of the compressed block is larger than the size of the original block, the original block is used as compressed data, and the block size is stored in the block header information. If the size of a block exceeds the original block size, it will not undergo decompression in the decompression process. The IP core should incorporate the AXI logic to perform read and write operations in DDR memory, using a burst length of 16 and a size of 4. Furthermore, it should adaptively adjust the number of burst transfers according to the size of each block. The header file of the compression/decompression module allows modification of the block size options. The CPU is tasked with getting the original image data from the SD card and executing the block function, before writing it into DDR. In addition, the CPU needs to access and configure registers to control the compression/decompression module.

The proposed hardware accelerator was developed utilizing the Vivado Design suite to evaluate its performance in compressing hyperspectral photos, based on an optimized version of JPEG2000. Xilinx offers a toolchain that encompasses Vivado HLS, a specialized tool aimed at optimizing the development of Intellectual Property (IP) components for FPGA-based solutions on Xilinx devices. The initial prototype employed the EBCOT algorithm and the MQ coder, which were specifically tailored for the variant XC7Z020-CLG484 of the Xilinx Zynq-7000 SoC. The FPGA was chosen for its cost-effectiveness, lightweight design, and great adaptability, making it an attractive option for incorporation into aerial vehicles such as drones. The purpose of this early prototype was to evaluate the efficiency of a mid-range reconfigurable FPGA for JPEG2000 compression. FPGA was used to implement the JPEG preprocessing accelerator specifically developed for picture categorization. This study engaged Verilog for the implementation of the Register-Transfer Level (RTL) and used Xilinx Vivado v2017.4 to compile the source code into the bitstream. The ImageNet Large Scale Visual Recognition Challenge 2012 (ILSVRC2012) was utilized as an input picture dataset. This dataset consists of 50,000 JPEG photos with various dimensions, ranging from 80×60 to 3657×2357, accompanied by multiple sampling compression algorithms. The efficiency of the proposed preprocessing accelerator was evaluated by employing a cost-effective 16 nm XCZU7EV chip. The preprocessing accelerator was synthesized with a clock frequency of 250 MHz.

A. Performance Analysis

Table I presents the performance comparison results of the JPEG2000 preprocessing techniques between the proposed hardware design and the 12 nm Nvidia GeForce RTX 2080Ti GPU with the Nvidia DALI library. This study evaluated and analyzed the power consumption, energy efficiency, and throughput of each of the three individual implementations. Thermal Design Power (TDP) and utilization data were used to evaluate the computational capability of a system based on a CPU. The Vivado Toolset and NVIDIA System Management Interface offer insight into the advantages of using a hardware accelerator that relies on FPGAs and GPU for implementation.

TABLE I. PERFORMANCE COMPARISON OF JPEG2000 PREPROCESSING

Metrics	Nvidia GeForce RTX 2080TI 12 nm	Xilinx XCZU7EV 16 nm (proposed)
FPS	984.45	875.67
Speedup	2.84×	2.52×
Power (W)	58	12.35
Joule per frame (J/F)	0.059	0.014
Energy efficiency	4.21×	1×

The results indicated that both the proposed FPGA hardware accelerator and GPU-based implementation surpassed the optimized CPU-based architecture by a factor of 2.52 and 2.84, respectively. The energy efficiency of the two options was evaluated and compared. The proposed FPGA-based hardware accelerator reduced the energy utilization for each photo preprocessing frame by a factor of 23.07 and 4.21, respectively, compared to the CPU and GPU. The performance findings illustrate that the proposed FPGS hardware accelerator effectively performs photo preprocessing.

B. Resource Utilization

Look-Up Table (LUT) resources and slice resources are distinct categories of logic resources found in FPGA devices,

each serving a specific purpose in the implementation of various functionalities. A LUT is a data structure that may contain truth tables, allowing for the implementation of combinational logic or distributed memory. A slice is a logical unit consisting of several LUTs and flip-flops and is capable of implementing sequential logic, arithmetic operations, data selectors, and other functions. In general, a lower utilization of resources indicates that the system has a greater amount of idle resources available to fulfill more requests or loads [16-17]. The XC7Z020-CLG484-1 Zynq7000 development board used in this study features 53200 LUTs, 106400 Flip-Flops (FFs), 4.9 Mb of Block RAM (BRAM), 512 MB of DDR memory integrated on the board, and an SD card port.

The cores of the processing system were completely integrated into the Zynq 7000 SoC XC7Z020-CLG-484-1 [18]. The entire system was synthesized and implemented at an ideal frequency of approximately 100 MHz. When examining performance and flexibility, it is possible to identify some physical constraints, such as the exhaustion of DSP blocks. The distribution of resources within the SoC depends on the complexity of the architecture being implemented. Three separate filter designs are used for the input resolution of 1920×1080 photos. Table II presents a concise overview of the system's resource utilization. The results hilight the advantages of using the Vivado 2017.4 HLS tool on the versatile SoC platform [18]. Approximately 6% of all logic cells were currently in use. The application of logic cells varies depending on the specific location and routing techniques. The frequency of the proposed architecture was 320 MHz.

TABLE II. ZYNQ DEVICE RESOURCE UTILIZATION REPORT OF THE PROPOSED IMPLEMENTATION

Resource	Utilized	Total(%)
No. of programmable logic cells	5217	6.05
Look Up Tables (LUTs)	3,692	4.28
Flip Flops (FFs)	4,218	4.89
Block RAMs (BRAMs)	21	15
Frequency (MHz)	320	

The IP created by the HLS must be integrated into an HW/SW architecture for the FPGA SoC, which was realized by the Xilinx Vivado Design Suite. After importing the exported "IP Catalog" as a new repository file into the Vivado catalog, the block diagram of the design was created and the accelerator design was added as a new IP element, as shown in Figure 6.

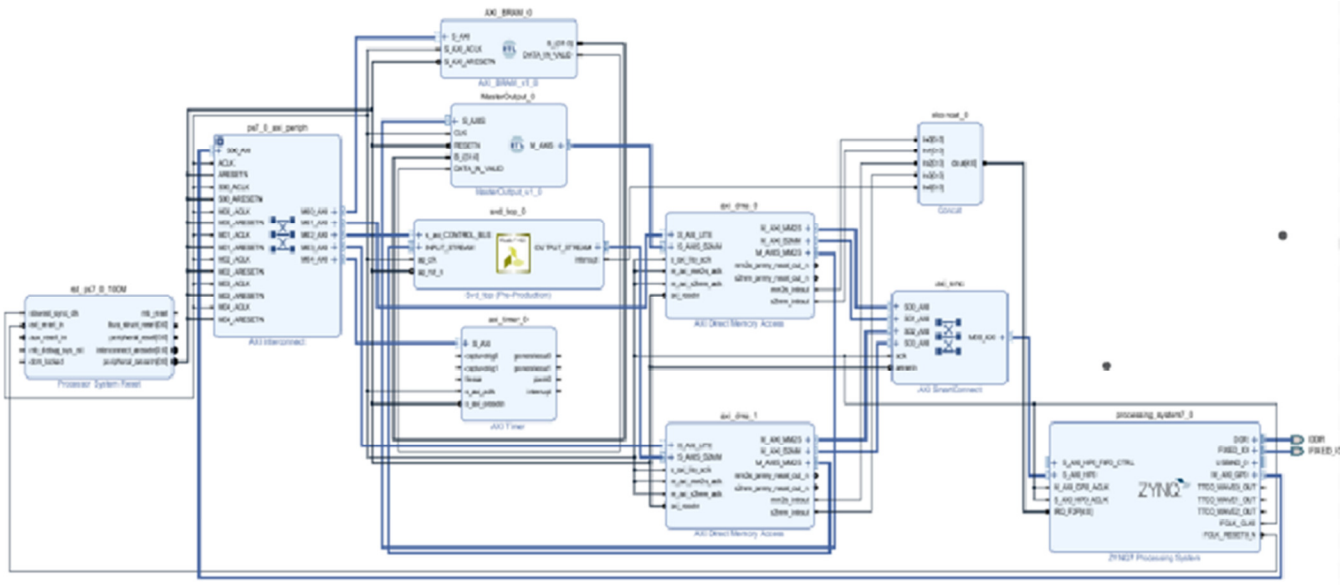

Fig. 6. RTL design for JPEG2000 architecture.

V. CONCLUSION

JPEG2000 is an advanced method that requires robust technology to achieve rapid real-time execution. The tier 1 coder in JPEG2000 is the most costly component because of the difficulty in optimizing code with unpredictable branching for ordinary processors. The algorithm's arithmetic and logic operations are very compatible with FPGA implementation because of their inherent simplicity. A swift and efficient design was created specifically for the highly skilled tier-1 coder in JPEG 2000, based on two fundamental concepts. The bit plane coder operates by simultaneously processing bits in batches of four, resulting in substantial performance improvement. A system comprising First-In-First-Out (FIFO) queues and buffers ensures a seamless stream of CxD pairs to the MQ-coder. Furthermore, the coder itself was painstakingly optimized using a pipelined technique. To avoid pipeline stalling, a strategy of consolidating multiple updates was utilized whenever possible, addressing a previous concern in design. This study presented an implementation of the proposed JPEG 2000 architecture on a Zylinx Zynq7000 board and compared it with an NVIDIA GeForce RTX 2080TI demonstrating approximately 4 times better energy efficiency. Within the field of hyperspectral imaging, the use of advanced lossy compression techniques allows real-time performance within the specified limits of the AVIRIS-ng sensor (30-72

MS/s, corresponding to a total of 491.52 Mb/s). This allows rapid compression of data at high speeds for immediate and long-lasting storage while preserving exceptional quality for future analysis.

ACKNOWLEDGMENT

The authors gratefully acknowledge the approval and support of this study by grant no. CSCR-2023-12-2062 from the Deanship of Scientific Research at Northern Border University, Arar, Saudi Arabia.

REFERENCES

[1] M. J. Khan, H. S. Khan, A. Yousaf, K. Khurshid, and A. Abbas, "Modern Trends in Hyperspectral Image Analysis: A Review," *IEEE Access*, vol. 6, pp. 14118–14129, 2018, https://doi.org/10.1109/ACCESS.2018.2812999.

[2] P. Ghamisi *et al.*, "Advances in Hyperspectral Image and Signal Processing: A Comprehensive Overview of the State of the Art," *IEEE Geoscience and Remote Sensing Magazine*, vol. 5, no. 4, pp. 37–78, Sep. 2017, https://doi.org/10.1109/MGRS.2017.2762087.

[3] M. Hernández-Cabronero *et al.*, "The CCSDS 123.0-B-2 'Low-Complexity Lossless and Near-Lossless Multispectral and Hyperspectral Image Compression' Standard: A comprehensive review," *IEEE Geoscience and Remote Sensing Magazine*, vol. 9, no. 4, pp. 102–119, Sep. 2021, https://doi.org/10.1109/MGRS.2020.3048443.

[4] O. Barkovska, I. Filippenko, I. Semenenko, V. Korniienko, and P. Sedlaček, "Adaptation of FPGA architecture for accelerated image preprocessing," *Radioelectronic and Computer Systems*, no. 2, pp. 94–106, May 2023, https://doi.org/10.32620/reks.2023.2.08.

[5] D. Báscones, C. González, and D. Mozos, "An FPGA Accelerator for Real-Time Lossy Compression of Hyperspectral Images," *Remote Sensing*, vol. 12, no. 16, Jan. 2020, Art. no. 2563, https://doi.org/10.3390/rs12162563.

[6] C. Fu, Y. Yi, and F. Luo, "Hyperspectral image compression based on simultaneous sparse representation and general-pixels," *Pattern Recognition Letters*, vol. 116, pp. 65–71, Dec. 2018, https://doi.org/10.1016/j.patrec.2018.09.013.

[7] P. K. Nath and S. Banerjee, "A high throughput pass parallel block decoder architecture for JPEG 2000 that prevents stalling in the decoding process," *Integration*, vol. 71, pp. 170–182, Mar. 2020, https://doi.org/10.1016/j.vlsi.2019.11.013.

[8] D. Taubman and M. Marcellin, JPEG2000 Image Compression Fundamentals, Standards and Practice: Image Compression Fundamentals, Standards and Practice. New York, NY, USA: Springer Science & Business Media, 2012.

[9] R. Ghodhbani, T. Saidani, L. Horrigue, and M. Atri, "An efficient pass-parallel architecture for embedded block coder in JPEG 2000," *Journal of Real-Time Image Processing*, vol. 16, no. 5, pp. 1595–1606, Oct. 2019, https://doi.org/10.1007/s11554-017-0666-7.

[10] B. Penna, T. Tillo, E. Magli, and G. Olmo, "Progressive 3-D coding of hyperspectral images based on JPEG 2000," *IEEE Geoscience and Remote Sensing Letters*, vol. 3, no. 1, pp. 125–129, Jan. 2006, https://doi.org/10.1109/LGRS.2005.859942.

[11] Y. Kang and X. Xu, "A System and Its Implementation Based on FPGA for Video JPEG2000 Codec and Network Transmission," in *Proceedings of the 2021 5th International Conference on Digital Signal Processing*, Chengdu, China, Jun. 2021, pp. 260–265, https://doi.org/10.1145/3458380.3458425.

[12] L. Horrigue, T. Saidani, R. Ghodhbani, J. Dubois, J. Miteran, and M. Atri, "An efficient hardware implementation of MQ decoder of the JPEG2000," *Microprocessors and Microsystems*, vol. 38, no. 7, pp. 659–668, Oct. 2014, https://doi.org/10.1016/j.micpro.2014.06.005.

[13] T. Saidani and R. Ghodhbani, "Hardware Acceleration of Video Edge Detection with Hight Level Synthesis on the Xilinx Zynq Platform," *Engineering, Technology & Applied Science Research*, vol. 12, no. 1, pp. 8007–8012, Feb. 2022, https://doi.org/10.48084/etasr.4615.

[14] L. Kechiche, L. Touil, M. Jemai, and B. Ouni, "A Power-Aware Real-Time System for Multi-Video Treatment on FPGA with Dynamic Partial Reconfiguration and Voltage Scaling," *Engineering, Technology & Applied Science Research*, vol. 12, no. 4, pp. 8996–9004, Aug. 2022, https://doi.org/10.48084/etasr.5099.

[15] "ISO/IEC 15444-1:2000 - JPEG 2000 image coding system." International Standards Organization, Geneva, Switzerland, 2000.

[16] R. Ghodhbani, T. Saidani, A. Alhomoud, A. Alshammari, and R. Ahmed, "Real Time FPGA Implementation of an Efficient High Speed Harris Corner Detection Algorithm Based on High-Level Synthesis," *Engineering, Technology & Applied Science Research*, vol. 13, no. 6, pp. 12169–12174, Dec. 2023, https://doi.org/10.48084/etasr.6406.

[17] L. A. Aranda, A. Sánchez, F. Garcia-Herrero, Y. Barrios, R. Sarmiento, and J. A. Maestro, "Reliability Analysis of the SHyLoC CCSDS123 IP Core for Lossless Hyperspectral Image Compression Using COTS FPGAs," *Electronics*, vol. 9, no. 10, Oct. 2020, Art. no. 1681, https://doi.org/10.3390/electronics9101681.

[18] J. Caba, M. Díaz, J. Barba, R. Guerra, J. A. de la Torre, and S. López, "FPGA-Based On-Board Hyperspectral Imaging Compression: Benchmarking Performance and Energy Efficiency against GPU Implementations," *Remote Sensing*, vol. 12, no. 22, Jan. 2020, Art. no. 3741, https://doi.org/10.3390/rs12223741.