Efficient Hardware Accelerator and Implementation of JPEG 2000 MQ Decoder Architecture

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

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

Albia Maqbool
Department of Computer Sciences, Faculty of Computing and Information Technology, Northern Border University, Saudi Arabia
albia.alam@nbu.edu.sa

Eman H. Abd-Elkawy
Department of Computer Sciences, Faculty of Computing and Information Technology, Northern Border University, Saudi Arabia | Department of Mathematics and Computer Science, Faculty of Science, Beni-Suef University, Egypt
eman.hassan@nbu.edu.sa

Jihane Ben Slimane
Department of Computer Sciences, Faculty of Computing and Information Technology, Northern Border University, Saudi Arabia
jehan.saleh@nbu.edu.sa

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

Faheed A. F. Alrslani
Department of Information Technology, Faculty of Computing and Information Technology, Northern Border University, Saudi Arabia
f.alrslani@nbu.edu.sa

Amjad A Alsuwaylimi
Department of Information Technology, College of Computing and Information Technology, Northern Border University, Saudi Arabia
amjad.alsuwaylimi@nbu.edu.sa

Marouan Kouki
Department of Information Systems, Faculty of Computing and Information Technology, Northern Border University, Saudi Arabia
marouan.kouki@nbu.edu.sa

Amani Kachoukh
Department of Information Systems Faculty of Computing and Information Technology, Northern Border University, Saudi Arabia
amani.khasookh@nbu.edu.sa

Received: 13 February 2024 | Revised: 23 February 2024 | Accepted: 24 February 2024

Licensed under a CC-BY 4.0 license | Copyright (c) by the authors | DOI: https://doi.org/10.48084/etasr.7065

ABSTRACT

Due to the extensive use of multimedia technologies, there is a pressing need for advancements and enhanced efficiency in picture compression. JPEG 2000 standard aims to meet the needs for encoding still pictures. JPEG 2000 is an internationally recognized standard for compressing still images. It provides a wide range of features and offers superior compression ratios and interesting possibilities when compared to traditional JPEG approaches. Nevertheless, the MQ decoder in the JPEG 2000 standard presents a substantial obstacle for real-time applications. In order to fulfill the demands of real-time processing, it is imperative to meticulously devise a high-speed MQ decoder architecture. This work presents a novel MQ decoder architecture that is both high-speed and area-efficient, making it comparable to previous designs and well-suited for chip implementation. The design is implemented using the VHDL hardware description language and is synthesized with Xilinx ISE 14.7 and Vivado 2015.1. The implementation findings show that the design functions at a frequency of 438.5 MHz on Virtex-6 and 757.5 MHz on Zync7000. For these particular frequencies, the calculated frame rate is 63.1 frames per second.

Keywords-JPEG2000; MQ decoder; FPGA; EBCOT; implementation

I. INTRODUCTION

The International Standard Organization/International Electrotechnical Commission (ISO/IEC) and the Joint Photographic Experts Group introduced the JPEG 2000 standard, which is a complex and high-performance digital image coding standard [1-4]. Positioned as the leading image compression standard, its goal is to surpass widely adopted standards such as JPEG. Numerous capabilities are included in JPEG 2000, including compressed domain processing, error resilience, continuous tone and bi-level compression, lossy and lossless compression, region of interest coding, and progressive transmission with pixel precision and resolution [2, 3]. The incorporation of these characteristics improves the efficiency of systems utilizing JPEG 2000 for picture compression. Although JPEG 2000 outperforms JPEG in terms of features and efficiency, it is notable for its considerably greater processing complexity. The improved functionality is accompanied by higher computational complexity and memory demands, mainly due to the utilization of the Embedded Block Coding with Optimized Truncation (EBCOT) technique, which is a fundamental component of JPEG2000 [5]. Multiple studies examining the JPEG2000 decoder algorithm [5, 6] have consistently found that entropy decoding is the operation that requires the most memory and processing time. Thus, the entropy decoder is the optimal choice for hardware acceleration. While the MQ decoding process inherently follows a sequential approach, achieving acceleration for this module is extremely difficult. Prior studies [7-12] have demonstrated that MQ decoder designs can achieve a one-clock-cycle decision, hence enhancing their efficiency. Nevertheless, these architectures exhibit sluggish performance and need a substantial amount of silicon space. As a result, the throughput of these MQ decoders is lower than anticipated.

This study presents a cost-effective and efficient architecture for a high-speed MQ decoder, implemented using the VHDL hardware description language. The outcomes of the proposed design are compared with those of multiple FPGA implementations.

II. JPEG2000 STANDARD

A. Block Diagram – Algorithmic Chain

Six distinct units are involved in the JPEG 2000 decoding process (Figure 1): a bitstream parser, an EBCOT, a dequantization unit, an Inverse Discrete Wavelet Transform (IDWT) unit, a color transform unit, and a tile combiner. The Tier 2 bitstream parser unit is specifically accountable for obtaining the compressed bitstream for every code block and transmitting it to the entropy block decoder by extracting pertinent data from image headers. Subsequently, the code stream is decoded into the wavelet domain by the EBCOT (Tier 1). The image is then arranged in raster order [3, 13] by the dequantization unit, which additionally increases the bit count of every coefficient [14]. The wavelet coefficients are then transformed into pixels using the IDWT, and these pixels are subsequently shifted from the YCbCr color system to the RGB color space using the color transform. Finally, to assemble the image in raster order, the tile combiner aligns the rows of every tile [3].

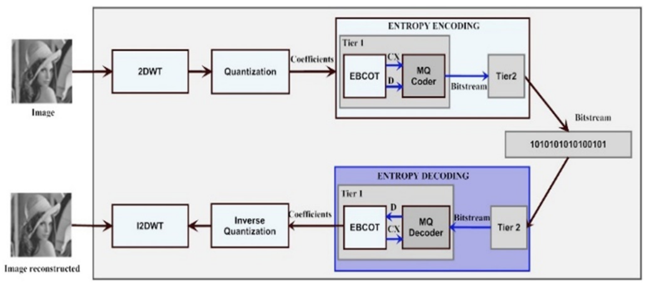

Fig. 1. Block diagram of the JPEG 2000 compression algorithm.

B. Entropy Block Decoder Overview

A simplified functional diagram of the entropy block decoder is illustrated in Figure 2. Two processing units comprise Tier 1: the decoding passes and the MQ decoder.

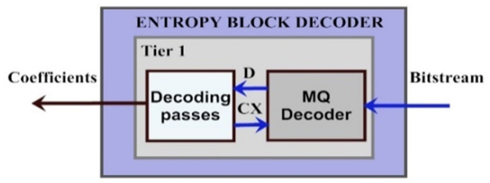

Fig. 2. The entropy block decoder.

a) Figure Labels

The decoding process utilizes the data from neighboring bits and the decoded judgments made by the MQ decoder to determine the importance of each bit in the image. During the decoding process, if a bit is found to be significant, the surrounding bits are used to offer contextual information to the MQ decoder for that particular bit. After obtaining the binary decision from the MQ decoder, the decoding operation must pause before processing the adjacent bits [15]. The decoding process has three separate phases: the Magnitude Refinement Pass (MRP), the Clean-Up Pass (CUP), and the Significance Propagation Pass (SPP) [1, 8]. SPP detects coefficients that are not labeled as significant but have at least one neighboring coefficient that is tagged as significant. The sign of the coefficient is established if the decoded bit is one. MRP focuses on coefficients that were previously determined to be significant in a previous bit plane and decodes the remaining bits of those coefficients. CUP decodes bits that are deemed inconsequential by the preceding two rounds.

b) The Binary Arithmetic Decoder

The binary arithmetic decoder used in JPEG 2000 is widely referred to as the MQ decoder. The likelihood estimate between the Most and the Least Likely Symbol (MPS and LPS, respectively) is considered, along with the context provided by the decoding steps, to calculate the output bit value. The resulting output value is subsequently utilized in the decoding process to assist in determining the importance of neighboring bits and their correct positioning in the final decoded image.

C. Binary Arithmetic Coding (BAC)

BAC executes entropy coding to construct the code stream based on the context data pairs from the bit plan coding. The BAC in JPEG 2000, frequently referred to as the MQ coder, is based on the QM adaptive arithmetic encoder in JBIG but employs the Q coder's byte emission [7, 12]. Like in the Q coder, 0 and 1 are dynamically assigned to MPS and LPS. The MQ coder, as depicted in Figure 3, is based on the recursive probability subdivision of Elias coding. With each binary decision, the current probability interval is divided into two smaller intervals. The code string is modified if needed to indicate to the lower bound of the probability sub-interval for the symbol that occurred. Although the coding method includes the addition of binary fractions rather than the concatenation of integer code words, more probable binary decisions can frequently be coded for less than one bit per decision [11].

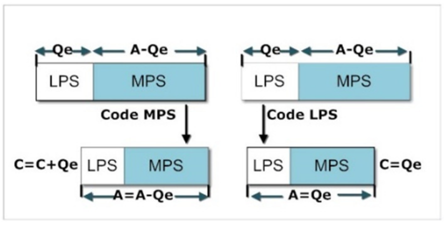

Fig. 3. MQ coder procedure.

D. MQ Decoder's Algorithm

The MQ decoder's algorithm is initialized using the INITDEC procedure, as shown in Figure 4. When all contexts have been read, the context (CX) is read and sent on to the DECODE procedure. The DECODE algorithm decodes the binary decision D and outputs a value of either 0 or 1. It is composed of four procedures: LPS_EXCHANGE, MPS_EXCHANGE, RENORMED, and BYTEIN. DECODE includes a probability estimation table that gives adaptive estimates of probability for every scenario. The compressed data have been decompressed once all CXs have been read, see [3, 5, 6, 11, 16] for a more detailed description of the algorithm.

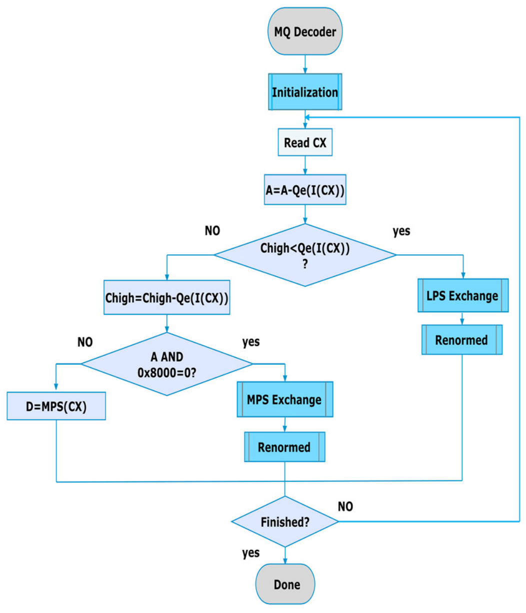

Fig. 4. The MQ-decoder's flowchart.

Figure 5 describes the arithmetic decoder's register configurations:

- In order to save the current interval value, the 16-bit A register is used.
- During the decoding procedure, the partially coded bits are stored in the C register, also known as the code register.
- By combining the Chigh register and the Clow register, we create a 32-bit C register. When we renormalize C, we move one bit of new data from Clow's bit 15 to Chigh's bit 0.

In the A register, the "a" bits stand for the fractional bits of the current interval value, and in the C register, the "x" bits imply the same. The "s" bits represent spacer bits that enforce crucial limitations on carry-over. The "b" bits indicate the specific locations from which entire bytes of data are taken from the C register. The letter "c" represents a carry bit.

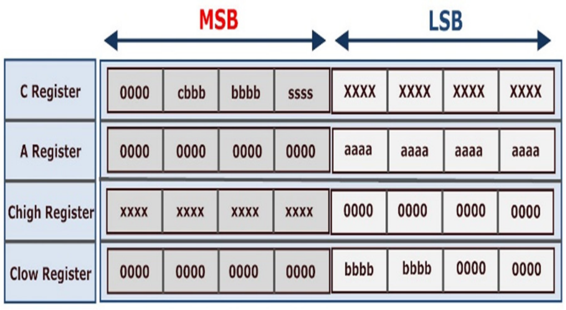
MSB
LSB
C Register
0000 cbbb bbbb ssss xxxx xxxx xxxx xxxx
A Register
0000 0000 0000 0000 aaaa aaaa aaaa aaaa
Chigh Register
xxxx xxxx xxxx xxxx 0000 0000 0000 0000
Clow Register
0000 0000 0000 0000 bbbb bbbb 0000 0000

Fig. 5. Register structures for the MQ decoder.

III. PROPOSED MQ DECODER ARCHITECTURE

The main approach to reduce the execution time of the MQ decoder is to minimize the number of clock cycles needed to generate all the contexts of a picture. Given this, we propose a low-power, efficient hardware solution that obtains a bitstream and a context, then renders a decision in 4 clock cycles. This section provides a comprehensive description of the hardware architecture of our recently launched MQ decoder. Figure 6 depicts a simplified and functional diagram of the MQ decoder.

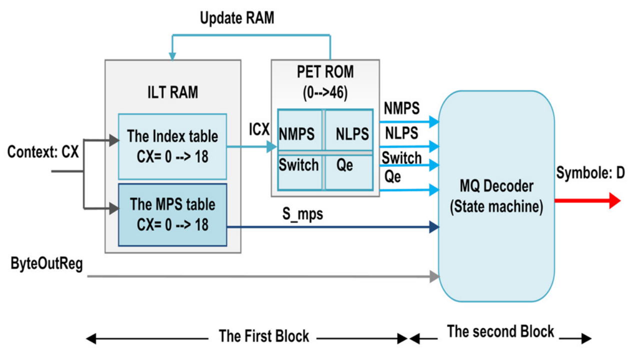

Fig. 6. MQ decoder functional diagram.

For every given CX value, the Bit Plane Coder (BPC) can produce one of nineteen possible outcomes [1]. The CX values are entered into a MQ decoder, where each context is associated with a corresponding item in the Index Look-Up Table (ILT). One field, ICX, is an index, and the other, S_mps, is the MPS value field. These two compose the ILT. While the ICX is used as a reference for the Probability Estimation Table (PET), the S_mps value decides if the symbol '0' or '1' is regarded as an MPS symbol. The initial values of the ILT are determined according to the standard JPEG 2000 and are later adjusted throughout the decoding process. The Positron Emission Tomography (PET) scan provides a detailed visualization of the decoder's internal structure [11, 16]. This architecture is comprised of two components: the probability estimator and the decoding block.

A. *Probability Estimator*

The context block prediction consists of the subsequent components: the ILT_RAM and the PET_ROM. These two principal components are composed, respectively, of 2 RAMs and 4 ROMs, as seen in Table I. The initial phase of the proposed architecture, as seen in Figure 7, commences with the retrieval of the context CX.

TABLE I. TTHE DIFFERENT COMPONENTS USED IN THE PREDICTION BLOCK

	RAM/ROM	Description/Use
ILT_RAM	RAM_ICX	It comprises 19 indexes, which correspond to the current index in the look-up table. Every index is encoded using 6 bits. In this scenario, the I (CX) functions as a pointer to the 4 ROMs.
	RAM_MPS	It has 19 MPSs (current Most Probable Symbols), and each MPS has a single bit of coding.
PET_ROM	ROM_NMPS	It has 47 NLPSs (Next Most Probable Symbols). Every NLPS has a 6-bit code.
	ROM_NLPS	It contains 47 NLPSs. Each NLPS is coded on 6 bits.
	ROM_Switch	It contains 47 elements (Switch). Each element is coded on 1 bit
	ROM_Qe	It consists of 47 components. Every element is encoded using 16 bits.

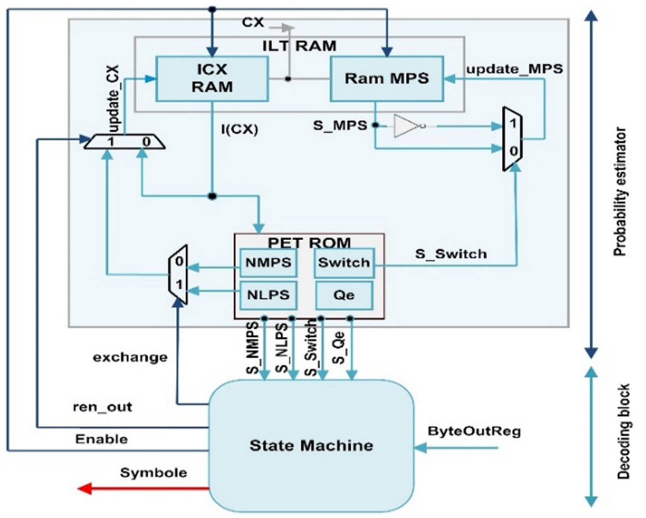

Fig. 7. Internal architecture of the MQ decoder.

To obtain the values of I (CX) and mps_D, the address bus is utilized to transfer the CX value to two RAMs, specifically RAM I (CX) and RAM MPS. Once the provided I(CX) value has been fed into each of the four ROMs (ROM_NMPS, ROM_NLPS, ROM_Switch, and ROM_Qe), the matching entries can be accessed. In cases where renormalization is required, the NMPS or NLPS value is copied and stored in the RAM (CX). Additionally, the S_MPS bit in RAM_MPS is modified when the S_Switch signal is very high.

B. The Decoding Block

In order to explain the functioning of the MQ decoder, we developed a state machine consisting of 11 states. We have created a hardware description of the process that is synchronized by the state machine, as depicted in Figure 8.

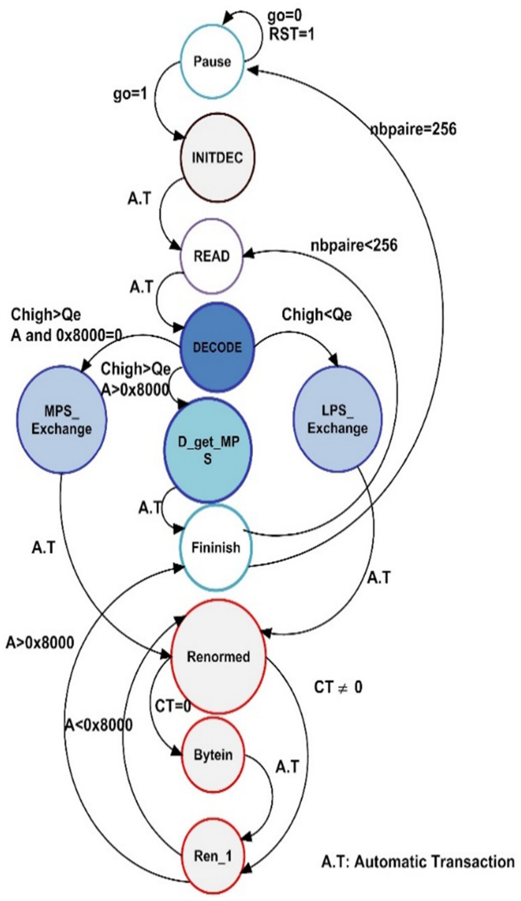

Fig. 8. The proposed state machine.

The state machine operates using a sequential pseudocode. Its primary states are:

- INITDEC: This state is utilized to initiate the MQ decoder. The compressed data pointer, BP (Byte Pointer B), is initialized to BPST (the starting value of BP). The initial byte of the compressed data is transferred to the least significant byte of Chigh, and subsequently, a new byte is retrieved. Ultimately, the register C undergoes a shift of 7 bits, the value of CT is decremented by 7, and the register A is set to the hexadecimal value 0x8000.
- BYTEIN: As seen in Figure 8, the bytein state is called by the RENORMD state when CT equals zero. This state handles the processing of a single byte of input, taking into consideration any extra bits that may come after the hexadecimal FF byte. The Clow and Chigh registers are concatenated to generate the Reg_C in Bytin. B is updated and put in the highest 8 bits of the Clow_register if it is not a hexadecimal FF byte. In this case, the value of BP is decremented. Indeed, B represents the byte specified by the compressed data buffer address BP. The byte B1, situated at the memory address BP+1, is examined to ascertain its equivalence to the hexadecimal value FF. When B1's value is greater than 8F in hexadecimal notation, the data byte in register C is incremented by adding the hexadecimal value FF, and bit counter CT is set to 8.
- Ren_1: During this state, both Reg_A and Reg_C are shifted to the left, and the counter CT is decremented by 1. If the value of reg_A is less than the hexadecimal value 8000, then the Renormed state is executed. Otherwise, the final state was executed successfully.
- FINISH: in this state if the number of pairs (nbpaire) is equal to the total number of CX, we will transition back to the pause state. Otherwise, the read operation will be repeated.

IV. IMPLEMENTATION AND RESULTS

A. Implementation

The proposed design was successfully tested on three platforms, namely Virtex5, Virtex4, and Spartan3A. This implementation utilized the default configurations of Xilinx ISE 14.7. Table II displays the maximum clock frequency and device consumption overview for the suggested MQ decoder design. It is possible for the proposed design to operate at 394.3 MHz, 320 MHz, and 176.8 MHz, respectively. The average decoding time for one decision is 4.10 cycles.

TABLE II. SYNTHESIS RESULTS FOR THE SUGGESTED DESIGN USING XILINX ISE 14.7

Used FPGA	XC5VLX50T	XC4VLX15	XC3SD3400a
Total slices LUTs	420	605	654
Total slices registers	288	322	345
Total FF pairs	197	289	290
Max. Frequency (MHz)	394.3	320	176.8

Using the default settings of Xilinx Vivado 15.1, the architecture's performance was assessed in terms of operating frequency and hardware requirements. The proposed MQ decoder architecture has been effectively implemented on three platforms: Zync7000, KINTEX7, and VIRTEX7, while adhering to all user-imposed constraints for implementation. Table III displays the summary of hardware requirements, Worst Negative Slack (WNS), data path delay, and requirements. Slack is determined by subtracting the arrival time from the required time [17]. WNS represents the amount of free time available once the timing constraints have been met. The time period for 1 clock cycle at a frequency of 100 MHz is 10 ns, as specified in the constraints file for the purpose of completing the execution.

TABLE III. SYNTHESIS RESULTS FOR SUGGESTED ARCHITECTURE USING VIVADO 15.1

Used FPGA	Zync7000 xc7z045	KINTEX7 XC7K70T	VIRTEX7 XC7VX415T
Total FF slices	260	260	260
LUT	372	384	386
Memory LUT	14	14	14
WNS (ns)	3.686	3.612	3.502
Data path delay (ns)	0.817	0.94	0.803
Requirements (ns)	5	5	5
Frequency (MHz)	757.7	720	667.5
Throughput (MB/s)	118.35	112.47	104.27

In order to ascertain the maximum clock frequency, it is necessary to initially establish the required clock period [17, 18]. Upon implementation of the proposed design on Zynq 7000, the required clock period is calculated as 1.32 ns (by subtracting 3.68 ns from 5 ns), which gives a frequency of 757.7 MHz and a throughput of 118 MB/s.

B. *Comparison*

In order to assess the efficiency of our architecture in terms of hardware demands and operational speed, we conducted comparative analysis with other pre-existing architectures that employed the identical FPGA. The maximum frequency values reached by the proposed design and existing architectures using the same FPGA are shown in Table IV.

TABLE IV. COMPARISON OF THE MQ DECODER WITH THE ARCHITECTURES FROM [8, 9]

Used FPGA	Architecture [8]	Architecture [9]		Proposed	
	Frequency (MHz)	Frequency (MHz)	FPS	Frequency (MHz)	FPS
Virtex-6 XC6VLX75T-3FF68	437.1	-	-	438.5	63.1
Virtex-4 XC4VLX15-12FF68	-	195.3	28.07	321.1	46.2
Virtex-5 XC5VLX30-3FF324	-	222.8	32.02	394.1	56.7

From the comparison, it can be inferred that the proposed MQ decoder runs at a greater frequency. For example, the operating frequency of the suggested design, when targeted at Virtex 5, is 1.76 times greater than that of current architecture [9]. The frame rate at 321.1 MHz is predicted to be 46.2 Frames Per Second (FPS). However, at 394.1 MHz and 438.5 MHz, the frame rates are 56.7 FPS and 63.1 FPS, respectively. These values are specific to high-definition TVs with a resolution of 1920p.

The frequency achieved with the Virtex2 XC2V6000-6 platform and the average clock cycle obtained by using a counter in state machine HDL to build the design are used to calculate the theoretical throughput. Equation (1) is utilized to ascertain the throughput of sources [11, 19]. The throughput and the average number of clock cycles from references [10, 11, 19, 20] are shown in Table V. Equation (1) calculates the throughput of the binary arithmetic decoder in MB/s. The terms w_{CB} and h_{CB} denote the width and height of each code block, measured in pixels, d_{CB} and f denote the pixel depth and design frequency in MHz, respectively, whereas μ represents the number of cycles needed per code block [11]. The design comparison uses (64 × 64) for the height and width per code block, and a byte for the depth of every each pixel. The processed data has a cumulative size of 4.096 bytes.

$$\text{Throughput (MB/s)} = w_{CB} \times h_{CB} \times d_{CB} \times f/\mu \qquad (1)$$

TABLE V. MQ DECODER THROUGHPUT

Used FPGA	Virtex-2 XC2V6000-6						
Design	[9]	[10]	Proposed	[11]	[19]	[20]	[21]
No slice registers	328	498	313	313	-	315	-
No slice LUTs	579	944	640	630	-	586	-
Frequency (MHz)	142	140.4	200.2	205.5	45.6	157.2	127.9
Frame rate supported (FPS)	20.41	-	350	29.56	-	-	-
Avg. no of clock cycles	-	74.23	26.22	28.44	32.7	68.53	-
Throughput (MB/s)	-	7.74	31.27	35.74	5.7	9.4	5.3

Table V presents the expected frame rate, maximum cycle count, maximum clock frequency, and logic utilization of the proposed MQ decoder design compared to other existing architectures implemented in Virtex 2. Each code block in the recommended design requires 26,223 clock cycles in total. The proposed design produces a throughput of 31.27 based on the outcomes of FPGA synthesis. Compared to the architecture indicated in [18], it has a throughput that is 5.48 times higher. Table V demonstrates a significant decrease of 36% in memory requirements for the proposed architecture in comparison to the architecture mentioned in [10]. The predicted frame rate of the suggested MQ decoder design is 1.41 times higher than that of the MQ decoder architecture in [9].

V. CONCLUSION

This paper presents a detailed explanation of the advanced design of an MQ arithmetic decoder, which is built upon the MQ decoder technique used in JPEG 2000. The VHDL hardware description language was used to implement the proposed design, which was then synthesized for FPGA devices. The zynq7000 device achieves a maximum working speed of 757.7 MHz. For high-definition TV with a resolution of 1920 pixels, the estimated frame rate is 63.1 FPS. Furthermore, the memory requirement for the proposed architecture is diminished by 37.1% in comparison to the other existing architectures, and the proposed design achieves the highest maximum frequency among the compared designs. Hence, the proposed design can be effortlessly transferred to Xilinx architectures and has the ability to decode 63.1 FPS of high definition (HD, 1920 × 1080 pixels). This makes it a promising option for deployment as a high-speed real-time

JPEG 2000 decoder in diverse applications such as medical imaging, satellite imagery, digital cinema, and mobile applications.

ACKNOWLEDGMENT

The authors extend their appreciation to the Deanship of Scientific Research at Northern Border University, Arar, KSA for funding this research work through the project number "NBU-FFR-2024-1662-02."

REFERENCES

[1] *JPEG 2000 Part I: Final Draft International Standard (ISO/IECFDIS15444-1)*. ISO/IEC JTC1/SC29/WG1 N1855, 2000.

[2] *ISO/IEC JTC 1 / SC 29 /WG 1, (ITU-T SG8) Coding of Still Pictures.* JBIG, 1999.

[3] D. S. Taubman and M. W. Marcellin, *JPEG2000 Image Compression Fundamentals, Standards and Practice*. Boston, MA, USA: Springer US, 2002.

[4] D. Santa-Cruz, R. Grosbois, and T. Ebrahimi, "JPEG 2000 performance evaluation and assessment," *Signal Processing: Image Communication*, vol. 17, no. 1, pp. 113–130, Jan. 2002, https://doi.org/10.1016/S0923-5965(01)00025-X.

[5] D. Taubman, "High performance scalable image compression with EBCOT," *IEEE Transactions on Image Processing*, vol. 9, no. 7, pp. 1158–1170, Jul. 2000, https://doi.org/10.1109/83.847830.

[6] A. Samet, M. B. Ayed, M. Loulou, and N. Masmoudi, "Comparison between JPEG and JPEG2000 still image compression standard," in *Proc. Visualization, Imaging, and Image Processing*, 2002.

[7] K. Sarawadekar and S. Banerjee, "VLSI design of memory-efficient, high-speed baseline MQ coder for JPEG 2000," *Integration*, vol. 45, no. 1, pp. 1–8, Jan. 2012, https://doi.org/10.1016/j.vlsi.2011.07.004.

[8] D. J. Lucking, E. J. Balster, K. L. Hill, and F. A. Scarpino, "FPGA implementation of the JPEG2000 binary arithmetic (MQ) decoder," *Journal of Real-Time Image Processing*, vol. 8, no. 4, pp. 411–419, Dec. 2013, https://doi.org/10.1007/s11554-011-0214-9.

[9] O. C. Kulkarni, K. Sarawadekar, and S. Banerjee, "VLSI implementation of MQ decoder in JPEG2000," in *IEEE Technology Students' Symposium*, Kharagpur, India, Jan. 2011, pp. 193–197, https://doi.org/10.1109/TECHSYM.2011.5783844.

[10] A. Descampe, F.-O. Devaux, G. Rouvroy, J.-D. Legat, J.-J. Quisquater, and B. Macq, "A Flexible Hardware JPEG 2000 Decoder for Digital Cinema," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 16, no. 11, pp. 1397–1410, Nov. 2006, https://doi.org/10.1109/TCSVT.2006.884573.

[11] 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.

[12] S. D. Jayavathi and A. Shenbagavalli, "FPGA-based Auxiliary Minutest MQ-coder architecture of JPEG2000," *Journal of Real-Time Image Processing*, vol. 16, no. 5, pp. 1765–1779, Oct. 2019, https://doi.org/10.1007/s11554-017-0683-6.

[13] K. Liu, Y. Zhou, Y. Song Li, and J. F. Ma, "A high performance MQ encoder architecture in JPEG2000," *Integration*, vol. 43, no. 3, pp. 305–317, Jun. 2010, https://doi.org/10.1016/j.vlsi.2010.01.001.

[14] T. Acharya and P.-S. Tsai, *JPEG2000 Standard for Image Compression: Concepts, Algorithms and VLSI Architectures*, 1st ed. Hoboken, NJ, USA: Wiley-Interscience, 2004.

[15] R. Ghodhbani, T. Saidani, L. Horrigue, A. M. Algarni, and M. Alshammari, "An FPGA Accelerator for Real Time Hyperspectral Images Compression based on JPEG2000 Standard," *Engineering, Technology & Applied Science Research*, vol. 14, no. 2, pp. 13118–13123, Apr. 2024, https://doi.org/10.48084/etasr.6853.

[16] N. Ramesh Kumar, W. Xiang, and Y. Wang, "Two-Symbol FPGA Architecture for Fast Arithmetic Encoding in JPEG 2000," *Journal of Signal Processing Systems*, vol. 69, no. 2, pp. 213–224, Nov. 2012, https://doi.org/10.1007/s11265-011-0655-1.

[17] 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.

[18] T. Saidani, R. Ghodhbani, A. Alhomoud, A. Alshammari, H. Zayani, and M. B. Ammar, "Hardware Acceleration for Object Detection using YOLOv5 Deep Learning Algorithm on Xilinx Zynq FPGA Platform," *Engineering, Technology & Applied Science Research*, vol. 14, no. 1, pp. 13066–13071, Feb. 2024, https://doi.org/10.48084/etasr.6761.

[19] H.-H. Chen, C.-J. Lian, T.-H. Chang, and L.-G. Chen, "Analysis of EBCOT decoding algorithm and its VLSI implementation for JPEG 2000," in *2002 IEEE International Symposium on Circuits and Systems (ISCAS)*, Feb. 2002, vol. 4, https://doi.org/10.1109/ISCAS.2002.1010457.

[20] D. J. Lucking, E. J. Balster, K. L. Hill, and F. A. Scarpino, "FPGA implementation of the JPEG2000 binary arithmetic (MQ) decoder," *Journal of Real-Time Image Processing*, vol. 8, no. 4, pp. 411–419, Dec. 2013, https://doi.org/10.1007/s11554-011-0214-9.

[21] T. Zhu, J. Zhou, and S. Liu, "Design and implementation of JPEG2000 arithmetic decoder based on Handel-C," in *2009 3rd International Conference on Anti-counterfeiting, Security, and Identification in Communication*, Hong Kong, China, Aug. 2009, pp. 505–508, https://doi.org/10.1109/ICASID.2009.5276988.