Design of a High Speed Architecture of MQ-Coder for JPEG2000 on FPGA

Taoufik Salem Saidani
Department of Computer Sciences
Faculty of Computing & Information Technology
Northern Border University, Rafha, Saudi Arabia

Hafedh Mahmoud Zayani
Department of Information System
Faculty of Computing & Information Technology
Northern Border University, Rafha, Saudi Arabia

***Abstract*—Digital imaging is omnipresent today. In many areas, digitized images replace their analog ancestors such as photographs or X-rays. The world of multimedia makes extensive use of image transfer and storage. The volume of these files is very high and the need to develop compression algorithms to reduce the size of these files has been felt.**

The JPEG committee has developed a new standard in image compression that now also has the status of Standard International: JPEG 2000. The main advantage of this new standard is its adaptability. Whatever the target application, whatever resources or available bandwidth, JPEG 2000 will adapt optimally. However, this flexibility has a price: the JPEG2000 perplexity is far superior to that of JPEG. This increased complexity can cause problems in applications with real-time constraints. In such cases, the use of a hardware implementation is necessary. In this context, the objective of this paper is the realization of a JPEG2000 encoder architecture satisfying real-time constraints. The proposed architecture will be implemented using programmable chips (FPGA) to ensure its effectiveness in real time. Optimization of renormalization module and byte-out module are described in this paper. Besides, the reduction in computational steps effectively minimizes the time delay and hence the high operating frequency.

The design was implemented targeting a Xilinx Virtex 6 and an Altera Stratix FPGAs. Experimental results show that the proposed hardware architecture achieves real-time compression on video sequences on 35 fps at HDTV resolution.

***Keywords*—MQ-Coder; High speed architecture; FPGA; JPEG2000; VHDL**

I. Introduction

The current development of computer networks and the dramatic increase in the speed of processors reveal many new potentialities for digital imaging. Whether in the medical, commercial or military field, new applications are emerging each with its specificities. The JPEG Group has developed a new, more flexible and better image encoding standard: JPEG2000 [1]. It is built around a wide range of image compression and display tools. This makes the algorithm appealing to many applications, whether for Internet broadcasting, medical imaging or digital photography [2].

The main JPEG2000 coding steps are shown in Fig. 1. Several features are available for encoding, such as progressive quality and/or resolution reconstruction, fast random access to compressed image data, and the ability to encode different regions of the image called regions of interest (ROI).

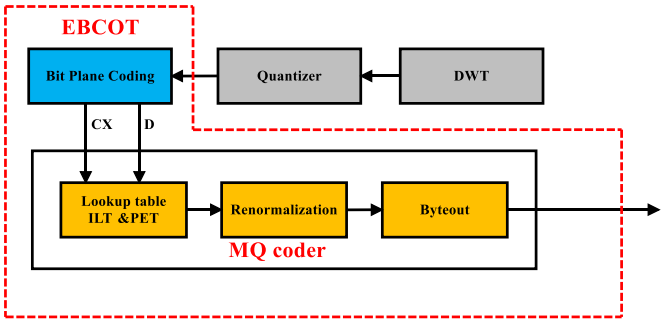

Fig. 1. Overview of JPEG2000 coding process.

The JPEG2000 standard can be broken down into several successive blocks. The original image is cut into tiles after the component transformation. All the tiles are then transformed into wavelets (transformation with or without loss), independently of each other [3]. The wavelets used in the JPEG2000 standard are bi-orthogonal, that is to say different wavelets are used for decomposition and reconstruction. Two types of bi-orthogonal wavelets are used: wavelets of Daubechies 9/7 and Le Gall 5/3 [4], [5]. These two wavelets are chosen according to the type of compression desired, lossless or lossy. Le Gall 5/3 wavelets used to perform a reversible transform are used for lossless compression. The wavelets of Daubechies 9/7 allowing realizing a reversible transform are used only for lossy compression.

The coefficients of the block-code undergo quantization and the quantized coefficients are decomposed into bit planes. The quantification minimizes the number of bits necessary for coding the supplied coefficients of the preceding block, by retaining only the minimum number of bits making it possible to obtain a certain quality level [6], [7].

Based on the wavelet decomposition technique, JPEG2000 is very different from previous standards and has many advantages that will allow it to be adopted in a wide range of applications, or even to be extended to video encoding. In contrast, this type of compression requires much more computational power than the original JPEG process, which makes software implementations irrelevant when very fast processing is required. Fig. 2 shows the comparison between JPEG and JPEG2000 in terms of performance. We note that the performance of JPEG2000 is greater than those of JPEG standard.

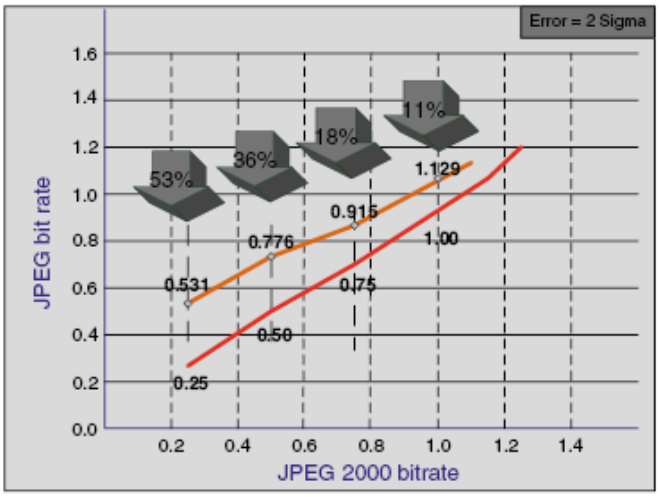

Fig. 2. Comparison between JPEG and JPEG2000 in terms of performance.

Due to many successive processes, JPEG2000 requires more computing power to achieve encoding and decoding speeds similar to JPEG. A hardware solution is therefore indispensable for fast applications.

The JPEG2000 image compression standard was created to meet the new requirements arising from the diversification of applications in the multimedia field. The many features that it offers bring a new breath to this sector. However, they have led to an increase in the complexity of the algorithm compared with existing standards. Faced with this complexity, a hardware encoder is the solution that allows satisfying the real-time constraints of certain applications.

This paper presents an FPGA-based accelerator core for JPEG2000 encoding. Comparison with various FPGA implementations is provided.

Contributions in this work are listed as follows:

1) The proposed high speed efficient MQ-coder architecture modifies the probability estimation (Qe) representation to minimize the memory consumption. The modification in probability estimation reduces the bitwise representation to 13 bit.

2) Due to the less memory occupation, the time and power required for the hardware-based JPEG2000 compression are reduced. Thereby, the operating speed is improved (more operating frequency) with the help of proposed MQ encoder for real time image processing.

3) The minimization in bitwise representation in proposed architecture of MQ coder reduces the count of memory elements to (32 9 13) 416 that leads to the preservation of silicon (Si) area further in the compact chip development.

4) The optimization of Renormalization and Byteout modules help speeding up the proposed architecture.

The remainder of this paper is decomposed into six sections. After the introduction, Section 2 details the JPEG2000 MQ encoder. Previously proposed hardware architectures for MQ-coder are described in Section 3. Section 4 describes the proposed hardware architecture of MQ coder. In Section 5, experiments and results are detailed. Finally, this paper is concluded in Section 6.

II. JPEG2000 MQ-CODER

The arithmetic coder used in JPEG2000, called the MQ encoder, takes as inputs the binary values D and the associated contexts CX resulting from the preceding step of binary modeling of the coefficients, and this in the order of the coding passes. Fig. 3 shows the arithmetic encoder inputs and outputs.

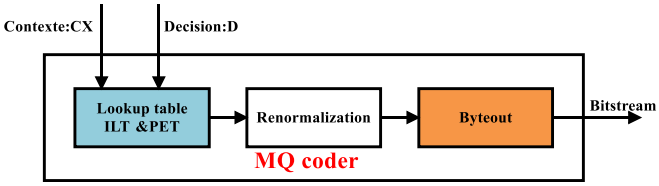

Fig. 3. The inputs and outputs of MQ-Coder.

Rather than representing the intervals associated with the probabilities of “0” or “1”, it was chosen to represent the data using the LPS (Less Probable Symbol) and MPS (More Probable Symbol) symbols, respectively representing the probabilities occurrence of the minority and majority species. Obviously, it is necessary to keep track of the meaning attributed to one or the other of the variables “0” or “1” is the minority species.

Thus the current interval is represented by the interval 1, which is then divided into two sub-intervals corresponding to the minority and majority species. From a representation point of view, LPS is always given as a lower interval. Each binary decision, represented by a bit, is divided recursively. The divisions are made to estimate the probability of Elias: MPS and LPS.

The binary sequence from the MQ is divided into a number of packets. Each of them contains the bit-stream corresponding to the same component, the same resolution level, the same quality layer and the same spatial zone of the resolution level. The spatial areas of each resolution level are called precincts. Each of the packets is preceded by a header containing information allowing identifying very precisely the data conveyed by this packet.

Four different progress orders are defined in JPEG 2000. They make it possible, during the decoding, to obtain in priority either the data of the same component, or those of the same resolution level, or those of the same quality layer, or those of the same spatial zone of the image.

In JPEG2000, the realization of the arithmetic coder is performed by means of an index table. The table represents the LPS probability estimate (Qe). For each input pair (decision, context), we look for the most probable symbol in a variable containing the different states. As each state is represented in the index table, the context can be associated with the index of the table. On its side, the decoder has the index replica of the table, which makes it possible to carry out the decoding.

The Finite state Machine (FSM) with 47 states defines the Probability Estimation Table structure clearly. The number of calculations to obtain the coding information and the utilization of resources are high that degrade the hardware performance. The hardware modeling of MQ-coder contains the following limitations:

5) Low clock frequency.
6) Consumption of LUTs and registers is more.
7) High hardware resource requirements.

A large number of context and decision pairs in MQ encoder shift the parallel operation into a serial operation; such a new architecture is called high-speed MQ encoder architecture. The storage of transformed coefficients in code block consumes more registers that lead to large flip-flop (FF) requirement. Hence, the reduction in code block based on the context pair probability estimation reduces the number of lookup tables (LUTs) and slice registers that leads to less memory consumption. The motivation behind the research work proposed in this paper is the reduction in memory, time and power consumption by reducing the size of the bitwise representation.

III. Related Works

The main advantage of JPEG 2000 was to combine most of these qualities, allowing using it in a very wide range of applications. This flexibility, coupled with a very high compression efficiency, unfortunately has a price. Moreover, some applications have real-time aspects which impose very high flow constraints.

An architecture composed of three stages is proposed by Mei *et al*. in [5]. When implemented on an APEX20K FPGA board, it operates with 37.27 MHz. Indeed in this architecture, if the state MPS occurs then two symbols will be coded simultaneously, if not a single symbol will be coded.

Shi *et al*. [8] proposed a MQ-coder hardware core that allows treating two symbols. Indeed, this architecture is based on the following hypothesis: a maximum of two offsets occurs when there is a renormalization operation. The architecture proposed in [9] is composed by three blocks. The first block is responsible for initializing register A at 0x8000, register C at 0, table MPS (Cx) at 0 and index table in ILT RAM, and perform all arithmetic operations. The second block is used to shift the registers A and the register C and to decrement the counter CT by 1. If CT = 0, the third block will be activated and the register B will be emitted as compressed data. The proposed architecture was implemented on a Startix FPGA and works at a frequency of 83.271 MHz.

The complexity of the JPEG 2000 algorithm is a problem for these real-time applications. In [10], the author indicates that in view of current technology, it is not possible for purely software implementations to respect the constraints imposed by these real-time applications. This is the reason why a growing number of companies and researchers are interested in (partially) hardware-related achievements of the standard, in which the computing resources have been optimized and the memory requirements reduced.

Below we give an overview of the hardware achievements to date and the results obtained.

As part of the PRIAM project, Thales Communications has developed an implementation of a JPEG 2000 encoder on an MPC74XX processor. This is studied in [11]. The MPC74XX processor is based on a PowerPC architecture (RISC type processor) to which is added a vector calculation unit called AltiVec. This allows multiple data sets to be processed in parallel in a single instruction.

Unlike the other blocks in the decoding chain, the entropy coder, due to its non-systematic behavior, is complex to optimize by means of vectorial instructions. This achievement gives overall very good results, but the entropy coder, requiring 400 cycles per 8-bit pixel, is truly the “bottleneck” of the system.

Bonaldi [12] has been working on the creation of a mixed software-hardware encoder. The medium used is the ARM-VIRTEX card of the DICE unit. An input rate of 6.6 Mbps for the entropy encoder is especially supported. Moreover, everything concerning the formation of the bit-stream is carried out in software, on the ARM. This approach of Co-Design is very judicious and is moreover widely supported by the literature.

The Amphion company offers an ASIC encoder-decoder available since 2003 [13]. Amphion announces speeds of 480 Mbps at encoding and 160 Mbps at decoding. This embodiment has interesting characteristics, such as the few constraints on the format of the input images, a division of tasks between hardware and software and an architecture compatible with the AMBA bus, which allows easy integration into other systems.

Analog devices [14] offer the ADV-JP2000. This circuit operates at a maximum 20 MHz frequency including a 5/3 wavelet transform (no 9/7) and an entropy encoder. The circuit is not fully compliant with the standard. The ADV-JP2000 offers two modes of operation: encodes and decodes. In the encode mode it accepts a single tile and generates the stream of code-blocks conforming to the standard. The ADV-JP2000 communicates via an asynchronous protocol but also allows an interrupt mode. Finally, the circuit supports the DMA mode.

Zhang *et al*. [15] proposed an architecture composed of four stages and three parts (P1, P2, P3). Indeed, P1 is implemented in Stage 1 to determine the new value of Qe when A < (0x8000). The P2 is called in Stage 2 and Stage 3, because the latter updates the Reg A and Reg C and also to perform the arithmetic operations and the offset operations. Finally, the P3 is used in Stage 4 to realize the bit stuffing when the counter CT is equal to 0. The processing frequency of this architecture was 110 MHz on an Altera FPGA card.

IV. Proposed Architechture

The proposed architecture of the encoder is shown by the block diagram of Fig. 4. The pairs (C, D) are received by the MQ coder as input and a sequence of bytes called ByteOutReg are provided as output. This architecture consists of two parts: the part of the prediction of the probability of the symbol to be

coded composed of 2 RAMs (ICX, MPS) and 4 ROMs (NMPS, NLPS, Switch, Qe), and the coding part which is composed of a state machine.

The four ROMs are not updated during coding operation. The pairs (CX, D) are first sequentially read. Subsequently, the CX context will be transmitted over the bus address of the ICX RAM and the MPS RAM. Then the value of I(CX) and MPS(CX) will be read. Then the I(CX) index will be delivered to the four ROMs. The mps_D will be executed with signal D, which causes the LPS_en signal to be generated. If this signal is equal to one then the CODELPS state will be carried out otherwise the CODEMPS state will take place.

The updating of the ICX RAM depends essentially on the signal Ren_out. Indeed this signal will set to one if the renormalization is carried out. However, the MPS RAM will update if the LPS_SW signal is equal to one. The Probability estimation architecture is shown in Fig. 5.

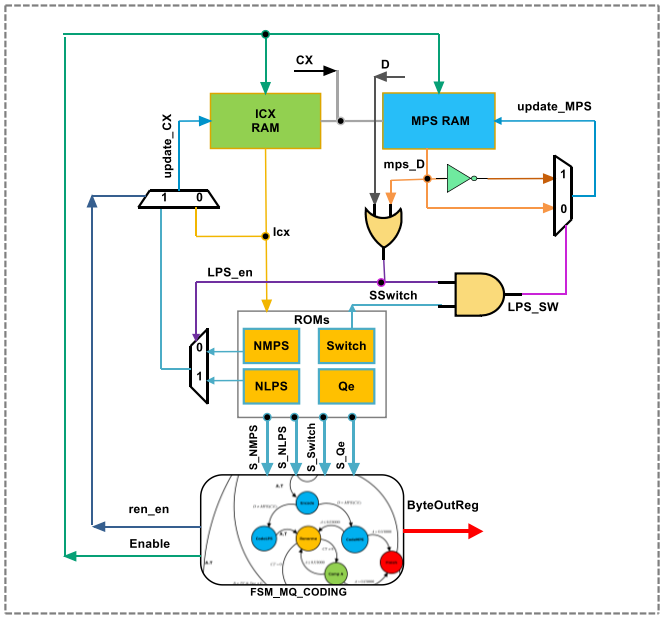

Fig. 4. MQ-Coder architecture.

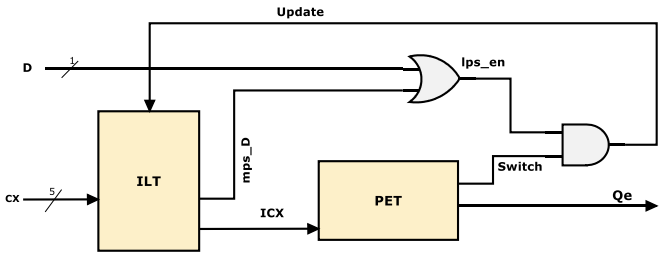

Fig. 5. Probability estimation architecture.

We are then interested in the coding part to manage the process of the coding by a machine of finite states by substituting the various sub-algorithms by states. The outputs depend on the current state and the inputs and react directly to changes in inputs. Fourteen states have been set up in order to describe the MQ encoder process. Fig. 6 shows the MQ-Coder state machine. The states used in this state machine are as follows:

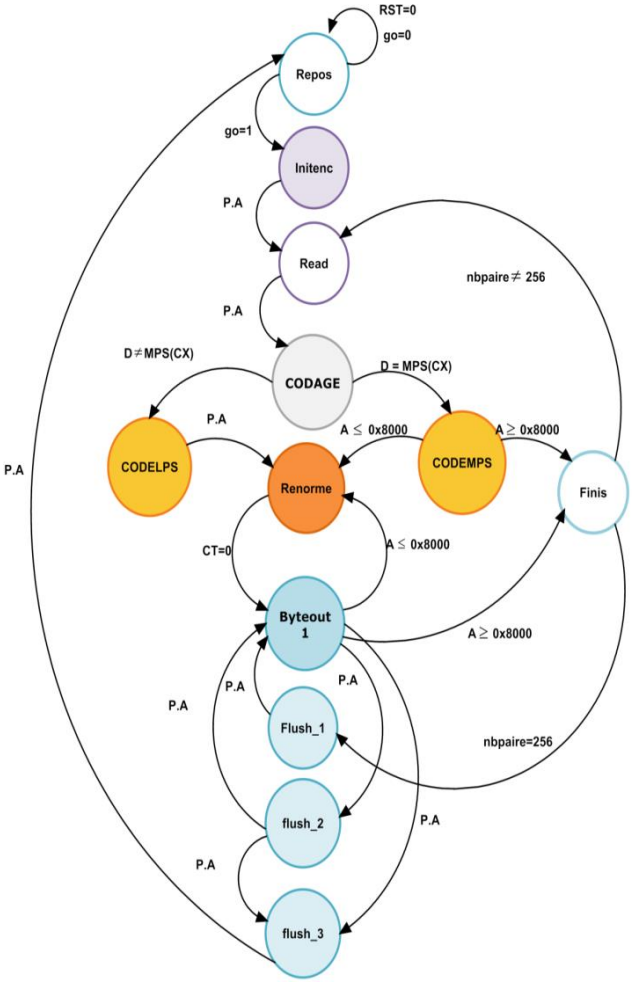

Fig. 6. MQ-Coder State machine.

8) **Repos:** This state essentially depends on the input "go", if go = 1 it switches to the state INITENC otherwise it remains in the same state.

9) **Initenc**: In this state, register Reg_A at (0x8000), register Reg_C at 0 and counter CT at 12 are initialized. Then the MPS RAM is filled with 0s and the RAM of the indexes by the 19 possible values of the context CX. Then you will automatically play (Read). The index/probability tables should be presented in the memory before coding begins.

10)**Read:** In this state, the context is read to deduce the value of the corresponding MPS (Cx) according to the table initialized in INITENC. We also read Decision D.

11)**CODAGE**: If Decision D = MPS (Cx), it switches to the CODEMPS state otherwise it goes to the CODELPS state.

12)**CODEMPS**: The register Reg_A is adjusted to Qe_reg. Then Reg_A is compared to (0x8000). If Reg_A is less than (0x8000), the index will be updated according to the NMPS table of the context index and the Renorme state will be used. If register Reg_A is greater than (0x8000), we add

the probability Qe_reg to the register Reg_C and we will pass to the Finis state.

13)**Finis**: In this state if the even number (number of pairs (CX, D) lu) is equal to 256 then we pass to the flush state, otherwise the next state will be Read.

14)**CODELPS**: In this state, register Reg_A to Reg_A-Qe_reg is adjusted. Then, if Reg_A is greater than Qe_reg then the register Reg_A takes the value of Qe_reg, otherwise we add the probability Qe_reg to the register Reg_C. The condition of inversion of the intervals is always checked. If a SWITCH is required, the direction of the MPS will be reversed. The index will take a new value according to the NLPS table and it will change to the Renorme state.

15)**Renorme**: The contents of Reg_A and Reg_C will be replaced by a simple left shift. This shift repeats until the value of Reg_A is raised above (0x8000). The counter CT containing the number of shifts of Reg_A and Reg_C will then be decremented at each offset. When the counter CT reaches 0 (CT was initially at 13, i.e., 13 left offsets were made at Reg_A and Reg_C), it will pass to byteout1 and if Reg_A is still less than (0 x 8000), we will return to the Renorme state as soon as we have finished with byteout. The optimization of the Renormalization procedure is presented in Fig. 7.

16)**Byteout1**: This state can be called in two states either in the Renorme state when the shift counter CT becomes equal to zero, or also at the end of the coding when the registers flush. The optimization of the Byteout procedure is presented in Fig. 8.

17)**FLUSH**: This is the state we reach towards the end of the encoding (in our case if nbpair = 256). The FLUSH procedure contains two calls to Byteout1 and two calls to Setbit; hence, the idea of subdividing it into three states: the first is flush_1 which ends with a call to byteout1, the second is flush_2 (same principle of fulsh_1) and the third is flush_3.

a) **Flush_1**: This state contains two sub-states, the first is the Setbit, the second is byteout1. First we make a call to the state Setbit then we apply an offset to the register Reg_C, then we make a call to byteout1 and we end by making a call to flush_2.

i) **Setbit:** In this state, the Reg_C register shift is automatically changed to byteout1, regardless of the Reg_C register value (i.e., Reg_C is lower or higher than TEMPC).

b) **Fluch_2**: This state has the same principle of the state flush_1 but this time we pass to state flush_3.

c) **Flush_3:** This state contains the end of the flush, when the first end marker 0xFF has to be inserted. The next state will be rest.

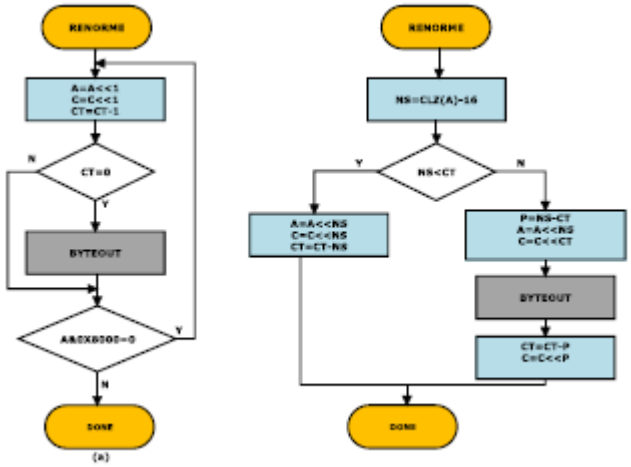

Fig. 7. (a) Original RENORME architecture (b) Optimized RENORME architecture.

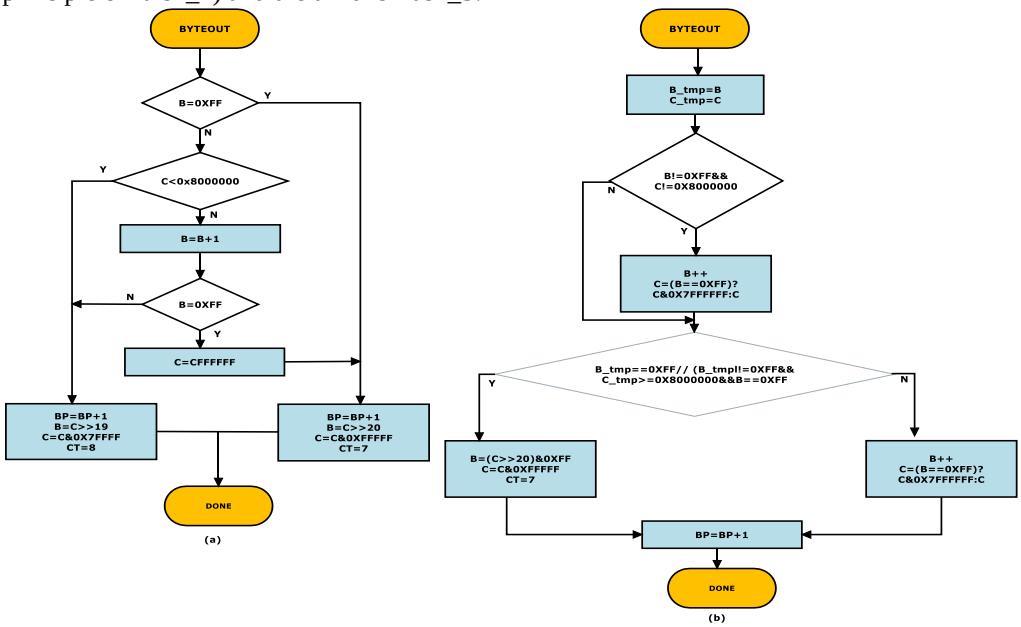

Fig. 8. (a) Original BYTEOUT architecture (b) Optimized BYTEOUT architecture.

V. Experimental Results

A. Simulation

Simulation of the proposed design, using VHDL HDL, is carried out with Mentor graphic.

ByteOutReg bytes coincide with those in column B and the parameters IDCX, MPS, Qe_reg, Reg_A, Reg_C and Reg_CT have evolved appropriately. This result has been well verified and we have chosen to take a sequence to visualize it in simulation and explain it in parallel. For the sake of clarity in Fig. 9, we have chosen to display some signals in the simulation flow that are the following: the compressed data Byteout_Reg, the index IDCX, the counter Reg_CT, the probability Qe_reg and states. Table 1 summarizes the simulation results of the MQ encoder: either from decision n° 28 to decision n° 34.

B. Synthesis Results

Implementation of the proposed design was made on Xilinx Virtex Family Platforms: XC6SLX75T, XC5LX30T and XC4VLX80 devices. We have used the Xilinx ISE tools version 14.1 .The synthesis results of the architecture is shown in Table 2. The proposed MQ encoder design gives the best result, in terms of hardware resources such as (the number of LUTs consumed, slices and Flip-Flop) and frequency of operation when implemented on a platform Virtex 6. Concerning the frequencies obtained, we note that our architecture meets the criteria real-time.

The design has a maximum frequency of 423.2MHz on the Virtex 6 (XC6SLX75T) device.

C. Comparison

A comparative study with other existing designs in the literature has been made. The Virtex 4 XC4VFX140 platform is used for this comparison. The performance comparison of our design with the architecture proposed in [16] is shown Table 3. Our proposed design codes frames in real time at a frequency of 244.475 MHz and requires only 455 slices.

The throughput of some architecture of MQ coders compared with our proposed architecture is presented in Table 4. It is calculated from the reported symbol consumption rate and operating frequency. It is found that our architecture encodes frames with a frequency 3.31 and 2.29 times higher than that of architecture [16] and [17] respectively.

Table 5 shows the comparisons of logic area, memory requirement, and estimated memory area of several previous works [5], [7], [15]-[20]. The total area of the proposed architecture is less than that of each previous work. However, the hardware cost of the word based architecture is larger than the proposed architecture. The proposed design can code 40 frames per second for high definition TV of 1920p at 254.84 Mhz on Stratix II.

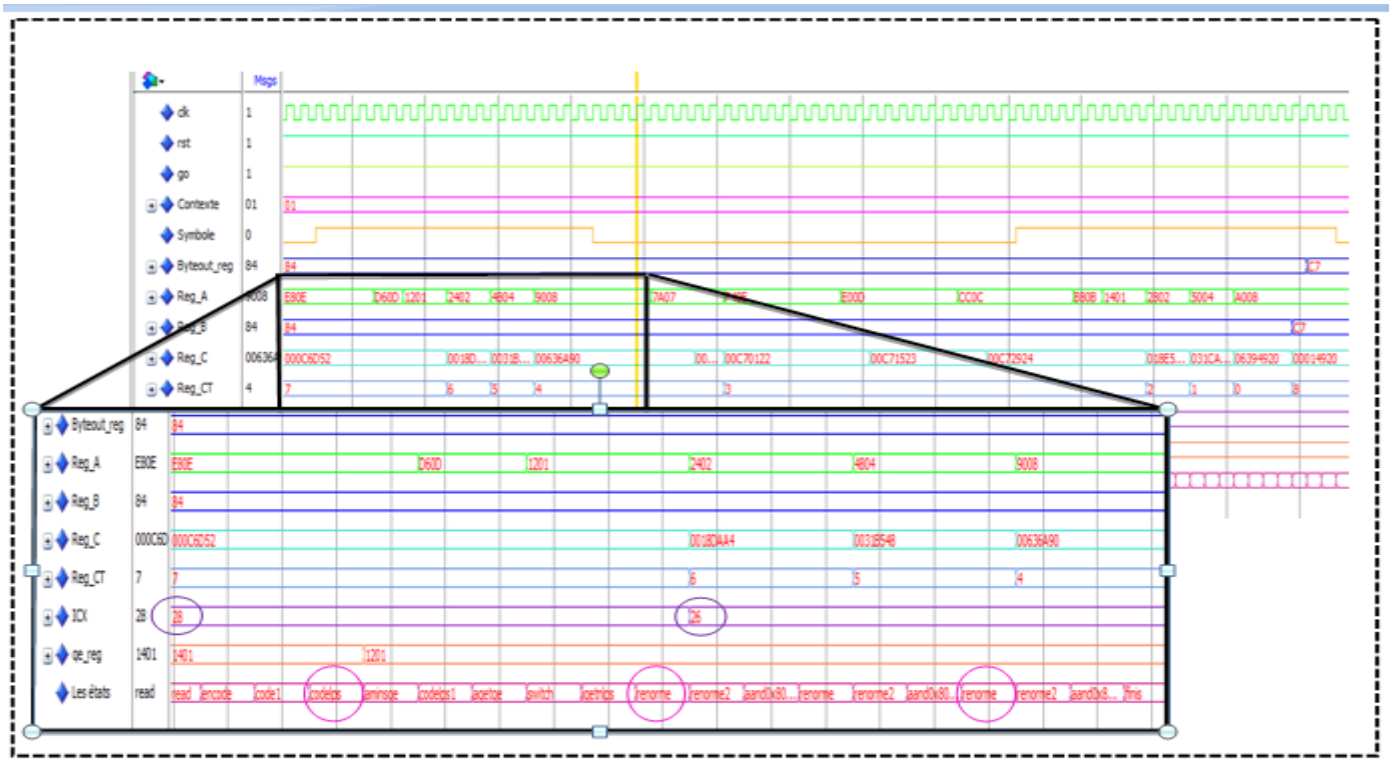

Fig. 9. Wave simulation for MQ coder architecture.

TABLE. I. EXAMPLE EXTRACTED FROM THE SIMULATION OF THE MQ ENCODER

Symbole	D	IDCX	Reg_A	Reg_C	CT	Byteout_Reg
10	1	28	0xE80E	0x000C6D52	7	0x84
11	0	26	0x9008	0x00636A90	4	0x84
12	0	27	0xF40E	0x00C70122	3	0x84
13	0	27	0xE00D	0x00C71523	3	0x84
14	1	27	0xCC0C	0x00C72924	3	0xC7
25	0	25	0xA008	0x00014920	8	0xC7
26	0	25	0x8807	0x00016121	8	0xC7

TABLE. II. RESULTS OF SYNTHESIS

Used Platform	XC6SLX75T	XC5VLX50T	XC4VLX80
Maximum Frequency (MHz)	423.2	336,304	264.2
No. of 4 input LUTs	540/343680	523/28800	766/71680
Total used slices	251/687360	247/28800	396/35840
Total FF slices	177/693	176/659	247/71680

TABLE. III. PERFORMANCE COMPARISON

Used FPGA	XC4VFX140	
Architecture	Proposed	Architecture [15]
Max. Frequency (MHz)	244.475	185.43
Used slices	455	495
Used FF slices	292	392
Used 4 input LUTs	865	893
Used BRAMs	1	2

TABLE. IV. THE THROUGHPUT OF SOME DESIGNS TESTED ON VIRTEX 4 X4VFX140

Used FPGA	XC4VFX140 (Virtex4)		
Design	Number of pairs	Frequency (Mhz)	Throughput (MS/s)
Design [7]	2	50.1	100.2
Design [21]	1	185.43	185.43
Design [18]	1.23	53.92	66.38
Design [19]	2	48.3	96.6
Proposed	1	244.475	244.475

VI. CONCLUSION

This paper discussed the problems in the real-time implementation of FPGA-based MQ coder architecture. The MQcoder utilization in both encoding and decoding stages performs the probability estimation of coefficients and optimization of Renorme and Byteout modules. The increase in computational overhead required more power and energy consumption. This paper provides the reduction in the bitwise computation to reduce the number of computational steps. The minimization in computational steps decrease the power and time delay. The proposed PET architecture reduced the bitwise representation from 13 bit to 12 bit that provided the reduction in memory elements from 416 to 348 compared to the existing MQ-coder architecture. Therefore, the size of PET ROM is 1376 bits. An embedded architecture of MQ Coder for JPEG2000 is designed and implemented in this paper. The implementations carried out during this work allowed us to know that the proposed architecture of the MQ encoder operates with a frequency of 423.2 MHz on Virtex6 XC6SLX4 device and that it can code 40 frames per second for the high-definition TV application. The proposed architecture is easily expandable to 2048×1080 resolution video at 45 fps. It can be used in several applications such as Internet broadcasting, medical imaging and digital photography. Moreover, the processing time was improved by about 13.6% in comparison with well-known architectures from literature.

ACKNOWLDGMENTS

The authors wish to acknowledge the approval and the support of this research study by the grant N° CIT-2016-1-6-F-5718 from the Deanship of the Scientific Research in Northern Border University, Arar, KSA.

TABLE I. COMPARISON WITH OTHER MQ CODER ARCHITECTURES

Architecture	FAGA family	Device used	Clk (MHz)	No. of LEs	Symbol/Clk	Throughput (MS/s)
[5]	APEX20K	EP20K600EFC672-3.	37.27	1256	2	74.54
[7]	Stratix	N/A	50.10	1596	2	100.2
[15]	Stratix	N/A	40.53	12649	2	81.6
[20]	Stratix II	EP2S15F484C3	106.2	1321	2	212.4
[22]	APEX20K	EP20K1000EFC672-1X.	9.25	14711	1	9.25
[23]	Stratix	N/A	27.05	761	1	57.05
[24]	Stratix	N/A	106.02	1267	2	210
[16]	Stratix	EP2S90F1020I4.	58.56	1488	2	117
[17]	Stratix	EP1S10B672C6.	145.9	824	1	145.9
Proposed	Stratix II	EP2S15F484C3	254.84	603	1	254.84

REFERENCES

[1] JPEG 2000 image coding system, ISO/IEC International Standard 15444-1. ITU Recommendation T.800, (2000).

[2] D. S. Taubman and M. W. Marcellin. JPEG2000 Image Compression Fundamentals, Standards, and Practice (2002).

[3] T. Acharya and P. Tsai, JPEG2000 Standard for Image Compression: Concepts, Algorithms and VLSI Architectures, J. Wiley & sons (2005).

[4] JASPER Software Reference Manual, ISO/IEC/JTC1/SC29/WG1N2415.

[5] K. Mei, N. Zheng, C. Huang, Y. Liu, Q. Zeng, VLSI design of a high-speed and area-efficient JPEG 2000 encoder, IEEE Transactions on Circuits and Systems for Video Technology 17 (8) (2007) 1065–1078.

[6] Horrigue, L., Saidani, T., Ghodhbani, R., Dubois, J., Miteran, J.,Atri, M.: An efficient hardware implementation of MQ decoder of the JPEG2000. Microprocess. Microsyst. 38, 659–668 (2014)

[7] L. Liu, N. Chen, H. Meng, L. Zhang, Z. Wang, H. Chen, A VLSI architecture of JPEG 2000 encoder, IEEE Journal of Solid-State Circuits 39 (11) (2004) 2032–2040.

[8] Jiangyi Shi, Jie Pang, Zhixiong D Yunsong Li. A Novel Implementation of JPEG2000 MQ-Coder Based on Prediction, International Symposium on Distributed Computing and Applications to Business, Engineering and Science, 2011. pp:179-182.

[9] Kishor Sarawadekar and Swapna Banerjee, VLSI design of memory-efficient, high-speed baseline MQ coder for JPEG 2000, Integration, the VLSI Journal, Elseiver. Vol 45, January 2012, Pages 1-8. DOI: 10.1016/j.vlsi.2011.07.004.

[10] J. Hunter. Digital cinema reels from motion JPEG 2000 advances, janvier 2003. http ://www.eetimes.com/story/OEG20030106S0034.

[11] C. Le Barz and D. Nicholson. Real time implementationof JPEG 2000 . june 2002.

[12] C. Bonaldi and Y. Renard. Conception et réealisation d'un codeur JPEG 2000 sur une carte Virtex-ARM . Laboratoire de Microélectronique (DICE), UCL, june 2001.

[13] Amphion. CS6590 JPEG 2000 codec preliminary product brief , October 2002. http ://www.amphion.com.

[14] D.Taubman and M.W.Marcellin, JPEG2000 - Image Compression Fundamentals, Standards and Practice, Kluwer Academic Publishers, Nov. 2001.

[15] K. Liu, Y. Zhou, Y. Song Li, J.F. Ma, A high performance MQ encoder architecture in JPEG2000, Integration, the VLSI Journal 43 (3) (2010) 305–317.

[16] P. Zhou, Z. Bao-jun, High-throughout hardware architecture of MQ arithmetic coder, in: 10th IEEE International Conference on Signal Processing (ICSP), 2010.

[17] K. Sarawadekar, S. Banerjee, An Efficient Pass-Parallel Architecture for Embedded Block Coder in JPEG 2000 . IEEE Trans. Circuits Systems. Video Technology, 22 (6) (2011) 825-836.

[18] Michael Dyer, David Taubman and Saeid Nooshabadi. Concurrency Techniques for Arithmetic Coding in JPEG2000. IEEE Transactions on Circuits and Systems for Video Technology, 2006, vol.53, pp. 1203–1213.

[19] Kishor Sarawadekar and Swapna Banerjee, "LOW-COST, HIGHPERFORMANCE VLSI DESIGN OF AN MQ CODER FOR JPEG 2000" ICSP2010, 2010, pp.397-400.

[20] Nandini Ramesh Kumar · Wei Xiang · Yafeng Wang, Two-Symbol FPGA Architecture for Fast Arithmetic Encoding in JPEG 2000, Journal of Signal Processing Systems, 69(2)(2012)213–224.

[21] Saidani, T., Atri, M., Khriji, L., Tourki, R.: An efficient hardware implementation of parallel EBCOT algorithm for JPEG2000. J. Real-Time Image Process. 11, 1–12 (2013).

[22] Varma,H.Damecharla,A.Bell,J.Carletta,G.Back,A fast JPEG2000 encoder that preserves coding efficiency:the splitarithmetic encoder, IEEE Transactions on Circuits and Systems—Part I:RegularPapers55(11)(2008) 3711–3722.

[23] M. Dyer, S. Nooshabadi, D. Taubman, Design and analysis of system on a chip encoder for JPEG 2000, IEEE Transactions on Circuits and Systems for Video Technology 19 (2) (2009) 215–225.

[24] N.R. Kumar, W. Xiang, Y. Wang, An FPGA-based fast two-symbol processing architecture for JPEG 2000 arithmetic coding, in: IEEE International Con- ference on Acoustics Speech and Signal Processing (ICASSP) 2010, 2010, pp. 1282–1285.