Analysis and VLSI High Speed of Parallel Causal EBCOT Architecture for Image Compression

[2]REFKA GHODHBANE, [2]TAOUFIK SAIDANI, [1]LAYLA HORRIGUE, , and [2]MOHAMED ATRI
[1]Electronics and Micro-Electronics Laboratory Faculty of Sciences Monastir University
TUNISIA
[2]Faculty of Computing and Information Technology, Northern Border University
refka.ghodhbane@nbu.edu.sa, Taoufik.Saidani@nbu.edu.sa, layla.horrigue@hotmail.fr,
Mohamed.Atri@leme.tn

Abstract: - Embedded block coding with optimized truncation (EBCOT) is a key algorithm in digital cinema (DC) distribution system. Though several high speed EBCOT architectures exist, all are not capable of meeting the DC specifications. With the augmentation in multimedia technology, demand for high speed real time image compression system has also increased. JPEG2000 is a relatively new image compression standard which builds and improves on its predecessor JPEG. In Jpeg 2000 the embedded Block Coding with Optimal Truncation (EBCOT) is the most important element to calculate the very hard portion in the compressing process of JPEG 2000 image compression standard. This paper proposes a Parallel Bit Plane Coding (BPC) architecture in which three coding passes operate in parallel and are allowed to progress independently.

Key-Words: - EBCOT, JPEG2000, VHDL, FPGA, VLSI..

1 Introduction

To address and improve on weaknesses in the JPEG standard, a new image compression scheme called JPEG2000 has recently been introduced. JPEG 2000 supports a rich set of features, including improved compression efficiency, optional lossless encoding, resolution and distortion (SNR) scalability, region of interest coding, and support for image editing on compressed images [1]. Although they share the same name, JPEG 2000 is fundamentally different from the original JPEG standard. First, JPEG 2000 replaces the JPEG discrete cosine transform (DCT) frequency decomposition with a discrete wavelet transformation (DWT). The DWT is a multi-resolution decomposition, which allows for resolution scalability within the embedded bit stream. In addition, the DWT exhibits better energy compaction than the DCT, allowing for superior compression efficiency. The DWT typically is applied on an image tile or on the entire image as a whole. This large scale application allows the DWT to minimize the blocking artifacts that plagued the 8x8 DCT in the original JPEG .The second major deviation of JPEG 2000 from its predecessor is the abandonment of the Huffman entropy encoding scheme for an adaptive binary arithmetic coder.JPEG 2000 uses the EBCOT (Embedded Block Coding and Optimal Truncation) algorithm to arithmetically encode the DWT coefficients [1, 2]. EBCOT works on a bit plane level, generating a neighborhood context for each bit that is coded. Because it must touch every bit of every bit plane in the image, the EBCOT dominates the processing time (50–75 %) depending on the source image [1]). In addition, the EBCOT works solely at the bit plane level, and is therefore fairly inefficient to implement in software. For these two reasons, the EBCOT algorithm warrants a custom. EBCOT role as the system bottleneck also ASIC implementation demands a fast implementation to speed the overall encoding process. In this paper, we propose a novel architecture for optimized EBCOT encoding. Our implementation builds son the work [1], who first proposed parallelizing the three EBCOT passes. Our dynamic buffering scheme extends the idea by allowing greater flexibility for each pass, permitting them to move relative to one another, rather than [1]. In March 2002, digital cinema initiative (DCI) was established [11] to develop new standard, suitable for digital cinema (DC) applications. In 2005, DCI officially selected JPEG 2000 part-1 as compression engine for DC applications [12]. The DC system consists of four parts: mastering, transport, storage and playback, and projection, as illustrated in Fig 1 [13]. The movie is compressed, encrypted, and packaged for delivery in the mastering stage. Next, these data are transported to the exhibition site where they are decrypted, decompressed, stored, and played back.

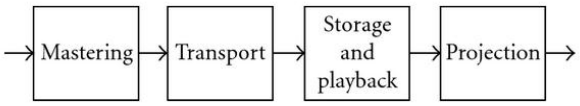

Fig 1. Block diagram of a digital cinema system

This paper is organized as follows: Section 2 provides an overview of JPEG 2000 standard. Section 3 describes the software analysis of EBCOT algorithm and some discussions. The Proposed architecture of EBCOT is presented in section 4.In section 5 experimental results are furnished and are compared with other architectures. Finally, brief conclusions are drawn in section 6.

2 Overview of JPEG 2000 encoding system

The JPEG 2000 algorithm has three main components; the DWT, Quantizer, and the EBCOT coder as seen in Fig 2. The DWT is a frequency transformation that decomposes the original image pixels into a multi-resolution representation. The DWT is important for compressing the image, since it compacts the energy of the original pixels into low amplitude wavelet coefficients [2] which can be represented with fewer bits of precision.

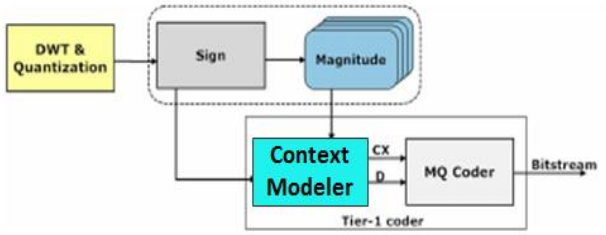

Fig2. Overview of JPEG 2000 Coding Proces

Typically, the DWT is implemented with a lifting algorithm, which has greater memory efficiency than the convolution based DWT, and can be implemented in hardware as banded matrix multiplication [3]. The transform is performed on the entire image or on a tile of the image when a lower memory utilization is desired. The JPEG 2000 standard specifies the integer (5,3) lifting based filter for lossless encoding and the (9,7) filter for lossy encoding [3]. Quantization is an optional step, and is performed on the wavelet coefficients when lossy mode is desired. JPEG 2000 makes use of a uniform quantizer with central dead zone [2]. The quantizer step size is a function of the dynamic range of sub band samples [3], and can be adjusted to take advantage of psycho-visual properties, similar to the original JPEG Q-table [2].

2.1 Bit Plane Coding

The DCI [1, 3, 4] standard has confined to use 5/3 DWT filter [4, 5]. DWT coefficients are input to the context coder, and these coefficients are then scanned at the bit plane level in the specific order shown in Fig 3. Bit planes are coded in order from the most significant bit plane to the least significant bit plane for a code block (CB).The CB memory comprises a sign plane and several magnitude planes, as shown in Fig 3(a). During encoding, the magnitude bit planes are scanned starting from the most significant bit (MSB) plane to the least significant bit (LSB) plane [6,7].

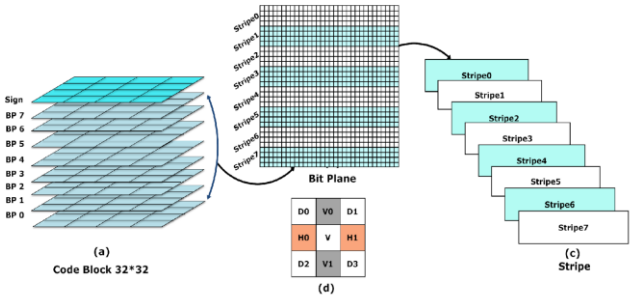

Fig 3. (a) Bit-plane representation of a code-block. (b) Each bit-plane consists of stripes made up of four rows. (c) Stripe-based scanning order for every pass. (d) Context window for a sample location

Each bit plane is further divided into stripes of four rows (refer to Fig 3(b)). Within a stripe samples are scanned column wise starting from left to right and within a column, from top to bottom as shown in Fig 3(c).To examine each sample in a CB three coding passes-cleanup (CUP),significant propagation pass (SPP) and magnitude refinement pass (MRP) are adopted in this algorithm [1, 3, 4]. Type of coding pass to be applied on a sample is determined by forming a context window surrounding it, as shown in Fig. 3(d), along with the three state variables σ, σ', and η. When MRP is run for the first time on a coefficient, σ' variable is set to one and if zero coding is applied on a coefficient in SPP, η_{16} is set to one. Initially, all state variables are assumed to be zero. The encoding start from the first non-zero magnitude bit plane assuming that all state variables are zero. Only CUP is run on this bit plane, whereas rest other bit planes are coded sequentially using SPP, MRP and CUP [3], [4]. All samples are encoded using four coding primitives: zero coding (ZC), sign coding (SC), run length coding (RLC), and magnitude refinement coding (MRC) and CxDs pairs are generated [10].

2.2 MQ coder

The compression technique adopted in JPEG2000 standard is a statistical binary arithmetic coding, which is also called MQ-coder [1, 3, 8].The MQ-coder utilizes the context (CX) to compress the decision (D).

In the MQ-Coder, symbols in a code stream are classified as either most-probable symbol (MPS) or least-probable symbol (LPS) [8]. The basic operation of the MQ coder is to divide the interval recursively according to the probability of the input symbols.

Figure 4 shows the interval calculation of MPS and LPS for JPEG2000 [4, 8]. We can find out whether MPS or LPS is coded, and the new interval will be shorter than the original one.

2.1.1 Sub-subsection

When including a sub-subsection you must use, for its heading, small letters, 11pt, left justified, bold, Times New Roman as here.

3 Analysis of Bit Plane Coding algorithm

In this section using the jasper implementation verification model software we will present the detailed run time analysis of JPEG2000 codec and an analysis of bit plane coding will be presented [10]. The size of the Lena image is 512*512 pixels (8bpp), and the compression is adjusted in basic mode (bank 9/7 filter and 5/3 filters, 3-level wavelet decomposition, 32 * 32 block size code). The bit plane coding processing (BPC) consumed 51.8 % and 55 % of the total execution time, respectively.

Therefore, it is the most intense part of the JPEG2000 coder. For coding a code block with size 32*32, 3xxNxNbp clock cycle is required (3: number of passes, N: number of clock cycle used to encode one bit plane, Nbp: number of bit plane to be encoded). For developing a new technique to minimize the large number of clocks cycles for processing. Analyzing of the intense block for JPEG2000 coder is performed. The PCB analysis is performed using four gray scale images ISO-Lena, Barbara, peppers and baboon with size 512*512. Using MATLAB environment, a legally wavelet transform [5, 6] is developed. As test image are decomposed up to three levels. Every Sub band of wavelets coefficient is decomposed to CBs with size 32x32 and stored in height bits. Finally with the help of MATLAB simulation, all these Cbs are encoded and (CX, D) pairs are generated. The number of (Cx, D)s generated for all test images is presented in Table 1. For this analysis, we choose three type of CB with size 32x32 who are: Code block where all coefficients has positive magnitude. CBs where all coefficients has negative magnitude. CBs where the maximum number of coefficient are equal at zeros. For Lena test image with size 512*512,there are 256 CBs, CB number 10,4 and 20 are choose for this analysis. CB number 10, all samples are positive sign. CB number 4, the sign of all samples is negative. CB number 20 have a large number of zeros coefficients 420). After analyzing, the similar relationships between the contents of CBs and the number of (CX, D) generated. The relationship between code block and the value of magnitude wavelets coefficient is shown in Fig 4.

Table 1. (CX ,D) NUMBER GENERATED FOR TEST IMAGES

Images	Number of (CX, D) pairs
Lena	1,282,176
Barbara	1,490,628
Peppers	1,397,547
Baboon	1,708,678
Average	1,463,765

For code block with all sample are positive Fig 4(a), the maximum and the minimum value are 521 and 12 respectively. This CB is derived from low resolution (LL3 sub band) from Lena image test. During CB encoding, MSB plane (bp 1) is skipped because all coefficients are insignificant. Here, the MRP passes will not generate any (Cx,D) pairs while encoding bit plane 6. This fact is demonstrated in Fig. 4(a). In this bit plane, large number of magnitude sample is significant. Here huge numbers of (CX, D) pairs are produced in SPP passes and a few pairs in CUP passes. The last coefficients become significant in bit plane 6, no (CX, D) pairs are generated by SPP and CUP.

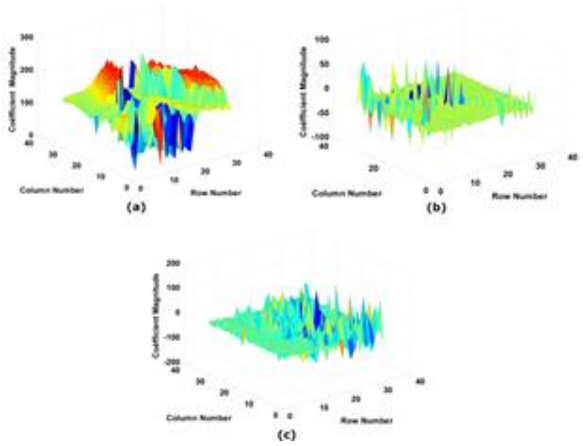

Fig 4. Contents of a CB in different type of block

In high resolution (LH3) a code block with all sample are negatives, the value range from -45 to -5, please refer to Fig 4(b).Much higher number of (CX, D) pairs generated in SPP passes while Bp 5 encoding. This fact is demonstrated in Fig 5(b).

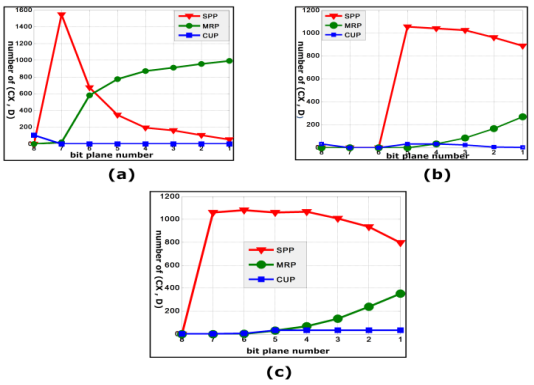

Fig 5.Statistics (Cx, D) generated from CBs analyzed.

In case of a CB with the maximum number of zeros wavelets, this maximum number needs one bit for representation magnitude. Contributions of CUP (Cx, D) are much higher in all bit planes except in the LSB, as shown in Fig 5(c). Every code block has 1024 number of samples. The number of bit plane to be processed by EBC and the number of code blocks are shown in Figure 6. Here the MSB plane is number eight; it is observed that a few number of CBs have zeros bit planes (ZBP) (9 blocks for Lena test).For Barbara image all codes blocks needs 6 bits (54 blocks) or 5 bits to presents the magnitude coefficients. Nulls numbers of CB have height or seven zeros bit planes for all image tests. According to this analysis, it can conclude that number of contexts generated in a bit plane depends on the contents of the CB. To attain specifications imposed by DCI not only pass parallel but also concurrent sample coding architecture for EBCOT must be adopted. Such architecture is presented in this paper.

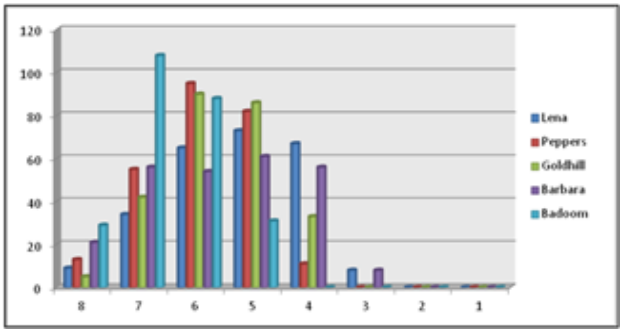

Fig 6. Relationship between contents block and zeros bit planes from images tests

4 Existing architecture

Many architectures Context Modeler have been proposed in the literature, to implement the JPEG2000 coder using hardware description language (HDL) implementations to be used in FPGAs [14, 15]. [14] is an optimized architecture of bit plane coder for Embedded Block Coding with Optimal Truncation (EBCOT). This proposed design operate at 67 MHz after post placement and routing on Xilinx XC2V1000 device.

The context formation engine used in EBCOT is analyzed and an architecture based on parallel processing of the three co01ding passes is proposed in [15]. This architecture is implemented on a XC2V1000 device, the design performs at 50 MHz after place and route.

5 Proposed architecture

The proposed Embedded Block Coding is shown in Fig 7. Architecture [7] adopts the normal mode but in our architecture we adopt the causal mode.

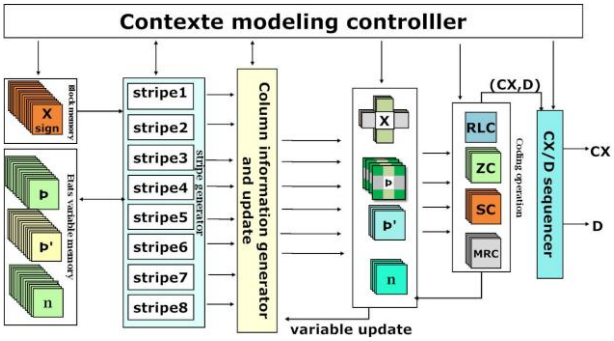

Fig 7. Top module of the proposed Column-parallel context modeling

After initialization of state variable (σ, σ', and η), the context modeling controller read a stripe, sign and state variable from six separate memory. Here this architecture use Single port RAM for the magnitudes and sign sample, and dual port RAM for states variables. The 32 column stripe are coded with Coding operations blocks, to investigate this fact, four ZC ,four SC ,four MRC ,and one RLC blocks are used in this architecture. In order to process all magnitude bits concurrently, entire magnitude column and the corresponding sign bits column are generated in one clock by column information generator. For each magnitude bit in a column stripe, four sign neighbors, eight σ, four σ' and four η neighbors are required. On one clock cycle, four to ten (CX, D) are generated simultaneously. These pairs are sent to MQ coder in order. This scheduling is done via CX/D sequencer. When the last column stripe is coded, states variable are updates for next stripe. The key building blocks of this architecture are as follows.

- Data organization and memory arrangement
- Column information generator
- Coding Operation
- CX/D sequencer
- Context Modeling Controller

5.1 Data organization and Memory arrangement

In order to reduce the required memory access clock cycle, magnitude and sign memories access and to achieve an efficient data and state variables, we present in this section a new data arrangement and memory organization to implement the parallel stripe coding for a Bit-plane coding. First each stripe for each bit plane as mapped with zeros (Fig 8).

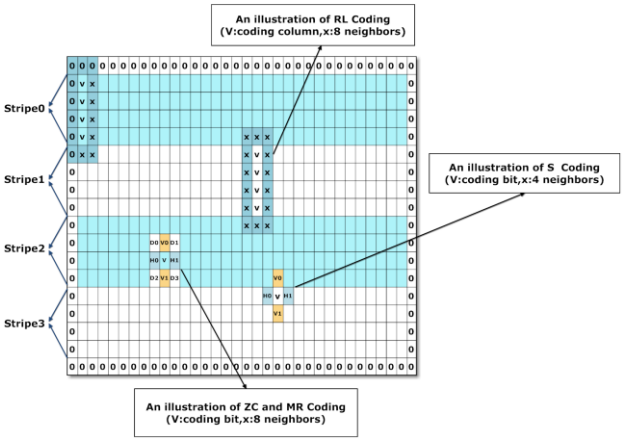

Fig 8. bit Plane lines and Bit Plane memories association

As shown in Fig 9 memory blocks are organized in sex partitions. MEM0 to MEM5 contain the stripe state variables data. Organization is adopted for magnitude bit plane and sign coefficients. By using this technique of memory arrangement, we can:

- Reduce the complexity of addressing
- Perform read and write operations at the same clock cycle
- Avoid the operational conflicts (simultaneous read or write)

Rom_0	Rom_1	Rom_2	Rom_3	Rom_4	Rom_5
Row 0	Row 1	Row 2	Row 3	Row 4	Row 5
Row 4	Row 5	Row 6	Row 7	Row 8	Row 9
Row 8	Row 9	Row 10	Row 11	Row 12	Row 13
Row 12	Row 13	Row 14	Row 15	Row 16	Row 17
Row 16	Row 17	Row 18	Row 19	Row 20	Row 21
Row 20	Row 21	Row 22	Row 23	Row 24	Row 25
Row 24	Row 25	Row 26	Row 27	Row 28	Row 29
Row 28	Row 29	Row 30	Row 31	Row 32	Row 33

Fig 9. Memory Organization for Bit Plane

5.2 Coding Operation

5.2.1 Zero coding (ZC) context block

The ZC context block generates the relevant contexts and decision bit based on the status of the significance states of the eight neighbors (σ_{00}, σ_{01}, σ_{02}, σ_{20}, σ_{21}, σ_{22}, σ_{10} and σ_{12}) of the current bit position X.

$$\begin{cases} CX[4]=0, \\ CX[3]=\sigma_{10}\wedge\sigma_{21} \\ CX[2]=Ph+\overline{H}\wedge\sigma_{01}\wedge\sigma_{01} \\ CX[1]=Pv\wedge\left(V+\overline{V}\wedge D\right)+\overline{H}\wedge\left(P_v+\overline{V}+Pd\right) \\ CX[0]=\sigma_{10}\oplus\sigma'_{12} \end{cases} \quad (1)$$

We present here the logic functions for generation of the 5-bit context CX [4:0] by zero coding the LL and LH sub-bands. Where + is the logical Or operation, is the logical AND operation, and represents the NOT of logic variable.

$$\begin{cases} H=\sigma_{10}+\sigma_{12} \\ V=\sigma_{01}+\sigma_{21} \\ D=\sigma_{00}+\sigma_{02}+\sigma_{20}+\sigma_{22} \\ Ph=\sigma_{10}\oplus\sigma_{12} \\ Pv=\sigma_{01}\oplus\sigma_{02} \\ Pd=\sigma_{00}\wedge\sigma_{02}+\sigma_{20}\wedge\sigma_{22}+(\sigma_{00}\wedge\sigma_{02})\wedge(\sigma_{20}\wedge\sigma_{22}) \end{cases} \quad (2)$$

When the H,V,D contributions is given in equation (2)

5.2.2 Sign coding (SC) context block.

Context and data are determined by horizontal reference value H and a vertical reference value V by the sign coding algorithm. Let assume that contributions of 0,+1, and -1 by the horizontal and vertical neighbors are expressed and respectively in (3) and (4).The contribution can be computed based on the status of the significance states of the horizontal neighbors and vertical neighbors and the corresponding sign From the Sign Memory block using the following logic equations.

$$\begin{cases} h_{c+}=\sigma_{10}\wedge\sigma_{12}\wedge\overline{\chi}_{10}\wedge\overline{\chi}_{21}+\overline{\sigma}_{10}\wedge\sigma_{12}\wedge\overline{\chi}_{12}+\sigma_{10}\wedge\overline{\sigma}_{12}\wedge\overline{\chi}_{10} \\ h_{c-}=\sigma_{10}\wedge\sigma_{12}\wedge\chi_{10}\wedge\chi_{21}+\sigma_{12}\wedge\sigma_{12}\wedge\chi_{12}+\sigma_{10}\wedge\overline{\sigma}_{12}\wedge\chi_{10} \\ h_{c0}=h_{c+}+h_{c-} \end{cases} \quad (3)$$

And

$$\begin{cases} v_{c+}=\sigma_{01}\wedge\sigma_{21}\wedge\overline{\chi}_{01}\wedge\overline{\chi}_{21}+\overline{\sigma}_{01}\wedge\sigma_{21}\wedge\overline{\chi}_{21}+\sigma_{01}\wedge\overline{\sigma}_{21}\wedge\overline{\chi}_{01} \\ v_{c-}=\sigma_{01}\wedge\sigma_{21}\wedge\chi_{01}\wedge\chi_{12}+\sigma_{01}\wedge\sigma_{21}\wedge\chi_{21}+\sigma_{01}\wedge\overline{\sigma}_{21}\wedge\chi_{01} \\ v_{c0}=v_{c+}\wedge v_{c-} \end{cases} \quad (4)$$

According the logic equation (5) function for the context bits Cx (i) , for i=0 to 4 based on the above values are as follows in equation (5).

$$\begin{cases} CX[4]=0 \\ CX[3]=1 \\ CX[2]=v_{c0}\wedge\left(h_{c+}+h_{c-}\right)+v_{c-}\wedge h_{c-}+v_{c+}\wedge h_{c+} \\ CX[1]=h_{c+}\wedge v_{c-}+h_{c-}\wedge v_{c+}+h_{c0}\wedge\left(v_{c+}+v_{c-}\right) \\ CX[0]=h_{c+}\wedge\left(v_{c0}+v_{c-}\right)+h_{c-}\wedge\left(v_{c-}+v_{c+}\right)+h_{c0}\wedge v_{c0} \end{cases} \quad (5)$$

5.2.3 Magnitude refinement coding (MRC) context Block.

The magnitude information in current bit-plane of the significant samples is encoded by this operation. The inputs to the MRC context block are σ ' [m, n] (value 1 indicate that it is not the first magnitude refinement for the current element) and the significance status of the eight neighbors (σ_{00} ,σ_{01} ,σ_{02} ,σ_{10} ,σ_{12} ,σ_{20} ,σ_{21} ,σ_{22}) (6) of the bit being encoded. According the logic function for the context bits CX[i] for i=0 t to 4 are as follows in (7).

$$\sum\sigma=\sigma_{00}\wedge\sigma_{01}\wedge\sigma_{02}\wedge\sigma_{10}\wedge\sigma_{12}\wedge\sigma_{20}\wedge\sigma_{21}\wedge\sigma_{22} \quad (6)$$

$$\begin{cases} CX[4] = \sigma_{m,n} \\ CX[3] = \overline{\sigma'}_{m,n} \\ CX[4] = \overline{\sigma'}_{m,n} \\ CX[4] = \overline{\sigma'}_{m,n} \\ CX[4] = \overline{\sigma'}_{m,n} \wedge \sum \sigma \end{cases} \quad (7)$$

5.3.3 Run-length coding (RLC) contexts.

This is used only when an immediately previously insignificant sample is found to be significant during ZC, SC or RLC operation. The sign information is encoded using one of the five different context states that depend on the sign and the significance of the immediate vertical and horizontal neighbors. The architecture for RLC coding is shown in Fig 10.

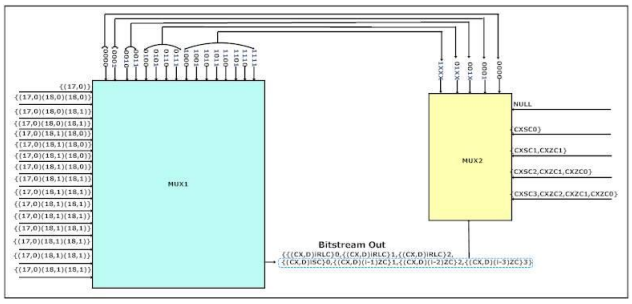

Fig 10. Architecture for RLC Module

5.3.4 Context Modeling Controller Block References:

This controller generates all the necessary control signals to enable all the modules in the architecture including loading the stripe from block Rom, reading and writing the stripe memory, controlling the pointers to access the CX/D sequencer, generating the control signals for the BPC encoder.

6 Experimental Results and Discussion

6.1 Simulation

Before our hardware implementation, a software-based EBCOT algorithm has been coded by MATLAB for system verification. The EBCOT algorithm has been verified by MATLAB software. The proposed system architecture has been implemented in VHDL, simulated with Modelsim 6.6d. The initialization of the program only occurs once at the beginning of the coding code-block. It reset all the variables involved in the coding. The simulation results of this scenario are shown in Fig 11.

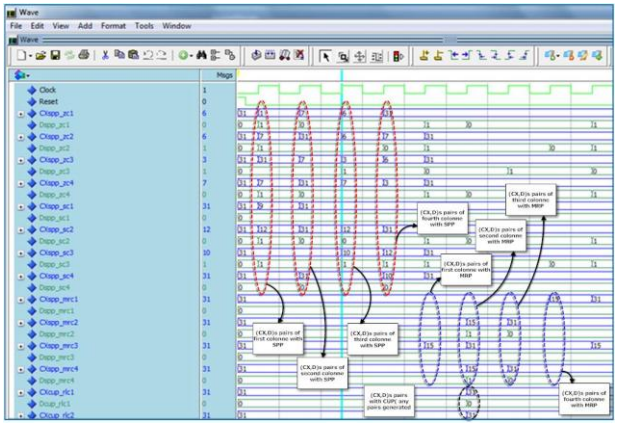

Fig 11. Wave simulation for EBCOT architecture

Once the first pair (CX, D) is read the coding process is triggered stops and that if he finishes the coding of all symbols presented at the entrance.

6.2 Results of Synthesis

The proposed architecture is described using VHDL language, synthesized with Xilinx ISE 13.1 and implemented on Virtex 4 Virtex5 FPGA and Spartan 3A DSP 3400. Table 2 present the resources for the proposed architecture. All samples in a stripe column are processed in single clock and to encode a bit plane, 138 clock cycles are required to coding one Bit Plane.

Table 2. PROPOSED EMBEDDED BIT PLANE CODING ARCHITECTURES

FPGA used	Virtex 5	Virtex 4	Spartan 3A DSP 3400
Max Frequency (MHz)	360	290	154
No of Input 4 LUTS	-	520	516
Total Slices used	310	290	274
Total FF Slices	26	28	28

Table 3 summarizes the runtime statistics for CBs with size 32x32. Each image comprises more than 587 zero bit planes. To encode an image on an average 183480 clock cycles are required (6291456 clock cycles for sequential architecture). The processing rate is obtained by computing the ratio of total number of clock cycles required per image to total coefficients in an image.

Table 3. Encoding Cycles Statistics for CB with size 32x32

Images	BP processed	Cycles/Images
Lena	1390	183480
Barbara	1717	226644
Peppers	1546	204072
Averages	1361	197652

To evaluate the performance of our architecture viewpoint hardware requirements and operating frequency, a comparative study was made with other existing architectures using the same FPGA Virtex-2 XC2V1000-6. Table 4 presents comparison of the

proposed BPC architecture with the existing architectures.

Table 4. Comparison with other BPC architectures

Architecture	[15]	[14]	Proposed
Frequency(MHz)	50	67	189
No. of 4 input LUTs	7.071	2.488	339
Total used slices	4.420	2.149	183
Total FF slices	1560	105	27

7 Conclusion

We have proposed an efficient VLSI architecture to implement the Bit-plane coding. The design was implemented in VHDL, and synthesized and routed in Virtex 5 FPGA platform. This new architecture is based on a parallel access to memories, parallel stripe coding and uses a new design of parallel context modeling. A working frequency of 362 MHz is achieved, and the number of the required clock cycles is reduced by 75%, which increase the processing speed, by comparison with the normal mode. With this speed the proposed EBCOT architecture is capable of encoding 2048x1080 size 42 frames of DC in a second.

Acknowledgments, Laboratory of Electronic and Microelectronic University Of Monastir.

[1] JPEG 2000 image coding system, ISO/IEC International Standard 15444 1.ITURecommendationT.800, (2000).

[2] ISO/IEC JTC1/SC29/WG1 N2678, document JPEG 2000 Part 1 020719 (final publication draft), (2002).

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

[4] T. Acharya and P. Tsai, JPEG2000 Standard for Image Compression: Concepts, Algorithms and VLSI Architectures, (2005).

[5] D. Lee, JPEG 2000: Retrospective and new developments, Proc. IEEE, 93 (1) (2005) 32–41.

[6] M. Rabbani and R. Joshi, An overview of the JPEG 2000 still image compression standard, Signal Processing: Image Communication., 17(1) (2002) 3–48.

[7] T. Saidani,M.Atri,L.Khriji and R.Tourki, An efficient hardware implementation of parallel EBCOT algorithm for JPEG 2000, Journal or Real Time Processing (January 2013)..

[8] A. Das, A. Hazra, and S. Banerjee, An efficient architecture for 3-D discrete wavelet transform, IEEE Trans. Circuits System Video Technology, 20(2) (2010) 286–296.

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

[10] JASPER software reference manual. ISO/IEC/JT.

[11] R. Wintner, Bits on the big screen, IEEE Spectrum, vol. 43,no. 12, pp. 42–48, 2006..

[12] Digital Cinema Initiatives, LLC Member Representatives Committee: Digital Cinema System Specification V1.0, Final Approval, July 2005..

[13] Digital Cinema Initiatives, LLC Member Representatives Committee: Digital Cinema System Specification V1.2, March 2008..

[14] Sarawadekar. K, Banerjee.S, A high-performance architecture of JPEG 2000 and its FPGA Implementation. In 17th European Signal Processing conference (EUSIPCO 2009), Septembre 2009.

[15] Gangadhar.M, Bhatia.D, FPGA based EBCOT arcitecture for JPEG 2000. Microprocess. Microsyst.29(8-9), 363-373(2005).