



**Abdulkreem Mohameed
 Salih ***

Ahlam Fadhil Mahmood

Computer Engineering Department
 University of Mosul
 Mosul
 Iraq

Tikrit Journal of Engineering Sciences

Design and Implementation of a Gray Scale JPEG CODEC on Spartan-3E

ABSTRACT

This paper presents the design and implementation of the hardware JPEG CODEC for gray scale images. The architecture is designed in a way based on modules that share between JPEG encoder and decoder circuit. Each module was designed to implement a forward and backward function and they have separate control signals. The JPEG CODEC (Compressor, Decompressor) architecture achieves high throughput with a deep and optimized pipeline, with a target to FPGA device implementation. The designed architectures are detailed in this paper and they are described in VHDL, simulated and physically mapped to XC3S500 FPGAs. The JPEG CODEC pipeline has a minimum latency of 166 clock cycles, that given the full modular pipeline depth. The CODEC could process a 512X 512 pixels still image in 5.2ms, reaching a maximum processing rate of 190 frames per second.

Keywords:

JPEG
 CODEC
 compressor
 decompressor
 FPGA
 DCT

ARTICLE INFO

Article history:

Received 3 January 2012
 Accepted 28 May 2012
 Available online 30 September 2017

© 2017 TJES, College of Engineering, Tikrit University

DOI: <http://dx.doi.org/10.25130/tjes.24.3.03>

تصميم وتنفيذ كبس وفك الكبس للصور الرمادية على رقاقة البوابات القابلة للبرمجة

الخلاصة

قدم هذا البحث تصميم وتنفيذ مادي لمعمارية ال JPEG CODEC للصور الرمادية. المعمارية صممت بطريقة تعتمد على تجزئة الخوارزمية الى وحدات هذه الوحدات كلها مشتركة بين دائرة الكبس وفتح الكبس. كل وحدة من هذه الوحدات صممت بطريقة بحيث تؤدي وظيفة الكبس وفك الكبس ولكل منها إشارات سيطرة منفصلة عن الأخرى. معمارة ال JPEG CODEC انجزت بإخراجية عالية مستخدمة خاصية خط الأنابيب باستخدام تقنية الFPGA. المعماريات المصممة موضحة في هذا البحث وموصوفة بلغة VHDL تم تنفيذ المحاكاة والتركييب على رقاقة نوع XC3S500 FPGAs. معمارة الكبس بخط الأنابيب تتأخر بمقدار 166 نبضة وتكون قادرة على كبس أو فتح كبس صورة بحجم 512x512 نقطة صورية وبزمن 5,2 ملي ثانية حيث تصل اكبر معالجة بحدود 190 إطار للثانية الواحدة.

1. INTRODUCTION

JPEG (Joint Photographic Expert Group) is a well-known method for compressing still image and has been adopted as the compression standard for still photographic images [1]. JPEG compression algorithm is very complex and supports different operation modes [1,2]. The professionals in a software and hardware designers implement baseline mode, the one most widely used across the industry [3]. The baseline mode will be used as a reference for the design and architecture.

Implementation of the JPEG CODEC on a hardware is a big challenge for most of the researchers as it requires a complex hardware. Volcan et al. provided a JPEG compressor for gray scale images directed to Altera Flex10KE FPGAs [4]. It processes an image of 640 x 480

pixels in 8.2 ms with minimum latency of 238 clock cycles. In 2007, Tumeo et al. [5] designed a JPEG encoder by using Virtex II-Pro XC2VP30, they proposed a mixed HW/SW architecture. Hardware JPEG CODEC was synthesized for Xilinx Virtex-II FPGA device on ARM926EJS emulation base board which can operate at frequencies up to 6MHz [2].

A parallel image compression system for high-speed camera was presented by Nishikawa et al. [6]. The proposed architecture requires much less hardware with high quality reconstructed image. In this paper, a high performance JPEG CODEC was proposed for gray images. All JPEG CODEC blocks were designed, optimized and coded in VHDL, a hardware description language. Full simulation of the design was done with ISE10.1, ModelSim which was chosen for synthesis. Xilinx™, mapping, placement and routing tool were used at final

* Corresponding author: E-mail : abdlat_1986@yahoo.com

Nomenclature

CODEC	COMpress DECompress
DLS	Encoder Level Shift
ELS	Encoder Level Shift
FDCT	Forward Discrete Cosine Transform
IDCT	Inverse Discrete Cosine Transform
IEntropy	Inverse Entropy
InvQuantization	Inverse Quantization
IZigzag	Inverse Zigzag
JPEG	Joint Point Expert Group

stages of prototyping. JPEG CODEC performance was tested on XC3S500 Starter kit.

2. JPEG CODEC ARCHITECTURE

The block diagram of the JPEG CODEC is shown in Fig. 1, it can work as encoder as well as decoder of JPEG system. JPEG CODEC utilizes Encoder Level Shift/Decoder Level Shift (ELS/DLS), FDCT/IDCT & Quantization/InvQuantization, Zigzag/IZigzag and

$$z(u, v) = \frac{1}{\sqrt{M}} \sum_{m=0}^{M-1} \sqrt{2} C(u) \cdot \cos \left[\frac{(2m+1)u\pi}{2M} \right] \left\{ \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} \sqrt{2} C(v) \cdot \cos \left[\frac{(2n+1)v\pi}{2N} \right] \cdot x(m, n) \right\} \quad (1)$$

$$(m, n) = \frac{2C(u)C(v)}{\sqrt{M \cdot N}} \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} z(u, v) \cdot \cos \left[\frac{(2m+1)u\pi}{2M} \right] \cos \left[\frac{(2n+1)v\pi}{2N} \right] \quad (2)$$

Entropy/IEntropy. All modules are shared between JPEG encoder and decoder, thereby considerably reducing the total size of the JPEG CODEC.

The designed baseline JPEG CODEC architecture can be broadly classified into five main blocks; they are described in detail below.

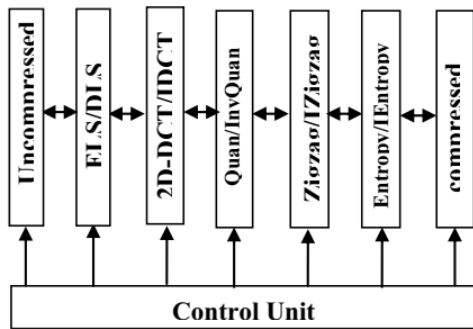


Fig. 1. Block diagram of JPEG CODEC.

3. ENCODER LEVEL SHIFT/DECODER LEVEL SHIFT (ELS/DLS)

This block reads the input samples from the input buffer. It generates a signal to subtract or add 128, according to the encoding or decoding application. The input to this block is 8-bit data while the encode operations vary from (0 to 255). Each input sample is subtracted by 128, thus changing the range between (-128 to 127). This block makes the values to zero-center and converts them from unsigned value to the signed value. When this block operates as a decoder its input ranges between (128- to 127+). Each input sample is added by 128, thus changing

the range from (0 to 255). This block converts the values from signed to unsigned value. The level shifted the output data which given to DCT/IDCT block.

4. 2D DCT/IDCT BLOCK

The 2D DCT is computationally intensive and as such there is a great demand for high speed, high throughput and short latency computing architectures. Several hardware design methods for the implementation of the 2D DCT have been developed in the recent years [7-10].

The proposed 2D DCT architecture targets power efficiency by minimizing the number of the arithmetic operations, and it is design using row column decomposition technique. The major concern in finding the 1D DCT/IDCT is the number of multipliers which are reduced by a factor of two.

The two dimensional 2D DCT in Eq. (1) transforms an (8x8) block of picture samples $x(m, n)$, into spatial frequency components $Z(u, v)$ for $0 \leq u, v \leq 7$. The IDCT in Eq. (2) performs the inverse transform for $0 \leq m, n \leq 7$. In Eqs. (1) and (2), $\alpha(0)=1/\sqrt{2}$ and $\alpha(j)=1, j \neq 0$.

This transformation can also be expressed in a matrix notation

$$Z = CYT, Y = CXT \quad (3)$$

where C is an N×N matrix whose basis vectors are sampled cosines. X is (8x8) input matrix and Y is the intermediate result. The 2D DCT/IDCT, as shown in Fig. 2, was decomposed in two 1D DCT/IDCT (as written in Eq. (3)), named as Row-DCT/IDCT and Column-DCT/IDCT and a Transpose Buffer, so Eq. (4) is rewritten in matrix form:

$$\begin{pmatrix} Y(0) \\ Y(2) \\ Y(4) \\ Y(1) \\ Y(3) \\ Y(5) \\ Y(7) \end{pmatrix} = \begin{pmatrix} A & A & A & A & 0 & 0 & 0 & 0 \\ B & C & -C & -B & 0 & 0 & 0 & 0 \\ A & -A & -A & A & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & D & E & F & G \\ 0 & 0 & 0 & 0 & E & -G & -D & -F \\ 0 & 0 & 0 & 0 & F & -D & G & E \\ 0 & 0 & 0 & 0 & G & -F & E & D \end{pmatrix} \begin{pmatrix} X(0)+X(7) \\ X(1)+X(6) \\ X(2)+X(5) \\ X(0)-X(7) \\ X(1)-X(6) \\ X(2)-X(5) \\ X(3)-X(4) \end{pmatrix} \quad (4)$$

As a result, the separable 2D DCT computation can be obtained by using 1D DCT computations as follows:

$$2D - DCT(x) = 1D - DCT(1D - DCT(x))^T \quad (5)$$

In 2D IDCT, similarly, a separable M×N point 2D IDCT can be obtained by row-column decomposition method. Thus the 2D-IDCT computation using 1D-IDCT computations is as follows.

$$2D - IDCT(z) = 1D - IDCT((1D - IDCT(z))^T) \quad (6)$$

$$X = TYT \quad (7)$$

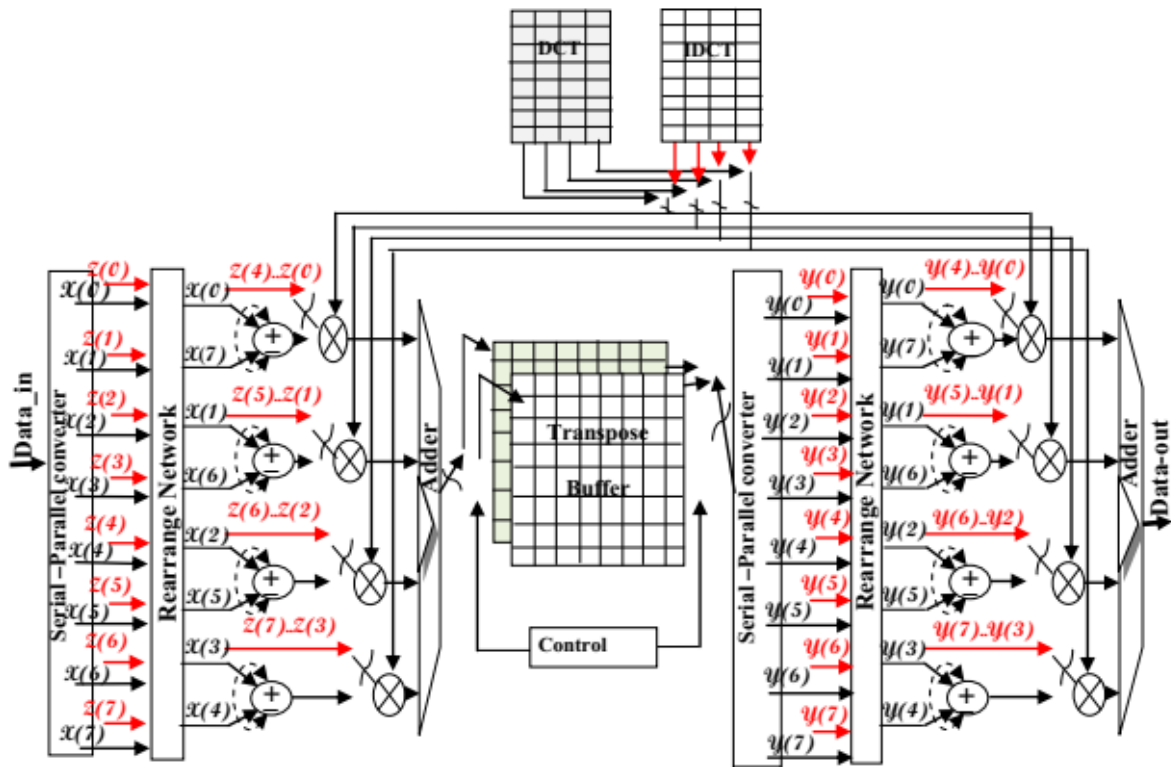


Fig. 2. The 2D DCT/IDCT unit architecture.

4.1. Quantization/Invquantization

Quantization is defined as a division of each DCT coefficient by the corresponding quantization value. Division operations are not efficient for a hardware resources so they are replaced by multiplication and shift operations. In inverse-quantization, the quantized DCT coefficient that multiplied by the corresponded coefficient in a quantization table thereby the two operations use a multiplication operation. Fig. 3 shows the quantization/inverse-quantization architecture. This architecture uses two ROMs memories and one multiplier. In quantization process, the values in the standard quantization tables used for division were transformed into multiplier values and stored in the ROM. The multiplier is shared between quantization and inverse-quantization process.

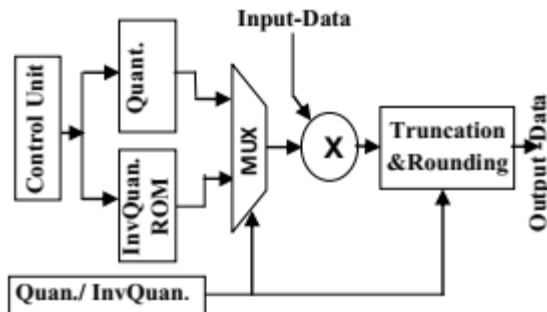


Fig. 3. The Quantization/InvQuantization unit.

4.2. Zigzag/Izigzag

In zigzag scanning, the quantized DCT coefficients read in a zigzag order. This scan puts the high frequency coefficients together, each of these coefficients is usually zero. The architecture for the zigzag/inverse zigzag is

shown in Fig. 4, consisting two transpose buffers RAM1 and RAM2, which are used to synchronize the procedure in pipeline manner, after 64 locations are written, RAM1 goes into read mode and RAM2 goes into write mode. The buffer latency is 64 clock cycles where the scanning order requires that some of the later coefficients that available in the beginning. Every 64 clock RAMs make change from write mode to read mode alternatively by Toggle2 which switch signal from '0' to '1' at every 64 clock cycles. The implementation of the zigzag/Inverse-zigzag on the same architecture makes use of more optimized resources, by sharing these resources where two RAMs are used in zigzag and inverse-zigzag instead of using two RAMs for each one.

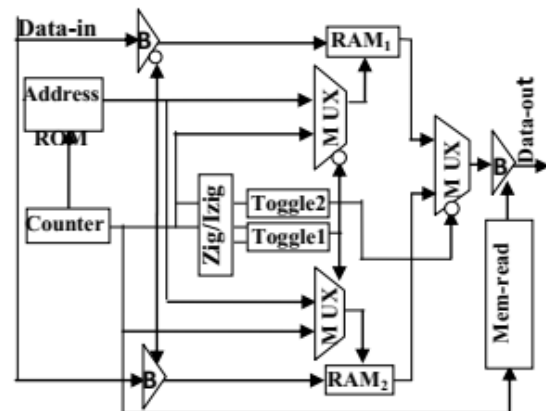


Fig. 4. The Block diagram of Zigzag.

4.3. Entropy/Inverse Entropy

The proposed entropy coding module consists of two interrelated modules, as shown in Fig. 5, the run length encoder and the Huffman encoder modules. The output of

zigzag is input to the run length encoder as data stream. There are three outputs of the run length encoder; the first output is an amplitude value, the second is a run length of zeros, and the third is the flag bit that indicates nonzero value. These outputs will provide input to Huffman encoder.

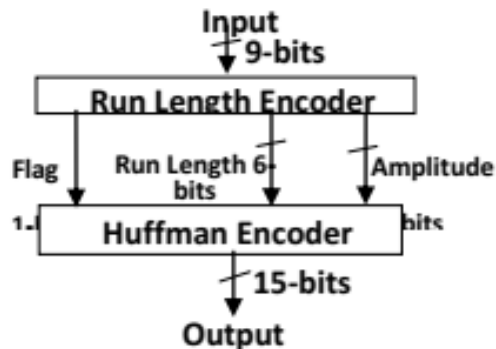


Fig. 5. Entropy encoder block diagram.

The run length encoder is a fairly simple concept which looks for runs of zeros in the data stream as shown in Fig. 6. The run length encoder will output an amplitude value, a run length of zeros, and flag as indicator for nonzero coefficients. Every coefficient which equals zero increases an internal counter which counts the run of zeros. Every nonzero coefficient input to the run length encoder will output a set of values.

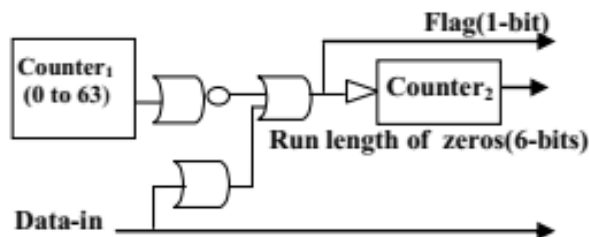


Fig. 6. The run length of the encoder architecture.

The run length, amplitude, and the flag values from the run length encoder are used as input to the Huffman encoder. Instead of assigning code word as in the standard Huffman Tables to produce the output which has unequal number of bits and the maximum is 16-bits as well as amplitude value, the proposed architecture will produce the run length of zeros with amplitude as output data. Where its size is 15-bits which are equal all the output, thereby will make the reconstruction of the compressed data to be less complex and easier to be predicted than in the case of Huffman tables. Another advantage is that there is no need for ROM to store Huffman tables.

The reversible operations are used in the entropy decoder. The compressed input data has an equal lengths of number of bits. The Huffman decoder separates the run length and amplitude. If the run length value is zero, it will send the amplitude directly as input to inverse zigzag stage, otherwise, it will send zero values as input and decrease the value of the run length of zeros, until the run length value becomes zero then it will send the amplitude value and read another input from the compressed data. Fig. 7 shows the entropy decoder.

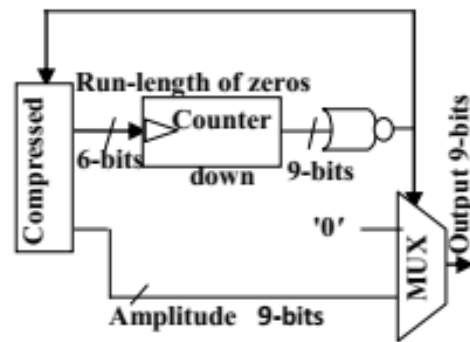


Fig. 7. The block diagram of the entropy decoder.

4.4. FPGA Implementation of the JPEG CODEC

The proposed architectures have been described by means of VHDL language. The results are summarized in Table 1.

The latency for the architecture is 166 clock cycles for the compressor and 165 clock cycles for the decompressor. Figs. 8 and 9 show the simulation results for the compressor and the decompressor respectively.

Table 1

Synthesis of JPEG CODEC for Xilinx Spartan- 3E XC3S500.

FPGA Resource	JPEG CODEC		
No. of Slices	3132	out of 4656	67%
No. of Slice Flip Flops	3209	out of 9312	34%
No. of 4 input LUTs	4442	out of 9312	47%
No. of bonded IOBs	18	out of 232	8%
No. of Slices	3132	out of 4656	67%
BRAMs	0		
Mult18x18s	9	out of 20	45%
Performance	78,92 MHz		

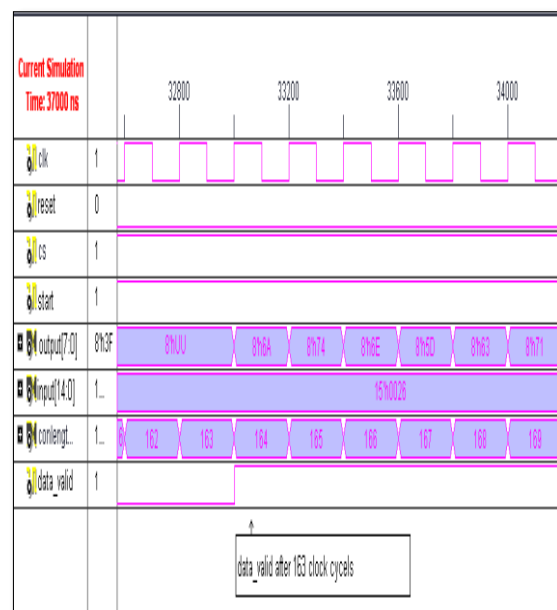


Fig. 8. Simulation results for JPEG Compressor.

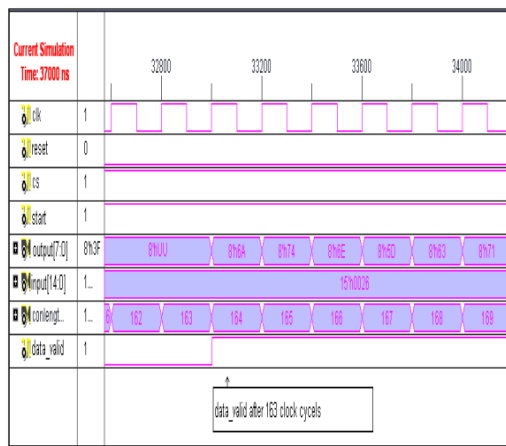



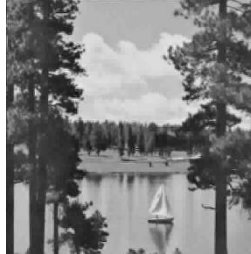


Fig. 9. Simulation results for JPEG DeCompressor.

Table 2 shows the PSNR and compression ratio (CR) for some standard of gray-scale (512×512) images.

Table 2
PSNR and CR for some of standard gray-scale images.

	
Goldhill PSNR=28,0620dB	Peppers SNR=27,9920dB
	
CR=16	CR=16,29
Boat PSNR=28,4104dB CR=15,93	Sailboat PSNR=26,6533dB CR=12,6

4.5. JPEG Architectures Comparison

This paper verifies an overall recent comparison for the JPEG CODEC which involves the required multipliers and the performance of the proposed architectures as compared to the previous JPEG CODEC architectures. All architectures are pipelined; the proposed architecture reduces the arithmetic operations by 45% as compared to the newer JPEG CODEC that is illustrated in Table 3.

4.6. System on Chip

Figure 10 shows the experimental setup for the JPEG CODE system. It consists of host terminal PC and Spartan-3E XC3S500 FPGA. Host terminal PC is connected to the soft processor through USB (Universal Serial Bus) cable.

Data in memory that is embedded on FPGA could be read and displayed on the computer screen by using a Microblaze processor which makes interface between the data buffer and I/O peripherals which are available on the board as RS232. This peripheral allows to upload the image from the memory in FPGA and displays it on the computer.

Table 3
Comparison for the JPEG CODEC.

Ref.	[4]	[11]	[2]	Proposed
No. of Multiplier	-	-	20	9
Latency (Cycles)	238	243	NA	166 for C 165 for D
Frame/sec	122,4	114	NA	190
Frequency (MHZ)	37,6	39,6	6	78,969
BRAMs	2	6	1 Dual Port	-
FPGA	Fle- x10KE	Fle- x10KE	XC2-V8000	XC3- S500E
Function	C	C	C&D	C&D

C=Compression Operation
D=Decompression Operation
NA=Not Available

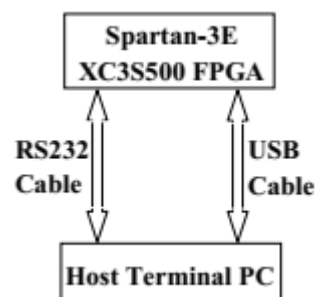


Fig. 10. Emulation setup.

5. CONCLUSIONS

A pipelined JPEG CODEC is designed and implemented on Spartan-3E FPGA for gray scale images format. The proposed architecture was designed in a modular that performs forward/backward function to allow the reuse of all modules in any other future designs. A (8x8) points low power 2D-DCT/IDCT is also implemented using a shared transpose buffers between DCT and IDCT. The quantization and inverse quantization unit are shared in one multiplier.

In the entropy encoder the Huffman tables that are required to reduce memory since no need for storing these tables in ROM, thus it makes the process reconstruction easier with suitable compression ratio as shown in Table 2. Other stages are also implemented in optimized method by sharing resources and latency for overall JPEG CODEC. The designed JPEG CODEC gives a higher throughput which is suitable for real time video applications since the latency equals to 166 clock cycles.

REFERENCES

- [1] Achaya T, Tsai P. JPEG2000 Standard for Image Compression Wiley-Interscience; 2005.
- [2] Tiwari T, Reddy SC. Performance measurement of a fully pipelined JPEG codec on emulation platform. 2nd International Advance Computing Conference (IACC), IEEE 2010: p. 167–171.
- [3] Yang H, Wang L. Joint optimization of run-length coding, huffman coding, and quantization table with complete baseline JPEG decoder compatibility. IEEE Transactions on Image Processing 2009; 18(1): 63–74.
- [4] Volcan L, Porto RC, Bampi S, Silva IS. A FPGA based design of a multiplierless and fully pipelined JPEG compressor. 8th Euromicro conference on Digital System Design (DSD'05) 2005; IEEE: p. 210–213.
- [5] Tumeo A, Monchiero M, Palermo G, Ferrandi F, Sciuto D. An internal partial dynamic reconfiguration implementation of the JPEG encoder for low-cost FPGAs. IEEE Computer Society Annual Symposium on VLSI (ISVLSI'07) 2007; p. 449–450.
- [6] Nishikawa Y, Kawahito S, Inoue T. A parallel image compression system for high-speed cameras. Imaging Systems and Techniques, IEEE International Workshop on 2007: p. 53–57.
- [7] Leong MP, Leong HW. A variable-radix digit-serial design methodology and its application to the discrete cosine transform. IEEE Transactions on Very Large Scale Integration (VLSI) Systems 2003; 11 (1): 90–104.
- [8] Gloster C, Gay JW, Amoo M, Chouikha M. Optimizing the design of a configurable digital signal processor for accelerated execution of the 2-D discrete cosine transform. 39th Hawaii International Conference on System Sciences 2006: p. 250c-250c.
- [9] Megalingam RK, Krishnan V, Sarma V, Mithun M, Srikumar R. Hardware implementation of low power, high speed DCT/IDCT based digital image watermarking. International Conference on Computer Technology and Development 2009: p. 535–539.
- [10] Sun CC, Donner P, Gotze J. Low-complexity multi-purpose ip core for quantized discrete cosine and integer transform. Circuits and Systems, ISCAS 2009. IEEE International Symposium: p. 3014–3017.
- [11] Agostini LV, Bampi S, Silva IS. High throughput architecture of JPEG compressor for color images targeting FPGAs. Electronics, Circuits and Systems, ICECS '06. 13th IEEE International Conference 2016: p.180–183.