TJES
ISSN: 1813-162X
Tikrit Journal of Engineering Sciences
available online at: http://www.tj-es.com

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

Abdulkreem Mohameed Salih [1,*] , Ahlam Fadhil Mahmood [2]
[1] *Computer Engineering Department, University of Mosul, Mosul, Iraq*
E-mail: abdlat_1986@yahoo.com
[2] *Computer Engineering Department, University of Mosul, Mosul, Iraq*
E-mail: ahlam.mahmood@gmail.com

(Received 3 January 2012, Accepted 28 May2012, Available online 30 September 2017)

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, all modules are sharing between JPEG encoder and decoder circuit . Each module was designed to implement 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 were described in VHDL, simulated and physically mapped to XC3S500 FPGAs. The JPEG CODEC pipeline has a minimum latency of 166 clock cycles, given the full modular pipeline depth. The CODEC can process a 512x512 pixels still image in 5.2ms, reaching a maximum processing rate of 190 frames per second.

***Keywords*:** JPEG, CODEC, compressor, decompressor, FPGA, DCT, IDCT, VHDL.

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. معمارية الJPEG CODEC انجزت بإخراجية عالية مستخدمة خاصية خط الأنابيب باستخدام تقنية إلFPGA. المعماريات المصممة موضحة في هذا البحث وموصوفة بلغة VHDLتم تنفيذ المحاكاة والتركيب على رقاقة نوعXC3S500 FPGAs . معمارية الكبس بخط الأنابيب تتأخر بمقدار 166نبضة وتكون قادرة على كبس أو فتح كبس صورة بحجم 512X512 نقطة صورية وبزمن 5,2 ملي ثانية حيث تصل اكبر معالجة بحدود 190 أطار للثانية الواحدة.

Symbols

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

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

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]. Software professionals and hardware designers implement baseline mode, the one most widely used across the industry [3]. The baseline mode will be used as reference for the design and architecture.

Implementation of the JPEG CODEC on 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 8.2ms 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, it can operate at frequencies up to 6MHz [2].

A Parallel image compression system for high-speed cameras 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 was chosen for synthesis. Xilinx™, mapping, placement and routing tool were used at final stages of prototyping. JPEG CODEC performance was tested on XC3S500 Starter kit.

JPEG CODEC Architecture

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

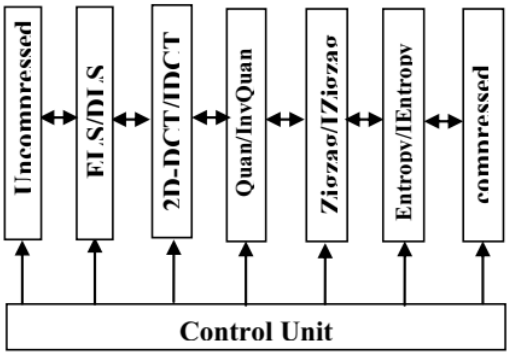


Fig. 1. Block diagram of JPEG CODEC

Encoder Level Shift/Decoder Level Shift (ELS/DLS)

This block reads the input samples from the input buffer. It generates signal to subtract or add 128, according to the encoding or decoding application. The input to this block is 8-bit data, in 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 signed value. When this block operate as decoder it's input ranging 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 value to unsigned value. The level shifted output data is given to DCT/IDCT block.

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 recent years [7-10].

The proposed 2D DCT architecture targets power efficiency by minimizing the number of 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 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 ≤ u,v ≤ 7. The IDCT in Eq. (2) performs the inverse transform for 0≤ m, n ≤ 7. In Eqs. (1) and (2), α(0)=1/√2 and α(j)=1, j≠0.

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

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

This transformation can also be expressed in 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) rewritten in matrix form:

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

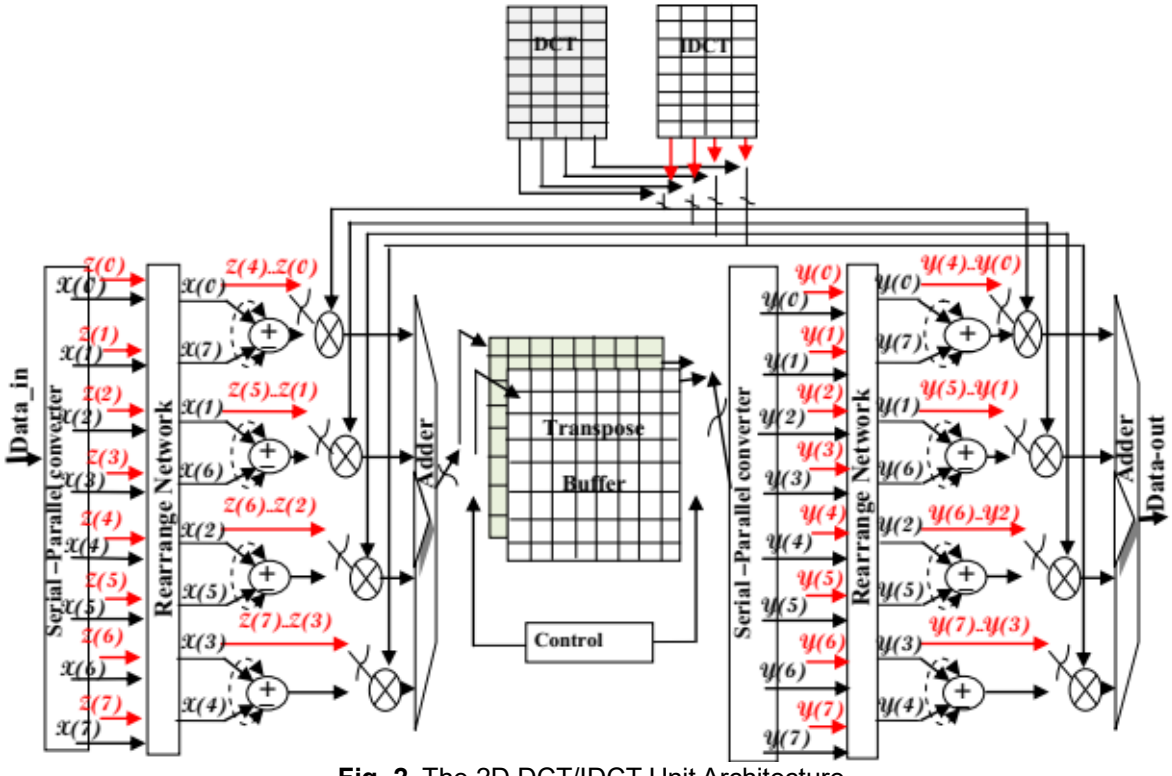


Fig. 2. The 2D DCT/IDCT Unit Architecture

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

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

In 2D IDCT, similarity, 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.

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

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

$$\begin{pmatrix} X''(0) \\ X''(1) \\ X''(2) \\ X''(3) \\ X''(4) \\ X''(5) \\ X''(6) \\ X''(7) \end{pmatrix} = 2x \begin{pmatrix} A & A & A & A & 0 & 0 & 0 & 0 \\ B & C & -C & -B & 0 & 0 & 0 & 0 \\ A & -A & -A & A & 0 & 0 & 0 & 0 \\ C & -B & B & -C & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & D & E & F & G \\ 0 & 0 & 0 & 0 & E & -G & -D & -F \\ 0 & 0 & 0 & 0 & F & -D & G & E \\ 0 & 0 & 0 & 0 & G & -F & E & -D \end{pmatrix}^{-1} \begin{pmatrix} z(0) \\ z(1) \\ z(2) \\ z(3) \\ z(4) \\ z(5) \\ z(6) \\ z(7) \end{pmatrix} \quad (8)$$

where

$X(0) = [X''(0) + X''(4)]/2,$
$X(1) = [X''(1) + X''(5)]/2,$
$X(2) = [X''(2) + X''(6)]/2,$
$X(3) = [X''(3) + X''(7)]/2,$
$X(4) = [X''(3) - X''(7)]/2,$
$X(5) = [X''(2) - X''(6)]/2$
$X(6) = [X''(1) - X''(5)]/2,$
$X(7) = [X''(0) - X''(4)]/2.$

Quantization/Invquantization

Quantization is defined as division of each DCT coefficient by the corresponding quantization value. Division operations are not efficient for hardware resources so it replaced with multiplication and shift operations. In inverse-quantization, the quantized DCT coefficient multiplied by the correspond coefficient in quantization table thereby the two operations use multiplication operation. Figure (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.

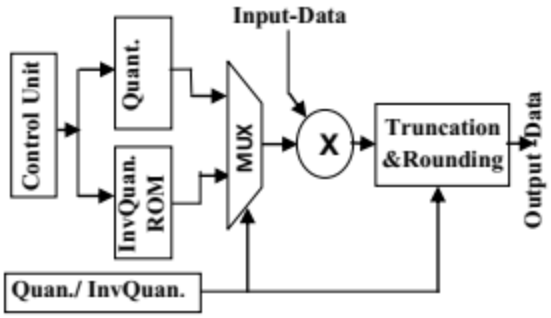


Fig. 3. The Quantization/InvQuantization unit.

Zigzag/Izigzag

In zigzag scanning, the quantized DCT coefficients read in zigzag order. This scan puts the high frequency coefficients together, each of these coefficients are usually zero. The architecture for the zigzag/inverse zigzag is shown in Fig. (4), consisting of two transpose buffer RAM1 and RAM2, which are used to synchronize the procedure in pipeline manner, after 64 locations are written into, RAM1 goes into read mode and RAM2 goes into write mode. The buffer latency is 64 clock cycles where the scanning order require that some of the later coefficients be 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. Implementation the zigzag/Inverse-zigzag on the same architecture make use of resources more optimized, by sharing these resources such as two RAMs are used in zigzag and inverse-zigzag instead of using two RAMs for each one.

Entropy/Inverse Entropy

The proposed entropy coding module consists of two interrelated modules, as shown in figure 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; first output is an amplitude

value, second is a run length of zeros, and third is the flag bit that indicates nonzero value. These outputs will input to Huffman encoder.

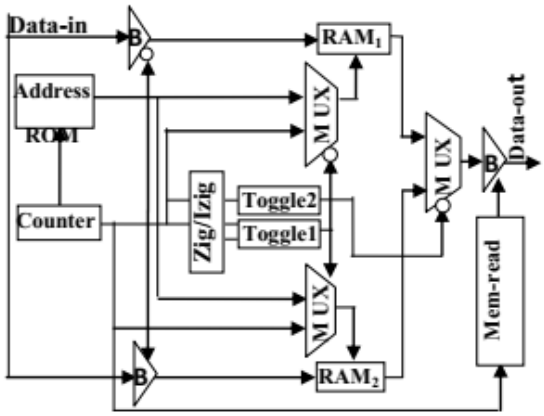


Fig. 4. The Block diagram of Zigzag.

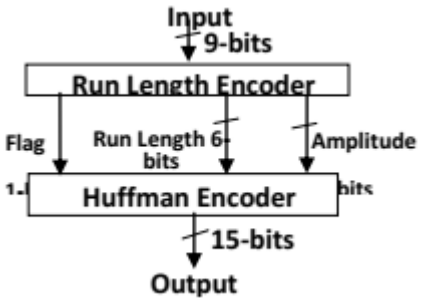


Fig. 5. Entropy Encoder Block diagram.

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 that equals zero, increments an internal counter which counts the run of zeros. Every nonzero coefficient input to the run length encoder will output a set of values.

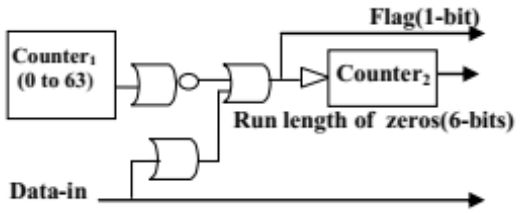


Fig. 6. The run length encoder architecture.

The run length, amplitude, and the flag values from the run length encoder are input to the Huffman encoder. Instead of assign code word as in standard Huffman Tables to produce the output which have unequal in 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 the size of it is 15-bits and this equal for all output, thereby will make the reconstruction of the compressed data is less complex and easy to predict than use Huffman tables, another advantage is that no need for ROM to store Huffman tables.

In entropy decoder, reversible operations are used. The compressed data input has equal length of number of bits. The Huffman decoder separate the run length and amplitude. If the run length value is zero, send the amplitude directly as input to inverse zigzag stage, or else it will send zero values as input and decrease the value of run length of zeros, until the run length value become zero, send the amplitude value and read another input from compressed data. Figure (7) shows the entropy decoder.

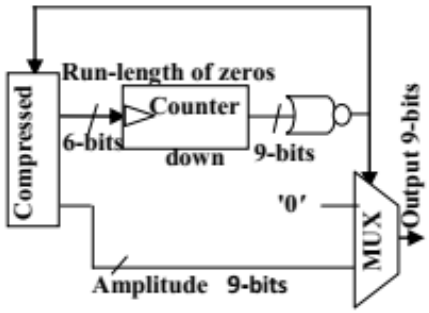


Fig. 7. The block diagram of entropy decoder.

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 compressor and 165 clock cycles for decompressor. Figures (8) and (9) show the simulation results for compressor and 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

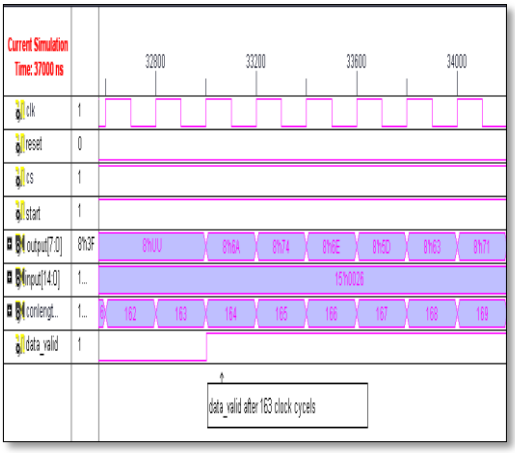


Fig. 8. Simulation results for JPEG Compressor

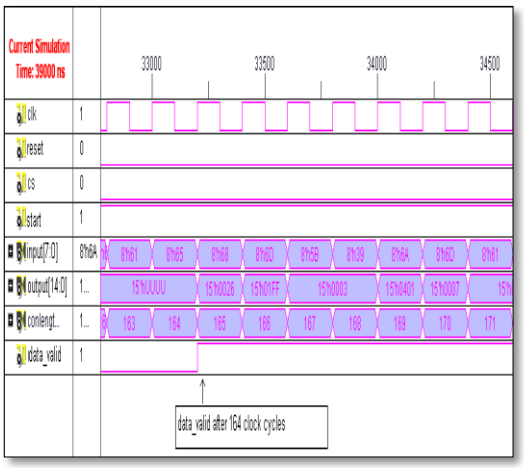
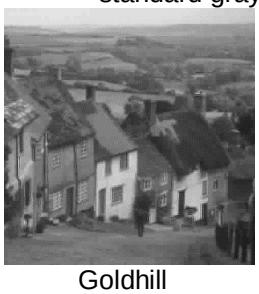


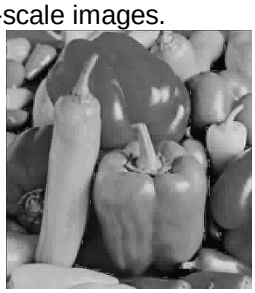
Fig. 9. Simulation results for JPEG DeCompressor.

Table (2) shows the PSNR and compression ratio (CR) for some standard of gray-scale (512x512) images.

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



Goldhill
PSNR=28,0620dB
CR=16



Peppers
SNR=27,9920dB
CR=16,29

Boat
PSNR=28,4104dB
CR=15,93

Sailboat
PSNR=26,6533dB
CR=12,6

JPEG Architectures Comparison

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

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 soft processor through USB (Universal Serial Bus) cable. Data in memory that embedded on FPGA can be read and display on the computer screen by using Microblaze processor which make interface between the data buffer and I/O peripherals that available on the board as RS232. This peripheral allows to upload image from memory in FPGA and display it on

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

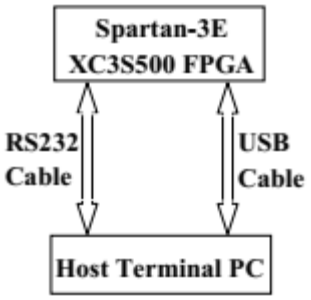


Fig. 10. Emulation setup.

Conclusions

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

In entropy encoder the Huffman tables that reduces memory is required since no need for store this tables in ROMs, make reconstructed process 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 give 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 Awiley-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. Proceedings of the 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.