Open access • Proceedings Article • DOI:10.1145/1838002.1838035

FPGA based implementation of baseline JPEG decoder — Source link

Jahanzeb Ahmad, Kamran Raza, Mansoor Ebrahim, Umar Talha

Institutions: Iqra University

Published on: 16 Dec 2009 - Frontiers of Information Technology

Topics: JPEG 2000, JPEG File Interchange Format, Soft-decision decoder, Lossless JPEG and JPEG

FPGA Based Implementation of Baseline JPEG Decoder

Jahanzeb Ahmad, Mansoor Ebrahim
Faculty of Engineering, Sciences and Technology, IQRA University, Karachi Pakistan.
jahanzeb.ahmad@iqra.edu.pk , mansoorebrahim99@hotmail.com

***Abstract*--The JPEG standard (ISO/ IEC 10918-1 ITU-T Recommendation T.81) defines compression techniques for image data. As a consequence, it allows to store and transfer image data with considerably reduced demand for storage space and bandwidth. From the four processes provided in the JPEG standard, only one, the baseline process is widely used. In this paper FPGA based High speed, low complexity and low memory implementation of JPEG decoder is presented. The pipeline implementation of the system, allow decompressing multiple image blocks simultaneously. The hardware decoder is designed to operate at 100MHz on Altera Cyclon II or Xilinx Spartan 3E FPGA or equivalent. The decoder is capable of decoding Baseline JPEG color and gray images. Decoder is also capable of downscaling the image by 8. The decoder is designed to meet industrial needs. JFIF, DCF and EXIF standers are implemented in the design.**

I. INTRODUCTION

Communication and storage cost are reduced by doing data compression. Data compression techniques can be divided into two categories "losy" and "lossless". Lossless compression model are based on entropy coding schemes.
This model is widely used for text and data compression. In lossless compression model exact data is obtained at the receiver. Lossy compression model produces close approximation of the original data at the receiver. Video, Image and audio compression commonly use lossy compression Compression ratio up to 100:1 can be achieved depending on the fidelity of the data.
There are several standards/formats for image compression/ decompression. Joint Photographic Experts Group (JPEG) [1, 17], Graphics Interchange Format (GIF) [7 8], Portable Network Graphics (PNG) [9], JPEG 2000 [10], Tagged Image File Format (TIFF) [11].
JPEG is a very well know image compression standard. It is widely adopted as compression standard for still images. Joint Photographic Expert Group (JPEG) is a joint workgroup of three international standard organizations, International Organization for Standardization (ISO), International Telegraph and telephone consultative committee (CCITT) and International Electrotechnical commission (IEC).
Enormous amount of data storage is required for digital images/video. An uncompressed color image requires 24 bits for each picture element (pixel). A 6 Mega pixel (3038 X 2012) camera requires 17.5 Mega Bytes, when stored uncompressed, same image when compressed with JPEG take almost 1.7 Mega bytes depending on the compression ratio. En-hui Yang, Longji Wang [18] proposed an algorithm which can further improve this ratio, the algorithm is iterative, which is more complex to implement in Hardware.
Digital devices are now more popular then analog devices especially in the field of multimedia (Audio, Video and Image) because of amazing improvement in digital signal processing algorithms and fast hardware. Digital storage media is more reliable and less effected by noise and distortion.
Real-time implementation of JPEG encoder or decoder requires efficient and fast hardware architecture. So architecture specific implementation is required to achieve real-time results. Variety of architecture designs capable of supporting real time image/video processing already exists such as ASIC, FPGA, Microprocessor and Digital signal processor based design, which implements different algorithms for image and video processing. But only a few efficient architectures are implemented for Image and video compression, decompression, processing [12, 13, 14, 15, 16, 19, 20, 21, 22, 23, 32]. Shizhen Huang and Tianyi Zheng [12] proposed an architecture for PNG image decoding, they used combination hardware and software approach which reduce the throughput of the system. Zulkalnain MohdYousof, et al. [13] proposed a Digital Signal processor based JPEG Decoder but it can only support small resolution images. R. P. Jacobi et al. [14] proposed an FPGA based JPEG decoder design but its maximum operating frequency is 38.7 MHz on Vertex 6 which is very slow for commercial design. Mario kovac and N. Ranganathan [15] presented encoder architecture which is capable of operating at 100 MHz and can support 1024x1024 spatial color image resolution. Mohammed Elbadri et al [16] also proposed a FPGA based design for JPEG decoder this design also has low operating frequency, 67 MHz. Kyeong-Yuk Min and Jong-Wha Chong [19] proposed an architecture for JPEG Encoder. Zulkalnain MohdYusof et al [20], proposed a Digital Signal Processor (DSP) based architecture, DSP based systems have low development time and cost but low throughput as compare to FPGA.
FPGA is relatively young technology. FPGA can provide speed, performance and flexibility because parallel and

pipelined implementation of Algorithm is possible. FPGA provide a better solution because hardware is designed for specific algorithm.

In this paper we proposed a FPGA based JPEG decoder architecture, which gives fast and efficient results. The paper is organized as follow: In section 2 we discuss JPEG in general. In section 3 JPEG stream is discussed. Hardware implementation is discussed in section 4. Synthesis reports are discussed in subsequent section. Finally, results and conclusions are discussed.

II. JPEG COMPRESSION OVERVIEW

Principles of JPEG can be explained better to take a look at the steps of encoding rather than decoding. Therefore, despite the fact that a decoder has been developed, due to better understanding the steps of encoding. The steps of decoding will be the inverse of the encoding steps but in reverse order (see Fig. 1 and Fig. 2).

The human eye is more sensitive to brightens then colors [33]. Almost no loss in visual perception quality can be achieved if chrominance component is stored in half resolution then luminance component [33]. JPEG images are stored in YCbCr color space rather then RGB. CCIR Rec 601 [6] defines the method of conversion between RGB and YCbCr.

Most JPEG encoders reduce the chrominance components to half of the resolution in both dimensions by taking the mean value of each 2x2 block. This sampling method is called "4:2:0". Another sampling method evolved from analog television signals [33] is "4:2:2" where chrominance components are reduced only in the horizontal dimension. For completeness the "4:4:4" method should be mentioned it does not reduce any component's resolution. For grayscale ("4:0:0") images only the Y component is processed. Fig.. 3 illustrate the described sampling methods. If the "4:2:0" or "4:2:2" sampling method is used this is one of two steps in the compression process where information is lost.

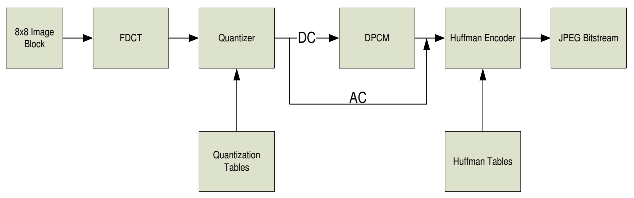


Fig. 1. JPEG Encoder

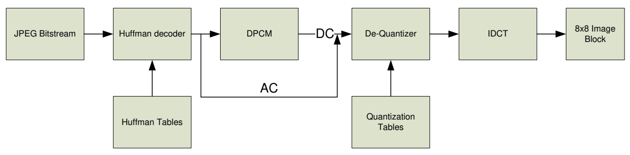


Fig. 2. JPEG Decoder

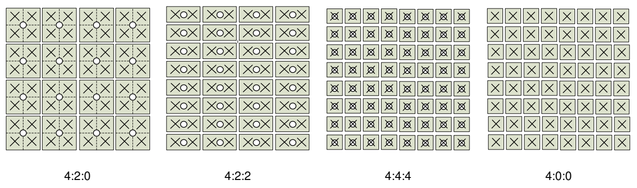


Fig. 3. Sampling

JPEG image is composed of smaller units. An image is composed of Minimum coded Units (MCUs) which consist of square blocks of 8x8 pixels. It depends on the chosen sampling method how many 8x8 blocks form an MCU, in Fig. 4 the 4:2:0 sampling is shown. The order in which the units will be processed is always from left to right and from top to bottom as shown in fig. 4. For the MCUs it is also important to keep the color-decomposition in mind.

Picture when displayed on screen or printed on paper is in spatial domain. DCT transforms a picture into frequency domain [34]. Human vision system is more sensitive to low frequency then higher frequency [33]. Since neighbor pixels are highly correlated and are in low frequency, the output of DCT result in most of the block energy being stored in the lower spatial frequencies. Higher frequencies will have values equal to or close to zero so they can be ignored without have significant loss in image quality.

The input data to be processed is a two-dimensional 8x8 block, therefore we need a two-dimensional version of the discrete cosine transformation. Since each dimension can be handled separately, the two-dimensional DCT follows straightforward form the one dimensional DCT. A one-dimensional DCT is performed along the rows and then along the columns, or vice versa.

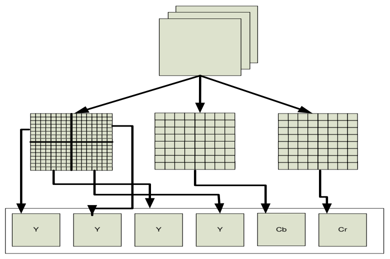


Fig. 4. 4:2:0 MCU

JPEG uses a zero-shift in the input samples to convert 8-bit image data from the range 0 to 255 to the range of -128 to +127. This is done by subtracting 128 before DCT is calculated. DCT is defined in equation (1) and IDCT is defined in equation (2)

FDCT: $$S_{vu} = \frac{1}{4} C_u C_v \sum_{x=0}^{7} \sum_{y=0}^{7} s_{yx} \cos\frac{(2x+1)u\pi}{16} \cos\frac{(2y+1)v\pi}{16} \qquad (1)$$

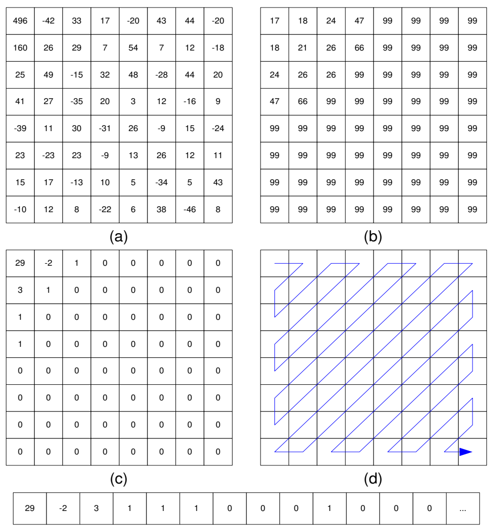
IDCT: $$s_{yx} = \frac{1}{4} \sum_{u=0}^{7} \sum_{v=0}^{7} C_u C_v S_{vu} \cos\frac{(2x+1)u\pi}{16} \cos\frac{(2y+1)v\pi}{16} \qquad (2)$$

$C_u, C_v = 1/\sqrt{2}$ for u,v = 0 $C_u, C_v = 1$ otherwise

The "Quantization" is a key step in the compression process since less important information is discarded.

The advantage of the representation in the frequency domain is that, unlike in spatial domain before the DCT, not every

dimension has the same importance for the visual quality of the image. Removing the higher frequencies components will reduce the level of detail but the overall structure remains, since it is dominated by the lower frequency components.
The 64 values of a 8x8 block will be divided according to the 64 values of an 8x8 matrix called the quantization table. There is no information lost in the division of the coefficients itself, but the result is then rounded to the next integer afterwards. The higher the divisor, the more information about the coefficient will be positioned after the decimal point hence lost in the rounding operation.
The two dimensional order of the DCT coefficients refers to the two dimensions that the 8x8 block has in spatial domain. After the quantization step most of the coefficients towards the lower right corner are zero. The Zigzag-Mapping - as shown in Fig. 5(d) - rearranges the coefficients in a one dimensional order, so that most of the zeroes will be placed at the end. This array with many consecutive zeroes at the end is now optimized to achieve high compression in entropy encoding.



496	-42	33	17	-20	43	44	-20
160	26	29	7	54	7	12	-18
25	49	-15	32	48	-28	44	20
41	27	-35	20	3	12	-16	9
-39	11	30	-31	26	-9	15	-24
23	-23	23	-9	13	26	12	11
15	17	-13	10	5	-34	5	43
-10	12	8	-22	6	38	-46	8

(a)

17	18	24	47	99	99	99	99
18	21	26	66	99	99	99	99
24	26	26	99	99	99	99	99
47	66	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99

(b)

29	-2	1	0	0	0	0	0
3	1	0	0	0	0	0	0
1	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

(c)

(d)

29	-2	3	1	1	1	0	0	0	1	0	0	0	...

(e)

Fig. 5.

The final step is a combination of three techniques: run length encoding, variable length encoding, and Huffman encoding.
The first coefficient is called "DC"(#0) all other coefficients are called "AC" (#1 till #63).
The first coefficient (DC) is the mean value of the original 8x8 block. There is a correlation between the DC coefficients of neighboring blocks.
It is very likely that the first coefficient has the largest value. This is the most significant coefficient and therefore usually the least reduced one in the quantization step.
Most zero coefficients appear at the end. The chance to find some consecutive zeroes followed by a non-zero component is good as well. Most non-zero coefficients have very small values.
The DC coefficient will be decoded slightly different than the AC coefficients. Respecting the correlation to the neighboring blocks, just for the first block the whole DC coefficient is processed. Later blocks will only encode the difference to the preceding block's DC component; this applies for each component separately. AC and DC coefficients have different Huffman tables.
Let's look at an example block of coefficients (the one from Fig. 5(e)):
29,-2, 3, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, . . .
Let's assume that the previously decoded block of the same component had the DC coefficient 22, therefore we decode the difference 29 - 22 = 7.
7,-2, 3, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, . . .
So now we take care of the zeroes using run length encoding. The tailing zeroes will be combined in one code, called "EOB". To each non-zero code we will stick the information about preceding zeroes, so we can remove the rest of the zeroes. For the DC coefficient there will be no preceding zeroes, however, unlike for the AC coefficients, "zero" is still a valid value that has to be concerned.

$$\begin{array}{cccccccc} 7, & -2, & 3, & 1, & 1, & 1, & \underbrace{0,0,0,1,} & \underbrace{0,0,0,0,\ldots} \\ []\,7, & [0]-2, & [0]\,3, & [0]\,1, & [0]\,1, & [0]\,1, & [3]\,1, & [\mathrm{eob}] \end{array}$$

The remaining coefficients will probably be very small so that variable lengths approach seams feasible. Therefore we switch to binary representation and add the minimum number of bits needed to represent the coefficients value to the information part. Negative values will be represented by negating every bit (one's complement). This can be done because we have the information about the length, so that every positive value starts with a 1.

$$\begin{array}{cccccccc} 7, & -2, & 3, & 1, & 1, & 1, & \underbrace{0,0,0,1,} & \underbrace{0,0,0,0,\ldots} \\ []\,7, & [0]-2, & [0]\,3, & [0]\,1, & [0]\,1, & [0]\,1, & [3]\,1, & [\mathrm{eob}] \\ [\,3]\,111, & [0\,2]\,01, & [0\,2]\,11, & [0\,1]\,1, & [0\,1]\,1, & [0\,1]\,1, & [3\,1]\,1, & [\mathrm{eob}] \end{array}$$

[EOB] is coded as [0 0], [ZRL] as [15 0]; there is no other code with the structure [X 0].
Since the coefficients are usually very small there is not much gain in compressing them further. However we have not thought about the information we attached to the coefficients yet. We use 4 bits for the preceding zeroes and 4 bits for the number of bits used to store the value. These 8 bits are compressed using a Huffman table which maps the frequently occurring values to shorter bit strings and the rarely occurring values to longer bit strings. How to choose the table is left to the encoder.
Let's assume we have built the Huffman table and find the following tables:

DC Table

Huffman code	Original Value	
⋮	⋮	
110	0x3	[3]
⋮	⋮	

AC Table

Huffman code	Original Value	
00	0x00	[eob]
01	0x01	[0 1]
100	0x02	[0 2]
101	0x11	[1 1]
1100	0x03	[0 3]
⋮	⋮	
1111110	0x31	[3 1]
⋮	⋮	

Now we can construct the final bit stream:

$$\begin{array}{ccccccccccccccc}
 & 7, & & -2, & & 3, & & 1, & & 1, & & 1, & \underbrace{0,0,0} & 1, & \underbrace{0,0,0,0,\ldots} \\
[\,] & 7, & [0] & -2, & [0] & 3, & [0] & 1, & [0] & 1, & [0] & 1, & [3] & 1, & [eob] \\
[3] & 111, & [0\,2] & 01, & [0\,2] & 11, & [0\,1] & 1, & [0\,1] & 1, & [0\,1] & 1, & [3\,1] & 1, & [eob] \\
110 & 111 & 100 & 01 & 100 & 11 & 01 & 1 & 01 & 1 & 01 & 1 & 1111110 & 1 & 00
\end{array}$$

The final bit stream:

11011110 00110011 01101101 11111110 100

So we compressed the 64 bytes of input data down to less than five bytes.

IV. JPEG STREAM

JPEG standard has many parts, only parts which are compliance with applicable parts of DCF [2], Exif [4] and JFIF [3] are implemented. The resulting stream is shown in Fig. 6. The variables and parameters are defined in JPEG [1].

III. HARDWARE IMPLEMENTATION

The system is consists of different blocks as shown in the block diagram in Fig. 8. The interface of the JPEG decoder is shown in Fig. 7. The design features are

Efficient Design.

Minimum clock speed of 100 Mhz.

Target Independent design.

Synchronous design

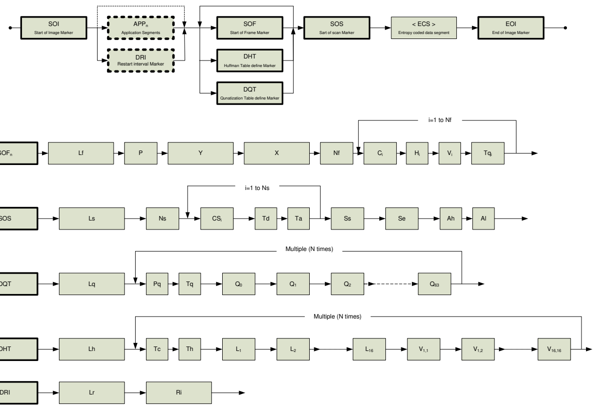


Fig. 6. JPEG Stream

The system works on 8-bit data input. When the start signal is asserted parser start to read data. If first two bytes are not Start of Image Marker parser generates the error. Otherwise it starts searching for the next marker.

JPEG stream parser, parse the input stream. Huffman tables, quantization tables and other information of the image is extracted and stored in the memory by this module. At the start of the entropy coded segment the control of the JPEG decoder is transferred from parser to Huffman decoder controller.

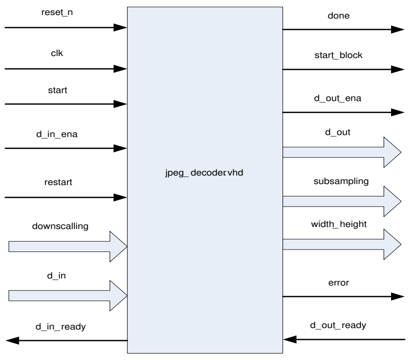


Fig. 7. JPEG decoder top VHDL Interface

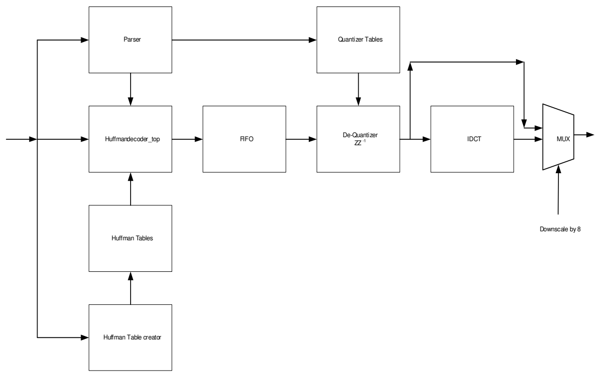


Fig. 6. Block Diagram of the VHDL blocks

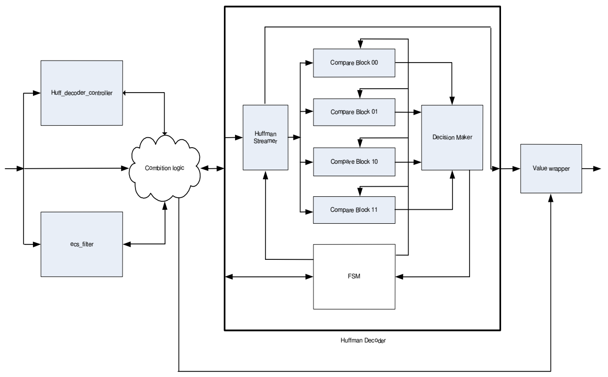


Fig. 7. Huffman decoder

Dequantization is one process where we lose information, this loss can be reduced by using other techniques [26, 27, 28, 29, 31], but these implementations are not the part of this project.

Dequantization and inverse-zigzag is done by one block. Inverse-zigzag was implemented by using simple lookup table. Dequantization was done by multiplying the value with the quantization table.

There are few efficient Huffman decoder architectures [24, 25, 30]. We have implemented our own architecture. Our Huffman decoder decodes the JPEG stream and generates the value in 11-bit. The input of the Huffman decoder is a 8-bit data, Huffman codes from Huffman memories block and image parameters from parser. The controller is inside the Huffman decoder top. Therefore, no extra controller is needed. Inside

Huffman decoder top there are 4 sub blocks Huffman_decoder_controller , ecs_filter , value_wrapper and huffmandecoder. Huffmandecoder has more blocks including huffmanstreamer, decisionmaker and compareblock as shown in Fig. 9.

FIFO stores the decoded codes from the Huffman decoder before it is dequantized and inverse zigzagged. JPEG decoder can also downscale image in size by the factor of 8 in both vertical and horizontal direction. Therefore in downscale by 8 mode the IDCT is bypassed. Bypassing IDCT increases the throughput of the decoder.

The 2D DCT/IDCT is based on the 1D fast DCT algorithm first described by Vetterli and Ligtenberg [5]. The input is 8x8 blocks of data in frequency domain and output is 8x8 block of data in time domain.

V. RESULTS/SYNTHESIS REPORT

Synthesis report of JPEG decoder is shown in Table IV and V.

TABLE I
SYNTHESIS REPORT (ALTERA)

Parameter	Value
Device	Altera cyclone II EP2C8T144C6
Synth/P& R Tool	Altera Quartus II v8.1
Synth/P& R settings	Timing constraint 100 MHz
Fmax	102.45
LEs	3996
Memory	15
Hard Multipliers	3
Clocks	1

TABLE II
SYNTHESIS REPORT (XILINX)

Parameter	Value
Device	Xilinx Spartan-3E xc3s500e-5-vq100
Synth/P&R Tool	Xilinx ISE v10.1
Synth/P&R settings	Time performance with physical synthesis
Fmax	101.8 MHz
Slices	3187
Memory	7 Block RAMs
Hard Multipliers	2
Clocks	1

VI. CONCLUSIONS

The goal of this project was to design an efficient JPEG decoder. The design was generic so it can be implemented on any FPGA. The project has four major modules: Parser, Huffman decoder, dequantizer/inv-zig-zag, IDCT. During the project it was noticed that the bottleneck for the throughput is IDCT. Therefore more efficient design of IDCT can increase the throughput. However in downscaling by "8" mode, IDCT is bypassed and throughput increases but in this case Huffman decoder or inverse zig-ziag block can be possible bottlenecks. Efficient design and pipelined implementation resulted in 100 MHz operating frequency and small size on silicon. The decoder can decode 6 mega pixel image in 200 msec to 600 msec depending upon image.

VII. ACKNOWLEDGEMENT

We are thankful to BitSim AB, Stockholm Sweden and Iqra University, Karachi Pakistan, for providing us a research environment and development facilities.

REFERENCES

[1] Digital Compression and coding of Continuous-Tone still images requirements and guidelines. T.81 (09/92), ITU-T.

[2] Design rules for Camera File system, DCF Version 2.0, 2003 Draft, Technical Standardization Committee on AV & IT storage systems and Equipments.

[3] JPEG File Interchange Format, Version 1.02, September 1, 1992, Eric Hamilton, C-Cube Microsystems.

[4] Exchangeable Image file format for digital still cameras, Exif Version 2.2, Technical Standardization Committee on AV & IT storage systems and Equipments.

[5] M. Vetterli and A. Ligtenberg, "A discrete Fourier-cosine transform chip", IEEE Journal on Selected Areas in Communications, Special Issue on VLSI in Telecommunications, Vol. 4, Nr. 1, pp. 49-61, 1986

[6] Studio encoding parameters of digital television for standard 4:3 and wide screen 16:9 aspect ratios. CCIR Rec.601 (BT.601-6 (01/07)) , ITU-T.

[7] GIF89a: Graphics Interchange Format Specification. Columbus, OH: CompuServe, Inc.

[8] C. W. Brown and B. J. Shepherd. Graphic File Formats; reference and guide. Manning Publications Co., 1995.

[9] T. Boutell and T. Lane. PNG (Portable Network Graphics) Specification, version 1.0. ftp://ftp.uu.net/graphics/png/documents/png-1.0-w3c.ps.gz.

[10] JPEG 2000 image coding system: Core coding system, ISO/IEC 15444-1:2004, JPEG Committee, 2004.

[11] ISO 12369. Tag image file format for image technology (TIFF).

[12] Shizhen Huang and Tianyi Zheng, "Hardware design for accelerating PNG decode" , IEEE International Conference on Electron Devices and Solid-State Circuits, December 8-10,Hong Kong, 2008.

[13] Zulkalnain Mohd Yusof, Zulfakar Aspar, Ishak Suleiman, "Field Programmable Gate Array (FPGA) Based Baseline JPEG Decoder" IEEE TENCON, 24-27 September 2000, Kuala Lumpur, Malaysia.

[14] R. P. Jacobi et al, "JPEG Decoding in an Electronic Voting Machine", 13th Symposium on Integrated Circuits and Systems Design, 18-24 September 2000, Manaus, Brazil.

[15] MARIO KOVAC AND N. RANGANATHAN, "JAGUAR: A Fully Pipelined VLSI Architecture for JPEG Image Compression Standard", Proceedings of the IEEE Volume 83, Issue 2, Feb. 1995 Page(s):247 – 258.

[16] Mohammed Elbadri et al, "HARDWARE SUPPORT OF JPEG" Canadian Conference on Electrical and Computer Engineering, 1-4 May 2005, Saskatoon, Sask.

[17] Gregory K. Wallace, "THE JPEG STILL PICTURE COMPRESSION STANDARD" , IEEE Transactions on Consumer Electronics, Vol. 38, No. 1, FEBRUARY 1992.

[18] En-hui Yang, Longji Wang, "Joint Optimization of Run-Length Coding, Huffman Coding, and Quantization Table With Complete Baseline JPEG Decoder Compatibility" IEEE TRANSACTIONS ON IMAGE PROCESSING, VOL. 18, NO. 1, JANUARY 2009.

[19] Kyeong-Yuk Min, Jong-Wha Chong, "A Real-Time JPEG Encoder for 1.3 Mega Pixel CMOS Image Sensor SoC", The 30th Annual Conference of the IEEE Industrial Electronics Society, November 2-6, 2004, Busan, Korea.

[20] Zulkalnain MohdYusof, et al, "Real Time Implementation Of Baseline JPEG Decoder Using Floating Point DSP TMS320C31" IEEE TENCON, 24-27 September 2000, Kuala Lumpur, Malaysia.

[21] Hiroki Mizosoe, Kazuhiro Maeda, Yuuichi Kubo, Yasutaka Tsuru, Kenichiro Kuroki, "An Advanced Multimedia Processing LSI Suitable For HDTV Applications", IEEE Transactions on Consumer Electronics, Vol. 47, No. 3, August 2001.

[22] SUNG-HSIEN SUN AND SHIE-JUE LEE, "A JPEG Chip for Image Compression and Decompression" Journal of VLSI Signal Processing 35, 43–60, 2003

[23] Markos E. Papadonikolakis, et al, "Efficient high-performance implementation of JPEG-LS encoder", Journal Real-Time Image Processing, 2008.

[24] S Lei, M Sun, "An Entropy Coding System for Digital HDTV Applications," IEEE TRANSACTION ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, no. 1, pp. 147-154, March, 1991

[25] Gopal Lakhani, "Modified JPEG Huffman Coding" IEEE TRANSACTIONS ON IMAGE PROCESSING, VOL. 12, NO. 2, FEBRUARY 2003

[26] Gopal Lakhani, "Adjustments for JPEG De-quantization Coefficients" Data Compression Conference, DCC 1998, March 30- April 1, 1998, Snowbird, Utah, USA. IEEE Computer Society, 1998.

[27] Giancarlo Calvagno, Gian Antonio Mian, Roberto Rinaldo, and Walter Trabucco, "Two-Dimensional Separable Filters for Optimal Reconstruction of JPEG-Coded Images" , IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, VOL. 11, NO. 7, JULY 2001.

[28] [28] A.A. Zaidee, Member IEEE, S. Fazdliana, Member IEEE, B.J. Adznan, "Content Access Control for JPEG Images using CRND Zigzag Scanning and QBP", The Sixth IEEE International Conference on Computer and Information Technology, September 20-22, 2006, Seoul, Korea.

[29] Hanli Wang, Sam Kwong, "Novel Variance Based Approach to Improving JPEG Decoding" ICIT 2005. IEEE International Conference on Industrial Technology, Hong Kong, 14-17 Dec. 2005.

[30] Shanq-Jang Ruan and Wei-Te Lin, "Bipartition Architecture for Low Power JPEG Huffman Decoder," The 12th Asia-Pacific Computer Systems Architecture Conference, Korea, pp. 235-243, 23-25 Aug. 2007.

[31] Chin-Chen Chang, Yung-Chen Chou, Jau-Ji Shen, "Improving Image Quality for JPEG CompressionF" 9th International Conference Knowledge-Based Intelligent Information and Engineering Systems, September 14-16, 2005, Melbourne, Australia.

[32] Markos Papadonikolakis, Vasilleios Pantazis and Athanasios P. Kakarountas "Efficient High-Performance ASIC Implementation of JPEG-LS Encoder", Journal of Real-Time Image Processing, Volume 3, Number 4 / December, 2008.

[33] Charles A. Poynton . Digital Video and HDTV: Algorithms and Interfaces. Morgan Kaufmann. ISBN 1558607927.

[34] N. Ahmed, T. Natarajan, and K. R. Rao, "Discrete Cosine Transform", IEEE Trans. Computers, 90-93, Jan 19.