International Journal of Engineering Science Invention
ISSN (Online): 2319 – 6734, ISSN (Print): 2319 – 6726
www.ijesi.org Volume 3 Issue 8 ǁ August 2014 ǁ PP.47-53

Matlab Implementation of Baseline JPEG Image Compression Using Hardware Optimized Discrete Cosine Transform

T. Vimal Prakash Singh
Department of Electrical Engineering, NERIST, Nirjuli, India

ABSTRACT : *In this paper, modeling of optimized Discrete Cosine Transform (DCT) with reduced hardware and implementation of JPEG Image compression in MATLAB is presented. DCT is the heart of the JPEG image compression. The computation of DCT is hardware intensive and power consumption is also very high. In JPEG Image Compression pipeline, a quantizer follows the DCT. Such structural pipeline is advantageous for reduced complexity in the entire JPEG compression/encoding. We compute DCT with no multipliers without any scaling operation and this scaling operation is absorbed in the JPEG compression flow itself. Performance of our implementation is also presented.*

KEYWORDS : *Discrete Cosine Transform, Joint Photographic Expert Group(JPEG), Optimization, MATLAB, Complexity, Compression, Baseline.*

I. INTRODUCTION

Wireless Image and video compression systems powered by battery need to be energy efficient in order to increase the operational life time. Data compression in these systems is required in order to reduce data transmission volume over the high power consumption wireless channel. Discrete Cosine Transforms is the main technique that are used in many image and video codecs such as JPEG,MPEG and H.264. Although DCT is computation intensive and requires lot of power , it is widely used because of its energy compaction property. So many people have proposed different implementation schemes with low complexity[1],[2].[3],[4],[5].It was proven that the theoretical lower bound on the number of multiplication operations in a DCT is 11 [6], [7]. Researchers came up with different implementation schemes and architectures with no multipliers which are hardware expensive. Some of the multiplier-less architectures are co-ordinate rotation digital computer (CORDIC) and binDCT [8],[9],[10]. In baseline JPEG compression pipeline, a quantizer follows the DCT . The DCT output is divided by quantizer constants that are stored in the quantization table. In such compression structures some of the arithmetic operations are absorbed in the quantization step [9]. CORDIC based DCT architectures require compensation that can not be absorbed in the codec quantizer.

This paper is organized as follows. In section II, overview of JPEG image compression is given and the following section III describes JPEG file format . In section IV, background of the DCT is given. In section V, Loeffler algorithm for computation of DCT is explained. Section VI describes the modeling of reduced hardware overhead Loeffler DCT algorithm in MATLAB is explained. In section VII implementation of JPEG compression using the optimized DCT is explained and performance of the JPEG compression using optimized DCT computation is given. In section VII , conclusion of our JPEG image compression is given.

II. JPEG IMAGE COMPRESSION OVERVIEW

The JPEG image compression standard developed by Joint Photographic Expert Group is a sophisticated lossy/lossless compression method for color or gray scale still images. It also works best on continuous tone images, where adjacent pixels have similar colors [11]. In JPEG image compression it is allowed to adjust the amount of data lost (and thus also the compression ratio) over a very wide range. JPEG unlike other formats like PPM (Predict, Probability Model), GIF (Graphics Interchange Format) and PGM, is a lossy compression technique, this means some visual information is lost . The baseline JPEG encoder/compressor structure diagram is shown in Fig. 1.

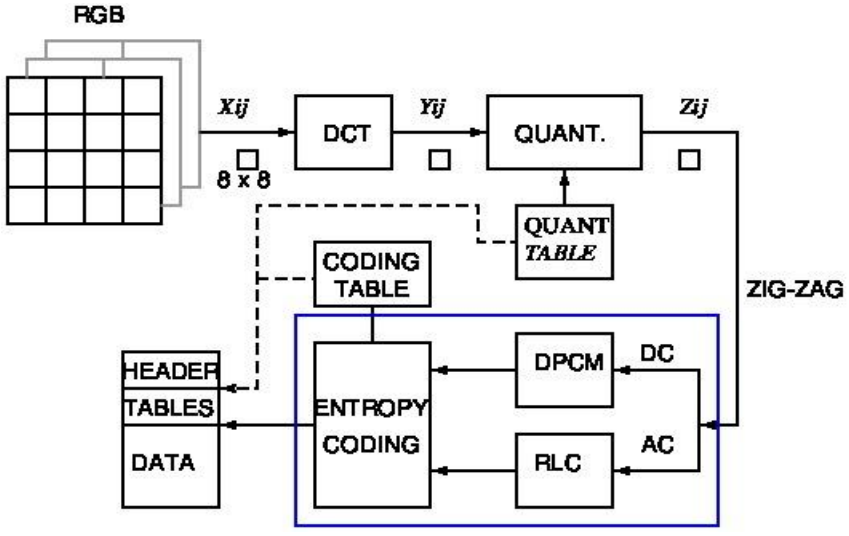

Figure 1: JPEG Image Compression Structure

An image is divided into non overlapping 8x8 blocks and DCT is applied on each of these 8x8 image data. Spatial variation of pixels in these blocks are very small. Those pixel values which are insensitive to human eye are removed with the help of a quantization step in the compression flow. And the quantization step size is determined by the acceptable visual quality of the compressed image. After quantization each of the 64 DCT outputs are arranged in increasing order of frequency in zig-zag pattern. The first component in these 64 data values is known as DC component and the remaining values are called AC component. The difference of DC coefficient between two adjacent blocks are coded with differential coding and the rest AC coefficients are coded with entropy coding with information of number of zeroes between two non-zero AC coefficients.[12]

III. JPEG FILE FORMAT

JPEG team did not specify any format during its development. It does not specify what an application needs to do in order to create an image while exchanging among different applications. Eric Hamilton filled the void left by the JPEG development team by creating JPEG File Interchange Format (JFIF). The JFIF standard defines the following:

- A signature for identifying JPEG files.
- Colorspace
- Pixel density
- Thumbnails
- Pixel relationship to sampling frequency

General layout of a JFIF file is shown in the Fig. 2 [13]. JFIF file specifies that the YCrCb colorspace be used. Therefore we need to convert image RGB to YCrCb as given in the equation (1). Markers are used to break a JPEG stream into its structures. These markers are of 2 byte length where the second byte is the code that specifies the marker type. The JPEG file must begin with SOI marker and end with EOI marker. The compressed component data does not occur within a marker. Compressed data immediately follows a SOS marker.

$$\begin{aligned}
Y &= 0.299R + 0.587G + 0.114B \\
Cb &= -0.1687R - 0.3313G + 0.5B + 2^{(SamplePrecision)/2} \\
Cr &= 0.5R - 0.4187G - 0.0813B + 2^{(SamplePrecision)/2}
\end{aligned} \tag{1}$$

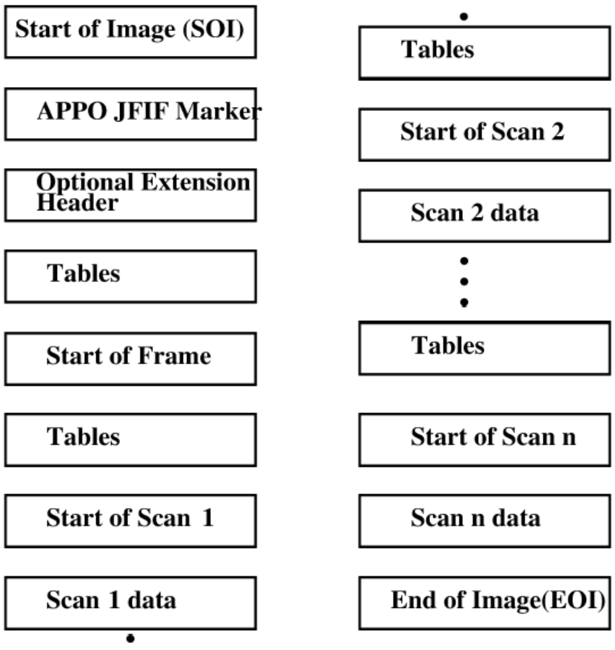

Figure 2: JFIF File Structure

IV. DCT BACKGROUND

The 1-D DCT of an 8 point image sampled data s(i) is given by equation (2) and for 8×8 its 2-D DCT is expressed as where $c[m]=1/\sqrt{2}$ for m=0 and $c[m]=1$ for m not equal to 0. The computation of 2-D DCT using equation (4) requires 4096 multiplications. To reduce these multiplication operations , it can be used the principle of separability of the DCT transform. The computation of 2-D DCT with intermediate transformation is shown in the Fig. 5 , and it can be computed with 1024 multiplications.

$$S[m]=\frac{1}{2}c[m]\sum_{i=1}^{8} s(i)\cos\left(\frac{(2i+1)m\pi}{16}\right) \tag{2}$$

$$c[m]=\left\{\begin{matrix}\frac{1}{\sqrt{2}} & for\ m=0 \\ 1 & for\ m\neq 0\end{matrix}\right\} \tag{3}$$

$$S[m,n]=\frac{1}{4}c[m]c[n]\sum_{i=1}^{8}\sum_{j=1}^{8} x(i,j)\cos\left(\frac{(2i+1)m\pi}{16}\right)\cos\left(\frac{(2j+1)m\pi}{16}\right) \tag{4}$$

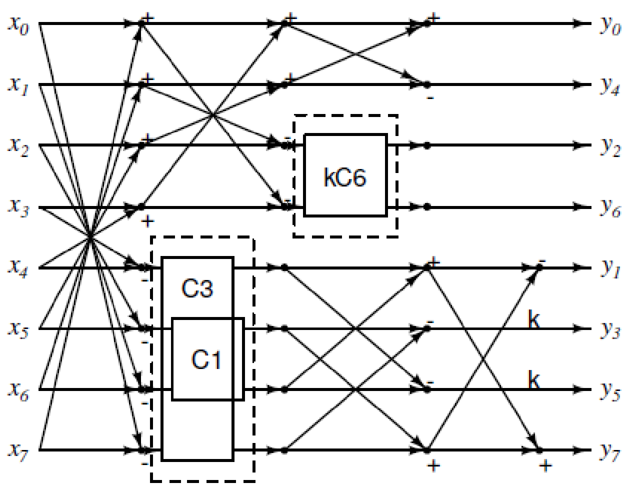

Figure 3: Signal Flow Graph

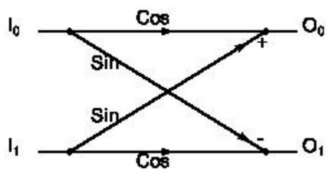

Figure 4: Butterfly Graph

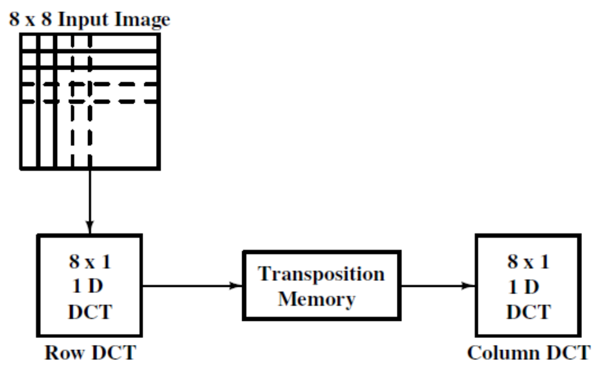

Figure 5: 2-D DCT computation

IV. LOEFFLER DCT

Loeffler proposed a scheme to compute 1-D DCT which requires only 11 multiplications. Its signal flow diagram is shown in figure 2. In the figure x_i are the inputs and y_i are the outputs. In the computation unit kCn , $k=\sqrt{2}, n=1,2,6$ and its mathematical equation is shown in equation (5). Computation units C1 and C3 are same with k=1. The equation (5) can be rewritten as equation (6) with appropriate substitution. Computation of equation (6) requires 4 multipliers and 2 adder. It can be rearranged as in equation (7) and it is seen that computation of this butterfly equation can be done with 3 adder and 3 multipliers.

It is also to be noted for a 1-D DCT the output y_i requires to be multiplied by . And 2-D Discrete Cosine Transform can computed using 1D DCT with intermediate transposition as shown in the Fig. 5. For 2D DCT computation, this scaling factor becomes 1/8.

$$\begin{aligned} y_2 &= x_2 . k . \cos\left(\frac{n.\pi}{16}\right) + x_3 . k . \sin\left(\frac{n.\pi}{16}\right) \\ y_6 &= - x_2 . k . \sin\left(\frac{n.\pi}{16}\right) + x_3 . k . \cos\left(\frac{n.\pi}{16}\right) \end{aligned} \tag{5}$$

$$\begin{aligned} y_2 &= A . x_2 + B . x_3 \\ y_6 &= A . x_3 - B . x_2 \end{aligned} \tag{6}$$

$$\begin{aligned} y_2 &= (B - A) . x_3 + A . (x_2 + x_3) \\ y_6 &= A (x_2 + x_3) - (A + B) x_2 \end{aligned} \tag{7}$$

Table 1: Binary Coefficient Representation in CSD

Constants	Fractional Equivalent	CSD equivalent
$\cos\left(\frac{6\pi}{16}\right)$	0.38268	$0.10\underline{1}00010\underline{1}$
$\sin\left(\frac{6\pi}{16}\right)$	0.92388	$1.000\underline{1}0\underline{1}001$
$\cos\left(\frac{3\pi}{16}\right)$	0.83147	$1.0\underline{1}0101001$
$\sin\left(\frac{3\pi}{16}\right)$	0.55557	$0.100100\underline{1}00$
$\cos\left(\frac{\pi}{16}\right)$	0.98079	$1.00000\underline{1}0\underline{1}0$
$\sin\left(\frac{\pi}{16}\right)$	0.19509	$0.010\underline{1}0010\underline{1}$

V. OPTIMIZED LOEFFLER DCT

Our MATLAB model of the DCT is based on the Loeffler DCT algorithm and we target a reduced power by reducing or removing the number of multiplier which consumes lots of power. By removing these multipliers, the complexity is also reduced. There were 11 multipliers and 29 additions Loeffler DCT architecture and there are three butterfly operations. The signal Graph of these butterfly operations are shown in Fig. 4. The angles for these butterfly operations are 67.5^0 , 33.75^0 and 11.25^0. In the first butterfly operation,C_6, we notice that there is a constant term k. To get the actual coefficient, we need to multiply the output of C_6 by the factor k. Then multiplication operation increases. So we take this out of the butterfly operation and combine with the scaling factor that need to be multiplied after the DCT output for the corresponding output. We also notice that in these butterfly equations, there are sine and cosine of angles. In our model these sine and cosine of angles are represented in Canonical Sign Digit(CSD) format and the multiplication is reduced to few additions with appropriate shifting operations. CSD equivalents of these butterfly angle constants are shown in Table 1. The computation of y_2 and y_6 is shown in equation (8) with appropriate variables. These representation of butterfly angles and addition and shifting operations are modeled appropriately in a MATLAB script function. It is mentioned here that the scaling operation in not included in the modeled DCT script function as appropriate scaling factors are absorbed in the next stage of the JPEG compression pipeline.

$$x_0 = \frac{69}{128} d_1$$
$$y_0 = \frac{48}{128} \left(c_1 + d_1\right) \quad (8)$$

Table 2: Comparison between different 2-D DCT

Algorithm	Multiplications(1-D)	Add/Subs(1-D)
DCT-SQ(15)	13	29
Chen DCT(1)	13	29
Wang DCT(16)	13	29
Loeffler DCT(6)	11	29
Proposed 1-D DCT model(MATLAB)	0	33

VI. JPEG COMPRESSION IMPLEMENTATION

Image pixels were obtained using MATLAB. As explained in the baseline JPEG image compression steps, this image was divided into non overlapping 8×8 image blocks. On these 8×8 image blocks DCT modeled using CSD was applied. It can be noted that in our MATLAB model of the DCT , its output was not multiplied . DCT outputs are divided constants in a quantization table and values of this table are chosen depending upon the visual perception and amount of information that can be compromised. It is again to be noted that this quantization values modified appropriately in our implementation in accordance with scaling factor of our DCT model as given by (7). JPEG compression results with optimized DCT model is shown in Fig. 6.

VII. CONCLUSION

In this paper, implementation of JPEG image compression in MATLAB using hardware optimized DCT model was presented. In our JPEG implementation, DCT was modeled and implemented with CSD representation of the butterfly operations. Since CSD butterfly requires no multiplications, it was implemented with with addition and shifting operations. The non uniform scaling factor of the DCT transform was also absorbed in the pipelines stages of the JPEG compression implementation. PSNR performance of our implementation in MATLAB was also presented. We also analyzed performance deviations with our CSD based DCT model JPEG compression and DCT function available in MATLAB.

$$DCTQ = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 36 & 55 & 64 & 81 & 194 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix} \quad (9)$$

Table 3: MATLAB Performance Comparison

Image	Pepper	Tulips	Monarch	Baboon	Fruits	Fruits
PSNR(dB)	11.8824	13.8240	13.0317	9.7251	23.5872	14.0194

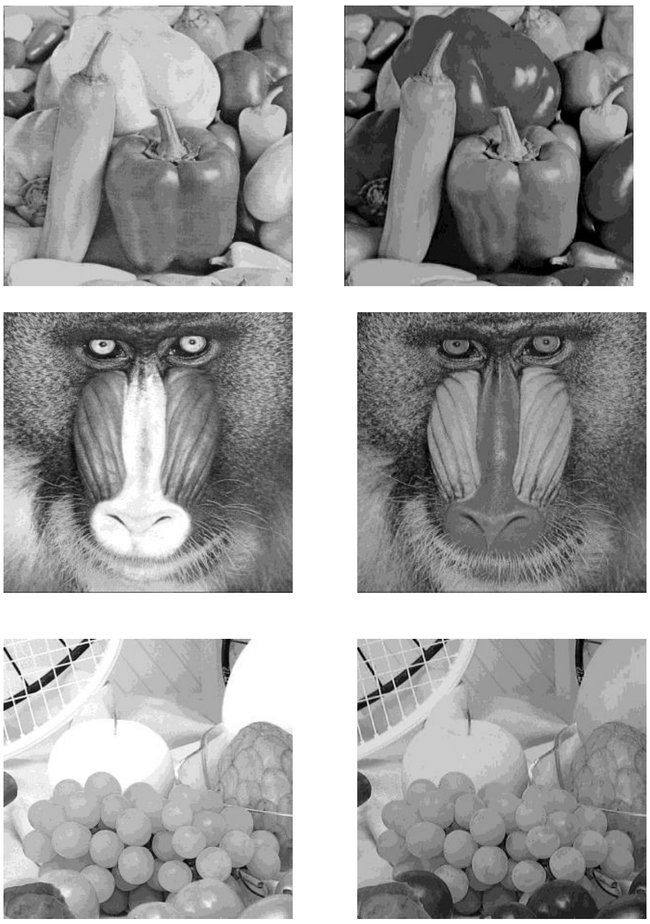

Figure 6: JPEG Compression

REFERENCES

[1] S. C. F. S. Chen, W. H., A fast computational algorithm for the discrete cosine transform, IEEE Trans. Communication, vol. 9, no. 25, pp. 1004–1009, 1977.

[2] J. Li and S. Lu, Low power design of two-dimensional DCT, IEEE Conf. on ASIC and Exhibit, 1996.

[3] J. August, N. and D. Ha, Low power design of DCT and IDCT for low bit rate video codecs, IEEE Trans. Multimedia, vol. 6, no. 3, pp.414–422, 2004.

[4] A.P.C.Thuyidies Xanthospouos, A low power DCT core using adaptive bitwidth and arithmetic activity exploiting signal correlation and quantization, IEEE JSSC, vol. 3, no. 35, pp. 740–750, 2000.

[5] S.R.K. Park, J. Kwon, Low power reconfigurable DCT design based on sharing multiplication,IEEE Int. Conf. on Acoustics, Speech, and signal Processing, vol. 3, pp. 3116–3119, 2002.

[6] I. A. M. G. S. Loeffler, C, Practical fast 1-D DCT algorithms with 11 multiplications, Proc. ICASSP, Glasgow, UK, vol. 2, pp. 988–991,1989.

[7] E. Feig and Winogard, On the multiplicative complexity of discrete cosine transform," IEEE Trans. Inform. Theory, vol. 38, July,1992.

[8] J. E. Volder, The CORDIC trigonometric computing technique, IRE Trans. Electron Comput., vol. 8, pp. 330–334, 1959.

[9] C. C. Sun, S.-J. Ruan, B. Heyne, and J. Goetze, "Low-power and high-quality Cordic-based Loeffler DCT for signal processing," Circuits, Devices Systems, IET, vol. 1, no. 6, pp. 453 –461, december 2007.

[10] T. D. Tran, "The binDCT: fast multiplierless approximation of the DCT, IEEE Signal Processing Lett., vol. 7, pp. 141–144, 2000.

[11] R. C. Gonzalez, R. E. Woods, Digital Image Processing (2nd.Ed.,Prentice Hall, 2002).

[12] W. B. pennebaker and J. L. Mitchel, JPEG Still Image data compression standard (New York: Van Nostrand Reinhold, 1993).

[13] T. A. Y. Arai and M. Nakajima, A fast DCT-SQ scheme for images, Trans. Of Institute of Electronics , Information and Communication Engineers, vol. E71, 1988.

[14] John Miano, Compressed Image File Formats, JPEG, PNG, GIF, XBM, BMP (Addison Wesley Longman, Inc ,1999)

[15] Y Arai,T Agui and M. Nakajima, A fast DCT-SQ Scheme for Images, Transactions of Institute of Electroninics, Information and Communication Engineers, vol E71, no. 11, pp. 1095-1097,1988.

[16] Z. Wang, Fast algorithms for the discrete W transform and for the discrete Fourier transform, IEEE Trans. Acoust, Speech, Signal Processing, vol. ASSP-32, pp. 803--816, Aug. 1984.