Hardware Implementation of Low Power, High Speed DCT/IDCT Based Digital Image Watermarking

Rajesh Kannan Megalingam, Venkat Krishnan B., Vineeth Sarma V., Mithun M., Rahul Srikumar
Department of Electronics and Communication
Amrita School of Engineering, Amrita Vishwa Vidyapeetham
Kollam, Kerala, India.
rajeshm@amritapuri.amrita.edu, venkatkrishnan.b@gmail.com, vineethisalways4u@gmail.com, mithun34@ieee.org, rahul44@ieee.org

Abstract— **This paper presents a comparison with the conventional watermarking technique and the novel 5-stage pipelined implementation of DCT/IDCT which is used in digital image watermarking. The most common method of Discrete Cosine Transform (DCT)-based digital image watermarking which is used for image authentication and copyright protection is the transpose method. In this method the 2-Dimensional DCT is obtained by taking two 1-dimensional DCTs in series. The image pixel value is first divided into 8x8 blocks and the row-wise 1D DCT of each block is taken. The transpose of the blocks is then determined and a column-wise 1D DCT is ascertained which gives the 2D DCT of the data. The major advantage of this design is that, unlike the conventional DCT-based watermarking technique, this method uses a 5-stage pipeline which can bring about a speed increase of close to 500% over the conventional method which is naturally a great advantage. This technique has been tested on the standard 'Lena' image. Both visible and invisible watermarking is implemented in hardware. The design is done in Verilog HDL and the simulation is done in Modelsim 6.3b. Matlab is used to produce the binary data file which is the input to the 1D DCT module. The hardware implementation is done in Xilinx XC3S4000 FPGA. The results of the comparison are discussed in the concluding sections.**

Keywords- DCT/IDCT, pipeline, watermarking, low power.

I. INTRODUCTION

Digital watermarking is about embedding digital information into another information/data, this process can be reversible or irreversible. The information can be audio, video or images. Digital watermarking can be visible or invisible and the visibility of the watermark can be varied according to the wish of the owner. Major applications of watermarking include copyright protection and steganography. In steganography people communicate secretly by embedding their information in digital signals. This technique has been used for decades, in one form or the other. Watermarking can be done using software or hardware. The efficiency of the watermark is determined by the capacity, robustness and perceptibility of the watermark, which are the most important qualities in watermarking. Based on these criteria, 'hardware watermarking' is found to be more efficient than 'software watermarking'. The watermarking that we have done in these experiments is purely image watermarking, both visible and invisible, and this finds many potent applications in copyright authentication. The security of the watermarking can be improved by including an embedded lock, which prevents anyone who does not have the digital key from accessing the information. Any methods in frequency domain can be used in watermarking, including those based on Discrete Cosine Transform (DCT), Discrete Wavelet Transform (DWT), Discrete Hartley Transform (DHT), etc. The discrete cosine transform (DCT) and the inverse discrete cosine transform (IDCT) are substantial performance bottlenecks in image watermarking since the time taken to watermark the image depends on the time required to get the DCT/IDCT of the images. We have done the DCT implementation by the transpose method (the most common method adopted for low power consumption), and we have also devised a method to accelerate the watermarking by bringing in a 5-stage pipeline which can give a maximum speed increase of 5 times the normal implementation.

II. TWO DIMENSIONAL DCT

A DCT expresses a finite number of data points in terms of the sum of cosine functions oscillating at different frequencies. The method that we have used for digital image watermarking is by DCT-based, where the 2D DCT of the images are determined and added to watermark an image on another image. This requires that, the 2-dimensional DCT of the images for the watermarking be performed. 2-D DCT is represented by the following equation (1) where n1 and n2 vary from 0 to 7 for an 8x8 block of data. The value of the constants k1 and k2 also vary from 0 to 7 [4].

$$X(k1,k2)=\sum_{n1=0}^{N1=1}\sum_{n2=0}^{N2=1} x(n1,n2)\cos[\pi/N1(n1+0.5)k1]\cos[\pi/N2(n2+0.5)k2] \qquad (1)$$

There are many methods for finding the 2D DCT of which the transpose method is the most common method.

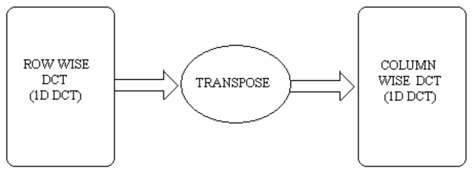

Figure.1. 2D-DCT using 1D-DCT

The digital data of the image is made into several 8x8 blocks and the 1D DCT of each block is determined. Then the transpose of the 8x8 block is taken and again the 1D DCT is determined, which, when done to all the blocks of the image gives the 2D DCT of the entire image. In other words, the row-wise block DCT is first ascertained followed by the column-wise DCT; these then combine to give the 2D DCT. Suppose that the image under consideration is of size 512x512. When divided into 8x8 blocks, the image gives 4096 blocks. Then the block-wise DCT is determined for each block. The "Fig.1" above illustrates the method of finding the 2D DCT of an 8x8 blocks of data. The same procedure is repeated for all the 8x8 blocks of data.

From the above explanation it is quite evident that the implementation of digital watermarking needs an efficient algorithm to find the 2D DCT/IDCT. This requires the implementation of 1D DCT. The implementation of 1D DCT is therefore discussed in the next section.

III. 1D DCT IMPLEMENTATION

The 1-D DCT of a sequence of length N is given by [7]

$$C(u) = \alpha(u) \sum_{x=0}^{N-1} f(x)\cos[\pi(2x+1)u/2N] \qquad (2)$$

For u=0, 1, 2, 3….. (N-1) and in the similar way the 1-D IDCT is defined as

$$f(x) = \alpha(u) \sum_{u=0}^{N-1} \alpha(u)\, C(u) \cos[\pi(2x+1)u/2N] \qquad (3)$$

For x=0, 1, 2, 3…. (N-1) and for both the equations

$$\alpha(u) = \begin{array}{ll} (1/N)^{0.5} & \text{for } u=0 \\ (2/N)^{0.5} & \text{for } u\neq 0 \end{array} \qquad (4)$$

Here we consider an 8x8 block of data. For this we will have to find the 1D-DCT of each row of 8 elements and each column of 8 elements separately. The major concern in finding the 1D DCT/IDCT is the number of multipliers and the adders that have to be used. In finding the 1D DCT of

$$\begin{bmatrix} y0\\y1\\y2\\y3\\y4\\y5\\y6\\y7 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} c4 & c4 & c4 & c4 & c4 & c4 & c4 & c4 \\ c1 & c3 & c5 & c7 & -c7 & -c5 & -c3 & -c1 \\ c2 & c4 & c6 & -c2 & -c2 & -c6 & c6 & c2 \\ c3 & -c7 & -c1 & -c5 & -c5 & c1 & c7 & -c3 \\ c5 & -c1 & c7 & c3 & -c3 & -c7 & c1 & -c5 \\ c6 & -c2 & c2 & -c6 & -c6 & c2 & -c2 & c6 \\ c7 & -c5 & c3 & -c1 & c1 & -c3 & c5 & -c7 \end{bmatrix} \begin{bmatrix} x0\\x1\\x2\\x3\\x4\\x5\\x6\\x7 \end{bmatrix}$$

Figure 2. Matrix method of implementing 1D-DCT/IDCT

$$\begin{bmatrix} y0\\y1\\y2\\y3\\y4\\y5\\y6\\y7 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} c4 & c4 & c4 & c4 & 0 & 0 & 0 & 0 \\ c2 & c6 & -c6 & -c2 & 0 & 0 & 0 & 0 \\ c4 & -c4 & -c4 & c4 & 0 & 0 & 0 & 0 \\ c6 & -c2 & c2 & -c6 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & c1 & c3 & c5 & c7 \\ 0 & 0 & 0 & 0 & c3 & -c7 & -c1 & -c5 \\ 0 & 0 & 0 & 0 & c5 & -c1 & c7 & c3 \\ 0 & 0 & 0 & 0 & c7 & -c5 & c3 & -c1 \end{bmatrix} \begin{bmatrix} x0+x7\\x1+x6\\x2+x5\\x3+x4\\x0-x7\\x1-x6\\x2-x5\\x3-x4 \end{bmatrix}$$

Figure 3. Matrix method which requires reduced number of multipliers and adders.

the 8 elements we have applied a similar method to that suggested in [1]. The 1D DCT is given by "Fig.2", where x(0) to x(7) represent the 8 data elements and Cx= cos(πx/16). In this implementation it needs the use of 64 multipliers, 56 adders, and shifters that perform arithmetic right shifts to divide by 2. This matrix can be further simplified to reduce the number of multipliers and adders [1]. This reduces the number of multipliers to 32 from 64 and adders to 24 from 56. This substantially reduces the power consumption. We have used a similar implementation of DCT/IDCT in watermarking, which includes a novel 5 stage pipelining method. We have compared the speed of both pipelined and non-pipelined implementations of DCT/IDCT have been done to bring about a comparison and have thereby been able to estimate the speed increase that can be achieved by the pipelined implementations of DCT/IDCT.

IV. FIVE STAGE PIPELINED 1D DCT IMPLEMENTATION

The only difference in the pipelined implementation is that there are registers being inserted in between each stage; the speed increase that can be obtained is nearly 5 times that of the non-pipelined method. For the pipelined implementation we divided the DCT/IDCT operation into 5 stages:

- Fetching of data from the memory to the local registers. This stage is called as 'the FETCH stage' and is represented by F.
- Adding or subtracting the data Eg: x(0) + x(7). This stage is called as 'ADDSUB stage' which is represented by AS.

1	2	3	4	5	6	7	8	9	10	11
F1	AS1	M1	A1	D1				CLOCK CYCLES		
	F2	AS2	M2	A2	D2					
		F3	AS3	M3	A3	D3				
			F4	AS4	M4	A4	D4			
				F5	AS5	M5	A5	D5		
					F6	AS6	M6	A6	D6	
						F7	AS7	M7	A7	D7

Figure. 4. Represents the space-time diagram of pipelined implementation

- Multiplication with the cosine values Eg: C_4 * [x(0) + x(7)], This stage is called 'the MULTIPLY stage' and is represented by M.
- Addition to get the output of matrix multiplication. This stage is known as 'the ADD stage' and this is represented by A.
- Divide by 2, which is obtained by signed right shift (or the arithmetic right shift) of the data elements. This stage is called 'the DIVIDE stage' and is represented by D.

Pipelining can be implemented by placing one register in between each stage. This is the only additional hardware required. In this method, first set of 8 data are fetched in the first clock cycle. In the second clock cycle when the data are being added and subtracted the second set of 8 data elements are fetched into the registers. In the third clock cycle the first set of 8 elements are multiplied when the second set is added and subtracted and at the same time the third set of data elements is fetched and so on. This procedure gives approximately one set of '8 elements output' per clock cycle. The pipelined implementation is as shown in the "Fig.4". It should be noted that the result of each stage has been registered.

V. Five Stage Pipelined 1D IDCT Implementation

The IDCT can also be implemented in the matrix method given in "Fig.2". This requires a modification in the matrix to be implemented which can be represented as in "Fig.6".

The output when IDCT is implemented using the matrix in "Fig.6" are not x0,x1,etc., but their linear combinations such as x0+x7, x1-x6, etc., as shown in "Fig.6". x0 to x7 can be determined from the linear equations that are obtained after the inverse operation is performed as shown in the matrix in "Fig.6".

For example if x0+x7= o1 and x0-x7=o2, then x0 is given by (o1+o2)/2 and x7 is given by (o1-o2)/2. In a Similar way all the x values can be found out. Here, in the case of IDCT also we implement a five stage pipeline. The various stages in the pipeline are as follows:

- Fetching the data from the memory. This stage is called 'the FETCH stage' and is represented by F

$$\begin{bmatrix} x0+x7 \\ x1+x6 \\ x2+x5 \\ x3+x4 \\ x0-x7 \\ x1-x6 \\ x2-x5 \\ x3-x4 \end{bmatrix} = 2\text{ X } \begin{bmatrix} y0 \\ y1 \\ y2 \\ y3 \\ y4 \\ y5 \\ y6 \\ y7 \end{bmatrix} \begin{bmatrix} c4 & c4 & c4 & c4 & 0 & 0 & 0 & 0 \\ c2 & c6 & -c6 & -c2 & 0 & 0 & 0 & 0 \\ c4 & -c4 & -c4 & c4 & 0 & 0 & 0 & 0 \\ c6 & -c2 & c2 & -c6 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & c1 & c3 & c5 & c7 \\ 0 & 0 & 0 & 0 & c3 & -c7 & -c1 & -c5 \\ 0 & 0 & 0 & 0 & c5 & -c1 & c7 & c3 \\ 0 & 0 & 0 & 0 & c7 & -c5 & c3 & -c1 \end{bmatrix}^{-1}$$

Figure. 5. Matrix representing IDCT

- Multiplication with the cosine values. Eg: C_4' * [y0]. This stage is called 'the MULTIPLY stage' and is represented by M. C_4' represents the inverse cosine value at x=4.
- Addition to get the output of matrix multiplication. This stage is represented by A.
- Multiply by 2 obtained by signed left shift (or the arithmetic left shift) of the data. This stage is represented by M'.
- Finding x0, x1, x2, etc from the result obtained. This stage is represented by P.

VI. Digital Image Watermarking Using DCT/IDCT

Watermarking of an image can be done using the DCTs of the images to be used. The DCT of both the images are taken and the intensity of the image that will appear as the watermark can be varied by the proportion in which the DCT is added. The basic process that is taking place is the addition of DCTs followed by the IDCT of the result which gives the watermarked image.

$$V_i' = V_i * (1 + \alpha * (X_i + \beta * W_i)) \qquad (5)$$

V_i' is the result of the added DCT of the two images. X_i is the DCT value of the image on which the watermarking is done and Wi is the DCT of the logo which is watermarked on the image. The constants α and β, decides the visibility of the watermark. If the value of β is very small we obtain an invisible watermark and as its value increases the visibility simultaneously increases. For the extraction of a watermark the value of α and β has to be known and hence the extraction of the watermarking cannot be done by anyone who does not know those values. To increase the security there are also methods wherein the values of α and β are varied for each block, and these values are known only to the owner. This principle used in image authentication and copyright protection.

VII. Hardware Implementation

The hardware implementation of DCT requires the cosine values as mentioned above in section III and the image pixel values. The pixel values of the images and cosine values are determined using Matlab and made into binary files. These two files are read into the HDL code and the data is

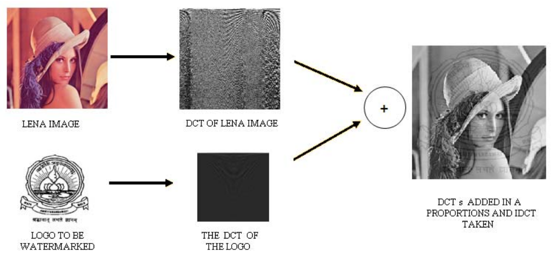

Figure.6 DCT/IDCT based digital watermarking

manipulated. Two memory modules are created to store the cosine values and the image pixel values and at each clock cycle the data is fetched from the memory into the registers. (This occurs in every 5 clock cycles in the case of non-pipelined implementation.) After the on-dimensional row-wise DCT is determined, the transpose of that result is taken by memory mapping. The existing value in the result memory is moved into the corresponding positions that it should take in the transposed form. The value from this memory block serves as the input to the next 1D DCT and the result of the 2D DCT is stored in the memory block. Next the 2D DCT of the second image is taken in a similar way (as mentioned in the section IV and V) and the data is added up in varying proportions. The resulting data is given to the IDCT block to obtain the pixel values of the watermarked image. IDCT can be implemented by the same procedure of DCT in reverse. The resulting values after adding two DCTs are the inputs to the IDCT module. The first step is to multiply those values by two. The resulting value is then multiplied by the inverse cosine matrix. Now the 1D-IDCT is obtained, and the transpose of the resulting values are taken. The ID-IDCT is then taken, which gives a set of 8 values from which the pixel values can be found. The value is written into a binary file which is converted into an image in Matlab. The flowchart in "Fig.7"explains the detailed hardware implementation.

VIII. RESULT AND DISCUSSION

The digital watermarking of different images is done using both pipelined implementation of DCT/IDCT and non-pipelined implementation. The original images used are given in "Fig. 8". The watermarked images for varying values of α and β are as given in "Fig. 9".

IX. PERFORMANCE ANALYSIS

The hardware implementation of both pipelined and non-pipelined 2D DCT/IDCT was done. The higher performance was obtained in the case of pipelined implementation of DCT. We already saw that there are five operations that are performed. Suppose the clock frequency is 1MHz, the clock time period is 1μs. In non pipelined implementation, each set of output is obtained in five clock cycles which means that the time taken is 5μs. Let this be represented as $t_{n.}$ In pipelined implementation each stage is registered and hence we get one set of 8 outputs in each clock cycle, which takes a time of 1μs, which is represented by $t_{p.}$ The increase in speed of the pipelined implementation can be found out using the formula [13].

$$S= (n*t_n)/ (k+n-1)*t_p \tag{6}$$

k is the number of stages in the pipeline and n is the number of tasks to be completed. Here the parent image under consideration is Lena. The number of tasks or the number of

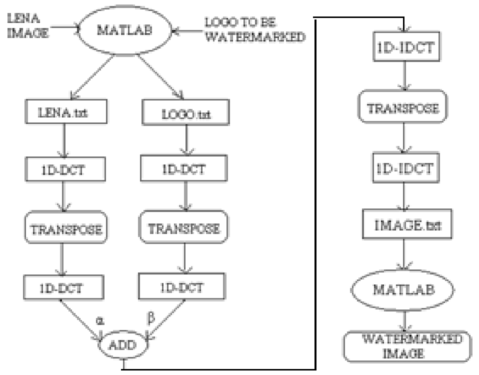

Figure.7 the various steps involved in digital watermarking

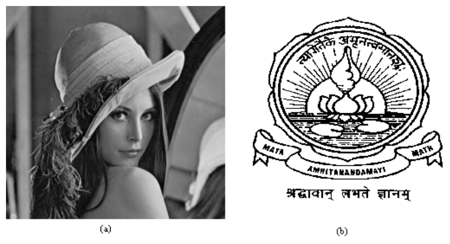

Figure 8. (a) Lena image (b) logo to be watermarked

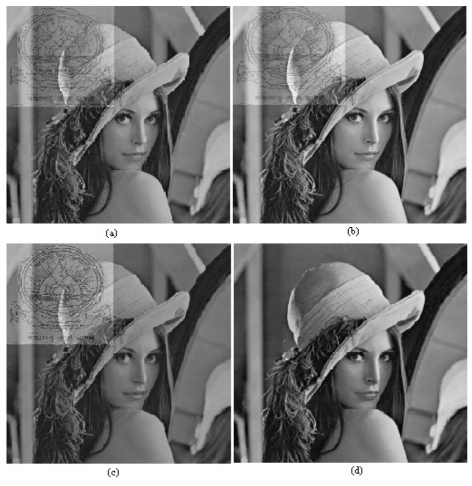

Figure 9. The images obtained after watermarking (a) for α=.97 and β=.07 (b) for α=.97 and β=.17 (c) for α=.97 and β=.27 (d) invisible watermarking

computations is purely dependant on the size of the images and the Table I (on the following page) gives the comparison of the speed-increase that can be achieved by using different images of different sizes. For example let us take the Lena image. It is of the size 512x512, and hence here the number of task is 32768 for 1D-DCT, while the size of the logo is 239x246. The speed increase calculated is as given in the Table I. As the number of tasks increases the speed-increase of the method also increases. The maximum speed increase obtainable is equal to the number of stages which is five in this case. The data given below has been obtained from the simulation carried out in Modelsim 6.3b.

TABLE I
THE COMPARISON OF THE SPEED FOR PIPELINED AND NON-PIPELINED IMPLEMENTATION

Image	Lena	Logo
Clock frequency	1 MHz	1 MHz
Image size	512X512	239X246
Number of data sets	32768	7350
Simulation time for non-pipelined DCT	163.84 µs	36.75µs
Simulation time for pipelined DCT	32.772µs	7.354µs
Speed up	4.9993	4.9972

TABLE II
COMPARISON OF PIPELINED AND NON-PIPELINED IMPLEMENTATION FOR FIRST 1D DCT

Parameter	Non-pipelined	Pipelined
Operating frequency	82.65 MHz	83.996 MHz
Registers	2303	2228
Number of LUTs	2159	2094
Number of slices	2132	2059

TABLE III
COMPARISON OF PIPELINED AND NON-PIPELINED IMPLEMENTATION FOR SECOND 1D DCT

Parameter	Non-pipelined	Pipelined
Operating frequency	55.107 MHz	55.107MHz
Registers	3699	3571
Number of LUTs	4651	4640
Number of slices	3743	3614

From Table I it can be seen that the speed increase is close to five. The speed increase in the table takes the case of 1D-DCT alone, but during digital image watermarking, we have 4 similar computations and each time a speed up of five can be obtained, which makes this novel implementation very high speed.

The comparison of pipelined and non-pipelined implementation of DCT/IDCT is done in terms of the operating frequency, number of slices, and number of LUTs as shown in the table. Table II shows the comparison for the 1D DCT that is first performed with the data from which is the row-wise DCT. Table III gives the comparison for the DCT that is performed after transposing the result obtained after 1D DCT, in other words, the comparison of the 1D DCT blocks which gives the final output of 2D DCT.

X. CONCLUSION

This paper presents a comparison of pipelined and non-pipelined implementations of 2D DCT/IDCT. From the results obtained, it is quite evident that the speed of digital watermarking can be increased by approximately 500% using the novel pipelined implementation of 2D DCT/IDCT. The paper also brings about a comparison of both pipelined and non-pipelined implementations of 2D DCT/IDCT. In our experiment, various images of different sizes were used in watermarking, and it was proven that this method gives approximately the same speed increase independent of the images used. As shown in the images used in this paper, the visibility of the watermark can also be varied and the output image is also shown in the paper. The advantage of the pipelined implementation of 2D DCT/IDCT is that it consumes less power and also gives a speed increase of approximately 500% over the conventional implementation.

ACKNOWLEDGMENT

We gratefully acknowledge the Almighty GOD who gave us strength and health to successfully complete this venture. The authors wish to thank Amrita Vishwa Vidyapeetham, in particular the Digital Library, for access to their research facilities and for providing us the laboratory facilities for conducting the research.

REFERENCES

[1] *D.W. Trainor J.P. Heron and R.F. Woods*," Implementation of the 2D DCT using a XILINX XC6264 FPGA", 0-7803-3806-5/97

[2] *S. Musupe and S. Arslun,"* Low power DCT implementation approach for VLSI DSP processors", 0-7803-5471 -0/99

[3] *S. An C. Wang* "Recursive algorithm, architectures and FPGA implementation of the two- dimensional discrete cosine transform", The Institution of Engineering and Technology 2008.

[4] *S. Musupe and S. Arslun*, "Low power DCT implementation approach for VLSI DSP processors", 0-7803-5471 -0/99

[5] *Saied Amirgholipour Kasmani, Ahmadreza Naghsh-Nilchi*, " A New Robust Digital Image Watermarking Technique Based On Joint DWTDCT Transformation", Third 2008 International Conference on Convergence and Hybrid Information Technology

[6] *A.Aggoun and I. Jalloh* "Two-dimensional DCT/SDCU architecture", 2003 IEE proceedings online no. *20030063*, *DO/:* 10.1049/ip-edt:20030063

[7] *Syed Ali Khayam*, "The Discrete Cosine Transform (DCT): Theory and Application", Department of Electrical & Computer Engineering, Michigan State University.

[8] *Khurram Bukhari, Georgi Kuzmanov and Stamatis Vassiliadis* "DCT and IDCT Implementations on Different FPGA Technologies", Computer Engineering Lab, Delft University of Technology.

[9] *Kuo-Hsing Cheng*, Chih-Sheng Huang# and Chun-Pin lin "*The Design and implementation of DCT/IDCT Chip with Novel Architecture" , ISCAS 2000 - IEEE international symposium on circuits and systems, may 28-31, 2000, Geneva, Switzerland

[10] *Christoph loeffler, Adriaan lieenberg, and George S. Moschytz*, "Practical fast 1-d DCT algorithms With 11 multiplications ", ch2673-2/89/0000-0098

[11] *Archana Cchidanandan, Joseph Moder, Magdy Bayoumi* "Implementation of neda-based DCT architecture using even-odd decomposition of the 8 x 8 DCT matrix", 1-4244-0173-9/06

[12] *Archana Chidanandan, Magdy Bayoumi*, "Area-efficient neda architecture for the 1-D DCT/IDCT", 142440469x/06/

[13] *MorrisMano*, "Computer System Architecture", 3rdEdition, PHI,2001