Pipelined Fast 2-D DCT Architecture for JPEG Image Compression

Luciano Volcan Agostini *agostini@inf.ufrgs.br*
Ivan Saraiva Silva* *ivan@dimap.ufrn.br*
Sergio Bampi *bampi@inf.ufrgs.br*
**Federal University of Rio Grande do Norte – DIMAp - Natal - Brazil*
Federal University of Rio Grande do Sul - Microelectronics Group
Caixa Postal 15 064 - Porto Alegre – Brazil

Abstract

This paper presents the architecture and the VHDL design of a Two Dimensional Discrete Cosine Transform (2-D DCT) for JPEG image compression. This architecture is used as the core of a JPEG compressor and is the critical path in JPEG compression hardware. The 2-D DCT calculation is made using the 2-D DCT separability property, such that the whole architecture is divided into two 1-D DCT calculations by using a transpose buffer. These parts are described in this paper, with an architectural discussion and the VHDL synthesis results as well. The 2-D DCT architecture uses 4,792 logic cells of one Altera Flex10kE FPGA and reaches an operating frequency of 12.2 MHz. One input block with 8 x 8 elements of 8 bits each is processed in 25.2µs and the pipeline latency is 160 clock cycles.

1. Introduction

Discrete Cosine Transform (DCT) is a mathematical tool that has a lot of electronics applications, from audio filters to video compression hardware. DCT transforms the information from the time or space domains to the frequency domain, such that other tools and transmission media can be run or used more efficiently to reach application goals: compact representation, fast transmission, memory savings, and so on.

The JPEG image compression standard [1], [2] was developed by Joint Photographic Expert Group [3]. The JPEG compression principle is the use of controllable losses to reach high compression rates. In this context, the information is transformed to the frequency domain through DCT. Since neighbor pixels in an image have high likelihood of showing small variations in color, the DCT output will group the higher amplitudes in the lower spatial frequencies [4]. Then, the higher spatial frequencies can be discarded, generating a high compression rate and a small perceptible loss in the image quality. The JPEG compression is recommended for photographic images, since drawing images are richer in high frequency areas that are distorted with the application of the JPEG compression [5].

The JPEG compression can be divided into five main steps, as shown in Fig. 1: color space conversion, downsampling, 2-D DCT, quantization and entropy coding. The first two operations are used only for color images.

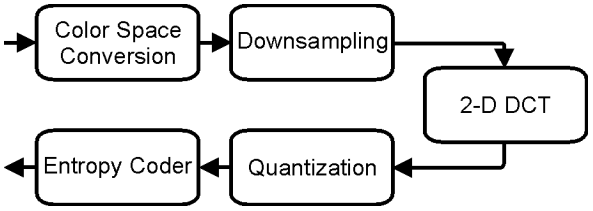

Figure 1 – Steps for JPEG compression of color images

The color space conversion transforms the RGB input image to a luminance and chrominance space color, such as the YCbCr representation. The downsampling operation reduces the sampling rate of the color information (Cb and Cr), because the chrominance components are less important to the human eye. The quantization operation discards the 2-D DCT high frequency and small amplitude coefficients. Finally, the entropy coding uses run-length encoding (RLE), Huffman, variable length coding (VLC) and differential coding to decrease the number of bits used to represent the image [1], [2].

The JPEG compression is a lossy compression, since downsampling and quantization operations are irreversible [5].

The first two steps in JPEG compression are related to color images. If the input image has no color information these two stages are discarded and the compression starts at the DCT calculation.

This paper focuses only in a pipelined hardware implementation of the main part of the JPEG standard: the Two Dimensional Discrete Cosine Transform. This is the most critical module to be designed in a JPEG compressor hardware because of its high algorithm complexity.

This work proposes initially a FPGA implementation for flexibility, time-to-finish and development purposes. The results from RTL level VHDL design can be reused for an ASIC implementation in the future and the designed VHDL for 2-D DCT calculation can be used as a core in other designs like video compression.

The paper sections present the algorithm used for the DCT calculation, the architecture proposed and the VHDL synthesis results after full completion of the design in Altera VHDL.

2. Algorithm Used for the DCT Calculation

The 2-D DCT calculation has a high degree of computational complexity. Since many authors have proposed simplifications to this calculation, as [6], [7], [8] and others, this complexity can be minimized according to the application needs. Specifically for image compression applications there are many algorithms to compute the 2-D DCT coefficientes and the algorithm chosen in this paper was proposed in [7] and modified in [9].

The algorithm chosen calculates the DCT in one dimension (1-D DCT) and the 2-D DCT calculation is made using its separability property. Thus, using two 1-D DCT steps it is possible to generate the 2-D DCT coefficients. In an 8x8 input matrix, the first 1-D DCT is performed row-wise and the second 1-D DCT is performed column-wise on the outputs of the first 1-D DCT. This simple decomposition reduces the complexity of the calculation by a factor of four [4]. The 2-D DCT algorithm executes 64 multiplications and 64 additions to calculate each DCT coefficient. Thus, 4096 multiplications and 4096 additions operations are need for each 8x8 pixels block. Using the row-column decomposition, one needs to compute 16 1D DCTs (eight for the rows and eight for the columns) for a total of 1024 multiplications and 1024 additions operations [4] for the same 8x8 block.

The algorithm proposed in [7] and [9] has other simplifications to reach higher performance. These simplifications make possible the use of just 29 additions and 5 multiplications to calculate an eight point 1-D DCT. Thus, just 80 multiplications and 464 additions are needed for the 2-D DCT calculation of one 8x8 block.

This algorithm does not make the complete 1-D DCT calculation because it is a scaled algorithm. Then, the 1-D DCT outputs are a scale of the real outputs. The DCT scale is corrected with a post-processing that is added to the quantization calculation step. Since the quantization operation and the scale correction are multiplications by constants, these operations can be performed in a single operation [4].

The complete algorithm is presented in Tab. 1, where:

- $m1 = \cos(4\pi/16)$
- $m2 = \cos(6\pi/16)$
- $m3 = \cos(2\pi/16) - \cos(6\pi/16)$
- $m4 = \cos(2\pi/16) + \cos(6\pi/16)$

Table 1 – 1-D DCT algorithm

Step 1		
$b0 = a0 + a7$	$b1 = a1 + a6$	$b2 = a2 - a4$
$b3 = a1 - a6$	$b4 = a2 + a5$	$b5 = a3 + a4$
$b6 = a2 - a5$	$b7 = a0 - a7$	
Step 2		
$c0 = b0 + b5$	$c1 = b1 - b4$	$c2 = b2 + b6$
$c3 = b1 + b4$	$c4 = b0 - b5$	$c5 = b3 + b7$
$c6 = b3 + b6$	$c7 = b7$	
Step 3		
$d0 = c0 + c3$	$d1 = c0 - c3$	$d2 = c2$
$d3 = c1 + c4$	$d4 = c2 - c5$	$d5 = c4$
$d6 = c5$	$d7 = c6$	$d8 = c7$
Step 4		
$e0 = d0$	$e1 = d1$	$e2 = m3 \times d2$
$e3 = m1 \times d7$	$e4 = m4 \times d6$	$e5 = d5$
$e6 = m1 \times d3$	$e7 = m2 \times d4$	$e8 = d8$
Step 5		
$f0 = e0$	$f1 = e1$	$f2 = e5 + e6$
$f3 = e5 - e6$	$f4 = e3 + e8$	$f5 = e8 - e3$
$f6 = e2 + e7$	$f7 = e4 + e7$	
Step 6		
$S0 = f0$	$S1 = f4 + f7$	$S2 = f2$
$S3 = f5 - f6$	$S4 = f1$	$S5 = f5 + f6$
$S6 = f3$	$S7 = f4 - f7$	

3. Two Dimensional DCT Architecture

There are many proposed architectures to calculate de 2-D DCT, as [10], [11], [9], [12] and others. The 2-D DCT architecture used in this paper is generically presented in Fig. 2. This architecture was designed to reach a high operating frequency and to allow the use of pipeline techniques and is based on the architecture proposed in [9] with some modifications. Thus, the architecture was divided into two 1-D DCT architectures and one transpose buffer. The two 1-D DCT architectures are similar but the bit widths at each pipeline stage are different. The 1-D DCT architectures are organized in a six stage pipeline, one stage for each algorithm step. The transpose buffer operates like a temporal barrier between the first and the second 1-D DCT, allowing the use of a 2-D DCT global pipeline

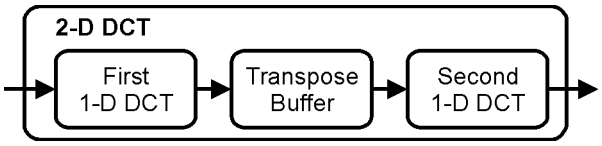

Figure 2 – The generic 2-D DCT architecture

The 2-D DCT inputs in our design are matrixes of 8x8 elements eight-bit wide each. The first 1-D DCT receives and processes this matrixes in a row-wise order. The transpose buffer receives the row-wise results and gives the column-wise inputs to the second 1-D DCT architecture. The second architecture processes the column-wise data and gives a column-wise data output.

Each 1-D DCT stage uses eight clock cycles and the 1-D DCT architecture latency is 48 clock cycles. A complete 8x8 matrix is processed at each 64 clock cycles when the pipeline is full. The 1-D DCT architecture proposed in [9] uses nine clock cycles per stage except for the stage with the multiplier that uses fourteen clock cycles. This 1-D DCT architecture has a latency of 59 clock cycles [9].

The transpose buffer latency is 64 clock cycles and a new transposed matrix are generated at each 64 clock cycles.

The designed 2-D DCT global latency is 160 clock cycles and a complete 8x8 matrix is processed at each 64 clock cycles when the pipeline is full. The 2-D DCT architecture proposed in [9] has a latency of 172 clock cycles.

The 2-D DCT input data must pass trough a level shifter before the DCT calculation starts to be used in the JPEG compression. This level shifter converts the unsigned binary input data representation to a two's complement representation. The basic operation is a subtraction of all input values by 128. This operation can be simplified to just one inversion of the most significant bit of the input.

3.1. One Dimensional DCT Architecture

The 1-D DCT architecture proposed in [9] and designed in this paper is presented in the Fig. 3

Considering the 1-D DCT algorithm steps, the use of a pipelined architecture between these steps becomes natural. Since the algorithm has six steps, the pipeline will have six stages, where five perform additions/subtractions and one performs multiplications. Each pipeline stage operates with the n-value sets (stored in a,b,c,d,e,f pipeline registers in Fig. 3) during eight clock cycles.

The five adders in the DCT architecture are ripple-carry and the multiplier is based on shift-add operations. This architecture uses a single arithmetic unit in each stage to perform all the necessary operations at that stage. Then the units inputs are connected by multiplexers to be selected which value must be used in each clock cycle. The mux controls are generated following the algorithm order.

Temporal barriers to allow the pipeline design are obtained with the use of ping-pong registers. These registers and the multiplier design will be focused in detail in the next sections of the paper.

Since two 1-D DCT architectures are used in the 2-D DCT calculation, the number of bits in each pipeline stage of the two 1-D DCT architectures are different. This difference between the output bit width of the pipeline stages in the two architectures is presented in the Tab. 2.

Table 2 – Bit width differences between the two 1-D DCT architectures

Pipeline Stage	First 1-D DCT bit width	Second 1-D DCT bit width
1	9	13
2	10	14
3	11	15
4	11	15
5	12	15
6	12	15

The simplified temporal diagram of the 1-D DCT pipeline is presented in Fig. 4. This figure presents the partial processing of two 8x8 matrixes, identified by the letters *X* and *Y*. The subscript numbers identify the input matrix elements that are used in each stage for the 1-D DCT calculation. The 1-D DCT pipeline latency is 48 clock cycles and a complete 1-D DCT calculation uses 64 clock cycles.

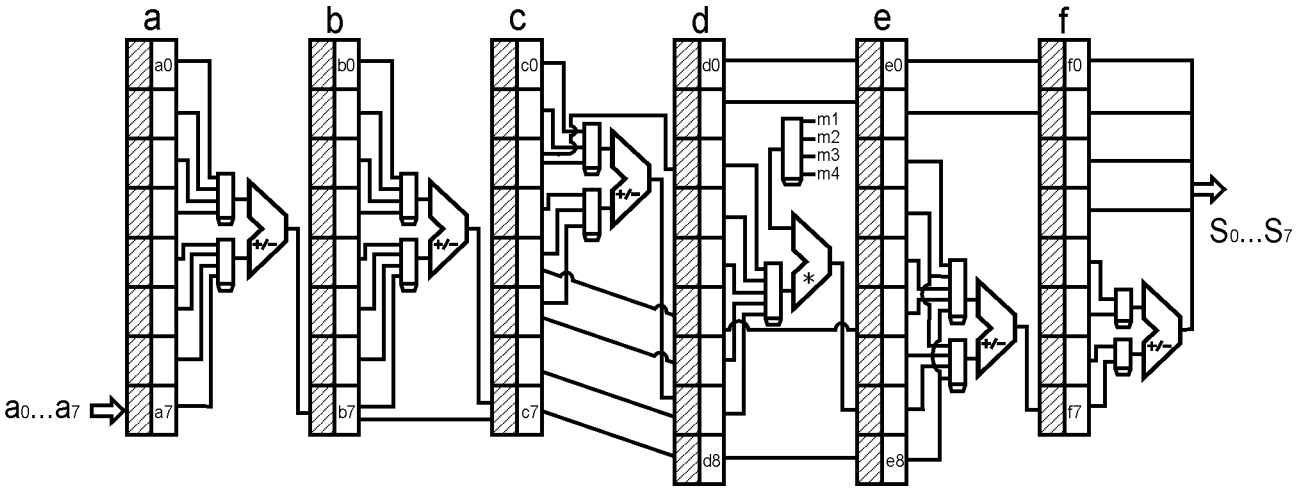

Figure 3 – One dimensional DCT architecture

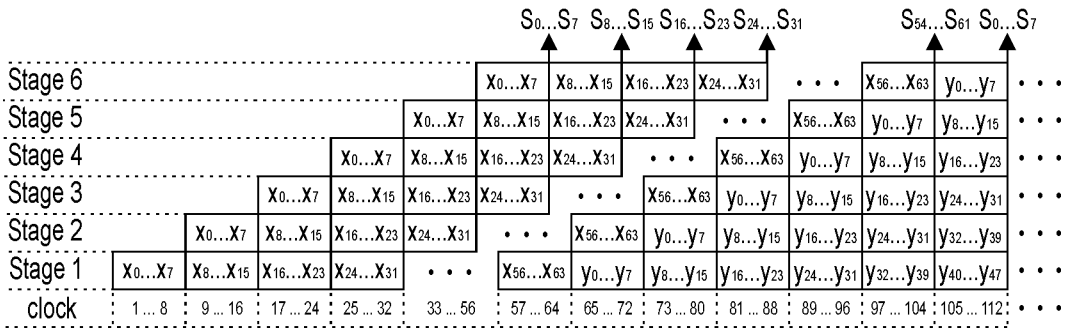

Figure 4 – Simplified time diagram of the 1-D DCT pipeline

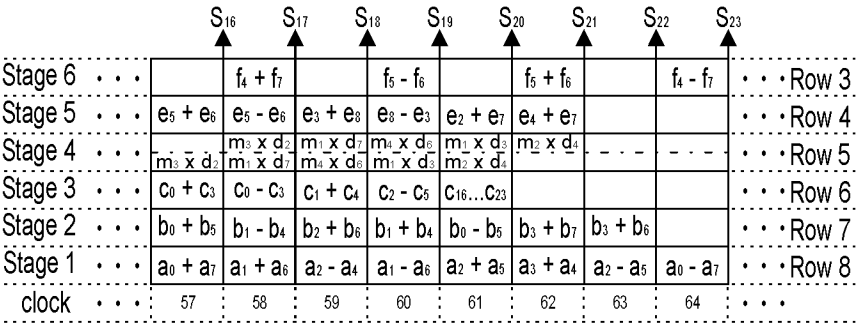

Figure 5 - Zoom into the clock cycles 57 to 64 in the time diagram

Fig. 5 presents a zoom in the diagram of Fig. 4, zooming on clock cycles 57 to 64. In this zoom it is possible to view which calculations are being performed in each pipeline stage. The last six lines of the *X* matrix are being calculated in the Fig. 5, and the third line is ready at the end of clock cycle 64. The stage 4 presents the multiplier pipeline that will be presented in higher details in the next item.

The 1-D DCT outputs must be serially generated by this architecture. While some outputs are generated in parallel (S0, S2, S4 and S6) and others are serially generated (S1, S3, S4 and S7), the 1-D DCT architecture must control the correct order of the output values.

The control block generates the signals to control the pipeline fill-up and emptying through a external signal that indicates if the input values are valid values for the image.

3.1.1. Multiplier Architecture

The multiplier used in the 1-D DCT architecture was decomposed in shifts and adds as a way to minimize the hardware. Since one of the multiplier inputs is always a constant, it is possible to preview the number of shifts necessary for each calculation. The shifters were designed as barrel shifters.

The multiplier architecture is different from the architecture proposed in [9]. The architecture proposed in [9] uses a six stage Wallace tree multiplier that uses 14 clock cycles to perform the five necessary multiplications. The multiplier designed in this paper, using shifts and adds, uses 6 clock cycles, saving 8 clock cycles.

The number of shift-adds was restricted to four to save arithmetic units. This restriction generates an error smaller than 0.6% in the constants values.

The multiplier architecture was designed to operate in a two-stage pipeline. This architecture is presented in Fig. 6.

Internally to the multiplier all the significant bits were considered to maximize the precision of this calculation but the multiplier outputs are truncated to discard the fractional part. The error generated has little significance for the JPEG compression quality, given that the next operation, after 2-D DCT operation completion, is an integer division by numbers higher than 10 and all the DCT outputs are operated on by such divider.

The designed multiplier makes the necessary 5 multiplications in six clock cycles, conveniently less than the eight cycles consumed by previous stages. Besides, the multiplier delay is similar to the delay in the adders of the other pipeline stages, once the multiplier main components are also adders.

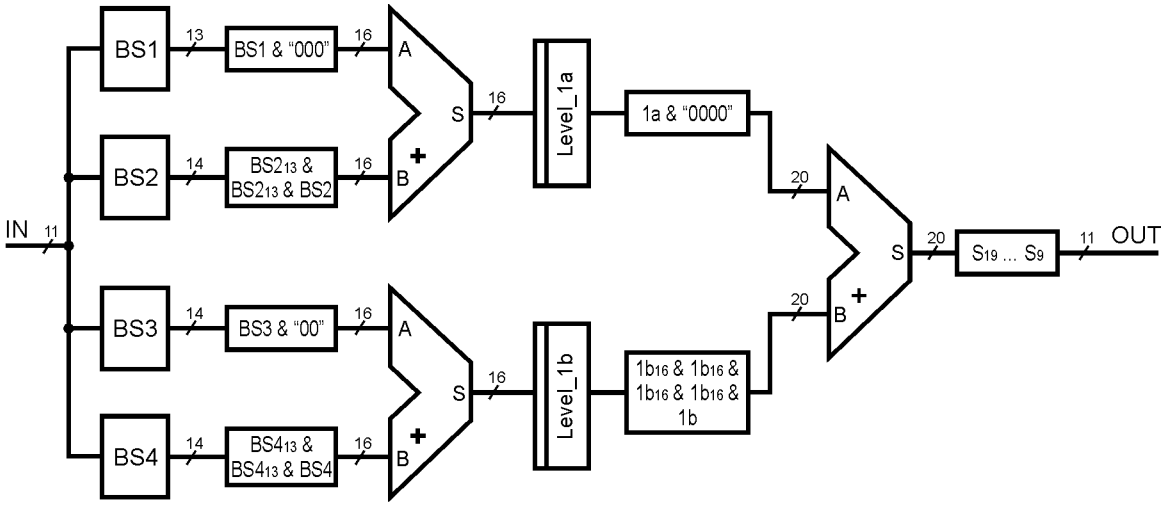

Figure 6 – Multiplier architecture in the DCT pipeline

3.1.2. Ping-Pong Buffers

The input data in each step of the scaled algorithm is stored in ping-pong buffers. The input data are serially generated with a rate of one new value at each clock cycle. All input values of each stage must be unchanged during eight clock cycles to allow the correct calculation. Then, ping-pong buffers are used to synchronize the serial data generation and the parallel data consumption, allowing the pipeline implementation.

Fig. 7 presents an architecture for an eight-value ping-pong buffer. The data enter serially and are stored in the ping registers. When the *Enable* signal is set to one, the ping registers data are transferred in parallel to the pong register.

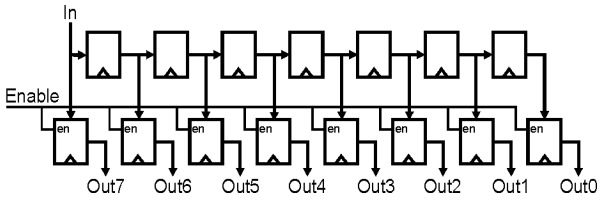

Figure 7 - Example of a ping-pong buffer

3.2. Transpose Buffer Architecture

The transpose buffer is used to connect the two 1-D DCT architectures once the first 1-D DCT outputs are row-wise and the second 1-D DCT inputs must be column-wise. Fig. 8 presents the transpose buffer, that was designed with two small 64-word 12-bit wide RAM.

The architecture proposed in [9] uses full custom registers to design the transpose buffer. The use of registers is better than the use of conventional RAM cells since their latency is lower and their performance is higher. The architecture designed in this paper uses RAM memory because the used FPGA device has internal RAM macroblocks. The use of registers in FPGAs takes a lot of logic cells and their performance is not better than the internal FPGA RAM performance. Other important reason to use internal memory is to save logic cells to be used in the other parts of the JPEG compressor.

The transpose buffer VHDL description is specific to be used with Altera [13] devices since internal FPGA memory are defined as Altera macroblocks.

When the first 1-D DCT architecture writes the results line by line in one memory (RAM1 or RAM2), the second 1-D DCT architecture reads the input values column by column from the other memory (RAM2 or RAM1). The read (*Rad* signal in Fig. 8) and write (*Wad* signal in Fig. 8) addresses are generated by a control block and this control block defines, by *Control* signal in Fig. 8, which memory is used to Read/Write at each memory access step.

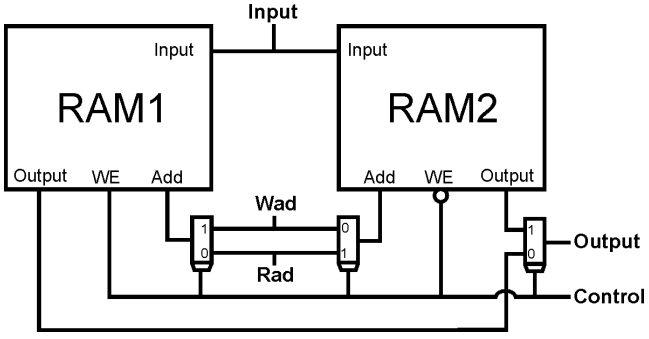

Figure 8 - Transpose buffer architecture

4. VHDL Synthesis Results

The 2-D DCT architecture was described in VHDL. This VHDL was synthesized into an Altera Flex 10KE family FPGA [13].

The VHDL description of the two 1D-DCT architectures is structural and device independent. This VHDL can be used for FPGAs or standard cell synthesis approach. The VHDL of the transpose buffer uses an Altera specific library that allows the use of internal memory, making it an Altera dependent description.

The complete synthesis results to Altera FPGAs are presented in Table 3, whose hardware was fit in an EPF10K100EQC208-1 device.

Table 3 – 2-D DCT VHDL synthesis results

	Logic Cells	Period (ns)	Memory Bits
1-D DCT 1	2051	73.2	0
1-D DCT 2	2473	80.8	0
Trans.Buf.	274	36.5	1536
2-D DCT	**4792**	**82.1**	**1536**

These results encourage the use of this architecture in one complete JPEG compressor, including the I/O and bus control functions which are omitted in this paper. Each 8x8 input matrix are processed in 5.6µs when the pipeline is full. One gray level image of 640 x 480 pixels is completely processed in just 25.2ms considering an empty pipeline, allowing a processing image rate of 39 images per second. Using the same image size, considering color images, one image is completely processed in 75.7ms , with a processing image rate of 13 images per second.

The 2-D DCT hardware was designed and described in approximately 5,250 lines of VHDL code. Hardware simulation and verification was performed module by module.

6. Conclusions

This paper presented the architecture of the 2-D DCT. The modules of the transpose 1-D DCT architecture were designed and synthesized. The control block hardware design is also completed. The detailed pipeline design, operators, and the final results of the synthesis of the modules were also presented, resulting in an architecture containing 4,792 logic cells, including the control block. The paper contributed with specific simplifications in the multiplier stage, by using shift-adds operations, which lead to hardware simplification and speed up over the original architecture proposed in [9].

The designed architecture performs the 2-D DCT calculation of a 640 x 480 pixels gray level image in 25.2ms, allowing its use in a JPEG compressor in hardware.

References

[1] The International Telegraph and Telephone Consultative Committee (CCITT). "Information Technology – Digital Compression and Coding of Continuous-Tone Still Images – Requirements and Guidelines". Rec. T.81, 1992.

[2] W. Pennebaker, J. Mitchell. *JPEG Still Image Data Compression Standard*, Van Nostrand Reinhold, USA, 1992.

[3] "Home site of the JPEG and JBIG committees" <http://www.jpeg.org/> (21/04/01).

[4] V. Bhaskaran, K. Konstantinides. *Image and Video Compression Standards Algorithms and Architectures – Second Edition*, Kluwer Academic Publishers, USA, 1999.

[5] J. Miano. *Compressed Image File For*mats – JPEG, PNG, GIF, XBM, BMP, Addison Wesley Longman Inc, USA, 1999.

[6] W. Chen, C. Smith, S. Fralick. "A Fast Computational Algorithm for the Discrete Cosine Transform". *IEEE Transactions on Communications*, v. COM-25, n. 9, p. 1004-1009, 1977.

[7] Y. Arai, T. Agui, M. Nakajima. "A Fast DCT-SQ Scheme for Images". *Transactions of IEICE*, vol. E71, n°. 11, 1988, pp. 1095-1097.

[8] E. Feig, S. Winograd. "Fast Algorithms for the Discrete Cosine Transform". *IEEE Transactions on Signal Processing*, v. 40, n. 9, p. 2174-2193, 1992.

[9] M. Kovac, N. Ranganathan. "JAGUAR: A Fully Pipeline VLSI Architecture for JPEG Image Compression Standard". *Proceedings of the IEEE*, vol. 83, n°. 2, 1995, pp. 247-258.

[10] A. Madisetti, A. Willson Jr. "A 100 MHz 2-D 8 x 8 DCT/IDCT Processor for HDTV Applications". *IEE Transactions on Circuits and Systems for Video Technology*, v.5, n.2, p.158-165, Apr. 1995.

[11] C. Wang, C. Chen. "High-Throughput VLSI Architectures for the 1-D and 2-D Discrete Cosine Transforms". *IEE Transactions on Circuits and Systems for Video Technology*, v.5, n.1, p.31-40, Feb. 1995.

[12] Y. Lee, T. Chen, L. Chen., M. Chen, C. Ku. "A Cost-Effective Architecture for 8 x 8 Two-Dimensional DCT/IDCT Using Direct Method". *IEEE Transactions on Circuits and Systems for Video Technology*, v.7, n.3, p.459-467, Jun. 1997.

[13] Altera Digital Library, Altera Corporation, June 2000.