THE FAST COMPUTATION OF DCT IN JPEG ALGORITHM

Miodrag Popović
Faculty of Electrical Engineering, University of Belgrade
Bulevar revolucije 73, 11120 Belgrade, Yugoslavia
e-mail: pop@el.etf.bg.ac.yu

Tomislav Stojić
Faculty of Mechanical Engineering, University of Belgrade
27 marta 80, 11120 Belgrade, Yugoslavia

ABSTRACT

In this paper two new parallel algorithms are proposed for the computation of discrete cosine transform (DCT) in JPEG algorithm on 32-bit computers. These algorithms are twice as fast as the fastest known DCT algorithm published to date. This is possible because 16-bit wordlength is sufficient for the computation of DCT, so the arithmetic unit is used to perform simultaneously two 16-bit operations. All operations in the DCT algorithm, that are not suitable for parallel processing, are substituted by equivalent operations.

1 INTRODUCTION

The International Standard Organization has adopted ISO/IEC-10918 standard for still image compression, widely known as the JPEG standard. The key part of this standard is the basic sequential method [1, 2], which is used in most practical implementations. The JPEG standard is based on the use of the Discrete cosine transform - DCT [3]. The DCT coefficients are quantized and entropy coded. Although the standard does not prescribe the way for the computation of the DCT, the test procedure is prescribed for checking compatibility of the results obtained by using different DCT algorithms.

The fast computation of DCT is of primary importance in software or hardware implementations of the JPEG standard. There are many fast DCT algorithms in literature, that reduce the number of arithmetic operations. In this paper new improvements of fast DCT algorithms are described, that are based on the use of parallel arithmetic. These algorithms are almost twice as fast on 32-bit computers as the fastest known DCT algorithm published to date. This is possible because 16-bit wordlength is sufficient for the computation of DCT [2], so that arithmetic unit can be used to perform simultaneously two 16-bit operations. But, since all operations needed to compute the DCT cannot be performed in parallel, a new formulation of the DCT algorithm is needed, in order to retain only these operations that can be executed in parallel. In this way, all operations in the DCT algorithm that are not suitable for parallel processing should be substituted by equivalent operations.

The paper is divided in three parts. In Section 2, the basic principles of parallel arithmetic are described. Then, in Section 3, the fastest known DCT algorithm is described [1, 4]. Modifications of the original AA&N algorithm lead to two new parallel algorithms PAR-I and PAR-II, described in Section 4, that speed up the DCT computation at least two times on 32-bit computers. Finally, some test results are shown in Section 5 in order to show the validity of this proposal.

2 PARALLEL ARITHMETIC

Parallel or twofold [5] arithmetic uses 32-bit arithmetic unit for simultaneous execution of two 16-bit arithmetic operations using only one 32-bit instruction. The idea is to use DCT computation of one block of 32-bit samples to obtain the DCT of two blocks of 16-bit samples. Therefore, the parallel processing simultaneously computes the DCT of two blocks of 8×8 samples.

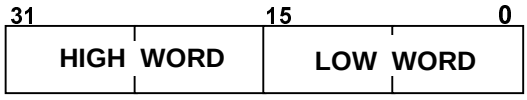


Fig. 1. *Packing of two* 16*-bit values in one* 32*-bit word.*

The basic restriction of parallel arithmetic is that operands and results of the operations must be integers. This requirement does not impose significant problems in DCT computation, since integer realizations of DCT fast algorithms are most frequent in practice due to their high speed. Therefore, we will assume that all numbers are integers in the *m*-bits 2nd complement representation. The sign of the number is represented by the MSB.

In Fig. 1, a 32-bit register is shown, whose contents defines two integer quantities, *HIGH WORD* and *LOW WORD*. Both numbers are represented in 16-bit 2nd complement representation. The *LOW WORD* occupies two lower bytes, and *HIGH WORD* occupies two higher bytes of this register.

The formation of the 32-bit quantity in a register is not performed by independent packing of two 16-bit numbers in higher and lower parts of the register, but by the following procedure. Let W_H and W_L denote two corresponding samples belonging to two different blocks. The value in the 32-bit register, W_{NET} has the following representation:

$$W_{NET} = 2^{16} W_H + W_L, \quad -2^{15} \le W_H, W_L \le 2^{15} - 1 \quad (1)$$

where W_H and W_L are 16-bit integer quantities.

Let us analyze first the 32-bit addition operation, $A + B \to C$, where:

$$A_{NET} = 2^{16} A_H + A_L \qquad B_{NET} = 2^{16} B_H + B_L \tag{2}$$

$$\begin{aligned} C_{NET} &= A_{NET} + B_{NET} \\ &= 2^{16}(A_H + B_H) + (A_L + B_L) = 2^{16} C_H + C_L \end{aligned} \tag{3}$$

This gives under requirement:

$$-2^{15} \le A_L, B_L, A_L + B_L, A_H, B_H, A_H + B_H \le 2^{15} - 1 \tag{4}$$

$$C_L = A_L + B_L, \qquad C_H = A_H + B_H \tag{5}$$

Therefore, the addition operation is transparent in parallel arithmetic. One 32-bit addition instruction realizes two 16-bit additions.

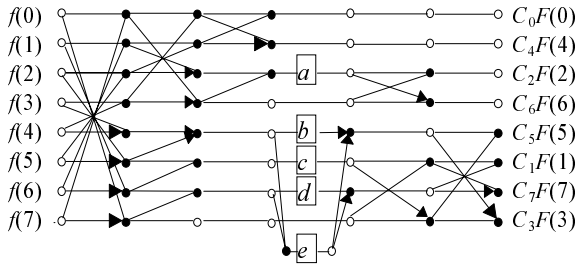
The arithmetic left shift operation, e.g. multiplication by a degree of 2, can be also executed in parallel, under the condition that the result does not exceed the range of 16-bit numbers.

The multiplication operation cannot be executed in parallel manner, since the product of two *m*-bit numbers contains *2m* bits, that can produce overflow in lower part of register and propagation of bits into higher part of register. The division cannot be executed in parallel for the same reason. The arithmetic right shift, e.g. division by a degree of 2, cannot be executed in parallel under any conditions, since the bits from the high word move into the low word of the register.

Therefore, the results of this analysis can be summarized as follows: the operations of addition or subtraction (ADD) and arithmetic left shift (SAL) can be executed twice faster under the restriction that both the operands and results are 16-bit quantities. The operations of multiplication (MUL), division (DIV), and arithmetic right shift (SAR) can not be accelerated by parallel execution.

3 DISCRETE COSINE TRANSFORM

The basic source of the compression in the JPEG algorithm is the Discrete cosine transform. In practical realizations, the computation of the DCT is performed by using fast algorithms. The most frequent approach uses the separability of the two-dimensional (2-D) DCT, where the computation of the DCT of a block of 8×8 samples reduces to computations of one-dimensional (1-D) DCTs of rows and columns, respectively. The fastest, but relatively less-known algorithm for the fast 1-D DCT computation of 8-sample sequence is so-called AA&N algorithm [1, 4]. This algorithm is represented by the flowgraph in Fig. 2. The black circles in the diagram represent the addition of the signals entering the node, and arrows represent the change of sign before the addition. The multiplication of the signal is denoted by a square in which the value of the multiplier is written. The output variables $F(k)$, scaled by factors C_k, are related to the DCT coefficients by the formulae [4]:



$a = c = 0.707106718$, $b = 0.541196100$,
$d = 1.306562963$, $e = 0.382683432$,
$C_0 = 8$, $C_1 = C_2 = C_3 = C_4 = C_5 = C_6 = C_7 = 16$.

Fig. 2. *AA&N DCT algorithm*.

$$C(0) = \sqrt{2} F(0) \tag{6}$$

$$C(k) = \frac{2F(k)}{\cos\frac{k\pi}{16}} \tag{7}$$

If the explicit values of the 8-sample DCT are to be obtained, the output values of Fig 2 should be multiplied by a set of constants, so that the total number of multiplications is 13. However, in JPEG algorithm, this final rescaling step can be combined with the quantization step, so that the total number of operations for 8-sample sequence is reduced to only 5 multiplications and 29 additions. Therefore, the AA&N method is the fastest known DCT algorithm for 8-sample sequences [1]. Unfortunately, this original AA&N algorithm still contains 5 multiplications with rational factors that make it unsuitable for the use of parallel arithmetic.

4 PARALLEL DCT ALGORITHMS PAR-I AND PAR-II

It is well known that fixed point (or integer) arithmetic is faster than floating point arithmetic. For example, our tests showed that computation of DCT in JPEG realizations can be made faster about three times, by simple changing all variables to integers. This change can be performed by multiplying rational constants in the algorithm by scaling factor and rounding the results. In this way the multiplications are made faster, but they must be completely eliminated from the algorithm in order to permit its execution in parallel.

The adaptation of a fast DCT algorithm to parallel processing need the replacement of multiplications by additions and left shifts that can be executed in parallel. The first attempt in this way was the use of the reparametrized Suehiro transform (RST) [6]. But, in the algorithm proposed in [5] it was necessary to use arithmetic right shift operation (SAR), that is equivalent to three additions and cannot be executed in parallel. Therefore, the benefits from the elimination of multiplications were not fully exploited.

In this paper two new methods are proposed that are based on the fact that all DCT coefficients can be

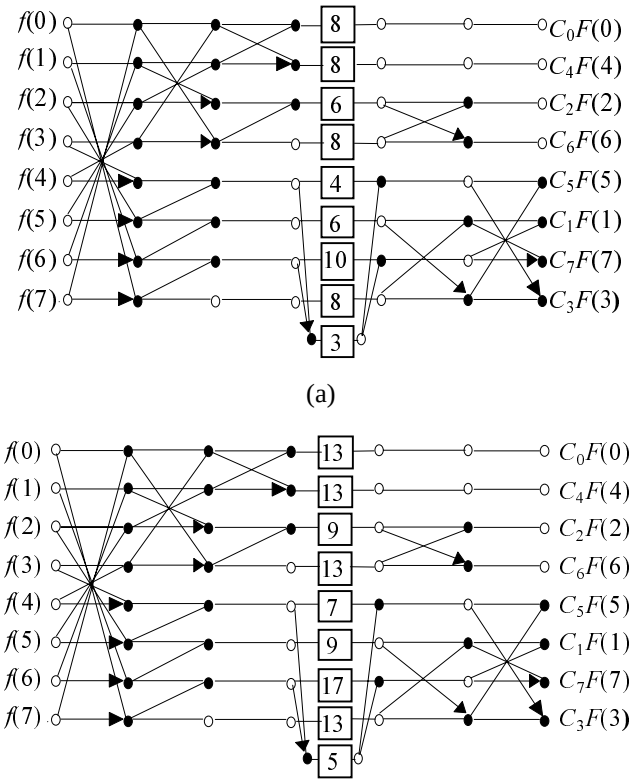


Fig. 3: (*a*) *Algorithm PAR-I*, (*b*) *Algorithm PAR-II*.

represented using at most 11 bits. Therefore, since 16 bits are available, the scaling constants must be represented by only five available bits. Two best approximations of DCT are obtained if all rational constants in AA&N algorithm are scaled by 8 (algorithm PAR-I), or by 13 (algorithm PAR-II,) and rounded to nearest integers. The flowgraphs of these algorithms are shown in Fig. 3. In the PAR-I case, the new set of multiplication factors is $a = c = 6$, $b = 4$, $d = 10$, $e = 3$, while in the PAR-II case the factors are $a = c = 9$, $b = 7$, $d = 17$, $e = 5$. Since all output values are larger 8 or 13 times, in order to prevent overflow it is necessary to scale all quantities by 64 between rows and columns processing.

In order to permit faster execution the multiplications should be replaced by additions (ADD) and arithmetic left shifts (SAL). The instruction sets of some processors, such as Intel 80386 and higher [7], permit still better saving of execution time. They contain the instruction LEA (or equivalent) that permits simultaneous nultiplication and addition, such as:

LEA EAX, [EAX+m*EAX], m = 2, 4, 8

so that multiplications with factors 3, 5, 9 and 17 can be realized using only one LEA instruction, as shown in Table 1. The multiplications with 6, 7, and 10 require one LEA and one ADD instructions, while multiplication with 13 requires one LEA, one ADD and one SAL instructions. Further, the multiplications with 4 or 8 require only one left shift instruction SAL. Therefore, in all cases it is possible to replace multiplications by a small number of simple operations that permit fully parallel computation.

Table 1: *Substitution of multiplications using LEA, SAL and ADD instructions.*

Multiplier	LEA	SAL	ADD
3	1	-	-
4	-	1	-
5	1	-	-
6	1	-	1
7	1	-	1
8	-	1	-
9	1	-	-
10	1	-	1
13	1	1	1
17	1	-	-

If the operands of these instructions come from the registers, it can be considered that the execution time for the instructions LEA and SAL is equal to the execution time of the ADD instruction. Hence, the execution time of the whole scaled DCT algorithm can be expressed by the number of so-called equivalent additions. In Table 2, three algorithms are compared: two new algorithms PAR-I and PAR-II and the RST algorithm proposed in [6]. The number of operations was counted assuming that two 8-sample blocks are simultaneously transformed.

Table 2: *Comparison of parallel 8-sample DCT algorithms.*

operation	RST	PAR-I	PAR-II
addition ADD	43	42	36
left shift SAL	3	6	5
right shift SAR	9×3=27	-	-
equivalent ADDs	73	48	41
equivalent ADDs per block	36.5	24	20.5

5 EXPERIMENTAL VERIFICATION

In order to verify the quality of the proposed solution several experiments were performed. As the basic JPEG implementation, a widely available source code in C distributed by the Independent JPEG group [8] was used. The PAR-I and PAR-II algorithms were implemented by modifications of C code, necessary to implement parallel processing. The results were compared to the results of the original JPEG algorithms.

First, the execution times of original and parallel algorithms were measured on a PC-486 66 MHz computer. Since the execution is fast, a special C program for measuring short time intervals, having the resolution of 10 μs, was used [9]. The results for the 512×512 24-bit color image *fruits* are presented in Table 3. It can be seen that the use of integer instead of floating point arithmetic in

the original AA&N algorithm makes the execution 2.5 times faster, while the algorithm PAR-II makes the execution faster 7.25 times compared to floating point, or 2.16 times compared to fixed point realization of the AA&N algorithm.

Table 3: *Time to compute the DCT of whole image* (s)

	AA&N (float)	AA&N (integer)	PAR-I	PAR-II
DCT	1.710	0.510	0.279	0.236

Since the new parallel algorithms represent only approximations of the true DCT algorithm, it is interesting to find the approximation error. The detailed analysis shows that the approximation error is smaller than the quantization error, even for the high values of the JPEG quality factor *Q*. In order to experimentally prove this conclusion, the peak signal-to-noise ratio (PSNR) was computed for several values of the *Q* factor, and the results are presented in Table 4. It can be seen that in the worst case (for $Q = 75$) the PSNR is reduced for only 0.08 dB (PAR-I), or 0.65 dB (PAR-II).

Table 4: *Signal-to-noise ratio PSNR* (dB).

Q	AA&N (float)	AA&N (integer)	PAR-I	PAR-II
75	30.54	30.54	30.46	29.89
45	28.94	28.94	28.92	28.57
15	26.99	26.99	26.97	26.93

Finally, it was examined whether the parallel execution of the DCT changes the compression ratio. Some representative results for the same values of *Q* as before are presented in Table 5. It can be seen that the number of bits needed to code an image is slightly increased, especially when PAR-II algorithm is used to compress images using high values of *Q* factor.

Table 5: *Bits-per-pixel ratio.*

Q	AA&N (float)	AA&N (integer)	PAR-I	PAR-II
75	0.910	0.903	0.938	1.006
45	0.566	0.566	0.565	0.599
15	0.309	0.309	0.309	0.312

The subjective tests do not show any visual difference between images processed by the original and by fast parallel algorithms.

Since all the test performed to examine the performances of the new parallel algorithms show that these algorithms are at least twice faster compared to the known implementations, and that they do not introduce any significant degradation in signal-to-noise ratio or compression ratio, it can be concluded that the use of the new fast parallel algorithms in software implementations of JPEG algorithms is fully justified.

6 CONCLUSION

In this paper two new algorithms are proposed for the computation of discrete cosine transform (DCT), suitable for parallel execution on 32-bit computers. These algorithms are at least twice as fast as the fastest known DCT algorithm published to date. This is possible because 16-bit wordlength is sufficient for the computation of DCT, so the arithmetic unit is used to perform simultaneously two 16-bit operations. In order to adapt the DCT algorithm for parallel processing, all operations in the DCT algorithm, that are not suitable for parallel processing, are substituted by equivalent operations. The results of objective and subjective tests show that the use of parallel arithmetic do not introduce any significant errors, so its use is fully justified.

REFERENCES

[1] W.B. Pennebaker, J.L. Mitchell, *JPEG Still Image Data Compression Standard*, Van Nostrand Reinhold, New York, 1993.

[2] G.K. Wallace, "The JPEG still picture compression standard", *Communications of the ACM*, Vol. 34, No. 4, pp. 30-44, April 1991.

[3] K.R. Rao, P. Yip, *Discrete Cosine Transform: Algorithms, Advantages, Applications*, Academic Press, New York, 1990.

[4] Y. Arai, T. Agui, M. Nakajima, "A fast DCT-SQ scheme for images", *The Transactions of the IEICE*, Vol. E 71, No. 11, pp. 1095-1097, Nov. 1988.

[5] J.D. Allen, "An approach to fast transform coding in software", *Signal Processing: Image Communication*, Vol. 8, No. 1, pp. 3-11, 1996.

[6] N. Suehiro, M. Hatori, "Fast algorithms for the DFT and other sinusoidal transforms", *IEEE Trans. Acoustics, Speech, Signal Processing*, Vol. 34, No. 3, pp. 642-644, June 1986.

[7] *Intel 80386 Programmer's Reference Manual 1986*, Intel Corporation, 1987.

[8] *Independent JPEG Group's Software, Release 5a*, IJG software, Dec. 1994.

[9] J. Jongerius, "Accurately timing WINDOWS events without timer reprogramming", *Microsoft Journal*, No. 4, pp. 1-9, 1991.