2–D DCT USING ON–LINE ARITHMETIC

Javier Bruguera

Department of Electronics
University of Santiago de Compostela
15706 Santiago de Compostela. SPAIN
bruguera@gaes.usc.es

Tomas Lang

Department of Electrical and Computer Eng.
University of California at Irvine
Irvine. CA 92717. USA
tomas@ece.uci.edu

ABSTRACT

We present a VLSI architecture for the evaluation of the (8x8)–point 2–D DCT with on–line arithmetic. The utilization of on–line arithmetic, in combination with an algorithm based on FCT and matrix multiplication, reduces the total hardware maintaining a data rate and a latency similar to approaches based on distributed or parallel arithmetic. The architecture has been integrated in a chip using a 1μ CMOS technology, occupying an area of $56.7mm^2$.

1. INTRODUCTION

The two dimensional Discrete Cosine Transform is considered an efficient technique for image compression and is being utilized as standard in several applications, including video compression, storing and transmission of still images (JPEG) and moving pictures (MPEG) and HDTV.

Since direct implementation of the 2–D DCT of an NxN real matrix is computationally intensive, it is usually implemented by means of the row–column decomposition technique (separated 2–D DCT), in which the N–point 1–D DCT of each column of the input data matrix is computed, and then followed by a N–point 1–D DCT of each row of the resulting matrix. There are in the literature several implementations which compute the 2–D DCT according to this method using parallel arithmetic [8], serial arithmetic [1] and distributed arithmetic [7] [9].

In this paper we present an VLSI architecture for the computation of the separated (8x8)-point 2–D DCT with on–line arithmetic [3]. We develop an on-line recurrence for the 1-D DCT and then integrate it into the overall algorithm. The architecture has been integrated in a chip using a $1\mu m$ CMOS technology, occupying an area of $56.7mm^2$. Although the on-line algorithm is digit serial, because it operates with the most-significant digit first it achieves the same latency as parallel and distributed-arithmetic implementations. The area of the on–line implementation is smaller than the area of the parallel implementation, since each multiplier is replaced by two 4-to-2 CSA adders and one 5-bit CPA adder. On the other hand, implementations with distributed arithmetic need a larger ROM area.

This work was supported in part by the Ministry of Education and Science (CICYT) of Spain under contract TIC92–0942. This work was performed while J. Bruguera was with University of California at Irvine

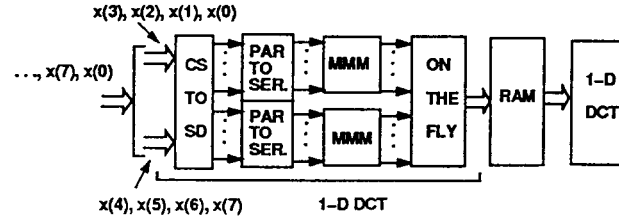

Figure 1: On–line 8x8 point 2–D DCT

2. ARCHITECTURE OF ON–LINE 2–D DCT

The 8–point 1–D DCT of an input sequence $\{x(0), \ldots, x(7)\}$ is defined as

$$y(m) = \frac{1}{2}u(m)\sum_{i=0}^{7} x(i)\cos\frac{(2i+1)m\pi}{16} \qquad 0 \le m \le 7 \quad (1)$$

being $u(0) = 1/\sqrt{2}$ and $u(m) = 1$ if $m \neq 0$

To reduce the number of multiplications in the computation of each 1–D DCT, two 4–point subsequences, $\{u(i)\}$ and $\{v(i)\}$ with $0 \le i \le 3$, are obtained as

$$\begin{aligned} u(i) &= x(i) + x(7-i) \\ v(i) &= x(i) - x(7-i) \end{aligned} \quad (2)$$

To achieve this the input sequence is ordered as {x(0), x(7), ..., x(3), x(4)}. Then, the transformed sequence is computed as,

$$y(i) = \sum_{j=0}^{3} c(i,j)z(j) \qquad 0 \le i \le 7 \quad (3)$$

being c(i,j) the elements of the matrix of cosine coefficients and $z(j) = u(j)$ when $i = 0, 2, 4, 6$ and $z(j) = v(j)$ when $i = 1, 3, 5, 7$.

Figure 1 shows the block diagram of the on–line architecture for the computation of the 8x8 2–D DCT. The subsequences $u(i)$ and $v(i)$ are represented in carry save (CS) format, where the sum word is $x(i)$ and the carry word is $x(7-i)$ or $-x(7-i)$. To obtain $-x(7-i)$ the input is complemented and the 1 is added in the CS–to–SD recoder. In this way, it is not necessary to compute the pre–additions.

Considering that the input and output are 9–bit and 12–bit data, respectively, as on–line arithmetic is a digit–serial arithmetic and the input sequence has to be processed in 8 cycles, radix 4 digits are used. Moreover, to simplify the multiplication in the implementation of the recurrence, the CS representation is converted into the symmetrical sign–digit format with digit set $\{-2, \ldots, 2\}$ [4]. Then, they are

0-7803-2431-5/95 $4.00

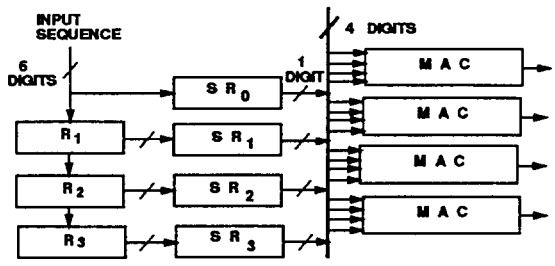

Figure 2: Digit serial matrix–matrix multiplication unit

converted to a digit–serial format. Equation (3) is computed in units MMM with an on–line implementation. Results obtained in the first 1–D DCT are converted to a parallel non–redundant representation, using an on–the–fly conversion unit [4], before being stored in the transposition RAM. The operation of the second 1–D DCT is similar.

The structure of the parallel–to–serial converter and the MMM unit for one of the subsequences is shown in figure 2. The bit–parallel data subsequence is shifted sequentially into the set of registers R. Then, the contents of the R registers are simultaneously bit–parallel loaded in the SR shift registers. Next, the data are digit–serial shifted out starting with the most significant digit, while a new sequence is loaded in the R registers.

Each element of the transformed sequence is obtained as a Multiply and Accumulate (MAC) operation (eq. (3)). These elements are computed concurrently using 4 MAC units. Each MAC operation is implemented with an on–line recurrence. All MAC units of each matrix–matrix multiplication block receive the same 4 input digits and use a different cosine coefficient set.

Then, to design an on–line unit for the computation of the DCT, we have to derive an on–line algorithm to evaluate equation (3).

3. ON–LINE IMPLEMENTATION

On–line arithmetic [3] is a digit–serial arithmetic in which operations are executed most–significant digit first. To generate the *p–th* digit of the result, up to digit $p+d$ of the operands are needed, where d is the *on–line delay*. On–line arithmetic requires the use of a redundant representation for the results.

The algorithm we develop computes the 1–D DCT in on–line format with respect to the inputs and the outputs, and cosine coefficients are stored in parallel non–redundant format. The input and output are expressed, as follows,

$$z(j) = \sum_{p=1}^{n+d} z_p(j) r^{-p}, \qquad z_p(j) = 0 \quad \text{for} \quad p \geq n$$

$$y(i) = \sum_{p=1}^{n+d} y_p(i) r^{-(p-d)}, \qquad y_p = 0 \quad \text{for} \quad p < d \qquad (4)$$

where r is the radix of the representation and n is the number of digits. Digits belong to a redundant set $\{-\rho, \ldots, \rho\}$, with $r/2 \leq \rho \leq r-1$. To obtain all the digits of the output, $n+d$ iterations must be performed. To simplify the notation we consider that input data and output coefficients are fractional numbers.

On–line arithmetic has an iterative nature. At step p, the on–line result obtained corresponds to the p most–significant digits of the desired result. In general, an on–line algorithm is specified recursively in terms of an internal variable or residual, w, and the on–line representation of operands and results, in such a way that iteration p uses a digit of each input data, $z_p(j)$, and the residual $w(i)[p-1]$, to produce the next residual $w(i)[p]$ and an output digit, $y_{p-d}(i)$.

The residual $w(i)[p]$ is defined as the difference between the scaled accumulation, computed with the p most significant digits of the inputs, and the output computed up to iteration $p-1$ [2] [5]

$$w(i)[p] = r^p \left(r^{-d} \left(\sum_{j=0}^{3} c(i,j) z(j)[p] \right) - y(i)[p-1] \right) \qquad (5)$$

where $z(j)[p]$ and $y(i)[p]$ are the representation of $z(j)$ and $y(i)$ with the p most significant digits. Then

$$w(i)[p] = r(w(i)[p-1] - y_{p-d-1}(i)) + r^{-d} \sum_{j=0}^{3} c(i,j) z_p(j) \qquad (6)$$

The operations involved in the recurrence are additions, multiplication by a single radix–r digit and shift. Because of addition, to achieve a fast implementation the residual is represented in a redundant format (carry save or signed digit). We select a carry–save form.

3.1. Selection function and on–line delay

The on–line delay depends on the radix, the redundancy ρ, the representation of the residual and the selection function for $y_p(i)$ [6]. Considering a carry–save representation for the residual and that the output digit $y_{p-d}(i)$ is obtained by rounding the carry–save form of the residual and selecting its integer part, the residual is bounded as follows,

$$-0.5 \leq (w(i)[p] - y_p(i)) \leq 0.5 + 2^{-t} \qquad (7)$$

where t bits of the sum and carry words of $w(i)[p]$ are assimilated to obtain $y_p(i)$. To evaluate the on–line delay, we calculate the maximum and minimum value of $w(i)[p]$ in equation (6) and substitute it in equation (7). First, we define

$$s = \max \left\{ \sum_{j=0}^{3} |c(i,j)| \quad 0 \leq i \leq 3 \right\} = 3.9 \qquad (8)$$

The maximum value of $w(i)[p]$ occurs when $w(i)[p-1]$ is maximum and the digits of the output and the inputs are ρ. With these conditions,

$$0.5 + 2^{-t} + \rho_{out} \geq r(0.5 + 2^{-t}) + s\rho_{inp} r^{-d} \qquad (9)$$

where ρ_{inp} and ρ_{out} are the digits sets of the input and output. In a similar way for the minimum

$$-0.5 - \rho_{out} \leq -0.5r - s\rho_{inp} r^{-d} \qquad (10)$$

Then, from equations (9) and (10),

$$d \geq \left\lceil \log_r \left(\frac{s\rho_{inp}}{\rho_{out} - (r-1)(0.5 + 2^{-t})} \right) \right\rceil \qquad (11)$$

As said before, a radix 4 ($r=4$) sign–digit format with $\rho_{inp} = 2$ is used. Then, to obtain a minimum delay and a fast recurrence, we chose

$$\rho_{out} = 3, \qquad t = 2 \qquad (12)$$

resulting in a on–line delay $d = 2$.

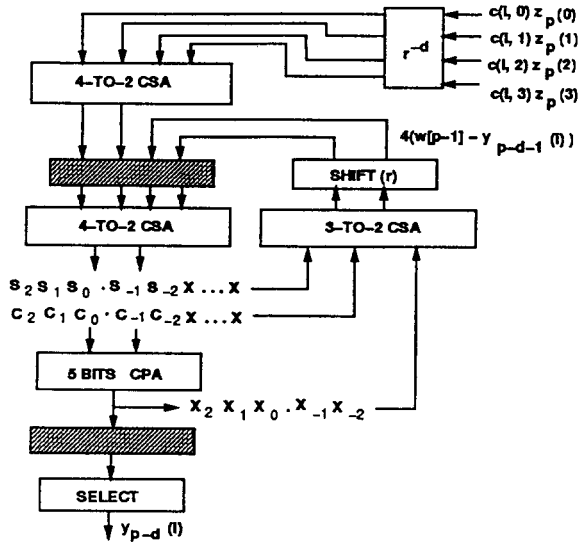

Figure 3: Block diagram of the recurrence

3.2. Implementation of the recurrence

Figure 3 shows a block diagram of the recurrence. We use 14 bits for the representation of the cosine coefficients. To represent the multiplication of the cosine times a input digit we add 3 integer bits. To perform the shift 4^{-d} we add another 4 bits. Then, the internal wordlength is 21 bits.

The clock period is limited by the delay of a 4–to–2 CSA, the CPA for the selection of the output digit and the 3–to–2 CSA for the subtraction of the output digit from the residual. As done in [6], to reduce the delay the CSA for the subtraction between the residual and the output digit is eliminated. This is done by expressing the term $4(w(i)[p-1] - y_{p-1}(i))$ of the residual as

$$\begin{array}{lccccc} \text{sum} & = & X_{-1} & X_{-1} & X_{-2} & .XXX\ldots X00 \\ \text{carry} & = & 0 & 0 & 0 & .XXX\ldots X00 \end{array} \quad (13)$$

where $X_2X_1X_0.X_{-1}X_{-2}$ is the result of the assimilation of the 5 most significant bits of $w(i)[p-1]$. Now, the critical path is formed only by the 4–to–2 CSA and the CPA.

A further improvement is to remove the CPA from the critical path. To do this, we express bits $X_{-1}X_{-1}X_{-2}$ of the output of the CPA as a function of the outputs of the 4–to–2 CSA, S and C. Figure 4 shows the resulting implementation. We denote the integer bits and 2 fractional bits of the output of the 4–to–2 CSA as,

$$S = S_2S_1S_0.S_{-1}S_{-2} \qquad C = C_2C_1C_0.C_{-1}C_{-2}$$

Since $X_2X_1X_0.X_{-1}X_{-2}$ is the output of the CPA,

$$X_{-2} = S_{-2} \oplus C_{-2} \qquad X_{-1} = S_{-1} \oplus C_{-1} \oplus S_{-2}C_{-2} \quad (14)$$

Moreover, the 4–to–2 CSA adds the S and C words of the residual with A and B. The operation performed is

$$\begin{array}{llll} A_2 & A_1 & A_0 & .\ X\ldots X\ X\ X \\ B_2 & B_1 & B_0 & .\ X\ldots X\ X\ X \\ X_{-1} & X_{-1} & X_{-2} & .\ X\ldots X\ 0\ 0 \\ 0 & 0 & 0 & .\ X\ldots X\ 0\ 0 \\ \hline S_2 & S_1 & S_0 & . \\ C_2 & C_1 & X & . \end{array}$$

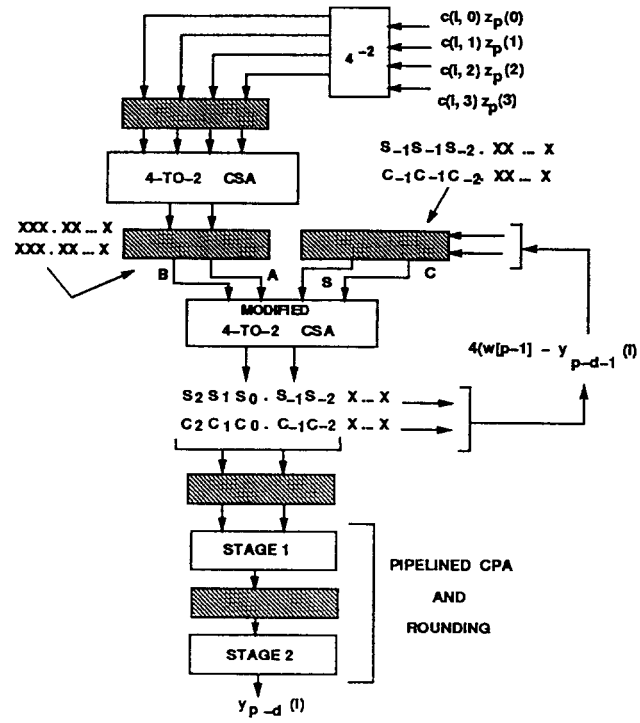

Figure 4: Implementation of the recurrence

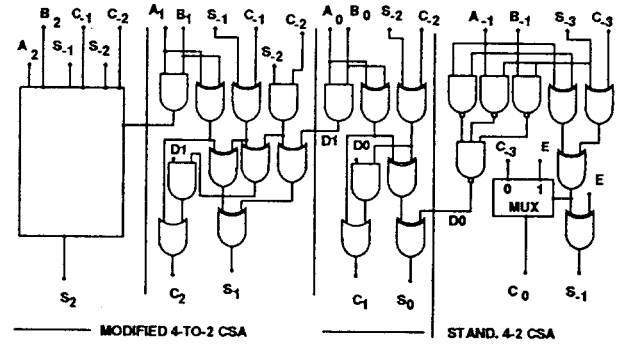

Figure 5: Modified 4–to–2 CSA

Then, as shown in figure 5, the three most significant bits of the 4–to–2 CSA are modified according with equation (14).

Since the CPA has been removed from the critical path, this path is formed only by a 4–to–2 CSA. The CPA outside of the recurrence is divided into two stages to adapt to the cycle time.

4. TIMING ANALYSIS

Figure 6a shows the timing of the on–line 8–point 1–D DCT. Each two cycles the elements of the carry save sequences are obtained and recoded to radix–4 signed digit format. In 7 cycles the sequence is stored in the SR registers and to discharge the SR registers 6 clock cycles are needed, one clock cycle per digit. The most significant digit of the transformed sequence is available 7 cycles later, due to the on–line delay, $d = 2$, and the pipelining of the MAC

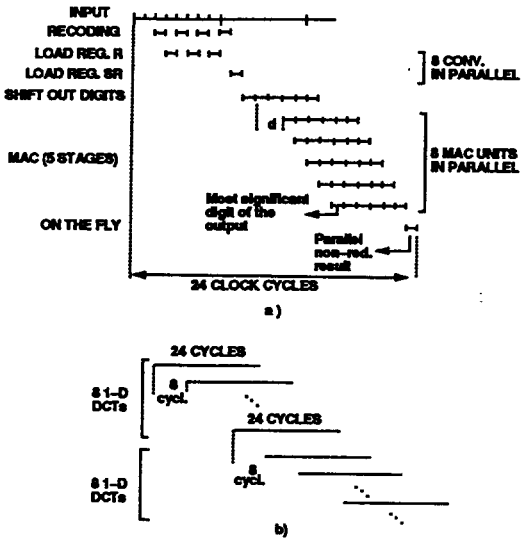

Figure 6: Timing of the 2-D DCT

unit. The transformed sequence is converted to a parallel non-redundant representation in the on-the-fly unit in 6 cycles, starting once the most significant digit of the redundant output has been obtained. This unit performs the conversion as the digits are produced and does not require carry propagation. The latency of the on-line 1-D DCT is 24 cycles.

Figure 6b shows the overlapping in the computation of the 8-point 1-D DCT over different input sequences. The computation of a new DCT starts each 8 cycles, without loss of cycles.

5. EVALUATION

In this section we compare the architecture we have developed with other architectures that implement the 2-D DCT. The comparison is based in the clock frecuency and silicon area.

The critical path of the architecture is composed of one 4-to-2 CSA and one register. This is the same as the critical path of implementations with distributed arithmetic and CS accumulation [7] and parallel implementations with CS arithmetic [8] and lower than other type of implementations.

The area of the on-line implementation is smaller than the area of the parallel implementations, because each multiplier is replaced by two 4-to-2 CSAs and one 5-bit CPA. Moreover, the interconnections among elements is digit serial, resulting in a lower routing overhead than bit parallel interconnections.

Now, we compare the area of the on-line and distributed arithmetic implementations. Table 1 shows the typical components. In distributed arithmetic, to maintain a continuous data flow, two adjacent bits of each input must be processed each cycle. But it is necessary to duplicate the ROM memory where partial products are stored. Then, each MAC unit needs two 4-to-2 CSAs and four 16-words ROM memories. This results in the same number of adders of the on-line implementation and a larger ROM memory.

In the on-line architecture, we need only two 16-words ROM memory modules to store the cosine coefficients, as the two 1-D DCTs can share the same ROM memory.

COMP.	ON LINE	DISTR.
RAM	2 mod. 64 words	2 mod. 64 words
ROM	2 mod. 16 words	64 mod. 16 words
4-2 CSA	32 of 21 bits	32 of 18 bits
CPA	16 of 5 bits	2 of 16 bits
CONV.	4	2
MUX	16x17 mux 3:1 1X12 mux 8:1	4x16 mux 8:1
CODERS	CSA to SDA	

Table 1: Comparison

Other components of the distributed arithmetic architecture are CPAs to convert from CS to non-redundant representation at the output of each 1-D DCT and parallel-to-serial converter at the input. This results in a higher silicon area than in the implementation with on-line arithmetic.

6. REFERENCES

[1] A. Artieri, S. Kritter, F. Jutand and N. Demassieux, "A One Chip VLSI for Real Time Two Dimensional Discrete Cosine Transform", Proc. ISCAS'88, pp. 710-704, 1988.

[2] R.H. Brackert, M.D. Ercegovac and A.N. Willson, "Design of an On-Line Multiply-Add Module for recursive Digital Filters", 9^{th} Symp. Computer Arithmetic, pp. 34-41, 1989.

[3] M.D. Ercegovac, "On-Line Arithmetic. An Overview", Proc. SPIE, vol. 495, Real Time Signal Processing VII, pp. 86-93, 1984.

[4] M. Ercegovac and T. Lang. "Division and Square Root: Digit-Recurrence, Algorithms and Implementations". Kluwer Academic Pub., 1994.

[5] M.D. Ercegovac and T. Lang, "Fast Arithmetic for Recursive Computation", VLSI Signal Processing V, IEEE Press, pp. 14-28, 1992.

[6] J.S. Fernando, "Design Alternatives for Recursive Digital Filters Using On-Line Arithmetic", Ph.D. Thesis, University of California at Los Angeles, 1993.

[7] U. Sjöström, I. Defilippis. M. Ansorge and F. Pellandini, "Discrete Cosine Transform Chip for Real-Time Video Applications", Proc. ISCAS'90, pp. 1620-1623, 1990.

[8] U. Totzek, F. Matthiesen, S. Wohlleben and T.G. Noll, "CMOS VLSI Implementation of the 2-D DCT with Linear Processor Arrays", Proc. ICASSP'90, pp. 937-940, 1990.

[9] S.I. Uramoto, Y. Inove, A. Tabakake, J. Takeda, Y. Yamashita, H. Terane and M. Yoshimoto, "A 100 MHz 2-D Discrete Cosine Transform Core Processor", IEEE J. Solid State Circuits, vol. 27, no. 4, pp. 492-499, 1992.