Journal of VLSI Signal Processing 45, 161–175, 2006
© 2006 Springer Science + Business Media, LLC. Manufactured in The Netherlands.
DOI: 10.1007/s11265-006-9764-7

# High Throughput Parallel-Pipeline 2-D DCT/IDCT Processor Chip

G. A. RUIZ, J. A. MICHELL AND A. BURÓN
*Departamento de Electrónica y Computadores, Facultad de Ciencias, Universidad de Cantabria, Avda. de Los Castros s/n, 39005, Santander, Spain*

*Received: 27 June 2005; Revised: 27 June 2005; Accepted: 2 May 2006*

**Abstract.** This paper presents a 2-D DCT/IDCT processor chip for high data rate image processing and video coding. It uses a fully pipelined row–column decomposition method based on two 1-D DCT processors and a transpose buffer based on D-type flip-flops with a double serial input/output data-flow. The proposed architecture allows the main processing elements and arithmetic units to operate in parallel at half the frequency of the data input rate. The main characteristics are: high throughput, parallel processing, reduced internal storage, and maximum efficiency in computational elements. The processor has been implemented using standard cell design methodology in 0.35 μm CMOS technology. It measures 6.25 mm$^2$ (the core is 3 mm$^2$) and contains a total of 11.7 k gates. The maximum frequency is 300 MHz with a latency of 172 cycles for 2-D DCT and 178 cycles for 2-D IDCT. The computing time of a block is close to 580 ns. It has been designed to meets the demands of IEEE Std. 1,180–1,990 used in different video codecs. The good performance in the computing speed and hardware cost indicate that this processor is suitable for HDTV applications.

**Keywords:** discrete cosine transform (DCT), inverse discrete cosine transform (IDCT), image compression, row column decomposition, parallel pipelined architectures, very large scale integration (VLSI)

## 1. Introduction

The Discrete Cosine Transform (DCT) is widely considered to provide a near optimal performance for transform coding and image compression, because it offers energy compaction, orthogonal separability and fast algorithms [1]. Thus, the DCT has been applied for most of recent still picture and moving picture international standards for sequential codecs [2, 3] as JPEG, MPEG, H.261 and H.263, as well as in high-definition television (HDTV) systems. The computation complexity requirements in many real-time applications often lead to the use of efficient dedicated hardware (ASIC's) operating at high speed with an acceptable cost in area.

Since the introduction of the DCT in the 1970s, a considerable amount of research has been performed on algorithms, architectures and processor design for computing of DCT. In the literature, there are many VLSI implementations proposed for DCT and its inverse (IDCT) which, to a greater or lesser extent, search for some of the following characteristics: low-cost area [4–9], regularity to reduce the design effort [10, 11], high throughput [4, 5, 9, 11–13] and low power [4, 14, 15]. Different approaches have been proposed to implement the 2-D DCT/IDCT: row–column decomposition method, the direct method and

This work was supported by the Spanish Ministry of Science and Technology (TIC2000-1289).

other minority alternatives based on transforms (as DFT [16] and DHT [17]), CORDIC algorithms [18] and systolic array implementations [19]. The row–column decomposition method uses the separability property of 2-D DCT to be broken into two sequential 1-D DCT, one along the row-wise block and the second along the column-wise block of previous row-wise processed blocks, which are stored in a transpose memory [4, 6, 8, 9, 11–15, 20, 21]. This method allows a 2-D DCT to be computed using fast algorithms and hardware developed for 1-D DCT. In some implementations, a single multiplexer 1-D DCT processor is used to perform both operations with the corresponding saving in hardware [8–10, 13, 20]. Roughly 90% of the survey implementations follow the row–column decomposition method because its regularity is highly suitable for VLSI implementation. The direct method requires fewer computations, but it incurs the irregularity [5, 7, 10]. However, the feature of low-computation complexity is still attractive and some regular structures have been researched recently.

This paper describes the architecture of an 8×8 2-D DCT/IDCT processor chip with a high throughput and a cost-effective architecture [22]. The 2D DCT/IDCT is calculated using the separability property, so that its architecture is made up of two 1-D processors and a transpose buffer (TB) as intermediate memory. This transpose buffer presents a regular structure based on D-type flip-flops with a double serial input/output data-flow highly suitable for pipeline architectures. The processor has been designed searching for high throughput, reduced hardware, parallel and pipeline architecture, and a maximum efficiency in all arithmetic elements. This architecture allows the processing elements and arithmetic units to work in parallel at half the frequency of the data input rate, except for the normalisation of the transform which is carried out in a multiplier operating at maximum frequency. Moreover, it has been verified that the precision analysis of the proposed processor meets the demands of IEEE Std. 1,180–1,990 [23] used in video codecs ITU-T H.261 [24], ITU-T H.263 [25] y ITU-T H.261+ [26]. The processor has been conceived using a standard cell design methodology and manufactured in a 0.35-µm CMOS CSD 3M/2P 3.3 V process (http://www.asic.austriamicrosystems.com). It has an area of 6.25 mm$^2$ (the core is 3 mm$^2$) and contains a total of 11.7 k gates, 5.8 k gates of which are flip-flops. A data input rate frequency of 300 MHz has been established with a latency of 172 cycles for 2-D DCT and 178 cycles for 2-D IDCT. The computing time of a block is close to 580 ns. This good performance in the computing speed as well as hardware cost, indicate that the proposed design is compact and suitable for HDTV applications.

The paper is organized as follows: Section 2 presents the principles and algorithm used to implement the 2-D DCT/IDCT. Section 3 addresses the architectural design and circuit design features of the basic processing elements. A description of a block diagram of the 2-D DCT/IDCT processor is presented in Section 4. Section 5 describes the main arithmetic elements and, finally, chip characteristics and comparisons with other previous DCT/IDCT processors are described in Section 6.

## 2. Two-Dimensional 8×8 DCT/IDCT

La 8×8 DCT transforms a block of the space domain, $\{x(n,m)\}_{n,m=0}^{7}$ , into its DCT domain components, $\{X(k,l)\}_{k,l=0}^{7}$ , according to the following equation:

$$X(k,l)=\frac{c(k)}{2}\frac{c(l)}{2}\sum_{n=0}^{7}\sum_{m=0}^{7}x(n,m)C(n,k)C(m,l), \quad \forall k,l=0,1,2,\ldots,7 \tag{1}$$

where $c(0)=1/\sqrt{2}$ , $c(k)$=1 for $k$>0, $C(i,j)=\cos\left(\frac{2i+1}{16}\cdot j\pi\right)$

The IDCT is defined by:

$$x(n,m)=\sum_{k=0}^{7}\sum_{l=0}^{7}\frac{c(k)}{2}\frac{c(l)}{2}X(k,l)C(n,k)C(m,l), \quad k,l=0,1,2,\ldots,7 \tag{2}$$

In matrix notation, let S$_{R8}$ the eight-point DCT matrix with rows reordered according to the sequence (0,4,2,6,1,5,3,7):

$$S_{R8}=\frac{1}{2}\begin{bmatrix} C_0 & C_0 & C_0 & C_0 & C_0 & C_0 & C_0 & C_0 \\ C_4 & -C_4 & -C_4 & C_4 & C_4 & -C_4 & -C_4 & C_4 \\ C_2 & C_6 & -C_6 & -C_2 & -C_2 & -C_6 & C_6 & C_2 \\ C_6 & -C_2 & C_2 & -C_6 & -C_6 & C_2 & -C_2 & C_6 \\ C_1 & C_3 & C_5 & C_7 & -C_7 & -C_5 & -C_3 & -C_1 \\ C_5 & -C_1 & C_7 & C_3 & -C_3 & -C_7 & C_1 & -C_5 \\ C_3 & -C_7 & -C_1 & -C_5 & C_5 & C_1 & C_7 & -C_3 \\ C_7 & -C_5 & C_3 & -C_1 & C_1 & -C_3 & C_5 & -C_7 \end{bmatrix} \tag{3}$$

where $C_0 = \frac{1}{\sqrt{2}}; C_i = \cos\frac{i\cdot\pi}{16}, i = 1, 2, \ldots, 7$ . Taking into account the properties of the cosine, the $C_1$, $C_3$, $C_5$ and $C_7$ elements of $S_{R8}$ can be expressed as:

$$\begin{aligned} C_1 &= C_4\cdot(C_3 + C_5); C_3 = C_4\cdot(C_1 + C_7); \\ C_5 &= C_4\cdot(C_1 - C_7); C_7 = C_4\cdot(C_3 - C_5); \end{aligned} \tag{4}$$

The elements of columns 2, 3, 6 and 7 of $S_{R8}$ can be decomposed by applying Eq. (4) in the following way:

$$\begin{aligned} S_{R8} &= P_{R8}\cdot J_{R8} = P_{R8}\cdot\begin{bmatrix} J_{RE4} & 0 \\ 0 & J_{RO4} \end{bmatrix}\cdot Q_{R8} \\ &= P_{R8}\cdot\begin{bmatrix} J_{SE4}Q_{R4} & 0 \\ 0 & J_{04B}J_{04C}J_{04D} \end{bmatrix} Q_{R8} \end{aligned} \tag{5}$$

where

$$P_{R8} = \frac{1}{2}\cdot\text{Diagonal}(C_0, C_4, C_2, C_2, C_1, C_5, C_5, C_1) \tag{6}$$

$$J_{SE4} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & T_2 & 1 \\ 0 & 0 & -1 & T_2 \end{bmatrix}, \quad Q_{R4} = \begin{bmatrix} I_2 & I_{D2} \\ I_{D2} & -I_2 \end{bmatrix} \tag{7}$$

$$J_{04B} = \begin{bmatrix} T_1 & 0 & 0 & 1 \\ 0 & T_5 & 1 & 0 \\ 0 & -1 & T_5 & 0 \\ -1 & 0 & 0 & T_1 \end{bmatrix}, \tag{8}$$

$$J_{04C} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix},$$

$$J_{04D} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -C_4 & C_4 & 0 \\ 0 & C_4 & C_4 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$Q_{R8} = \begin{bmatrix} I_4 & I_{D4} \\ I_{D4} & -I_4 \end{bmatrix} \tag{9}$$

where $I_{D4}$ is the matrix resulting from permuting the $I_4$ rows according to the ordering sequence (3,2,1,0), $I_{D2}$ ordering according to (1,0), $T_1=C_7/C_1$, $T_2=C_6/C_2$ and $T_5=C_3/C_5$. Figure 1 shows the flow graph of Eq. (5), frequently referenced as Chen's fast algorithm [27] which has been used in some implementations [12, 13]. It involves multiplication by four different constants and the outputs are scaled which require a further multiplication. Moreover, $J_{R8}$ is made up by two 4×4 sub-matrices which can be computed in parallel.

Since $S_{R8}$, $Q_{R8}$ and $P_{R8}$ are orthogonal, after Eqs. (3) and (5) we get:

$$S_{R8}^{-1} = J_{R8}^t P_{R8} = Q_{R8}\begin{bmatrix} Q_{R4}J_{SE4}^t & 0 \\ 0 & J_{04D}J_{04C}J_{04B}^t \end{bmatrix} P_{R8} \tag{10}$$

The flow chart of inverse DCT is shown in Fig. 2. In the compute of this algorithm, the role of the inputs and outputs are reversed and the $J_{RE4}$ and $J_{RO4}$ must be replaced by $J_{RE4}^t$ and $J_{RO4}^t$, respectively.

The 8×8 2-D DCT can be expressed on the basis of the nuclei of the $S_{R8}$ matrix transform as:

$$X_R = S_{R8}\cdot x\cdot S_{R8}^t = K_8\bullet\left(J_{R8}\cdot x\cdot J_{R8}^t\right) \tag{11}$$

where (●) represent the Hadamard product and $K_8$ is the normalization matrix defined as

$$K_8 = \frac{1}{4}\begin{bmatrix} C_0C_0 & C_0C_4 & C_0C_2 & C_0C_2 & C_0C_1 & C_0C_5 & C_0C_5 & C_0C_1 \\ C_4C_0 & C_4C_4 & C_4C_2 & C_4C_2 & C_4C_1 & C_4C_5 & C_4C_5 & C_4C_1 \\ C_2C_0 & C_2C_4 & C_2C_2 & C_2C_2 & C_2C_1 & C_2C_5 & C_2C_5 & C_2C_1 \\ C_2C_0 & C_2C_4 & C_2C_2 & C_2C_2 & C_2C_1 & C_2C_5 & C_2C_5 & C_2C_1 \\ C_1C_0 & C_1C_4 & C_1C_2 & C_1C_2 & C_1C_1 & C_1C_5 & C_1C_5 & C_1C_1 \\ C_5C_0 & C_5C_4 & C_5C_2 & C_5C_2 & C_5C_1 & C_5C_5 & C_5C_5 & C_5C_1 \\ C_5C_0 & C_5C_4 & C_5C_2 & C_5C_2 & C_5C_1 & C_5C_5 & C_5C_5 & C_5C_1 \\ C_1C_0 & C_1C_4 & C_1C_2 & C_1C_2 & C_1C_1 & C_1C_5 & C_1C_5 & C_1C_1 \end{bmatrix} \tag{12}$$

Similarly, the 8×8 IDCT is obtained as:

$$x = S_{R8}^t\cdot X_R\cdot S_{R8} = J_{R8}^t\cdot(K_8\bullet X_R)\cdot J_{R8} \tag{13}$$

Eqs. (11) and (13) allow the multiplication by $K_8$ coefficients to be done at output for forward DCT or at input for IDCT, reducing the number of inner multiplications. This scheme is very attractive when this transform is incorporated into the decoding

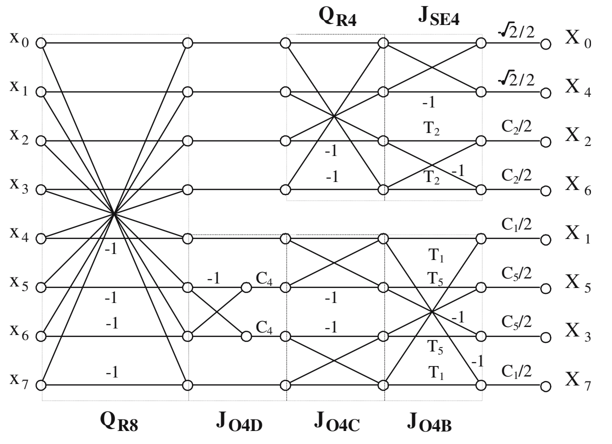

*Figure 1.* Signal flow graph of 1-D forward DCT.

process in an adaptive transform coding system. Therefore, the quantization process at the receiver or at the transmitter can include this normalization in the decoding/encoding lookup table. In this case, the normalization can be completely eliminated and thus a significant reduction in hardware is obtained.

## 3. Architecture of $J_{R8}/J^t_{R8}$ Processor

Figure 3 shows the architecture of the $J_{R8}/J^t_{R8}$ processor designed to implement the matrix decompositions defined in Eqs. (5) and (10) respectively. It is made up of a double-input $Q_{R8}$ processor, and the processors $Q_{R4} - J_{SE4}/J^t_{SE4}$ and $J_{O4D} - J_{O4C} - J_{04B}/J^t_{04B}$ operating in parallel. The configuration of forward DCT or IDCT of the architecture is performed by means of the control of the multiplexers and of the operation of processors $J_{SE4}/J^t_{SE4}$ and $J_{04B}/J^t_{04B}$ . One important characteristic is that the input data $I_E$ and $I_O$ are processed in parallel and thus the output data $O_E$ and $O_O$ are also generated in parallel. In this way, the operation frequency of the $J_{R8}/J^t_{R8}$ processor is reduced to $f_s/2$, where $f_s$ is the input data sampling frequency. Figure 4 shows the architecture of each of the basic processors specified in Fig. 3 which are derived from Eqs. (7)–(10). The control is very simple and is carried out using four signals: Clk1, main clock at frequency $f_s$, Clk2, internal clock at frequency $f_s/2$, and the multiplexer selection signals $S_1$ at frequency $f_s/4$ and $S_2$ at

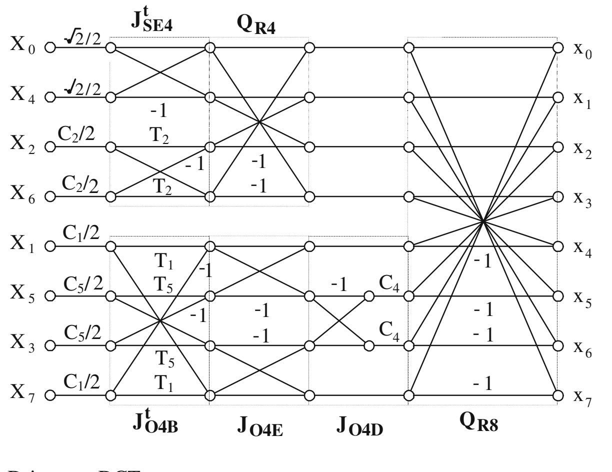

*Figure 2.* Signal flow graph of 1-D inverse DCT.

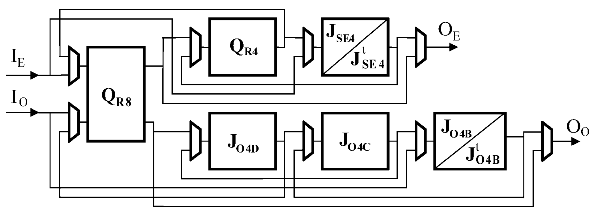

*Figure 3.* Architecture of $J_{R8}/J^t_{R8}$ processor.

frequency $f_s/8$. All of the basic processors have been conceived to work with four input data introduced in series and whose four output data are thus also in series, these outputs being compatible with the next processor. The Forward/Inverse (F/I) signal modifies the operation of the processor to perform the transformation or its inverse. These basic processors are made up of shift registers (S-R), multiplexers (MUX), carry incrementer adders/subtracters and hardwired multipliers and they have been designed aiming at an efficiency of 100% in the arithmetic elements in most cases.

## 4. 8×8 2-D DCT/IDCT

The 2-D 8×8 DCT/IDCT is implemented by the row–column decomposition technique according to Eq. (11) for DCT and to Eq. (13) for IDCT. Figure 5 shows the block diagram of the proposed architecture composed of two 1-D processors ($J_{R8}/J^t_{R8}$), a transpose buffer (TB) for storing the intermediate data, one down-sampling (D-S) unit and another up-sampling (U-S) unit, and a multiplier for performing the normalization of the transform according to matrix $K_8$ described in Eq. (12). One of the main characteristics of this architecture is that it operates at half the frequency of the input data rate ($f_s/2$), except for normalization with $K_8$ performed at the output for the forward DCT and at the input for the IDCT. This multi-rate operation involves D-S and U-S modules: D-S multiplexes in parallel and at frequency $f_s/2$ the input data, while the U-S performs the opposite process.

### *4.1. Architecture of TB*

The 8×8 intermediate data generated by the first 1-D DCT processor has to be stored and transposed in the TB before the second 1-D DCT is performed. This TB allows simultaneous read and write operations between the two processors while performing matrix transposition. To achieve this, the data are read out of the memory column-wise if the previous intermediate data were written into the memory row-wise, and vice versa.

The TB based on D-type flip-flops has been found to be adequate for pipeline architectures, unlike other proposed architectures based on RAM memories. The schema of this circuit is shown in Fig. 6a. It has a regular serial input/output structure made up of eight 16-bit shift-registers and multiplexers. The control signals are: R/C, which selects the input of shift-registers to store data in row-wise (R/C=0) or column-wise (R/C=1), $W_j$ (j=1 to 8), which selects the jth shift-register to make a write operation, and $R_k$ (k=1 to 3), which selects the jth shift-register specified by $W_j$ to read out the data. The write & read operation is simultaneously made with two serial input data, $\{I_3I_2I_1I_0\}$ and $\{I_7I_6I_5I_4\}$, and generates two serial output data in parallel, $\{O_3O_2O_1O_0\}$ and $\{O_7O_6O_5O_4\}$. The control signals allow the data to be stored alternatively row-wise and column-wise in order to perform data transposing and to avoid loss of data. For the sake of clarity, Fig. 6b and c show the configuration of shift-registers and the arrangement of stored input data when a writing process in column-wise or in row-wise is made. Both storing data processes are easy to configure from the state of the TB control signals. Figure 6d shows the timing diagram of the state of these variables as a function of the number of Clk2 clock cycles. First, the input data are stored column-wise (in Fig. 6b, they are filled downward) to complete each of the vertical halves which the memory is divided. To do this, $W_j$ recurs sequentially to each of the shift-registers so that only four Clk2 cycles are required to store a whole row. At the same time as the outputs, the data previously stored column-wise are read out. To do

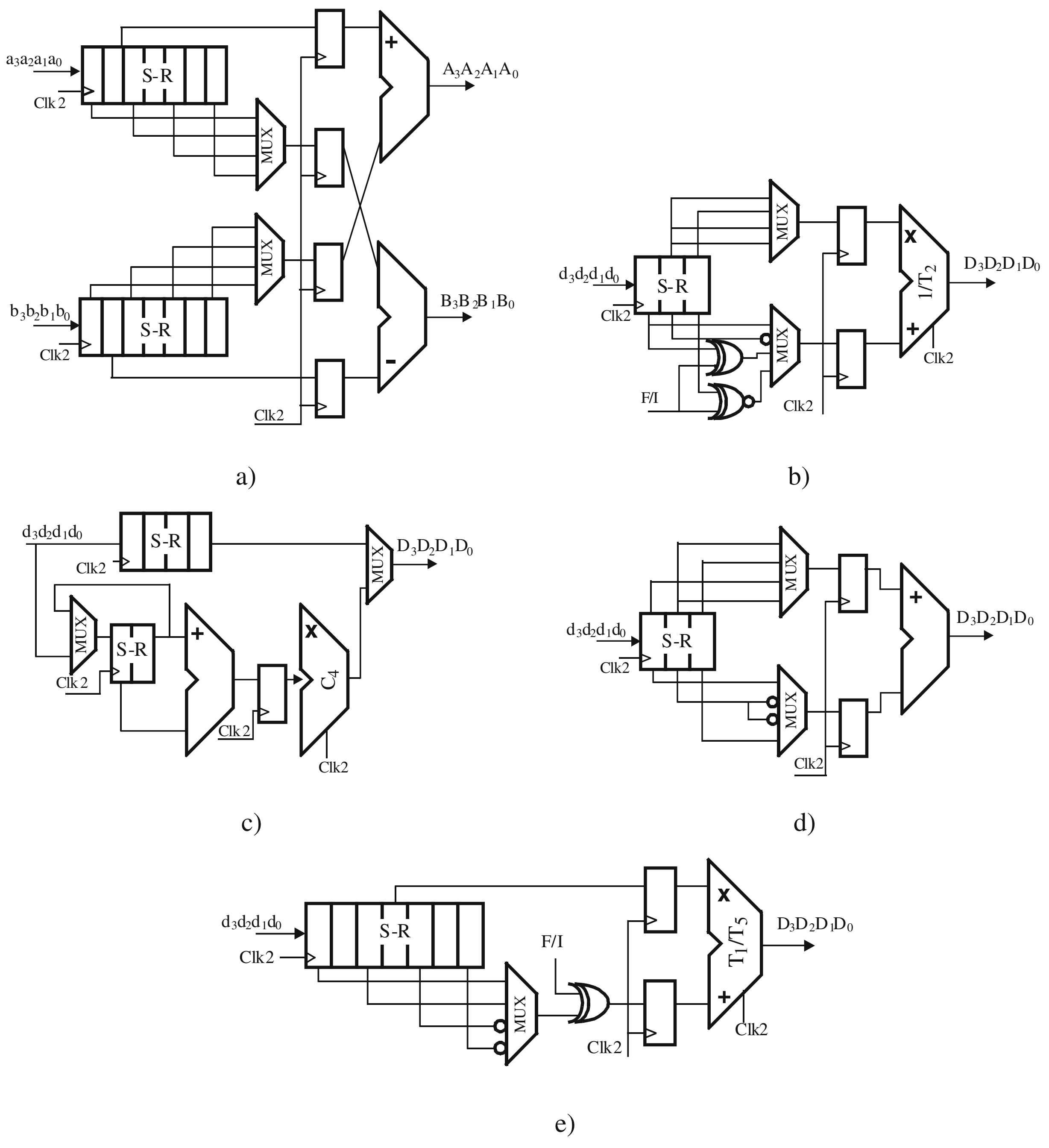

*Figure 4*. Architecture of basic processors: **a** $Q_{R8}$, **b** $J_{SE4}\big/J^t_{SE4}$ , **c** $J_{O4D}$, **d** $Q_{R4}$ and $J_{O4D}$, and **e** $J_{O4B}\big/J^t_{O4B}$.

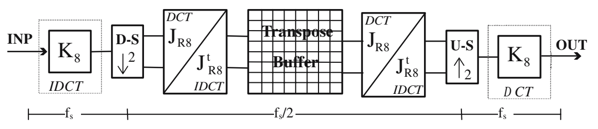

*Figure 5*. Block diagram of 2-D 8×8 DCT/IDCT processor.

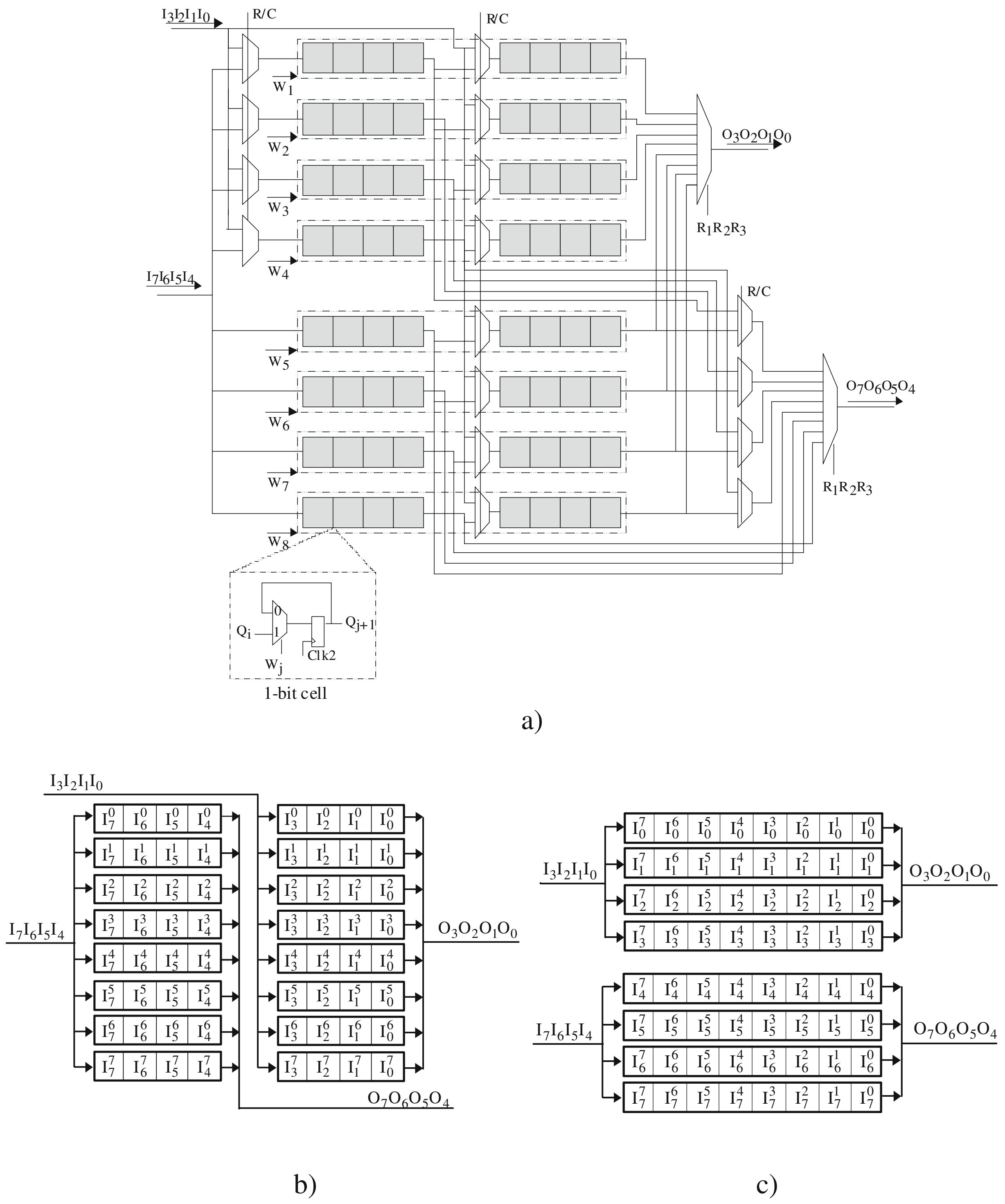

*Figure 6.* TB circuit: **a** Schematic, **b** writing data in row-wise, **c** writing data in column-wise, and **d** timing diagram in term of number of Clk2 cycles.

|  | Clk2 | $W_1$ | $W_2$ | $W_3$ | $W_4$ | $W_5$ | $W_6$ | $W_7$ | $W_8$ | $R_1$ | $R_2$ | $R_3$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **(R/C=0) Writing in row-wise** | 4x{ | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | 4x{ | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
|  | 4 x{ | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
|  | 4 x{ | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
|  | 4 x{ | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
|  | 4 x{ | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |
|  | 4 x{ | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 |
|  | 4 x{ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| **(R/C=1) Writing in column-wise** | 8x{ | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
|  |  | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
|  |  | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
|  |  | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 |

d)

*Figure 6* (Continued).

this, the writing process, performed using $W_j$, and the reading process, performed using $R_k$, are synchronized. In the next block, the input data are stored column-wise (in Fig. 6c, they are filled rightward) and the outputs now read out the data previously stored row-wise. In this case, the data are written sequentially in each of the horizontal halves into which the memory is divided, repeating eight times the sequence specified in Fig. 5d. As a result, the outputs are continuously transposed in parallel and 32 Clk2 cycles are required to fill up all of the memory.

## 4.2. Pipeline Scheme

The DCT/IDCT processor uses a pipelining scheme to shorten the cycle time and perform real-time processing for applications with high pixel rates. The pipeline registers are inserted in the critical path to improve the operation speed with minimal overhead. Figure 7 shows the data cycle timing for calculating the two 1-D DCT/IDCT and TB transposing operations for the different blocks. The input data are fed in row-wise order at 1 pixel/clock and the output data are produced in column-wise order. The first 1-D DCT requires 45 cycles and the 1-D IDCT 67 cycles, since normalization is necessary. The second 1-D DCT requires 63 cycles and the 1-D IDCT 47 cycles. The total latency for 2-D DCT is 172 cycles and for 2-D IDCT is 178 cycles. This small difference in the number of cycles is due to the synchronization between the different processors of the circuit.

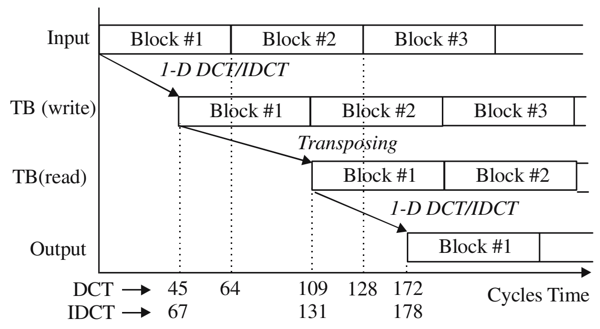

*Figure 7.* Data timing of 2-D DCT/IDCT processor.

*Table 1.* Computation precision of IDCT according to standard [23].

| | MSE (<0.02) | ME (<0.0015) | PMSE (<0.06) | PME (<0.015) |
|---|---|---|---|---|
| [−256, 255] | 0.016733 | 0.000008 | 0.0224 | 0.0028 |
| [255, −256] | 0.017011 | 0.000136 | 0.022500 | 0.0029 |
| [−300, 300] | 0.014727 | 0.000083 | 0.018300 | 0.0029 |
| [300, −300] | 0.014872 | 0.000075 | 0.018700 | 0.0024 |
| [−5, 5] | 0.011494 | 0.000209 | 0.013600 | 0.0027 |
| [5, −5] | 0.011422 | 0.000181 | 0.0144 | 0.0025 |

*MSE* Overall mean square error, *ME* overall mean error, *PMSE* peak mean square error, *PME* peak mean error

### *4.3. Accuracy Specifications*

The accuracy of the computing of the DCT is an important characteristic of this hardware. The DCT kernel components are real numbers so that truncation or rounding errors are inevitably introduced during computation. There are two inherent errors in a DCT implementation: (1) finite internal wordlength and (2) coefficient quantization error. The standards H.261, H.263 and H.263+ defined for videoconference applications, establishes the accuracy specifications in the computing of the IDCT. The fulfilment of the specifications ensures the compatibility between different implementations of the IDCT [28]. This standard has been used to define the accuracy of the data-path and wordlength of the coefficients of the processor.

Table 1 summarizes the simulation results carried out with MATLAB according to the procedure described in [23]. This specification allows the errors caused by finite wordlength in IDCT to be evaluated. Thus, the IDCT architecture must be excited with 10,000 $8\times8$ blocks of random numbers in the ranges [−5, 5], [−256, 255] and [−300, 300]. The simulations find the following minimum architectural requirements: 20-b for data-path, 12-b for coefficients wordlength and 13-b for normalization coefficients of $K_8$.

## 5. Arithmetic Circuits

The arithmetic circuits limit the speed of the processor. They have been carefully selected to find

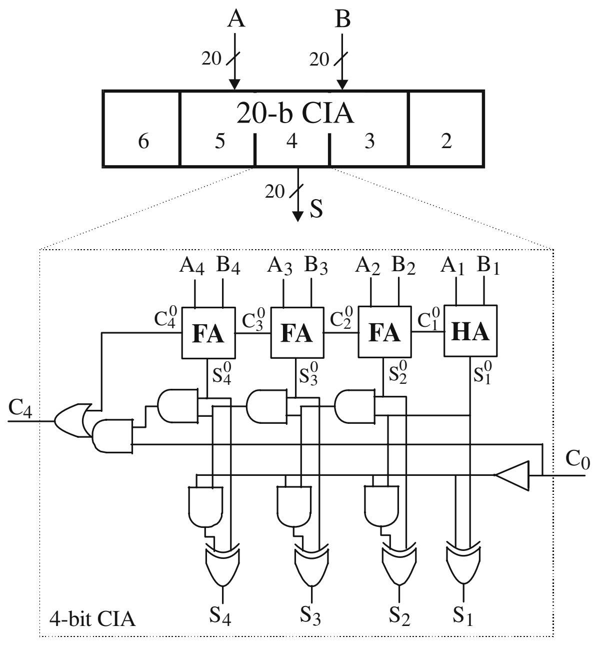

*Figure 8.* 20-b CIA detailed for 4-b.

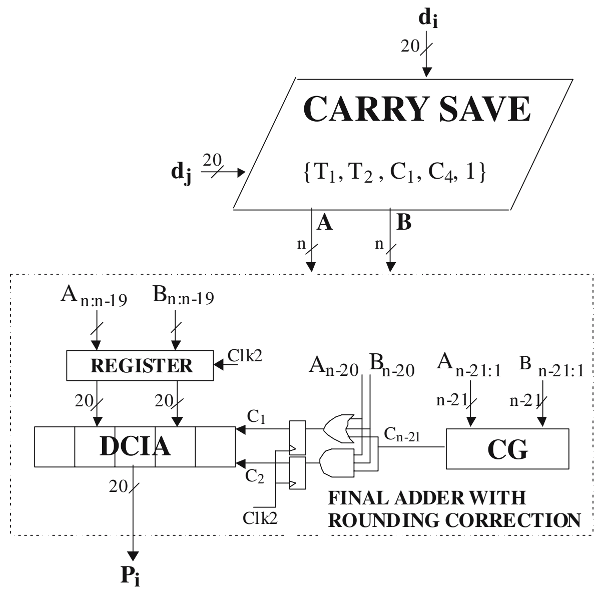

*Figure 9.* General structure of hardwired multipliers.

best compromise in area, throughput and latency. The $J_{R8}/J^t_{R8}$ processor uses a total of six 20-bit carry incrementer adders and three hardwired multipliers. In the design of these circuits, fast architectures with minimal overhead, radix representation and pipeline stages have been used to provide balanced critical paths. However, this does not mean a great effort in design since these operate at half the frequency of the input data rate. Only a fine grain pipeline architecture is required for the normalization multiplier $K_8$ since it is operating at input data rate frequency. In this section, the main arithmetic circuits are described.

### 5.1. Carry Incrementer Adder

The carry incrementer adder (CIA) has been chosen because it has an asymptotic performance with O(n) area and O($\sqrt{n}$ ) time, and provides a compromise between a ripple-carry adder (RCA) and a carry look-ahead adder. It has a short critical path at the expense of a small increase in area in comparison with RCA. The CIA is made up of an RCA divided into blocks and some additional selection logic and it is a modification of the adder presented in [29]. Figure 8 shows the 20-bit CIA built from five blocks of different lengths {6, 5, 4, 3, 2}. For the sake of clarity, a 4-bit block is shown in detail, its output being obtained from the following equations:

$$\begin{aligned} S_1 &= C_0 \oplus S_1^0 \\ S_2 &= \left(C_0 S_1^0\right) \oplus S_2^0 \\ S_3 &= \left(C_0 S_2^0 S_1^0\right) \oplus S_3^0 \\ S_4 &= \left(C_0 S_3^0 S_2^0 S_1^0\right) \oplus S_4^0 \end{aligned} \tag{14}$$

where $S_i^0$ is the RCA output and $C_0$ the input carry for this block. The output carry, which forms the input carry for the next block, is defined as:

$$C_4 = C_4^0 + S_4^0 S_3^0 S_2^0 S_1^0 C_0 \tag{15}$$

### 5.2. Hardwired Multipliers

The concept of hardwired multiplication and binary signed digit representation for fixed coefficients has been adopted to simplify the hardware complexity for realizing multiplication through a carry-saver adder scheme. The multiplication by fixed-coefficients is computed in three types of configurable multipliers which perform the following arithmetic operations: P=d$_i \cdot \{T_1$ or $T_5\}$+d$_j$, P=d$_i \cdot \{1$ or $T_2\}$+d$_j$ and P=d$_i \cdot$ C$_4$, where d$_i$ and d$_j$ are input data. These multipliers, whose general structure is shown in Fig.

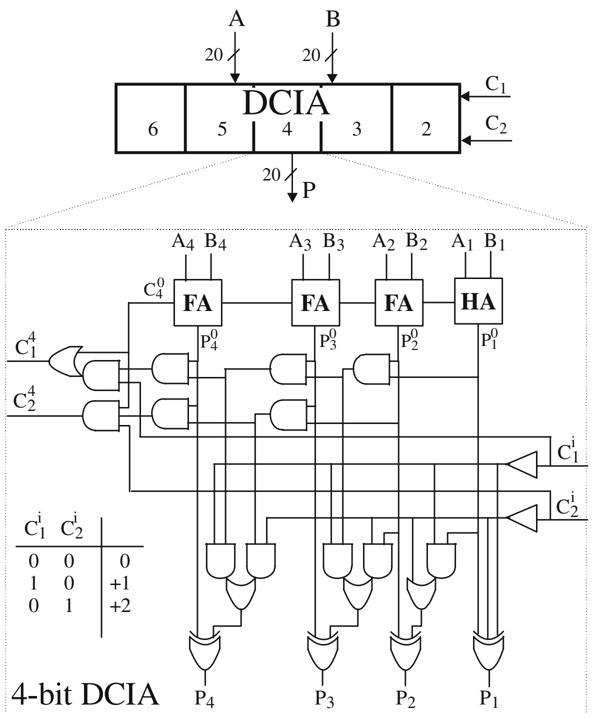

*Figure 10.* Structure of DCIA detailed for 4-bit.

9, are made up of a carry save adder tree based on 4:2 and 5:3 compressors and configured according to the type of coefficient, and a final adder with rounding correction. To limit the critical path, pipeline stages and a radix-8 encoding are used for $T_1$ and $T_5$. The fixed coefficients are expressed in the carry save structure as:

$$\begin{aligned}
T_1 &= \frac{815}{4096} = 0.198974609375 = 3{\cdot}2^{-4} + 3{\cdot}2^{-8} \\
&\qquad -2^{-12} \\
T_5 &= \frac{6130}{4096} = 1.49658203125 = 3{\cdot}2^{-1} - 2^{-8} + 2^{-11} \\
C_4 &= \frac{2896}{4096} = 0.70703125 = 2^{-1} + 2^{-3} \\
&\qquad +2^{-4} + 2^{-6} + 2^{-8} \\
T_2 &= \frac{1696}{4096} = 0.4140625 = 2^{-2} + 2^{-3} + 2^{-5} + 2^{-7}
\end{aligned} \tag{16}$$

For example, $P = x\cdot T_1 + y = (3x)\cdot 2^{-4} + (3x)\ 2^{-8} - x\cdot 2^{-12} + y$. The term 3x is precomputed by adding and shifting operations, $2\cdot x + x$, in a CIA.

These multipliers generate a larger output than the data-path of the processor. This necessarily implies the use of less significative bits to adapt the output of the multiplier to the 20-bit size of the data-path. However, it has been verified through simulation with MATLAB that if a rounding correction operation rather than a truncation operation is performed on this final adder, the size of the data-path required to verify the IEEE Std. 1,180–1,990 requirements is reduced from 22-b to 20-b. This result is important because it leads to a significant saving in the total area of the processor.

The final adder with rounding correction is made up of a carry generator (CG) with a structure of a high-speed binary carry-look ahead [30], and a 20-bit double carry incrementer adder (DCIA). The DCIA must compute the input carries, $C_1$ and $C_2$, which indicate the type of increase to be made: +0 for $C_1$=$C_2$=0,+1 for $C_1$= and $C_2$=0, and +2 for $C_1$=0 and $C_2$=1. $C_1$ y $C_2$ can easily be generated from the following expressions:

$$\begin{aligned}
C_1 &= A_{n-20} + B_{n-20} + C_{n-21} \\
C_2 &= A_{n-20}B_{n-20}C_{n-21}
\end{aligned} \tag{17}$$

where $C_{n-21}$ is the carry generated in the CG.

*Table 2.* Distribution in terms of number of gates for the different blocks in 2D DCT processor.

| Block | No. of gates |
|---|---|
| $Q_{R8}$ processor | 549 |
| $Q_{R4}$ processor | 318 |
| $J_{04C}$ processor | 320 |
| $J_{O4B}/J^t_{O4B}$ processor | 600 |
| $J_{SE4}/J^t_{SE4}$ processor | 372 |
| $J_{O4D}$ processor | 502 |
| TB | 2,563 |
| $K_8$ multiplier | 1,528 |
| Others | 2,300 |
| Total | 11,740 |

Figure 10 shows the schema of the 20-b DCIA composed of five blocks of variable length {6, 5, 4, 3, 2} detailed for the case of 4-bits. The outputs, $P_i$, of this block are given by the following expressions:

$$\begin{aligned} P_1 &= C_2^0 \oplus C_1^0 \oplus P_1^0 \\ P_2 &= \left(C_2^i + C_1^i P_1^0\right) \oplus P_2^0 \\ P_3 &= \left(C_2^i P_2^0 + C_1^i P_1^0 P_2^0\right) \oplus P_3^0 \\ P_4 &= \left(C_2^i P_2^0 P_3^0 + C_1^i P_1^0 P_2^0 P_3^0\right) \oplus P_4^0 \end{aligned} \tag{18}$$

and the output carries are defined as:

$$\begin{aligned} C_1^4 &= C_4^0 + P_1^0 P_2^0 P_3^0 P_4^0 C_1^i \\ C_2^4 &= P_2^0 P_3^0 P_4^0 C_4^0 C_2^i \end{aligned} \tag{19}$$

## 5.3. Normalization Multiplier

The Hadamard product for the normalization described by Eqs. (11) and (13) are performed in a Booth-multiplier with fine pipeline. The multiplier is

*Table 3.* Comparison between the proposed architecture and existing architectures which verify the standard [23].

| Ref. | Year | Function | Tech. CMOS | Area in mm$^2$ | Gates/Trans. | Frequency/latency | Architecture |
|---|---|---|---|---|---|---|---|
| [12] | 1992 | DCT/IDCT | 0.8 μm (FC) | 21 (core) | 102 k Tr. | 100 MHz @5 V/1.28μs | RC TRAM DA |
| [20] | 1992 | IDCT | 1 μm (GA) | 110 | 31 k gates | 40 MHz @ 3.4 V/24 cycles | MUXRC ERAM HM |
| [13] | 1995 | DCT/IDCT | 0.8 μm (FC) | 10 (core) | 67 k Tr. | 100 MHz | MUXRC TRAM HM |
| [4] | 1996 | DCT/IDCT | 0.3 μm (FC) | 4 | 120 K Tr | 150 MHz @ 0.9 V/112 cycles | RC TRAM DA |
| [14] | 1998 | IDCT | 0.5 μm (GA) | | 200 k Tr | 27 MHz @ 3.3 V/80 cycles | RC TRAM HM |
| [5] | 1999 | IDCT | 0.6 μm (SC) | 12.3 (core) | 10 k gates/78 k Tr | 100 MHz @ 3.3 V/86 cycles | DM TB DA |
| [21] | 1999 | DCT/IDCT | 0.8 μm (SC) | 33 | 15 K gates + RAM | 36 MHz | RC TRAM DA |
| [6] | 1999 | DCT/IDCT | 0.5 μm (SC) | | 10 k gates + RAM | 33 MHz | RC TRAM DA |
| [7] | 2000 | DCT/IDCT | 0.6 μm (SC) | 0.78 (core) | 1.5 k gates + RAM | 33 MHz/1,208 cycles | DM TRAM DA |
| [8] | 2001 | DCT/IDCT | 0.65 μm (SC) | | 9.2 k gates | 50 MHz | MUXRC TB DA |
| [15] | 2002 | DCT/IDCT | 0.18 μm (SC) | | 28 k gates | 33.6 MHz @ 1.6 V/227 cycles | RC TRAM DA |
| [9] | 2003 | IDCT | 0.6 μm (SC) | 21 (core) | 7.5 gates + RAM | 100 MHz | MUXRC TRAM HM |
| [11] | 2004 | DCT/IDCT | 0.25 μm (SC) | 1.5 (core) | 52 k gates | 150 MHz/64 cycles | RC RF HM |
| Ours | 2004 | DCT/IDCT | 0.35μm (SC) | 3 (core) | 11.7 k gates | 300 MHz @ 3.3 V /178 cycles | RC TB HM |

*FC* Full-custom, *SC* semi-custom, *GA* gate-array, *RC* row–column decomposition, *MUXRC* multiplexed row–column decomposition, *DM* direct method, *TRAM* transpose RAM, *ERAM* external transpose RAM, *TB* transpose, *RF* register file, *HM* hardwired multiplier, *DA* distributed arithmetic.

made up of an optimised Booth-decoder, a seven-stage pipeline carry-save structure and a pipeline final adder with rounding correction based on a DCIA and a GC. The counter selects a specific element of $K_8$ and the Booth-decoder encodes the 10 different elements of normalization which are defined in 13-b as: $C_1C_1$=7,880/8,192, $C_2C_1$=7,423/8,192, $C_4C_1$=5,681/8,192, $C_5C_1$=4,464/8,192, $C_2C_2$=6,992/8,192, $C_4C_2$=5,352/8,192, $C_5C_2$=4,205/8,192, $C_4C_4$=4,096/8,192, $C_5C_4$=3,218/8,192 and $C_5C_5$=2,529/8,192. This encoding has been optimized looking for common sharing terms of these fixed-coefficients in order of save area. This multiplier contains 1,528 cells of which 760 are flip-flops. It has a latency of 17 Clk1 clock cycles.

## 6. Implementation and Comparisons

A prototype of an 8×8 2-D DCT processor chip has been designed using standard cells in a semi-custom methodology. It uses 9-b input data and 12-b output data for DCT and 12-b and 9-b for IDCT. The processor was implemented with a 0.35 μm CMOS CSD 3M/2P 3.3 V technology of *Austria-Microsystem* (http://www.asic.austriamicrosystems.com). The chip has an area of $2.5\times2.5\approx6.25$ mm$^2$ (the core is $1.75\times1.75\approx3.06$ mm$^2$). It contains a total of 11.7 k gates, 5.8 k gates of which are flip-flops and 826 gates are FA/HA. Table 2 shows the hardware cost in terms of number of gates for the different blocks of this processor. More details about chip implementation can be found in [22]. A maximum operating frequency of about 300 MHz has been established. The latency for 2-D DCT is 172 Clk1 cycles and for 2-D IDCT is 178 Clk1 cycles. The computing time of a block is close to 580 ns.

In the literature, there are many implementation styles for DCT and IDCT. For proposes of comparison, Table 3 lists features of the proposed processor and other DCT implementations selected from among those which fulfil the specifications of the IEEE standard. This Table shows the following parameters: year of publication, function indicating whether it implements the forward DCT and the IDCT or only the IDCT, technology (all are in CMOS) and design methodology (FC for full-custom, SC for semi-custom, GA for Gate Array), area in mm$^2$ of die or of core, complexity in terms of number of gates or transistors (some designs include additional RAM), frequency and latency, and finally, some basic specifications of the architecture. Three parameters have been taken into account in the specification of the architecture [31]:

- Implementation based on row–column decomposition method or direct method. In the first case, the property of separability of the 2D DCT is used to separate its computation into two sequential 1-D DCT (RC) and transpose memory, or into a single 1-D DCT (MUXRC) processor which performs both operations. In the second case, the direct formula of the 2D DCT is used. Table 3 shows clearly that the RC or MUXRC implementation is superior to the direct method.
- Memory. The implementation of the la DCT requires intermediate memory which, in most cases, is a RAM, either external, ERAM, or internal, TRAM. Other alternatives are the register file (RF) configuration based on flip-flops or transpose buffer (TB).
- Computation based on hardwired multipliers (HM) or on distributed arithmetic (DA).

As compared with the existing design listed in Table 3, the proposed processor is clearly superior in terms of speed even for those processors that use a better technology [11, 15]. The parallel-pipeline architecture and arithmetic units operating at half the frequency gives an input data rate of 300 MHz, far higher than that of the fastest processor listed in this table [11]. This speed does not imply any additional cost in terms of the number of gates since it is similar to that of the other designs proposed which offer an efficient hardware complexity [5, 6, 8, 9, 13, 20]. Thus, [9, 13, 20] use the MUXRC approach to reduce hardware, [5, 9] implement only the IDCT and others present the same complexity in number of gates but require additional RAM [6]. In this respect, notice should be taken in particular of the highly area-efficient processor described in [7] which combines a software-oriented controller with a hardware unit. This processor is not a pure hardware-oriented approach unlike the rest of the processors.

## 7. Conclusions

This paper describes the architecture of an 8×8 2-D DCT/IDCT processor chip with high throughput, reduced hardware, parallel and pipeline architecture operating at half the frequency of input data rate, and a maximum efficiency in all arithmetic elements. This processor is clearly superior in terms of speed without increasing hardware complexity in comparison with others processors which also meet the demands of IEEE Std. 1,180–1,990. This good performance in the computing speed as well as hardware cost, indicate that the proposed design is suitable for HDTV applications.

One advantage of the proposed architecture is that the $K_8$ normalization can be incorporated into the decoding process in an adaptive transform coding system. Therefore, the quantization process at the receiver or at the transmitter can include this normalization. Then, the $K_8$ multiplier can be completely removed in the 2D DCT processor, reducing area by 13% and latency by 10%.

## References

1. K. R. Rao and P. Yip, “Discrete Cosine Transform: Algorithms, Advantages and Applications,” Boston/San Diego/New York/London/Sydney/Tokyo/Toronto: Academic, 1990.
2. V. Bhaskaran and K. Konstantinides, “Image and Video Compression Standards: Algorithms and Architectures,” Boston/Dordrecht/London: Kluwer, 2nd Edition, 1997.
3. K. R. Rao and J. J. Hwang, “Techniques and Standards for Image Video and Audio Coding,” New Jersey: Prentice Hall PTR, 1996.
4. T. Kuroda, T. Fujita, S. Mita, T. Nagamatsu, S. Yoshioka, K. Suzuki, F. Sano, M. Norishima, M. Murota, M. Kako, M. Kinugawa, M. Kakumu, and T. Sakurai, “A 0.9-V, 150 MHz, 10-mW, 4 mm2, 2-D Discrete Cosine Transform Core Processor with Variable Threshold-voltage (VT) Scheme,” *IEEE J. Solid-state Circuits*, vol. 31, no. 11, 1996, pp. 1770–1779.
5. T. H. Chen, “A Cost-effective 8×8 2-D IDCT Core Processor with Folded Architecture,” *IEEE Trans. Consum. Electron.*, vol. 45, no. 2, 1999, pp. 333–339.
6. I. K. Kim, J. J. Cha, and H. J. Cho, “A Design of 2-D DCT/IDCT for Real-time Video Applications,” *6th Int. Conf. on VLSI and CAD*, 1999, pp. 557–559.
7. T. S. Chang, C. S. Kung, and C. W. Jen, “A Simple Processor Core Design for DCT/IDCT,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 10, no. 3, 2000, pp. 439–447, April.
8. Dae-Won-Kim, Taek-Won-Kwon, Jung-Min-Seo, Jae-Kun-Yu, Suk-Kyu-Lee, and Jung-Hee-Suk, “A Compatible DCT/IDCT Architecture Using Hardwired Distributed Arithmetic,” *IEEE Int. Symp. Circuits Syst. Proc.*, vol. 2, 2001, pp. 457–460.
9. J. I. Guo and J. C. Yen, “An Efficient IDCT Processor Design for HDTV Applications,” *J. VLSI Signal Process.*, vol. 33, 2003, pp. 147–155.
10. Y. P. Lee, T. H. Chen, L. G. Chen, M. J. Chen, and C. W. Ku, “A Cost-effective Architecture for 8×8 Two Dimensional DCT/IDCT using Direct Method,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 7, no. 3, 1997, pp. 459–466.
11. D. Gong, Y. He, and Z. Cao, “New Cost-efective VLSI Implementation of a 2-D Discrete Cosine Transform and its Inverse,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 14, no. 4, 2004, pp. 405–415.
12. S. I. Uramoto, Y. Inoue, A. Takabatate, 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, 1992, pp. 492–499.
13. A. Madisetti and A. N. Willson, “A 100 MHz 2-D 8×8 DCT/IDCT Processor for HDTV Applications,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 5, no. 2, 1995, pp. 158–165.
14. R. Rambaldi, A. Uguzzoni, and R. Guerrieri, “A 35 μW 1.1 V Gate Array 8×8 IDCT for Video Technology,” *Proc. ICASSP*, vol. 5, 1998, pp. 2993–2996.
15. L. Fanucci and S. Saponara, “Data Driven VLSI Computation for Low Power DCT-based Video Coding,” *9th IEEE Int. Conf. on Electronics-Circuits-and-Systems*, vol. 2, 2002, pp. 541–544.
16. B. D. Tseng and W. C. Miller, “On the Computing the Discrete Cosine Transform,” *IEEE Trans. Comput.*, vol. C-27, no. 10, 1978, pp. 966–968.
17. H. Malvar, “Fast Computation of Discrete Cosine Transform Through Fast Hartley Transform,” *Electron. Lett.*, vol. 22, no. 7, 1986, pp. 352–353.
18. S. Yu and E. E. Swartzlander, “A Scaled DCT Architecture with the CORDIC Algorithm,” *IEEE Trans. Signal Process.*, vol. 50, no. 1, 2002, pp. 160–167.
19. Y. T. Chang and C. L. Wang, “A New Fast DCT Algorithm and its Systolic VLSI Implementation,” *IEEE Trans. Circuits Syst., 2 Analog Digit. Signal Process.*, vol. 44, no. 11, 1997, pp. 959–962.
20. P. A. Ruetz, P. Tong, D. Bailey, A. D. Luthi, and P. H. Ang, “A High-performance Full-motion Video Compression Chip Set,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 2, no. 2, 1992, pp. 111–122.
21. L. Fanucci, R. Saletti, and F. Vavala, “A Low-complexity 2-D Discrete Cosine Transform Processor for Multimedia Applications,” 1999, pp. 449–452.
22. G. A. Ruiz, J. A. Michell, and A. M. Burón, “Parallel-pipeline 2D DCT/IDCT Processor Architecture,” *SPIE Symp. on Microtechnologies for the New Millennium*, pp. 774–784, May 2005.
23. “IEEE Standard Specifications for the Implementations of 8×8 Inverse Discrete Cosine Transform,” Institute of Electrical and Electronics Engineers, New York, March 1991.

24. Video Codec for Audio-Visual Services at px64 kbits/s, ITU-T H.261, 1993.
25. ITU-T recommendation H.263. Video Coding for Low Bit-Rate Communication, 1996.
26. ITU-T recommendation H.263+, Jan. 27, 1999, Draft 21.
27. W. Chen, C. H. Smith, and S. Fralick, "A Fast Computation Algorithm for the Discrete Cosine Transform," *IEEE Trans. Commun.*, vol. 25, 1977, pp 703–709.
28. S. Kim and W. Sung, "Optimum Wordlength Determination of $8\times8$ IDCT Architectures Conforming to the IEEE Standard Specifications," Conference Record of The 29th Asilomar Conference on Signals, Systems and Computers, vol. 2, 1996, pp. 821–825.
29. T. Y. Chang and M. J. Hsiao, "Carry-select Adder Using Single Ripple-carry Adder," *Electron. Lett.*, vol. 34, no. 22, 1998, pp. 2101–2103.
30. R. P. Brent and H. T. Kung, "A Regular Layout for Parallel Adders," *IEEE Trans. Comput.*, vol. C-31, no. 3, 1982, pp. 260–264.
31. G. S. Taylor and G. M. Blair, "High Design for the Discrete Cosine Transform in VLSI," *IEE Proc. Comput. Digit. Tech.*, vol. 145, no. 2, 1998, pp. 127–133, March.

**Gustavo A. Ruiz** was born in Burgos, Spain, in 1962. He received the M.Sc. degree in physics in 1985 from the University of Navarra, Spain, and the Ph.D. degree in physical science in 1989 from the University of Cantabria, Santander, Spain. Since 1985, he has been with the Department of Electronics and Computers at the University of Cantabria, where he is currently an Associate Professor. His current research interests are mainly focused on VLSI architectures for signal processing and high-speed arithmetic circuits.

**Juan A. Michell** was born in Cáceres, Spain, in 1952. He received the M.S. and the Ph.D. degrees in physical sciences from the University of Cantabria, Spain, in 1974 and 1977, respectively. Since 1974 he has been with the Department of Electronics and Computers at the University of Cantabria, where he was appointed Professor in Electronics in 1991. His current research interests are VLSI architectures and integrated circuit design for digital signal processing applications.

**Angel M. Burón** was born in Córdoba, Spain, in 1948. He received the M.S. and the Ph.D. degrees in physical sciences from the University of Sevilla, Spain, in 1969 and 1973, respectively. Since 1973 he has been with the Department of Electronics and Computers at the University of Cantabria, where he was appointed Professor in Electronics in 1982. His current research interests are integrated circuit design and VLSI/ULSI architectures for digital signal processing.