Fast RNS-based DCT Computation with Fewer Multiplication Stages

Pedro G. Fernández [(1)], Antonio García [(2)], Javier Ramírez [(2)], Luis Parrilla [(2)] and Antonio Lloris [(2)]

[(1)] Dpto. Ingeniería Eléctrica
Universidad de Jaén, 23071 Jaén
e-mail: pfernan@ujaen.es

[(2)] Dpto. Electrónica y Tecnología de Computadores
Universidad de Granada, 18071 Granada
e-mail: {agarcia, jramirez, lparrilla, lloris}@ditec.ugr.es

Abstract

This paper presents a Field-Programmable Logic (FPL) implementation of the Discrete Cosine Transform (DCT) using the Residue Number System (RNS). It is based on the use of the separability property of the two-dimensional DCT (2D-DCT) and a Fast Cosine Transform (FCT) algorithm. Each 1D-DCT in this decomposition technique requires only 14 Look-up Tables (LUT) and 28 adders or substractors. This paper shows that the RNS-based FCT implementation on FPL devices provides a throughput improvement of up to 56.09% over the equivalent binary system. This is achieved due to the synergy between RNS and modern FPL device families.

1. Introduction

The Discrete Cosine Transform (DCT) [1] is considered the most efficient technique for image encoding and compression schemes, being the preferred transform for most codes due to its decorrelation and energy compaction properties. It has been used in most of the coding standards for image compression, including CCITT H.261/H.263, JPEG (Joint Photographic Experts Group) and MPEG (Motion Pictures Experts Group). The most commonly used transform is the two-dimensional DCT (2D-DCT), used as a standard processing component in many applications [2]. Several hardware-design methods have been developed for 2D-DCT implementation. The most efficient method for hardware implementation is based on the use of the DCT separability property, thus computing the 2D-DCT through several one-dimensional DCTs (1D-DCT). A solution for the 1D-DCT computation is the fast cosine transform (FCT) algorithm.

On the other hand, RNS is a powerful solution for maintaining constant performance and increasing precision in fixed-point arithmetic applications [3]. Field-Programmable Logic (FPL) provides Digital Signal Processing (DSP) arithmetic support with fast carry chains (Xilinx XC4000, Altera FLEX) which are used to implement Multiply-Accumulate (MAC) blocks at high speed [4]. Two main advantages are obtained when FPL is used to develop DSP applications: parallelism (implementing multiple MAC calls) and pipelining (each logic element has a register, and therefore the pipeline requires no additional resources). Pipelined digital filters and other DSP applications have been implemented using RNS on FPL devices [5] showing improved results over the corresponding binary systems and, thus, illustrating the synergy between RNS and FPL devices.

An RNS-based direct implementation of the separable DCT has recently been reported [6], showing substantial throughput increase when compared to both binary distributed arithmetic and conventional binary implementations. In this paper, a FCT implementation for the calculation of the DCT based on the RNS is presented and compared with its corresponding binary systems on FPL devices. This paper shows that the RNS-based 1D-DCT provides a throughput improvement over the equivalent binary systems of up to 56.09% when 8-bit moduli are used.

2. RNS Background

An RNS system is defined by a set of positive relatively prime integers $\{m_1, m_2,..., m_L\}$, called moduli. The dynamic range of the RNS system is M, where:

$$M = \prod_{i=1}^{L} m_i \qquad \textbf{(1)}$$

Any integer $X \in [0, M-1]$ is represented by the L–tuple $[x_1,..., x_L]$, where x_i is the residue of X modulo m_i (i=1, 2, ..., L). RNS arithmetic operations are defined over the ring of integers modulo M. Thus, for $0 \le X, Y, Z < M$:

$$\begin{aligned} Z &= \left| X \lozenge Y \right|_M \equiv [z_1,..., z_L] \\ z_i &= \left| x_i \lozenge y_i \right|_{m_i} \qquad i = 1, 2,..., L \end{aligned} \qquad \textbf{(2)}$$

where $\lozenge$ represents either addition, subtraction or multiplication. Equation (2) shows the parallel nature of the RNS. In this way, a large dynamic range binary system can be partitioned into faster parallel small dynamic range RNS channels, with

arithmetic modulo M being reduced to a set of operations over smaller moduli.

3. Discrete Cosine Transform

The DCT transforms a set of samples into coefficients over a scaled set of discrete cosine basis functions. Each $N \times N$ block is transformed separately into N^2 coefficients for reconstruction from the DCT basis functions. The properties of the DCT are that the coefficients are highly decorrelated and most of the information content is in the low frequency coefficients. These properties support the reduction of both redundant and irrelevant information. The DCT has been the preferred transform for most codes due to these properties.

The basic 1D-DCT of an N-point sequence $\{x(0), x(1),..., x(N-1)\}$ can be expressed as [2]:

$$X(u) = \sqrt{\frac{2}{N}}\ e(u) \sum_{i=0}^{N-1} x(i) \cos \frac{u(2i+1)\pi}{2N} \qquad \textbf{(3)}$$

$$u = 0, 1, ..., N-1$$

where:

$$e(u) = \begin{cases} \dfrac{1}{\sqrt{2}} & \text{if } u = 0 \\ 1 & \text{otherwise} \end{cases}$$

The most commonly used transform for image processing is the 2D-DCT. Typically, the image is divided into blocks of fixed size (8×8 or 16×16), so hardware development has concentrated on these basic block sizes. When applied to an image block of size $N \times N$, the 2D-DCT transforms a real $N \times N$-point matrix $x(i, j)$ into another matrix $X(u, v)$ of coefficients, whose elements are given by:

$$X(u,v) = \frac{2e(u)e(v)}{N} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} x(i, j) \cos \frac{u(2i+1)\pi}{2N} \cos \frac{v(2j+1)\pi}{2N}$$

$$u,v = 0, 1, ..., N\text{-}1$$

Several design methods can be used for the 2D-DCT computation. Most of these designs makes use of the separability property of the 2D-DCT. Several implementations based on row-column decomposition can be found in the literature. The basic block diagram for these schemes is shown in Fig. 1.

Therefore, a 2D-DCT of $N \times N$ points can be computed by means of $2N$ N–point 1D-DCTs as follows: first, the N-point 1D-DCT of each row is computed. Then, the intermediate result is transposed and the N–point 1D-DCT of each column is computed in an identical fashion. This property can be used to reduce the arithmetic complexity, since $2N$ N-point 1D-DCTs require substantially fewer arithmetic operations than one N^2-point DCT. Thus, a 2D-DCT of an $N \times N$-pixel matrix can be performed with N N-point 1D-DCTs, followed by a matrix transposition and N N-point 1D-DCTs of the transposed matrix. Thus, our main goal is to develop efficient 1D-DCT processors.

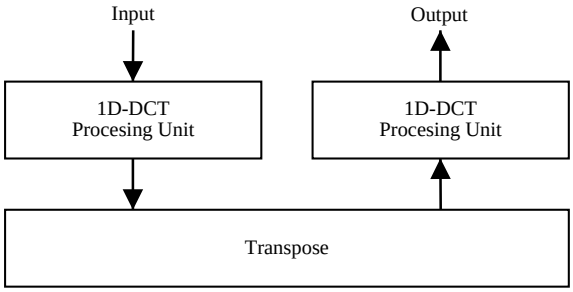

Fig. 1. 2D-DCT block diagram.

4. Fast Cosine Transform (FCT)

The DCT was originally introduced by Ahmed, Natajaran and Rao [1] in 1974. After this first proposal numerous fast algorithms for the efficient computation of the 1D-DCT have been proposed [9], [10], [11]. The aim of developing these fast algorithms is to reduce the number of arithmetic operations and, consequently, to simplify hardware realizations. These fast algorithms for DCT have been based on FFT, real arithmetic and recursive algorithms. When fast algorithms are implemented, floating-point multiplications should be avoided due to their negative effect on computational speed. Many algorithms that require fewer multiplications, as Lee's algorithm [9], require multiple stages of multiplications and, consequently, require wider multiplications and a longer computation time. Therefore, many systems implement the DCT using integer multiplication. However, this introduces a round-off error in each multiplication stage and a fast algorithm with fewer multiplication stages is necessary [12]. This algorithm minimizes the propagation error by reducing the number of multiplication stages and ensures a sufficient accuracy with a small amount of hardware. Also, this algorithm performs most multiplications at the final stage and requires fewer multiplications stages than other algorithms.

The number of additions for this algorithm is less than other fast 1D-DCT algorithms such as Lee's [10] or Hou's [11], while the number of multiplications is the same. The maximum number of multiplication stages is defined as the maximun number of multiplier components that are passed sequentially to derive a DCT coefficient. For the 8-point DCT this number is 2, smaller than for other algorithms. This reduction in the number of multiplicatins stages minimizes the error in the

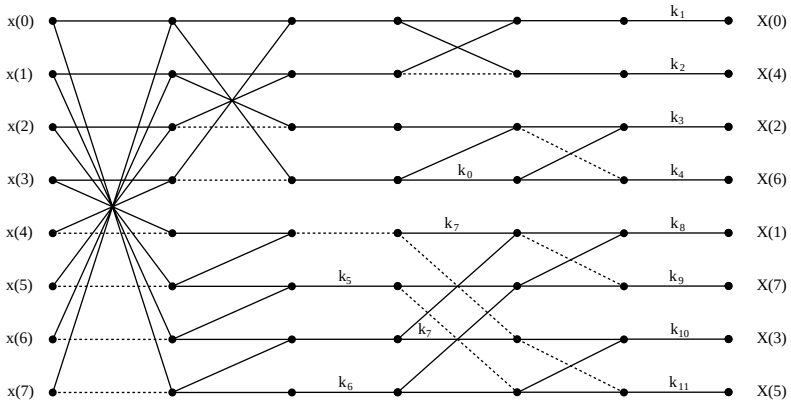

Fig. 2. 8×8 DCT signal flow graph.

computation. The signal flow graph of Fig. 2 describes the algorithm for the computation of the 8×8 DCT.

This algorithm is stated below:

Step 1:

$$a_1=x(0)+x(7) \qquad a_2=x(1)+x(6)$$
$$a_3=x(2)+x(5) \qquad a_4=x(3)+x(4)$$
$$a_5=x(3)-x(4) \qquad a_6=x(2)-x(5)$$
$$a_7=x(1)-x(6) \qquad a_8=x(0)-x(7)$$

Step 2:

$$b_1=a_1+a_4 \qquad b_2=a_2+a_3$$
$$b_3=a_2-a_3 \qquad b_4=a_1-a_4$$
$$b_5=a_5+a_6 \qquad b_6=k_5(a_6+a_7)$$
$$b_7=a_7+a_8 \qquad b_8=k_6 a_8$$

Step 3:

$$c_1=b_1+b_2 \qquad c_2=b_1-b_2$$
$$c_3=b_3+b_4 \qquad c_4=k_0 b_4$$
$$c_5=k_7 b_5+b_7 \qquad c_6=b_6+b_8$$
$$c_7=k_7 b_7-b_5 \qquad c_8=b_8-b_6$$

Step 4:

$$X(0)=k_1 c_1 \qquad X(1)=k_8(c_5+c_6)$$
$$X(2)=k_3(c_3+c_4) \qquad X(3)=k_{10}(c_7+c_8)$$
$$X(4)=k_2 c_2 \qquad X(5)=k_{11}(c_8-c_7)$$
$$X(6)=k_4(c_4-c_3) \qquad X(7)=k_9(c_6-c_5)$$

where:

$x(i)$ input elements ($0 \le i \le 7$)
$X(i)$ DCT coefficients ($0 \le i \le 7$)

and k_i coefficients:

$$k_0=1/C_4^1 \qquad k_1=\sqrt{2}/4 \qquad k_2=C_4^1/2$$
$$k_3=C_4^1/4C_8^1 \qquad k_4=C_4^1/4C_8^3 \qquad k_5=C_4^1/C_8^1$$
$$k_6=1/C_8^1 \qquad k_7=C_8^3/C_8^1 \qquad k_8=C_8^1/4C_{16}^1$$
$$k_9=C_8^1/4C_{16}^7 \qquad k_{10}=C_8^1/4C_{16}^3 \qquad k_{11}=C_8^1/4C_{16}^5$$

for C_m^n defined as $C_m^n=\cos(\pi n/m)$.

5. RNS Implementation of the DCT

As shown in section 3, 2D-DCT is reduced to several 1D-DCTs, which consist of a large number of fixed coefficient multiplications and additions. Thus, RNS may be considered for the implementation of the fast 1D-DCT processor. On the other hand, an often used block size is 8×8, since this represents a standard size for image and video coding. Therefore, an RNS-based 8-point 1D-DCT fast processor has been implemented on FPL devices.

Altera FLEX10K [3] devices have been shown to be ideal tools for the implementation of RNS-based DSP applications [5], [6] over programmable devices. Thus, this device family was chosen for the implementation of the proposed RNS DCT processor and its comparison with the binary alternative. FLEX10K devices include assorted Logic Elements (LE) distributed in 8-LE Logic Array Blocks (LAB). Each LE provides a $2^3 \times 1$ LUT and a fast carry chain support when working in arithmetic mode. On the other hand, FLEX10K devices also include a number of Embedded Array Blocks (EAB), which are 2K-bit tables, configurable as either $2^8 \times 8$, $2^9 \times 4$, $2^{10} \times 2$ or $2^{11} \times 1$. These EABs can be used to create ROMs, FIFOs or synchronous, asynchronous and dual-port RAMs. The presence in the same chip of fast arithmetic resources and dedicated tables enables the synergy between FPL

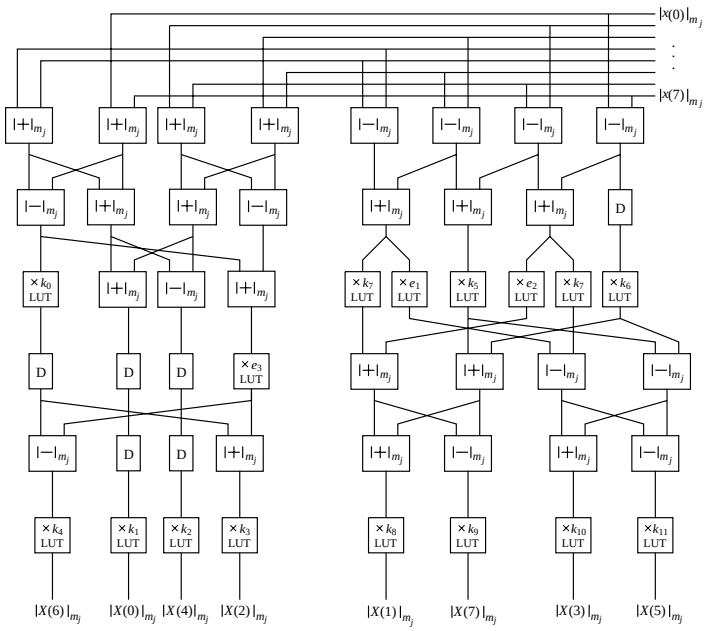

Fig. 3. Modulo m_j 8-pixel RNS 1D-DCT channel.

devices and RNS.

First of all, a 2's complement binary implementation of the algorithm shown in section 3 was carried out. The 8-point input sequence {$x(0)$, $x(1)$, $x(2)$, $x(3)$, $x(4)$, $x(5)$, $x(6)$, $x(7)$} is initially represented with 8-bit precision, while each fixed coefficient is represented with 10-bit precision. 10-bit fixed coefficients and a 32-bit dynamic range ensure sufficient accuracy.

Since Altera FLEX10K devices provide 2^8×8-bit EABs, if 8-bit moduli are considered it is possible to store fixed coefficient multiplications in a single EAB. From the algorithm of Fig. 2, it is deduced that the maximum number of fixed coefficient products is two for each transformed point calculation, so two EABs can accommodate this fixed coefficient multiplications required for the computation of each transformed points.

An FPL-based implementation of the RNS 1D-DCT using Altera FLEX10K devices has been developed using the algorithm shown in section 3. Using this FCT algorithm, the modulo m_j 8-pixel RNS 1D-DCT can be represented by the architecture shown in Fig. 3. Each arithmetic instance is a RNS component consisting of a modular adder, a modular subtractor or a look-up table multiplier. Fast and easily implementable LUT multipliers are used for the multiplication [13].

The modulus set {256, 255, 253, 251} is chosen to implement an RNS based 8-point 1D-DCT, with 8-bit inputs and 10-bit fixed coefficients. 8-bit modulus parallel channels are chosen in order to ensure fast internal RNS processing. The RNS–based computation of the DCT is executed in parallel over these four channels. The benefit gained from this architecture is a more efficient realization of hardware due to the replacement of multipliers by memory look-up tables.

Modulo 256 offers hardware reduction in the binary-to-RNS conversion and a considerable memory reduction in the processing channel, since n-bit power-of-two modular adders and multipliers are efficiently implemented with conventional n-bit binary adders and multipliers, respectively.

Binary and RNS implementations of the 8-point DCT were carried out on FPL devices in order to compare hardware complexity and speed performance. This architecture requires only 14 LUTs and 28 adders/subtractors. Detailed information about the LE and EAB usage for the 8-bit RNS, the modulo 256 RNS and binary 2's complement 8-pixel 1D-DCT processor and their throughput, measured in millions of DCTs per second, for several device speed grades is shown in Tables 1, 2 and 3 and is summarized as follows:

a) RNS implementation provides a higher throughput due to the efficiency of the RNS multiplication through LUTs. Specifically, for grade –3 speed FLEX10KE devices, the performance on FPL devices provides a 56.09% throughput improvement over equivalent binary system.
b) The limited number of LEs required by the binary implementation, despite the large number

of binary multipliers, is due to the constant input for each binary multiplier.

c) A modulus 256 offers a considerable memory reduction in the processing channel.

On the other hand, the practical implementation of RNS-based systems has a serious limitation in the conversion stage. A solution to overcome the conversion drawback is the auto-scaling RNS-to-binary converter (ε-CRT) proposed by Griffin *et al* [14], that uses only LUTs and conventional binary adders.

	#LEs	#EABs	Throughput
Grade-1	625	14	95.23
Grade-2	625	14	80.64
Grade-3	625	14	60.97

Table 1. Performance and LE and EAB usage results for a 8-bit RNS channel architecture.

	#LEs	#EABs	Throughput
Grade-1	833	-	149.25
Grade-2	833	-	136.98
Grade-3	833	-	99.00

Table 2. Performance and LE and EAB usage results for a modulo 256 RNS channel.

	#LEs	#EABs	Throughput
Grade-1	1884	-	62.11
Grade-2	1884	-	51.81
Grade-3	1884	-	39.06

Table 3. Performance and LE and EAB usage results for a pipelined binary architecture (two stage).

6. Conclusions

The validity of RNS arithmetic for the development of high-performance DSP applications over FPL devices has been shown. Intensive LUT-based arithmetic has been the main limitation for the practical implementation of RNS-based systems, but the advent of new FPL device families and their increasing densities allow the mapping of full DSP applications in a single chip, while taking advantage of RNS properties for maximum performance. This paper extends these ideas and shows that RNS is an efficient tool for the development of high-performance DCT processors. Using the Altera FLEX10K device family, a RNS-based implementation of a FCT 1D-DCT processor have been developed with an 8-bit modulus set, thus taking full advantage of the device's built-in tables. The 1D-DCT requires only 14 LUTs and 28 adders/subtractors, fewer than other designs. Parameters such as area and performance were assessed by synthesizing both binary 2's complement and RNS 1D-DCT processor using VHDL and Altera FLEX10KE devices. Thus, the proposed RNS-based 1D-DCT processor provided a throughput advantage of up to 56.09% over its binary counterpart. This advantage is due to the efficient use of the device's built-in tables for the mapping of fixed coefficient products for 8-bit moduli. Finally, RNS-to-binary conversion can be carried out through an auto-scaling ε–CRT converter.

Acknowledgements

The authors were supported by the Dirección General de Enseñanza Superior (Spain) under project PB98-1354. CAD tools and supporting material were provided by Altera Corp. under the Altera University Program.

References

[1] N. Ahmed, T. Natarajan, and K. R. Rao, "Discrete Cosine Transform", *IEEE Trans. on Computers*, vol. C-23, pp. 90-93, Jan. 1974.

[2] K. R. Rao, P. Yip, *Discrete Cosine Transform. Algorithms, Advantages, Applications.* Academic Press Inc., 1990.

[3] M. Soderstrand, W. Jenkins, G. A. Jullien, F. J. Taylor, *Residue Number System Arithmetic: Modern Applications in Digital Signal Processing*. IEEE Press, 1986.

[4] Altera Corporation, "1998 Data Book", Jan. 1998.

[5] A. García, U. Meyer-Bäse and F. J. Taylor, "Pipelined Hogenauer CIC Filters Using Field-Programmable Logic and Residue Number System". *Proc. 1998 IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, Seattle, WA, vol.5, pp. 3085-3088, May 1998.

[6] P. G. Fernández, A. García, J. Ramírez, L. Parrilla and A. Lloris, "A New Implementation of the Discrete Cosine Transform in the Residue Number System". *Proc. 3rd Asilomar Conf. on Signals, Systems and Computers*, Pacific Grove, CA, vol. 2, pp. 1302-1306, Oct. 1999.

[7] U. Totzet, F. Matthiesen, S. Wohlleben and T. G. Noll, "CMOS VLSI Implementation of the 2D-DCT with Linear Processor Arrays", *Proc. 1990 IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, Albuquerque, NM, pp. 937-940, Apr. 1990.

[8] M. T. Sun, T. C. Chen and A. M. Gottlieb, "VLSI Implementaion of 16×16 Discrete Cosine Trasform", *IEEE Trans. on Circuits and Systems*, vol. 36, no. 4, pp. 610-617, Apr. 1989.

[9] W. Chen, C.H. Smith, and S. C. Fralick, "A Fast Computational Algorithm for the Discrete Cosine Transform", *IEEE Trans. on Communications*, vol. COM-25, pp. 1004-1009, Sep. 1977.

[10] B. G. Lee, “A new algorithm to compute the discrete cosine transform”, *IEEE Trans. on Acoustics, Speech and Signal Processing*, vol. 32, no. 6, pp. 1243-1245, 1984.

[11] H. S. Hou, “A fast recursive algorithm for computing the discrete cosine transform”, *IEEE Trans. on Acoustics, Speech and Signal Processing*, vol. 35, no. 10, pp. 1445-1461, 1987.

[12] Y. Jeong, I. Lee, H. S. Kim, and K. T. Park, “Fast DCT algorithm with fewer multiplication stages”, *Electronics Letters*, vol. 34, no. 8, pp. 723-724, Apr. 1998.

[13] J. Ramírez, A. García, P. G. Fernández, L. Parrilla and A. Lloris, “A New Architecture to compute the Discrete Cosine Transform using the Quadratic Residue Number System”, *Proc. 2000 IEEE Int. Symp. on Circuits and Systems*, Geneva, vol. 5, pp. 321-324, May 2000.

[14] M. Griffin, M. Sousa and F. Taylor, “Efficient Scaling in the Residue Number System”, *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. 1075-1078, 1989.