A DCT Architecture based on Complex Residue Number Systems

J. RAMÍREZ [1], A. GARCÍA [2], P. G. FERNÁNDEZ [3], L. PARRILLA [1], A. LLORIS [1]

[1] Dept. of Electronics and Computer Technology
University of Granada
Campus Universitario Fuentenueva
18071, Granada, SPAIN

[2] Dept. of Computer Sciences
Universidad Autónoma de Madrid
Ciudad Universitaria de Cantoblanco
28049, Madrid, SPAIN

[3] Dept. of Electrical Engineering
University of Jaén
Escuela Politécnica Superior
23071, Jaén, SPAIN

Abstract: - In this paper a high-performance 1-D DCT processor is shown. It is based on a FFT algorithm that benefits from complex residue arithmetic to reduce the number of operations required. The array is based on the Quadratic Residue Number System (QRNS) which enables the implementation of complex adders (multipliers) with only two modular adders (multipliers). Thus, the number of additions and multiplications for the 8-point (16-point) processor was reduced in 21.43% (24.18%) and 24% (30.56%), respectively. An 8-point DCT array was developed for comparison using binary 2's complement and the QRNS. The systems were modelled using VHDL and synthesized over Altera FLEX10KE Field-Programmable Logic (FPL) devices. The proposed array based on the QRNS yields a performance advantage of up to 81.84% over the equivalent system based on 2's complement arithmetic.

Key-Words: - Residue Number System, DCT, QRNS, DSP, Field Programmable Logic.

1 Introduction

Currently there are design barriers inhibiting the implementation of high-precision digital signal processing (DSP) objects with field programmable logic (FPL) devices. This paper explores overcoming these barriers by fusing together the Quadratic Residue Number System (QRNS) architecture with the Discrete Cosine Transform for use in FPL-centric designs.

The Discrete Cosine Transform (DCT) [1, 2] is a subject of study of modern digital image processing technology. Its inclusion in standards for image and video compression (JPEG, MPEG, H.261) as the transform-coding technique has led to numerous contributions focussing its fast implementation with reduced hardware complexity.

The DCT of a N-point sequence $\{x(0), x(1), \ldots, x(N-1)\}$ is defined by:

$$X(m) = \sqrt{\frac{2}{N}} K_m \sum_{n=0}^{N-1} x(n) \cos\left(\frac{(2n+1)m\pi}{2N}\right) \qquad m = 0,\ 1,\ \ldots,\ N-1 \qquad \textbf{(1)}$$

$$K_0 = \frac{1}{\sqrt{2}} \qquad K_1 = K_2 = \ldots = K_{N-1} = 1$$

Its direct computation through equation (1) requires N^2 multiplications and $(N\text{-}1)^2$ additions. In order to reduce computational complexity, the transform is expressed as an MVM (Matrix Vector Multiplication) algorithm and the resulting $N \times N$ cosine matrix is factored. These structures lead to improved architectures that accept N parallel inputs and, in several addition and multiplication stages, produce the resulting N outputs in parallel. Thus, different algorithms exist that exploit this design methodology and reduce the number of operations required. Many of these algorithms exploit the relation between the DCT and the Discrete Fourier Transform (DFT) to take advantage of the Cooley-Tukey Fast Fourier Transform (FFT) algorithms [3-5].

These algorithms lead to architectures with reduced computational complexities but with greater dynamic ranges than the direct computation since more than one successive multiplication stages are required in the signal path. These structures use pipelining to improve the performance of large word-width carry-propagate adders and multipliers.

On the other hand, the use of Quadratic Residue Number System (QRNS) [6, 7] arithmetic in FFT-based 1-D DCT processors leads to an even higher reduction in the number of additions and multiplications. Addition and multiplication of complex integers can be computed efficiently using the QRNS with only two modular additions and multiplications, respectively. At the same time, the

use of a number of parallel small word-width (6-, 7- or 8-bit) QRNS channels enable to accelerate this kind of 1-D DCT processors that consist of modular arithmetic adders and multipliers avoiding the use of slow carry propagation delay binary 2's complement adders and multipliers

An RNS (Residue Number System) [6, 8] implementation of an arithmetic-intensive system leads to several copies of the original performing modular arithmetic and working over small dynamic ranges but globally handling the total one. It is possible to consider the mapping of RNS systems into Field-Programmable Logic (FPL) devices. The LUT (Look-Up Table) requirements for some RNS operations have been a serious obstacle for the development of such structures in the past. However, the new programmable device families, such as Altera FLEX10K or APEX20K [9], Xilinx Virtex [10] and Actel ProASIC [11], have overcome this problem by means of small built-in embedded memories and logic blocks with fast carry and cascade chains support [12, 13].

This paper shows the design methodology of a QRNS enabled 8-point 1-D DCT processor. Special attention is given to the algorithm derivation, selection of moduli and performance and hardware complexity analysis. QRNS and binary 2's complement 1-D DCT processors were implemented through architectural level VHDL synthesis to assess hardware requirements and throughput. Altera FLEX10KE modern FPL devices were considered. The proposed processor reduces the number of additions and multiplications in a factor of 21.43% and 24%, respectively, while the throughput increase is of 81.84% over the massively pipelined binary 2's complement version.

2 RNS-based complex arithmetic

Let the modulus set $\{m_1, m_2, \ldots, m_L\}$ consists of L pairwise relatively prime integers and $M = \prod_{i=1}^{L} m_i$. Thus, any positive integer $X \in Z(M)$ is uniquely represented by the L-tuple $[x_1, x_2, \ldots, x_L]$ of its residues, where $x_i = X \bmod m_i$.

Given two integers X, $Y \in Z(M)$, and their corresponding residue representations $[x_1, x_2, \ldots, x_L]$ and $[y_1, y_2, \ldots, y_L]$, arithmetic in the RNS is defined by:

$$X \circ Y \leftrightarrow \left[\left| x_1 \circ y_1 \right|_{m_1}, \ \ldots, \ \left| x_L \circ y_L \right|_{m_L} \right] \qquad \textbf{(2)}$$

where $\circ$ represents either addition, subtraction or multiplication.

The Complex Residue Number System (CRNS) [14], $C(M)$, is defined over $Z(M) \times Z(M)$ and any complex number Z is uniquely represented by:

$$\begin{aligned} Z &= X + jY \qquad 0 \le X, Y \le M-1 \\ Z &= (Z_1, Z_2, \ldots, Z_L) \\ Z_i &= X_i + jY_i \qquad i = 1, 2, \ldots, L \end{aligned} \qquad \textbf{(3)}$$

with $j = \sqrt{-1}$, $X_i = X \bmod m_i$ y $Y_i = Y \bmod m_i$. Arithmetic is defined in the usual form by operations defined to be modulo m_i. Quadratic Residue Number System (QRNS) [7] is an extension of the RNS for complex calculus. It is defined by an isomorphic mapping between the ring of complex integers modulo m and the set of pairs of elements in the ring of integers modulo m. QRNS exists if the prime factor decomposition of m has only primes of the form $4K+1$ or, equivalently, if the equation x^2+1 can be factorized as $(x-r)(x+r)$ with $r \in \{0, 1, \ldots, m-1\}$. Let $q_1 + jq_2$ be a complex number with $0 \le q_1, q_2 < m$, and r a root of the equation $x^2+1 = 0$ in the ring of integers modulo m. Thus, QRNS is defined by the isomorphic mapping:

$$\begin{aligned} q \equiv q_1 + jq_2 \ &\xrightarrow{\text{QRNS}} \ \bar{q} \equiv (\bar{q}_1, \bar{q}_2) \\ \bar{q}_1 &= \left| q_1 + rq_2 \right|_m \\ \bar{q}_1 &= \left| q_1 - rq_2 \right|_m \end{aligned} \qquad \textbf{(4)}$$

Arithmetic in the QRNS is defined as follows:

$$(\bar{q}_1, \bar{q}_2) \lozenge (\bar{p}_1, \bar{p}_2) = \left(\left| \bar{q}_1 \lozenge \bar{p}_1 \right|_m, \left| \bar{q}_2 \lozenge \bar{p}_2 \right|_m \right) \qquad \textbf{(5)}$$

where $\lozenge$ represents addition, subtraction or multiplication. The inverse QRNS mapping is given by:

$$\begin{aligned} (\bar{q}_1, \bar{q}_2) \ &\xrightarrow{\text{QRNS}^{-1}} \ q_1 + jq_2 \\ q_1 &= \left| 2^{-1} (\bar{q}_1 + \bar{q}_2) \right|_m \\ q_2 &= \left| (2r)^{-1} (\bar{q}_1 - \bar{q}_2) \right|_m \end{aligned} \qquad \textbf{(6)}$$

Thus, RNS-to-QRNS and QRNS-to-RNS conversion only require one adder, one subtractor and one LUT each. On the other hand, according to (5), a relevant property of the QRNS is that it allows complex integer multiplication through only two integer multiplications while maintaining the RNS speed advantage.

3 FFT-based 1-D DCT array

The efficient Cooley-Tukey algorithm for the DFT has been used to compute the DCT through the DFT. Ahmed [1] introduced the first fast algorithm to

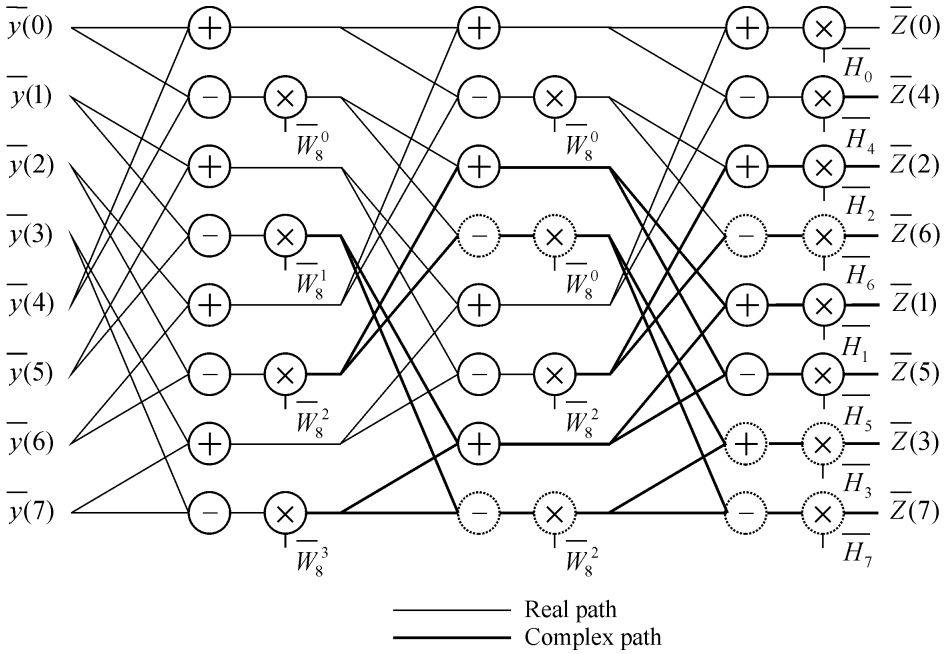

Fig. 1. Array to compute the 1-D DCT in the QRNS.

compute the N-point DCT through the computation of the real part of a $2N$-point DFT scaled by a complex exponential constant. The N-length input sequence is padded with N zeros and the N-point DCT of the original sequence can be computed through the calculation of a $2N$-point DFT with a fast transform algorithm, as derived from:

$$X(m) = \sqrt{\frac{2}{N}} K_m \,\mathrm{Re}\left\{ e^{-j\frac{m\pi}{2N}} \sum_{n=0}^{2N-1} x(n)\, W_{2N}^{mn} \right\} \quad \textbf{(7)}$$

$$W_{2N} = e^{-j\frac{2\pi}{2N}} \qquad x(n) = 0 \qquad n = N,\ ...,\ 2N-1$$

After this first proposal many new algorithms have been developed for the efficient computation of the DCT with a reduced number of real multiplications and additions. Another possible $2N$-point sequence to compute the DCT through a scaled FFT is the specular reflection of the original N-point sequence. Thus, the N-point DCT can be computed with $2N \cdot \log_2(2N)$ complex MAC operations. Haralick [3] shows that the N-point DCT can be computed with two N-point FFTs obtaining a reduction of $2N$ complex MACs. The procedure is based on the fact that the FFT of a real even data sequence of length $2N$ can be obtained by computing two FFTs of N-point data sequences. Tseng and Miller [4] have shown an algorithm with only $N/2 \cdot (\log_2 N+1)$ complex operations which calculate only the real part of the FFT of a $2N$-point real specular-reflection sequence. Narasimha and Peterson [5] have proposed a more efficient algorithm to compute the N-point DCT with a rearrangement of the input sequence and a radix-2 algorithm for the DFT. With this algorithm, the resulting number of complex multiplications is $(N \cdot \log_2 N - N + 1)/4$, which leads to an implementation that is twice as fast as the previous proposals.

The computation of the 1-D DCT making use of the Narasimha and Peterson algorithm by means of the QRNS is described in detail in this paper. The use of QRNS-based complex arithmetic enables reducing the number of operations and increasing the performance due to the parallelism between high-throughput small wordwidth modular channels.

4 QRNS-based DCT architecture

The algorithm used for the QRNS-based DCT computation is as follows. Initially, the N-point input sequence $\{x(0), x(1), ..., x(N-1)\}$ is reordered to obtain the sequence $\{y(0), y(1), ..., y(N-1)\}$ defined by:

$$\begin{aligned} y(n) &= x(2n) \\ y(N-n-1) &= x(2n+1) \end{aligned} \qquad n = 0, ..., \frac{N}{2}-1 \quad \textbf{(8)}$$

Let $\{Y(0), Y(1), ..., Y(N-1)\}$ be the DFT of the sequence $\{y(0), y(1), ..., y(N-1)\}$. It can be shown [5] that the DCT sequence $\{X(0), X(1), ..., X(N-1)\}$ of the original sequence $\{x(n)\}$ can be obtained through the real part of $Z(n)$, defined by:

$$Z(n) = H_n Y(n) = \sqrt{\frac{2}{N}} K_n W_{4N}^n Y(n) \qquad W_{4N} = e^{-j\frac{2\pi}{4N}} \tag{9}$$

If the following property:

$$Z(N-n) = -jZ^*(n) \quad \Leftrightarrow \quad \mathrm{Re}[Z(N\text{-}n)] = -\mathrm{Im}[Z(n)] \tag{10}$$

is used, it is only necessary to compute $N/2+1$ values of $Z(n)$. Thus, the $N/2+1$ set of points {$Z(0)$, $Z(1)$, ..., $Z(N/4)$, $Z(N/2)$, Z($N/2+1$), ..., Z($3N/4-1$)} is proposed to minimize the number of operations in the N-point DCT computation. With this set, $N-3$ fewer adders and $N/2-2$ fewer multipliers are required. In this way, using this set and taking into account the property given in equation (10), the N-point DCT of the sequence {$x(0)$, $x(1)$, ..., $x(N-1)$} is given by {Re[$Z(0)$], Re[$Z(1)$], ..., Re[$Z(N/4)$], −Im[$Z(3N/4-1)$], −Im[$Z(3N/4-2)$], ..., −Im[$Z(N/2+1)$], Re[$Z(N/2)$], Re[$Z(N/2+1)$], ..., Re[$Z(3N/4-1)$], −Im[$Z(N/4)$], −Im[$Z(N/4-1)$], ..., −Im[$Z(1)$]}.

The computation of the DCT using complex arithmetic and the algorithm described above requires a high number of binary additions and multiplications. However, this drawback can be overcome by taking advantage of the reduced complexity QRNS multiplication. Thus, the architecture for a prearranged dynamic range is composed of a number of parallel channels with pairwise relatively prime moduli satisfying the QRNS mapping requirements. Fig. 1 shows the architecture to compute the 8-point DCT for a channel of the QRNS. The array is based on the Decimation In Frecuency (DIF) FFT [15, 16] algorithm with regular connections between adders and multipliers in all stages. Note that dotted components are not required by the algorithm and that, depending on the input (real or complex), they can be simplified according to (5). QRNS adders and multipliers are used instead of complex binary adders and multipliers. According to (5), each QRNS adder (multiplier) consists of two modular adders (two modular multipliers). A QRNS butterfly consists of an adder, a subtractor and a multiplier.

Regarding the 1-D DCT computation, the fact that the input sequence is real reduces each QRNS adder/subtractor with *real inputs* to only one real RNS adder/subtractor. In addition, each real "twiddle" factor multiplier with real input requires only one modular multiplier. Thus, the number of *real input* QRNS additions is:

$$A = \sum_{i=1}^{\log_2 N} 2^i \tag{11}$$

A direct calculation of the binary adders (*BA*), binary multipliers (*BM*), modular adders (*MA*) and modular multipliers (*MM*) required to compute the N-point 1-D DCT leads to:

$$\begin{aligned} BA &= (3N-1)(\log_2 N - 1) - 2A + 10 \\ BM &= (2N-1)(\log_2 N - 1) - A + 9 \\ MA &= (2N-1)(\log_2 N - 1) - A + 6 \\ MM &= (N-1)(\log_2 N - 1) + 5 \end{aligned} \tag{12}$$

Table 1 shows the number of real binary additions and multiplications required for the computation of the binary 8-point and 16-point DCTs, as well as the modular additions and modular multiplications required by each channel of the proposed QRNS alternative. QRNS provides a reduction in the number of operations required while maintaining the speed advantage of the RNS. Reduction in the additions required is 21.43% and 24.18% for the 8- and 16-point DCTs, respectively, while the reduction in the multiplications required is 24% and 30.56%.

N	*BA*	*BM*	*MA*	*MM*
8	28	25	22	19
16	91	72	69	50

Table 1. Arithmetic operations required by the binary 2's complement and QRNS 1-D DCT processors.

5 VHDL synthesis results

The implementation of an 8-point 1-D DCT processor was considered by its use in image and video coding standards (JPEG, MPEG) for multimedia applications. The synthesis of binary 2's complement and QRNS-based 1-D DCT processors through structural VHDL over Altera FLEX10KE devices were carried out to assess hardware requirements and compare throughput. These devices contain an embedded array to implement memory and specialized logic functions and a logic array to implement general logic. The embedded array consists of a series of EABs (Embedded Array Blocks). When an EAB is used as a memory, it provides 4096 bits. The logic array consists of LEs

(Logic Elements). Each LE provides a four-input LUT, a programmable flip-flop and dedicated signal paths for carry and cascade functions.

The 8-point input sequence {$x(0)$, $x(1)$, $x(2)$, $x(3)$, $x(4)$, $x(5)$, $x(6)$, $x(7)$} was initially represented with 8-bit precision, while complex "twiddle" factors and scale factors were represented with 10-bit precision. The wordwidth was extended as required in each stage, obtaining a final 32-bit dynamic range for the 1-D DCT computation. The necessary registers between arithmetic instances for a pipelined implementation were added. Regarding a QRNS-mapped implementation of the architecture described above, the required 32-bit dynamic range is achieved with the four 8-bit moduli {221, 229, 233, 241}, and the roots for the QRNS isomorphic mappings (5) and (7) are {47, 107, 89, 177}, respectively. 8-bit modulus channels are selected in order to ensure fast internal QRNS processing. Thus, it is previously necessary to convert the 8-bit real inputs {$x(0)$, $x(1)$, $x(2)$, $x(3)$, $x(4)$, $x(5)$, $x(6)$, $x(7)$} to the QRNS. According to (5), conversion of a real integer to the QRNS is easily simplified to just a residue computation only requiring an 8-bit comparator and an 8-bit subtractor. Thus, conversion to QRNS with an 8-bit modulus set is easily accomplished with few resources. The resulting QRNS architecture is shown in Fig. 1.

The results for both architectures are shown in Table 2. Hardware requirements were assessed in terms of the number of LEs and EABs while performance was evaluated in terms of the register-to-register maximum delay path. The throughput is measured in millions DCTs per second. The performance of the proposed QRNS-based 1-D DCT processor was up to 96.58% better when compared to equally pipelined traditional arithmetic implementations.

6 Conversion stage

The practical implementation of RNS-based systems encounters a difficulty in the conversion stage. Different solutions were addressed to overcome the conversion drawback. Binary-to-RNS and RNS-to-QRNS conversions are fast operations since the input and the selected moduli are 8-bit wide. According to (4) and (6), RNS-to-QRNS and QRNS-to-RNS conversions only require one adder, one substractor and one LUT. However, conversion from RNS to binary implies the use of 32-bit word-length modular adders and large multipliers. Thus, a direct implementation of the Chinese Remainder Theorem (CRT) results in excessive hardware usage and slow performance. The auto-scaling RNS-to-binary converter (ε-CRT) proposed by Griffin *et al* in [17] allows to overcome these drawbacks using only LUTs and conventional binary adders. In this way, for a scaled n-bit binary output and a k-bit modulus set, this converter needs one $2^k \times n$ LUT per modulus and a binary n-bit adder tree to add the LUT outputs. The converters were designed to have 8-, 16-, and 24-bit output and require 56, 163 y 321 LEs and 4, 4, and 9 EABs, respectively. 1Using pipeline in the binary adder tree the performance of the ε-CRT converter with 24, 16 and 8 bits output can reach the IO rate required and the high performance of the proposed QRNS-based DCT architecture is not degraded when converters are added.

7 Conclusion

This paper shows the design of a QRNS-enabled 1-D DCT processor for high performance image processing applications. The use of complex arithmetic benefits from the reduced complexity QRNS multiplication, thus reducing hardware complexity and leading to a high speed architecture suitable for VLSI implementation. The efficient QRNS computation of the radix-2 DIF FFT using only modular adders and small reduced latency LUTs, provides the intrinsic RNS speed and reduction in resource usage. When the proposed processor is compared with the corresponding binary 2's complement version, reduction in the number of additions required is 21.43% and 24% for the 8- and 16-point DCTs, respectively, while the reduction in the number of multiplications is 24.18% and 30.56%. Simulations of the proposed QRNS-based 8-point DCT architecture and the binary version using FPL devices have corroborated a sustained increase of up to 81.84% in performance using an 8-bit modulus set. Moreover, the use of a special class of auto-scaling converters means that binary-to-QRNS and QRNS-to-binary conversions do not to degrade the performance of the entire system.

Acknowledgement

The authors were supported by the Dirección General de Enseñanza Superior (Spain) under project PB98-1354.

		Binary 2's complement 1-D DCT Processor (Pipeline stages in the scaling multiplier)				QRNS 1-D DCT Processor
		(1)	(2)	(3)	(4)	
#LEs		2602	2579	2629	2879	4×672
#EABs		-	-	-	-	4×14
Throughput for several speed grade devices (MDCTPS)	**-1**	35.33	51.81	64.51	71.42	129.87
	-2	27.32	42.55	51.54	58.82	98.03
	-3	19.84	31.44	37.59	42.55	69.93

Table 2. Throughput and resource usage for both binary 2's complement and QRNS 1-D DCT processors.

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] R. M. Haralick, "A Storage Efficient Way to Implement the Discrete Cosine Transform", *IEEE Trans. on Computers*, vol. C-25, pp. 764-765, June 1976.

[4] B. D. Tseng and W. C. Miller, "On Computing the Discrete Cosine Transform", *IEEE Trans. on Computers*, vol. C-27, pp. 966-968, Oct. 1978.

[5] M. J. Narasimha and A. M. Peterson, "On the Computation of the Discrete Cosine Transform", *IEEE Trans. on Communications*, vol. COM-26, pp. 934-946, Jun. 1978.

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

[7] M. C. Vanwormhoudt, "Structural Properties of Complex Residue Rings Applied to Number Theoretic Fourier Transforms", *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 26, pp. 447-455, 1978.

[8] N. Szabo and R. Tanaka, *Residue Arithmetic and its Applications to Computer Technology*. McGraw-Hill, 1967.

[9] Altera Corporation, *1998 Data Book*, Jan. 1998.

[10] Xilinx Inc., The Programmable Logic Data Book, 1999.

[11] Actel Corporation, *ProASIC 500K Family Data Sheet,* 2000.

[12] 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. of 2000 IEEE Intl. Symposium. on Circuits and Systems*, vol. 5, pp. 321-324, May 2000.

[13] J. Ramírez, A. García, P. G. Fernández, A. Lloris, "FPL Implementation of a SIMD RISC RNS-enabled DSP", *4th World Multiconference on Circuits, Systems, Communications and Computers* (CSCC 2000), pp. 1281-1286, 2000.

[14] F. Taylor, C. Huang, "A Comparison of DFT Algorithms Using a Residue Architecture", *International Journal on Computers and Electrical Engineering*, vol. 8, no. 3, pp. 161-173, 1981.

[15] V. Boriakoff, "FFT Computation with Systolic Arrays, A New Architecture", *IEEE Trans. Circuits and Systs. II*, vol. 41, no. 4, 1994, pp. 278-284.

[16] J. Choi and V. Boriakoff, "A New Linear Systolic Array for FFT Computation" *IEEE Trans. Circuits and Systs. II*, vol. 39, no. 4, 1992, pp. 236-239.

[17] M. Griffin, F. J. Taylor and M. Sousa, "New scaling algorithms for the Chinese Remainder Theorem.", *Proc. of 22nd Asilomar Conference on Signals, Systems and Computers* (Pacific Grove, CA), 1988.