Journal of Circuits, Systems, and Computers

A FAST QRNS-BASED ALGORITHM FOR THE DCT AND ITS FIELD-PROGRAMMABLE LOGIC IMPLEMENTATION

JAVIER RAMÍREZ
Department of Electronics and Computer Technology,
University of Granada,
Campus Universitario Fuentenueva
Granada, 18071, Spain

ANTONIO GARCÍA
Department of Electronics and Computer Technology,
University of Granada,
Campus Universitario Fuentenueva
Granada, 18071, Spain

Received (received date)
Revised (revised date)
Accepted (accepted date)

This paper assesses the arithmetic benefits provided by the Residue Number System (RNS) for building Digital Signal Processing (DSP) systems with Field Programmable Logic (FPL) technology. The quantifiable benefits of this approach are studied in the context of a new Fast Cosine Transform (FCT) architecture enhanced by using the Quadratic Residue Number System (QRNS). The system reduces the number of adders and multipliers required for the N-point Discrete Cosine Transform (DCT) and provides high throughput. For an FPL-based implementation, the proposed design gets significant improvements over an equivalent 2C structure. By using up to 6-bit moduli, an overall increase in the system performance of about 140% is achieved. If this speed increase is considered along with the penalty in device resources, the presented QRNS-based FCT system provides an improvement in the area-delay figure factor of about 20%. Finally, the conversion overhead was carefully studied and it was found that the quantifiable benefits of the proposed design are not affected when converters are included.

1. Introduction

The design of high-performance, high-precision, real-time Digital Signal Processing (DSP) systems, such as those associated with Discrete Cosine Transform (DCT) image processing, is a challenging problem. The evolution of the DSP market and technology makes necessary considering not only cell-based Application Specific Integrated Circuits (ASICs) but also modern Complex Programmable Logic Device (CPLD) families, such as Altera FLEX10K [1] or Xilinx Virtex [2], in the design and implementation of modern DSP systems.

Traditional numbering systems are commonly used to build DSP systems with commercially available Field Programmable Logic (FPL) technology. The two's complement (2C) system has been adopted for a wide range of real-time applications

including digital communications, image, video and speech processing, multimedia systems, networking, etc. However, a review of the FPL vendor supplied application notes [3, 4] shows that these devices suffer from weak arithmetic performance when compared to carefully designed standard-cell based ASICs. While FPL houses champion their technology as a provider of system-on-a-chip (SOC) DSP solutions, engineers have historically viewed FPL as a prototyping technology. In order for FPL to begin to compete in areas currently controlled by low-end standard-cell ICs, a means must be found to more efficiently implement DSP objects.

An arithmetic system capable of surmounting this barrier is the Residue Number System, or RNS [5, 6]. Computer arithmeticians have long held that the RNS offers the best speed-area advantage in multiply and accumulate (MAC) intensive applications [7]. This paper develops a mechanism for achieving synergy within an FPL-defined environment to implement arithmetic intensive DSP solutions. FPL devices are organized in channels (typically 8-bits wide). Within these channels are found short delay propagation paths and dedicated memory blocks with programmable address and data spaces, which are commonly used to synthesize small RAM and ROM functions. Performance rapidly suffers when carry bits and/or data have to propagate across a channel boundary. We call this the channel barrier problem [8]. Existing 2C designs encounter the channel barrier problem whenever precision exceeds the channel width. An alternative design paradigm is advocated in this paper. The advantage is gained by reducing arithmetic to a set of concurrent operations that reside in small wordlength non-communicating channels. The quantifiable benefits of this approach are studied in the context of a design example, a Fast Cosine Transform (FCT) architecture enhanced by using the Quadratic Residue Number System (QRNS) [6]. This work will build upon previous works [9] and RNS-FPL design studies [10].

2. Residue Number System

Some of the well-known attributes of weighted numbering systems are: *i*) easy algebraic comparison, *ii*) dynamic range extension, *iii*) multiplication (division) by simple arithmetic shifts, and *iv*) simplified overflow and sign detection. However, the disadvantage of these systems is that carry information needs to be propagated from digits of lesser significance to those of greater significance. On the contrary, the RNS [5, 6] is a carry-free system defined in terms of a set of relatively prime moduli $\{m_1, m_2, ..., m_L\}$. Any integer X in the residue class $Z(M)$, being $M = m_1 m_2 ... m_L$, has a unique representation given by its residues $\{x_i = X \bmod m_i, i = 1, 2, ..., L\}$. If X and Y have RNS representations given by $[x_1, x_2, ..., x_L]$ and $[y_1, y_2, ..., y_L]$, respectively, and using $\diamond$ to represent $+$, $-$ or $\times$, operations modulo $M, Z = (X \diamond Y) \bmod M$, are computed in parallel over reduced wordwidth integer rings and are given by the L-tuple $[z_1, z_2, ..., z_L]$, where $z_i = (x_i \diamond y_i) \bmod m_i$. Thus, a significant throughput improvement is provided by the RNS since arithmetic is performed in parallel over reduced wordlength, high-performance RNS channels. The individual modular arithmetic RNS operations are normally reduced to small Look-Up Table

(LUT) calls, thus resulting adequate for FPL assimilation by taking full advantage of the built-in device resources. Moreover modulo adders [11] benefit from short carry propagation paths and are able to reside within a logic array block in which fast carry chains are defined, thus avoiding the propagation of carries or information across the device routing networks [8].

2.1. *Quadratic Residue Number System*

An extension of RNS for complex calculus is the QRNS [6], which is defined by an isomorphic mapping between the ring of complex integers modulo m_i and $Z(m_i) \times Z(m_i)$. The QRNS requires that the prime factor decomposition of m_i has only primes of the form $4K_i + 1$ or, equivalently, if equation $x^2 + 1$ can be factorized as $x^2 + 1 = (x - r_i)(x + r_i)$ with $r_i \in \{0, 1, ..., m_i - 1\}$.

Let $q_1 + jq_2$ be a complex number with $0 \leq q_1, q_2 < m_i$, and r_i a root of the equation $x_2 + 1 = 0$ in the ring of integers modulo m_i. Thus, QRNS is defined by the isomorphic mapping:

$$\begin{array}{c} q \equiv \ q_1 + jq_2 \overset{\text{QRNS}}{\longrightarrow} \bar{q} \equiv \ (\ \bar{q}_1,\ \bar{q}_2) \\ \bar{q}_1 = |q_1 + r_i q_2|_{m_i} \\ \bar{q}_2 = |q_1 - r_i q_2|_{m_i} \end{array} \tag{1}$$

The QRNS defines a domain transformation in which complex arithmetic is performed with a minimum complexity. Arithmetic in the QRNS is defined as follows:

$$(\bar{q}_1,\bar{q}_2) \diamondsuit (\bar{p}_1,\bar{p}_2) = \ \left(\ |\bar{q}_1 \diamondsuit \bar{p}_1|_{m_i}\ ,\ \ |\bar{q}_2 \diamondsuit \bar{p}_2|_{m_i}\right) \tag{2}$$

where $\diamondsuit$ represents addition, subtraction or multiplication. Finally, the inverse QRNS mapping is defined as:

$$\begin{array}{c} (\bar{q}_1,\ \bar{q}_2) \overset{\text{QRNS}^{-1}}{\longrightarrow} q_1 + jq_2 \\ q_1 = \ \left|\ 2^{-1}\ (\bar{q}_1 + \bar{q}_2)\ \right|_{m_i} \\ q_2 = \left|\ (2r_i)^{-1}\,(\bar{q}_1 - \bar{q}_2)\ \right|_{m_i} \end{array} \tag{3}$$

Thus, RNS-to-QRNS and QRNS-to-RNS conversion only require one adder, one subtractor and one LUT each. According to (2), a clear advantage of the QRNS is that a complex integer multiplication operation requires only two modular operations that can be performed in parallel over reduced wordwidth modular integer rings. However, a disadvantage of the QRNS is the hardware overhead introduced by conversion. It has been shown that, for fully parallel high-order DSP systems, the area overhead is maintained below 10% while speed is not affected when output converters are inserted in the system if an adequate design is used [12].

3. Design of the QRNS-enabled FCT processor

The DCT [15, 16] is a subject of study of modern digital image processing technology. Its inclusion in standards for image and video compression, such as JPEG

or MPEG, as the transform-coding technique, has motivated an increasing interest and numerous contributions have been proposed focusing on its fast implementation with reduced hardware complexity. Interest in the DCT derives from the fact that it approaches the statistically optimal Karhunen-Loeve Transform (KLT) providing a more efficient computational framework [16]. The DCT was introduced in 1974 by Ahmed [15] and, for an N-point sequence $x(n)$, is defined as:

$$\begin{array}{c} X(m) = \sqrt{\frac{2}{N}} K_m \sum_{n=0}^{N-1} x(n) \cos\left(\frac{(2n+1)m\pi}{2N}\right) \; m = 0,\ 1,\ ...,\ N-1 \\ K_0 = \frac{1}{\sqrt{2}}\ K_1 = \ K_2 = \ ... = \ K_{N-1} = \ 1 \end{array} \tag{4}$$

Algorithms for the fast computation of the Discrete Fourier Transform (DFT) have been widely used to compute the DCT [15, 17-19]. Ahmed [15] introduced the first algorithm to compute the N-point DCT through the DFT of a $2N$-point sequence. Haralick [17] showed that the N-point DCT could be computed with two N-point FFTs thus reducing the number of complex MAC operations. Tseng and Miller [18] proposed an algorithm with only $N/2{\cdot}(\log_2 N{+}1)$ complex operations. Narashima and Peterson [19] have proposed a more efficient algorithm based on a rearrangement of the input sequence that requires $(N{\cdot}\log_2 N{-}N{+}1)/4$ complex multiplications. Furthermore, several papers have shown the advantages of the RNS for the computation of the DCT and other discrete transforms [20-22]. In order to overcome the disadvantage of using complex arithmetic for those FFT-based algorithms, the use of QRNS [6] arithmetic is examined in this paper. The advantages of QRNS are RNS-like speed and reduced multiplication complexity. This paper will show a QRNS-based VLSI architecture to compute the N-point DCT with a reduction in hardware complexity and sustained performance increase.

3.1. *DCT algorithm*

The algorithm for the computation of the DCT using the QRNS is presented below. Initially, the N-point integer input sequence $x(n)$ is reordered as the sequence $y(n)$ defined by:

$$\begin{array}{l} y\,(n) = x\,(2n) \;\; n = 0,\ ...,\ \frac{N}{2} - 1 \\ y\,(N-n-1) = x\,(2n+1) \end{array} \tag{5}$$

The DCT sequence, namely $X(m)$, defined by equation (4), can be formulated as:

$$X(m) = \sum_{n=0}^{N/2-1} y(n) \cos\left[\frac{(4n+1)m\pi}{2N}\right] + \sum_{n=0}^{N/2-1} y(N-n-1) \cos\left[\frac{(4n+3)m\pi}{2N}\right] \tag{6}$$

It can be shown [19] that the transformed sequence, $X(m)$, can be computed by means of an N-point scaled DFT sequence. Let $Y(m)$ be the DFT of the sequence

$y(n)$. Thus, $X(m)$ is obtained through the real part of $Z(m)$, defined as follows:

$$\begin{gathered} Z\left(m\right) = \ H_m Y\left(m\right) = \ \sqrt{\tfrac{2}{N}} K_m W_{4N}^m Y\left(m\right) \ \ W_{4N}^m = \mathrm{e}^{\ -\ j\frac{2_{\pi m}}{4_N}} \\ K_0 = \tfrac{1}{\sqrt{2}} K_1 = K_2 = ... = K_{N-1} = 1 \end{gathered} \tag{7}$$

The algorithm described above, in opposition to other proposals [15, 17, 18], reduces efficiently the DCT computation to a single $N-$point DFT calculation and only needs to compute $(N{\cdot}\log_2 N - N{+}1)/4$ complex MAC operations when a radix-2 FFT algorithm is used. Note that if the following property:

$$Z(N-m) = -jZ^*(m) \Longleftrightarrow Re[Z(N-m)] = -Im[Z(m)] \tag{8}$$

is used, it is only necessary to compute $N/2{+}1$ values of the sequence $Z(m)$. In order to minimize the number of multiplications required, we have proposed the set of $N/2{+}1$ points $\{Z(0),\ Z(1),\ ...,\ Z(N/4),\ Z(N/2),\ Z(N/2{+}1),\ ...,\ Z(3N/4\text{-}1)\}$. With this set, $N-3$ fewer complex adders and $N/2$-2 fewer complex multipliers need to be computed and, finally, using the relation given in equation (8), the N-point DCT of the input sequence $x(n)$ is given by {Re[$Z(0)$], Re[$Z(1)$], ..., Re[$Z(N/4)$], -Im[$Z(3N/4\text{-}1)$], $-$Im[$Z(3N/4\text{-}2)$], ..., -Im[$Z(N/2{+}1)$], Re[$Z(N/2)$], Re[$Z(N/2{+}1)$], ..., Re[$Z(3N/4\text{-}1)$], -Im[$Z(N/4)$], -Im[$Z(N/4\text{-}1)$], ..., $-$Im[$Z(1)$]}.

Thus, the DCT is obtained by means of a highly efficient algorithm in two steps:

(i) compute the DFT of the reordered sequence using a radix-2 Decimation In Frequency (DIF) FFT algorithm and,
(ii) multiply by the necessary scaling factors.

Step (i) is simplified by using the relation shown in equation (8), while scaling by the H_m coefficients is only required for $N/2{+}1$ output values.

3.2. *QRNS-DCT merged architecture*

The computation of the DCT using conventional 2C arithmetic and the algorithm described above requires a large number of binary additions and multiplications. This drawback can be overcome by taking advantage of the arithmetic benefits provided by the QRNS. The architecture for a prearranged dynamic range is composed of a number of parallel channels with moduli satisfying the QRNS mapping requirements. Fig. 1 shows the QRNS channel architecture needed to compute the 8-point DCT. The array is based on the radix-2 DIF FFT [23, 24] algorithm, which benefits from regular connections between intermediate processing stages. As it was previously mentioned, it is only necessary to compute five values of the $Z(m)$ sequence: $Z(0)$, $Z(1)$, $Z(2)$, $Z(4)$ and $Z(5)$. Because of this algorithm simplification,

shadowed components in the figure are eliminated from signal paths.

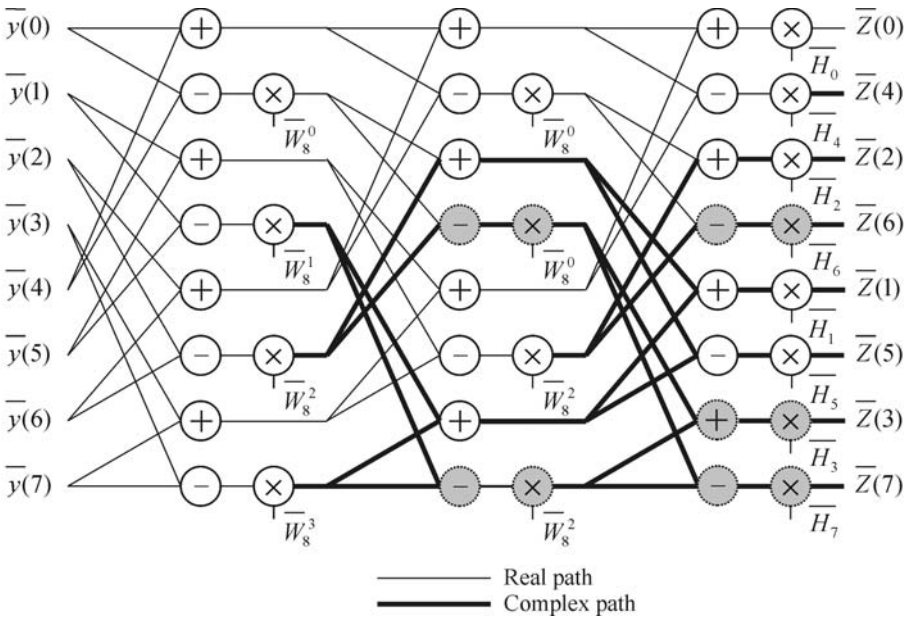

Fig. 1. QRNS-based radix-2 DIF FFT architecture for the 8-point DCT.

This structure, when compared to traditional FFT architectures, replaces complex binary adders and multipliers by QRNS adders and multipliers. Each QRNS adder, subtractor, or multiplier consists of two modular adders, subtractors, or multipliers, respectively. Since the input sequence and many of the *twiddle* factors are real, there are QRNS arithmetic modules in the array that have real inputs and, thus, can be reduced to a single modular adder, subtractor or LUT. A QRNS butterfly, shown in Fig. 2, consists of an adder, a subtractor and a constant coefficient modular multiplier. These arithmetic modules are easily mapped onto hardware providing the best area-speed performance.

3.3. *Complexity analysis*

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.

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:

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

where A, the number of *real input* QRNS additions, is:

$$A = \sum_{i=1}^{\log_2 N} 2^i \tag{10}$$

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% and 24% for the 8- and 16-point DCTs, respectively, while the reduction in the multiplications required is 24% and 31%.

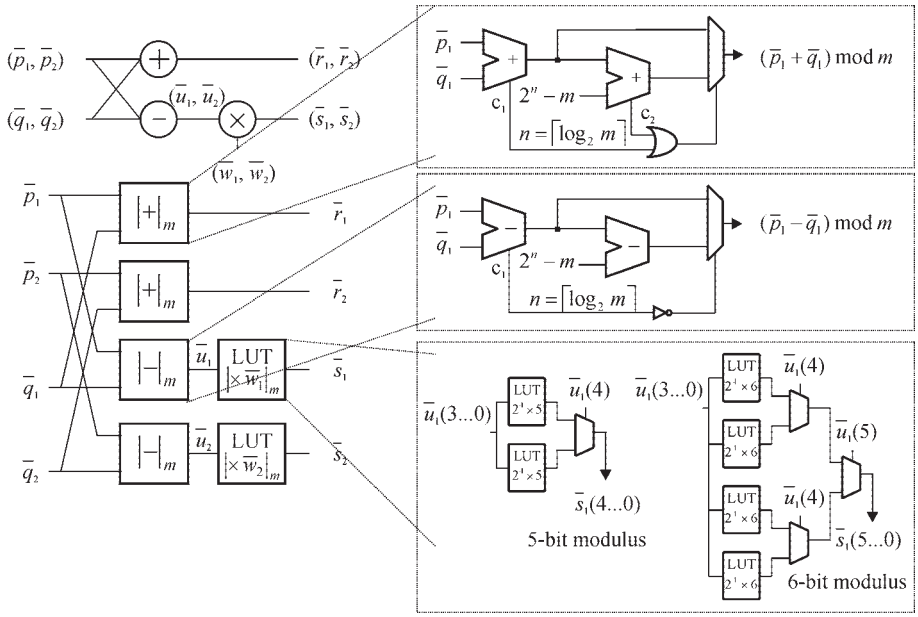

Fig. 2. Design of QRNS-based radix-2 DIF FFT butterflies.

It is interesting to analyze accurately the complexity of a QRNS-based design in terms of a radix-2 FFT butterfly. Fig. 2 shows the best implementation of such a system. For modulo adders and subtractors, a two-stage CPA (carry propagate adder)-based design [11] is normally the best choice and requires $2(n+1)$ FAs (full

Table 1. Arithmetic operations required by 2C and QRNS DCT processors.

N	**Additions**		**Multiplications**	
	2C	**QRNS**	**2C**	**QRNS**
8	28	22	25	19
16	91	69	72	50
N	$(3N\text{-}1)(\log_2 N\text{-}1)\text{-}2A\text{+}10$	$(2N\text{-}1)(\log_2 N\text{-}1)\text{-}A\text{+}6$	$(2N\text{-}1)(\log_2 N\text{-}1)\text{-}A\text{+}9$	$(N\text{-}1)(\log_2 N\text{-}1)\text{+}5$

adders) and one multiplexor, with n being the modulus width. A more complex design computes the two possible results in parallel and uses carry save adders (CSAs) for the three-term summation. It requires n+1 FAs, an n-bit CSA and one multiplexor. This design has a short delay but it requires more resources, thus resulting only appropriate for an ASIC design since, normally, for FPL devices carry chains provide the best area-speed performance. For the two modulo multipliers of the radix-2 butterfly, LUTs are the best option. Multiplication by the *twiddle* factors is reduced to a LUT call and built using high-speed small ROMs. Fig. 2 shows the design of the LUT-based multiplier for 5- and 6-bit modulus. The $2^5\times5$ and $2^6\times6$ tables are built with smaller LUTs and multiplexors to take advantage of the FPL target technology. In this way, the $2^5\times5$ LUT requires two $2^4\times5$ LUTs (5 logic elements, LEs) and a multiplexor, while the $2^6\times6$ LUT needs four $2^4\times6$ LUTs (6 LEs) and three multiplexors. For smaller moduli it is not required to break the table, while for larger moduli, up to 8-bit wide, other built-in FPL memory resources become available. Moreover, this procedure does not result optimal only for FPL-centric designs, but it is equally interesting for standard-cell-based VLSI CMOS chips [25].

4. RNS-FPL synergic implementation

As it was argued in the introduction section, FPL devices suffer from weak arithmetic performance when compared to standard cell VLSI technology. When the implementation of a 2C multiplier using an Altera FLEX10K reference device is considered, speed is affected by the increasing precision. Even with an adequate segmentation, speed limitations are caused by the increasing length of carry chains [3, 4]. This drawback represents a serious implementation obstacle for most of the emerging DSP systems demanding high-precision and high-performance real-time data processing. RNS arithmetic modules are able to operate at higher data rates. Fast RNS-FPL merged adders have been proposed recently [8]. In this paper, new QRNS-FPL arithmetic modules are defined and the implementation of the widely used 8-point FCT is carried out to assess the advantages and disadvantages of the proposed methodology. In order to facilitate efficient RNS-centric FPGA designs, a collection of RNS VHDL models for modular adders, QRNS arithmetic modules and RNS converters were developed for the Altera FLEX10K target technology, specifically for FLEX10KE devices. The resources found in these devices are ideal for RNS-based systems. Modulo adders are defined within short carry chains logic array blocks (LABs) (8-bit long), pipelining does not require additional resources

and memory blocks (providing up to 4096 bits) are optimal to implement LUT calls at high speed.

The implementation of an 8-point FCT processor was considered because of its wide use in image and video coding systems including JPEG and MPEG standards for multimedia applications. The QRNS-based 1-D DCT processors was built on a FPGA-based system by means of a defined RNS-FPL VHDL model optimised for the FLEX10K target technology. The VHDL high-level synthesis of an equivalent 2C design was carried out to assess hardware requirements and compare throughput. The 8-point input sequence $\{x(n)\}$ (n= 0, 1, ..., 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 output sequence. The resulting QRNS architecture is shown in Fig. 1, commented above, and makes use of the FPL-enhanced QRNS butterfly shown in Fig. 2. Two different alternatives were used for the QRNS-based system. The first option takes advantage of the built-in EABs (Embedded Array Block) providing up to 4K-bit LUTs for high-speed modulo multiplication . The 32-bit dynamic range of the system is ensured by using four 8-bit moduli {221, 229, 233, 241}, with roots {47, 107, 89, 177} for the QRNS isomorphic mappings (1) and (3), respectively. 8-bit modulus channels were selected in order to provide fast internal QRNS processing. Thus, it is previously necessary to convert the 8-bit real inputs to the QRNS. According to (1), conversion of a real integer to the QRNS is easily simplified to just a residue computation only requiring one 8-bit comparator and one 8-bit subtractor [9]. An alternate design does not make use of embedded memory blocks and reduce the modulus wordwidth to a maximum of six bits. Thus, the $2^n \times n$ LUTs are synthesized through a number of $2^4 \times n$ LUTs (requiring only n LEs) and a multiplexing scheme similar to that shown in Fig. 2. For the binary-to-RNS conversion, input-block decomposition was used. 4-bit blocks were found to be an attractive choice, since it allows the direct mapping of the converter LUTs on LEs.

Table 2. Throughput and resource usage for both 2C and QRNS 1-D DCT processors.

		2C				**QRNS**	
		# stages (scaling multiplier)				Modulus set	
		1	2	3	4	{221, 229, 233, 241}	{53, 41, 29, 25, 17, 13, 5}
#LEs		2602	2579	2629	2879	4×672	5449
#EABs		-	-	-	-	4×14	-
F(MHz)	-1 grade	32	52	65	71	130	167
	-2 grade	27	43	52	59	98	141
	-3 grade	20	31	38	43	70	108

Table 2 compares the QRNS-based FCT processor with the 2C equivalent system. The table includes the resources required for both structures and their maximum operating frequency. For a 2C design, pipelined constant coefficient multipliers were used. The synthesis process followed Altera's recommendations, but it was noticed that performance rapidly degrades because of the carry propagation delays

across the multiplier. Even with a deep segmentation strategy, the 2C version was found to be much slower than the proposed QRNS-based FCT system. Particularly, the maximum operating frequency of a 2C design with 4-stage pipelined scaling multipliers was found to be 71, 59 and 43 MHz for –1, –2 and –3 speed grade FLEX10KE devices, respectively. For the proposed QRNS-based FCT processors, two-modulus set were explored: *i*) 8-bit moduli, and *ii*) an unbalanced modulus set consisting of up to 6-bit moduli. For the first modulus set, the $2^8 \times$8-bit LUT multipliers are mapped on EABs. Since each EAB provides a double precision $2^8 \times$16-bit LUT and some of the multipliers have the same address, the 19 multipliers of the system required only 14 EABs. The system can operate at 130, 98 and 70 MHz for –1, –2 and –3 speed grade FLEX10KE devices, respectively. Thus, the proposed QRNS-based processor is 83%, 66% and 63% faster than a classical 2C implementation for the studied speed grades. However, this QRNS-enabled system requires a considerable number of EABs. In order to palliate this drawback and to maximize the throughput of the system, a second option was explored. This alternative uses up to 6-bit moduli with the LUTs being mapped on LEs by decomposing the LUT multipliers in $2^4 \times n$ LUTs, where n is the modulus width in bits. Thus, the system is synthesized using only LEs and speed is increased to 167, 141 and 108 MHz for –1, –2 and –3 speed grade devices, respectively. In this case, the overall increase of the system performance is 135%, 139% and 151% over equivalent 2C processors. On the other hand, if the speed increase is considered along with the penalty in device resources, the QRNS-based FCT system provides an increase in the area-delay figure factor of about 23%, 21% and 15% for –1, –2 and –3 speed grade devices, respectively. The practical implementation of RNS-based systems encounters a serious 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. According to (1) and (2), RNS-to-QRNS and QRNS-to-RNS conversions only require one adder, one subtractor and one LUT. However, conversion from RNS to binary implies the use of 32-bit modular adders and large multipliers. Thus, a direct implementation of the Chinese Remainder Theorem (CRT) results in excessive hardware usage and considerable performance degradation. The auto-scaling RNS-to-binary converter (ε-CRT) [12], originally reported in [13] and later discussed in [14], 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, this converter needs one $2^k \times n$ LUT per modulus and a binary n-bit adder tree to add the LUT outputs. Table 3 shows hardware requirements for this converter when mapped on –1, –2 and –3 speed grade Altera FLEX10KE devices. Concretely, ε-CRT converters with 24-, 16- and 8-bit outputs operate at 135.13 MHz in a –1 speed grade device by means of a 3-, 2- and 1-stage pipelined adder tree, respectively. Moreover, it was found that the converters surpass the performance of the QRNS architecture for all the considered speed grades and precisions. On the other hand, if serial output converters are used, 321 LEs and 9 EABs are required for a 24-bit output while for 16- and 8-bit output converters, only 163 LEs and 4 EABs or 56 LEs and 4 EABS

are needed, respectively. Thus, simulations showed that the high performance of the proposed QRNS-based DCT architecture is not degraded when converters are added.

Table 3. Implementation of auto-scaling ε-CRT RNS-to-2C converter.

	Different output precision ε-CRT converters					
	{221, 229, 233, 241}			{53, 41, 29, 25, 17, 13, 5}		
	24-bit	16-bit	8-bit	24-bit	16-bit	8-bit
#LEs	321	163	56	898	548	248
#EABs	9	4	4	-	-	-

5. Conclusion

This paper considers the design and implementation of FCT architectures using the QRNS and FPL technology. To achieve maximum performance, a new algorithm was developed and its efficient implementation on FPL devices was shown. With this, and other innovations, the presented QRNS architecture was shown to be well suited for integrating digital image processing systems within FPL devices. To assess the improvement of the proposed methodology, an 8-point DCT was used as a standard. For that concrete processor, the proposed FCT architecture reduces the number of arithmetic instances and, when area and speed are compared to a classical 2C implementation, it leads to a 140% improvement in speed, while the area-delay figure factor is about 20% better. Finally, the conversion overhead was carefully assessed and the conducted simulations showed that the hardware-speed benefits provided by this system are not affected when optimized converters are included.

Acknowledgements

The authors were supported by the Comisión Interministerial de Ciencia y Tecnología (CICYT, Spain) under project PB98-1354. CAD tools and supporting material were provided by Altera Corp., San Jose, CA, under the Altera University Program.

References

1. Altera Corp., "FLEX10K Embedded Programmable Logic Device Family v. 4.1", *http://www.altera.com/literature/ds/dsf10k.pdf.* (2001).
2. Xilinx Inc., "Virtex 2.5V Field Programmable Gate Arrays Data Sheet v. 2.6" *http://www.xilinx.com/partinfo/ds003-2.pdf.* (2001)
3. Altera Corp., "Implementing FIR Filters in FLEX Devices v.1.01", *http://www.altera.com/literature/an/an073.pdf.* (1998) .
4. Xilinx Inc., "Transposed Form FIR Filters v. 1.2", *http://www.xilinx.com/xapp/xapp219.pdf.* (2001).
5. N. S. Szabo and R. I. Tanaka, *Residue Arithmetic and Its Applications to Computer Technology*, McGraw-Hill, NY, 1967.

6. M. A. Soderstrand, W. K. Jenkins, G. A. Jullien and F. J. Taylor, *Residue Number System Arithmetic: Modern Applications in Digital Signal Processing*, IEEE Press, 1986.
7. F. J. Taylor, "Residue arithmetic: a tutorial with examples", *IEEE Computer*, **17 (5)** (1984) 50–62.
8. U. Meyer-Bäse, A. García and F. J. Taylor, "Implementation of a Communications Channelizer using FPGAs and RNS Arithmetic", *Journal of VLSI Signal Processing*, **28 (1/2)** (2001) 115–128.
9. 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 the 2000 International Symposium on Circuits and Systems*, **(5)** (2000) 321–324.
10. J. Ramírez, A. García, P. G. Fernández, L. Parrilla and A. Lloris, "RNS-FPL Merged Architectures for Orthogonal DWT", *Electronics Letters*, **36 (4)** (2000) 1198–1199.
11. M. Dugdale, "VLSI Implementation of Residue Adders Based on Binary Adders", *IEEE Transactions on Circuits and Systems II*, **39 (5)** (1992) 325–329.
12. M. Griffin, F. J. Taylor and M. Sousa, "New scaling algorithms for the Chinese Remainder Theorem", *Proceedings of the 22nd Asilomar Conference on Signals, Systems and Computers*, (1988).
13. I. Ja. Akushskii, V. M. Burcev and I. T. Pak, "A new positional characteristic of nonpositional codes and its applications", *Coding Theory and Optimization of Complex Systems* (1977).
14. J. Gonella, "The application of core functions to residue number systems," *IEEE Transactions on Signal Processing*, **39 (1)** (1991) 284–288.
15. N. Ahmed, T. Natarajam and K. R. Rao, "Discrete Cosine Transform", *IEEE Transactions on Computers*, **23 (1)**, (1974) 90–93.
16. K. R. Rao and P. Yip, Discrete Cosine Transforms. Algorithms, Advantages, Applications, *Academic Press. Inc* , (1990).
17. R. M. Haralick, "A Storage Efficient Way to Implement the Discrete Cosine Transform", *IEEE Transactions on Computers*, **25 (6)** (1976) 764–765.
18. B. D. Tseng and W. C. Miller, "On Computing the Discrete Cosine Transform", *IEEE Transactions on Computers*, **27 (10)**, (1978) 966–968.
19. M. J. Narashima and A. M. Peterson, "On the Computation of the Discrete Cosine Transform", *IEEE Transactions on Communications*, **(26)**, (1978) 934–946.
20. V. S. Dimitrov, G. A. Jullien and W. C. Miller, "A Residue Number System Implementation of Real Orthogonal Transforms", *IEEE Transactions on Signal Processing*, **46 (3)**, (1998) 563–570.
21. A. L. Bequilard and S. D. O'Neil, "Systolic RNS computation of the two-dimensional DCT in a ring of algebraic integers", *Proceedings of the 20th Annual Conf. Inform. Sci. Syst.*, (1986).
22. 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", *Proceedings of the 33rd Asilomar Conference on Signals, Systems and Computer*, **(2)**, (1999) 1302–1306.
23. V. Boriakoff, "FFT Computation with Systolic Arrays, A New Architecture", *IEEE Transactions on Circuits and Systems II*, **41 (4)** (1994) 278–284.
24. J. Choi and V. Boriakoff, "A New Linear Systolic Array for FFT Computation", *IEEE Transactions on Circuits and Systems II*, **39 (4)** (1992) 236–239.
25. J. Ramírez, P. G. Fernández, U. Meyer-Bäse, F. Taylor, A. García, A. Lloris, "Index-based RNS DWT Architectures for Custom IC Designs", *Proc. of the IEEE Workshop on Signal Processing Systems*, (2001) 70–79.